# AdaPool: Exponential Adaptive Pooling for Information-Retaining Downsampling

Alexandros Stergiou, *Student Member, IEEE*, and Ronald Poppe, *Senior Member, IEEE*

***Abstract*— Pooling layers are essential building blocks of convolutional neural networks (CNNs), to reduce computational overhead and increase the receptive fields of proceeding convolutional operations. Their goal is to produce downsampled volumes that closely resemble the input volume while, ideally, also being computationally and memory efficient. Meeting both these requirements remains a challenge. To this end, we propose an adaptive and exponentially weighted pooling method: *adaPool*. Our method learns a regional-specific fusion of two sets of pooling kernels that are based on the exponent of the Dice-Sørensen coefficient and the exponential maximum, respectively. AdaPool improves the preservation of detail on a range of tasks including image and video classification and object detection. A key property of adaPool is its bidirectional nature. In contrast to common pooling methods, the learned weights can also be used to upsample activation maps. We term this method adaUnPool. We evaluate *adaUnPool* on image and video super-resolution and frame interpolation. For benchmarking, we introduce *Inter4K*, a novel high-quality, high frame-rate video dataset. Our experiments demonstrate that adaPool systematically achieves better results across tasks and backbones, while introducing a minor additional computational and memory overhead.**

***Index Terms*— Pooling, downsampling, upsampling.**

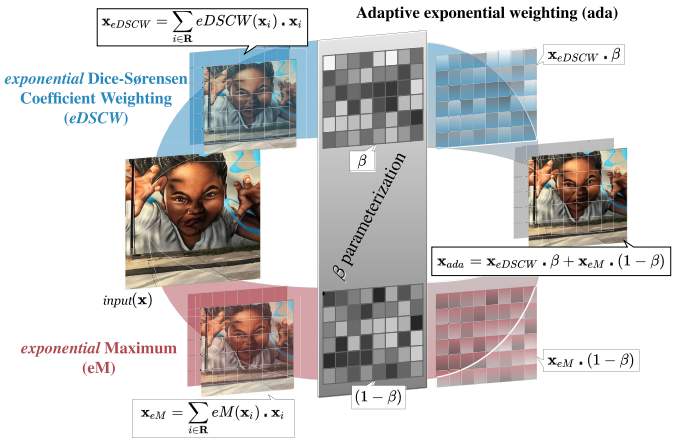



Fig. 1. **AdaPool downsampling**. The output is the combination of two processes. The first uses exponential Dice-Sørensen Coefficient Weighting (*e*DSCW) downsampling, based on a region's mean ($\overline{\mathbf{x}}$). The second downsamples using the exponential maximum (*e*M). Both outputs ($\mathbf{x}_{eM}$,$\mathbf{x}_{eDSCW}$) are summed with region-based weight masks $\boldsymbol{\beta}$ and (1-$\boldsymbol{\beta}$) to produce the adaptively weighted output ($\mathbf{x}_{ada}$).

## I. Introduction

POOLING methods downsample spatial input to a lower resolution. Their goal is to minimize the computational overhead of subsequent network operations and to increase their receptive fields. Pooling operations are essential in image and video processing approaches, including those based on CNNs. An important aspect of pooling is that it introduces a loss of information within the model. Thus, the retainment of detail in the structural aspects of the input, such as contrast and texture, can become challenging. As pooling is a key component in virtually all popular CNN architectures, it is necessary to ensure that this information loss does not incur a cost in performance.

A range of pooling methods has been proposed, each with different properties (see Section II). Most architectures use maximum or average pooling, both of which are fast and memory efficient but leave room for improvement in terms of retaining information. Other approaches use trainable subnetworks. Such methods have shown some improvements over average or maximum pooling, but they are typically less efficient and not generally applicable because their parameters need to be determined beforehand.

In this work, we study how the shortcomings of pooling methods can be addressed with low-computational approaches based on exponential weighting. We introduce methods to weigh kernel regions, either based on the softmax-weighted sum of activations [9], or based on the exponent of the similarity between each activation and the mean activation within the kernel region obtained by the Dice-Sørensen Coefficient [7], [8]. We then propose *adaPool* as the learned fusion of both methods, schematically visualized in Figure 1. AdaPool does not average over high-frequency patterns as in average pooling, nor does it focus exclusively on such patterns as in maximum pooling. Instead, adaPool provides a balance between retaining informative detail and the local image structure.

Manuscript received 25 October 2021; revised 21 June 2022 and 23 September 2022; accepted 22 November 2022. Date of publication 12 December 2022; date of current version 21 December 2022. The article was supported by the Netherlands Organization for Scientific Research (NWO) through the TOP-C2 grant "Automatic recognition of bodily interactions" (ARBITER). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Chandra Sekhar Seelamantula. *(Corresponding author: Alexandros Stergiou.)*

Alexandros Stergiou was with Utrecht University, 3584 CC Utrecht, The Netherlands. He is now with the Department of Computer Science, University of Bristol, BS8 1UB Bristol, U.K. (e-mail: alexandros.stergiou@bristol.ac.uk).

Ronald Poppe is with the Department of Information and Computing Sciences, Utrecht University, 3584 CC Utrecht, The Netherlands (e-mail: r.w.poppe@uu.nl).

Digital Object Identifier 10.1109/TIP.2022.3227503

This work is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 License.
For more information, see https://creativecommons.org/licenses/by-nc-nd/4.0/

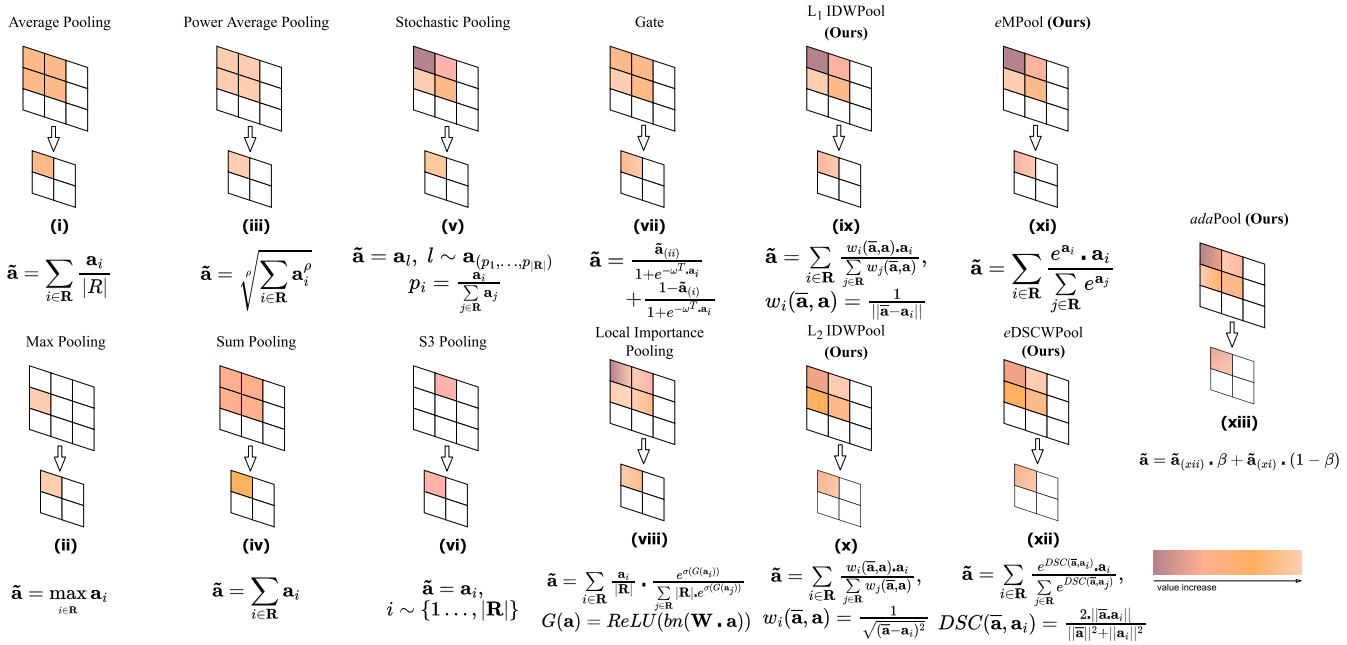



Fig. 2. **Pooling variants**. **R** denotes the kernel neighborhood as a set of pixels. (i-ii) **Average** and **maximum pooling** are based on the average or maximum activation value of the kernel region. (iii) **Power-average pooling** [1], [2] is proportional to average pooling raised to the power of $\rho$. When $\rho \to \infty$ the output equals maximum pooling, while $\rho = 1$ equals average pooling. (iv) **Sum pooling** is also proportional to average pooling with all kernel activations summed in the output. (v) **Stochastic pooling** [3] samples a random activation from the kernel region. (vi) **Stochastic Spatial Sampling** (S3Pool) [4] samples horizontal and vertical regions given a specified stride. (vi) **Gate pooling** [5] uses max-average pooling based on a gating mask ($\omega$) and a sigmoid function. (viii) **Local Importance Pooling** (LIP) [6] uses a trainable sub-net $G$ to enhance specific features. (ix-x) **$L_1$ and $L_2$ Inverse Distance Weighting Pooling** (IDW, ours) weighs kernel regions based on their inverse distance to the mean activation ($\overline{\mathbf{a}}$). (xi) **Exponential maximum pooling** (*e*mPool/SoftPool, ours) exponentially weighs activations using a softmax kernel. (xii) **Exponential Dice-Sørensen Coefficient Weighting Pooling** (*e*DSCWPool, ours) uses the exponent Dice-Sørensen Coefficient [7], [8] of the kernel activations ($\mathbf{a}_i$) and their average ($\overline{\mathbf{a}}$) as weights. (xiii) **Adaptive exponential pooling** (adaPool, ours) combines (xi) and (xii) with a trainable mask of weights $\boldsymbol{\beta}$.

Many tasks, including instance segmentation, image generation and super-resolution, require upsampling of inputs or signals, which has the inverse goal of pooling. With the exception of LiftPool [10], pooling operations cannot be reversed as this would lead to sparse upsampling results (e.g., using maximum pooling [11]). Common upsampling approaches such as interpolation, transposed convolutions and de-convolution approximate, rather than reconstruct, the higher-resolution features. The lack of inclusion of prior knowledge is an obstacle as the encoding of information to a lower resolution comes at a loss of local information. Instead, we argue that introducing prior local knowledge benefits the upsampling process. Based on the same formulation as adaPool, we propose *adaUnPool* for upsampling.

We demonstrate the favorable effects of adaPool in preserving descriptive features. Consequently, this allows models with adaPool to consistently improve classification and recognition performance. AdaPool maintains a low computational cost and provides an approach to retain prior information. We further introduce adaUnPool and address super-resolution and interpolation tasks. Summarized, we make the following contributions:

- We adapt Inverse Distance Weighting (IDW) [12] for pooling and extend it by using a similarity measure through the Dice-Sørensen Coefficient (DSC), by utilizing its exponent *e*DSC to weigh kernel elements.
- We propose adaPool, a parameterized learnable fusion of portions from the smooth approximation of the maximum and average. Using the inverse formulation, we develop upsampling process adaUnPool.
- We introduce a collection of 1,000 4K videos with high frame-rates, *Inter4K*, to benchmark frame super-resolution and interpolation algorithms.
- We experiment on multiple global and local-based tasks including image and video classification, and object detection. We show consistent improvements by replacing original pooling layers with adaPool. We also demonstrate the improved performance of adaUnPool on image and video super-resolution and video frame interpolation.

The remainder of the paper is structured as follows. We first discuss related work. We then detail our downsampling methods *e*DSCPool, *e*MPool, and adaPool as well as upsampling method adaUnPool (Section III). We introduce Inter4K in Section IV and evaluate on global and local-based image and video tasks (Section V). We conclude in Section VII.

## II. Related Work

### A. Pooling Hand-Crafted Features

Downsampling has been widely used in hand-coded feature extraction. In Bag-of-Words (BoW, [13]), images are represented as groups of local patches that are pooled and then encoded as vectors [14]. Based on this approach, Spatial Pyramid Matching (SPM) [15] aims at preserving spatial information. Later works extend this approach with linear SPM [16] that selects the maximum SIFT features in a spatial region. Most of the early works on feature pooling have focused on


Fig. 3. **Example of detail preservation with different pooling methods**. Common methods such as average and maximum pooling result in a distorted signature with unrecognizable details such as numbers or characters. Exponential weighting through either normalized local maximum (*e*M) or similarity-based measures (*e*DSCW) better capture details. Further improvements in the detail and representation quality are observed when introducing an adaptive fusion between both of these exponential weighting methods (adaPool).

max-pooling based on the max-like behavior of biological cortex signals [17]. Maximum and average pooling studies in terms of information preservation by Boureau et al. [18] have suggested that max-pooling produces comparatively more representative results in low feature activation settings.

### *B. Pooling in CNNs*

With the prevalence of learned feature approaches in various computer vision tasks, pooling methods have also been adapted to kernel-based operations. In CNNs, pooling has been mainly used to create condensed feature representations to reduce the model's computational requirements, and in turn to enable the creation of deeper architectures [19].

More recently, the preservation of relevant features during downsampling has taken a more prominent role. Initial approaches include stochastic pooling [3], which uses a probabilistic weighted sampling of activations within a kernel region. Other pooling methods such as mixed pooling are based on a combination of maximum and average pooling, either probabilistically [20] or through a combination of portions from each method [5]. Power Average ($L_p$) [2] utilizes a learned parameter $p$ to determine the relative importance of average and maximum pooling. With $p = 1$, sum pooling is used, while $p \to \infty$ corresponds to max-pooling.

Some approaches use grid-sampling. S3Pool [4] randomly samples rows and columns of the original feature map to create the downsampled version. Methods can also employ learned weights such as in Detail Preserving Pooling (DPP, [21]) that uses average pooling while enhancing activations with above-average values. Local Importance Pooling (LIP, [6]) utilizes learned weights within a sub-network attention mechanism. A visual and mathematical overview of the operations performed by different pooling methods appears in Figure 2.

The majority of the pooling work reported in the literature cannot be inverted for upsampling. Badrinarayanan et al. [11] proposed an inversion of the maximum operation by tracking the in-kernel position of the selected maximum input while the other positions are populated by zero values in the upsampled output. This ensures that the original values are used, but the output is inherently sparse. Recently, Zhao and Snoek [10] proposed LiftPool based on the use of four learnable sub-bands of the input. The produced output is composed as a mixture of the discovered sub-bands. They also propose an upsampling inversion of their approach (LiftUpPool). Both methods are based on sub-network structures that limit their usability as a computation and memory-efficient pooling technique.

Most of the aforementioned methods rely on combinations of maximum and average pooling, or the inclusion of sub-networks that prohibit low-compute and efficient downsampling. Instead of combining existing methods, our work is based on an adaptive exponential weighting approach to improve the retention of information and to better preserve details of the original signal. Our proposed method, adaPool, is inspired by Luce's choice of axiom [22]. We thus weigh kernel regions based on their relevance without being affected by the neighboring kernel vectors. This is in contrast to both average and maximum pooling. AdaPool uses two sets of pooling kernels. The first uses the channel-wise similarity of individual kernel vectors to their mean in order to determine their relevance. Similarities are calculated based on the Dice-Sørensen coefficient. The second is based on softmax weighting to amplify feature activations of greater intensity [9]. Finally, outputs from both kernel operations are parametrically fused to a single volume. Parameters are specific to each kernel location thus making our approach regionally-adaptive.

A key property of adaPool is that gradients are calculated for each kernel vector during backpropagation. This improves the network connectivity. In addition, downsampled regions are less likely to exhibit a vanishing trend of activations, as observed by equal-contribution approaches such as average or sum pooling. We demonstrate how adaPool can adaptively capture details in Figure 3, where the zoomed-in region displays a signature. AdaPool shows improvements in the clarity and recognizability of the letters and numbers.

## III. Methodology

In this section, we introduce the two processes (Sections III-A and III-B) that make up the final adaPool

method (Section III-C. We subsequently introduce the inverse adaUnPool method in Section III-D).

We start by introducing the basic operations of our pooling method. We define the local kernel region $\mathbf{R}$ as part of activation map $\mathbf{a}$ of size $C \times H \times W$, with $C$ channels, height $H$ and width $W$. For notation simplicity, we omit the channel dimension and assume that $\mathbf{R}$ is the set of relative position indices corresponding to the activations in the 2D spatial region of $k! \times k$ (i.e., $|\mathbf{R}| = k^2$). We denote the pooling output as $\widetilde{\mathbf{a}}$ and the corresponding gradients as $\nabla\widetilde{\mathbf{a}}_i$, at the $i^{th}$ coordinate within region $\mathbf{R}$.

### A. Smooth Approximated Average Pooling

Average pooling uses equal weights for all input vectors within a kernel region. The combined outputs are therefore strongly affected by outliers within the region. We argue that improvements in the calculation of the regional average can limit the effect of outlier values in both the creation of pooled volumes in the forward pass, as well as gradient calculations in the backward pass.

Inverse Distance Weighting (IDW) is widely applicable as a weighted average approach for multivariate interpolation [23], [24]. The assumption is that geometrically close observations exhibit a higher degree of resemblance than geometrically more distant ones. We extend IDW to kernel weighting for pooling by using the distance of each activation $\mathbf{a}_i$, with coordinate index $i \in \mathbf{R}$, to the mean activation $\bar{\mathbf{a}}$ of $\mathbf{R}$. The resulting pooled region $\underset{IDW}{\widetilde{\mathbf{a}}}$ is formulated as:

$$\underset{IDW}{\widetilde{\mathbf{a}}} = \begin{cases} \sum\limits_{i \in \mathbf{R}} \dfrac{\underset{IDW}{w}(\bar{\mathbf{a}}, \mathbf{a}_i) \cdot \mathbf{a}_i}{\sum\limits_{j \in \mathbf{R}} \underset{IDW}{w}(\bar{\mathbf{a}}, \mathbf{a}_j)}, \; if \; d(\bar{\mathbf{a}}, \mathbf{a}_i) \neq 0 \; \forall \; i \in \mathbf{R} \\ \mathbf{a}_i, \; if \; d(\bar{\mathbf{a}}, \mathbf{a}_i) = 0 \; \exists \; i \in \mathbf{R} \end{cases} \tag{1}$$

The weights $\underset{IDW}{w}(\cdot, \cdot)$ are based on the inverse of the distance $d(\cdot, \cdot)$ between each activation and the mean activation:

$$\underset{IDW}{w}(\bar{\mathbf{a}}, \mathbf{a}_i) = \frac{1}{d(\bar{\mathbf{a}}, \mathbf{a}_i)} \tag{2}$$

Distance function $d(\cdot, \cdot)$ can be calculated by any geometric distance approach. Further details and limitations of IDWPool are discussed in Appendix VII-A.

As distance methods can produce artifacts when directly applied in input regions (see Appendix VII-A), the use of similarity measures is a better suited solution for the region-based nature of pooling. For the widely-used cosine similarity, an issue arises when the similarity between the two vectors is 1 even if one of the two vectors is infinitely large [25]. Other dot-product methods for vector volumes such as the Dice-Sørensen Coefficient (DSC) overcome this limitation by taking into account the vector lengths.

Given the IDW approach of Equation 1, zero-valued distances or coefficients will be assigned a zero weight. Therefore, our second extension is the use of the exponent ($e$) of the similarity between the activation vector and the average activations. This effectively makes the pooling method differentiable during backpropagation as at least a minimum gradient will be calculated for every location. It also reduces the possibility for the vanishing gradients problem to arise. Based on the introduction of the exponent of the similarity coefficient, we re-formulate Equation 1 as:

$$\underset{eDSC}{\widetilde{\mathbf{a}}} = \sum_{i \in \mathbf{R}} \frac{e^{\underset{DSC}{w}(\bar{\mathbf{a}}, \mathbf{a}_i)} \cdot \mathbf{a}_i}{\sum\limits_{j \in \mathbf{R}} e^{\underset{DSC}{w}(\bar{\mathbf{a}}, \mathbf{a}_j)}} \tag{3}$$

It is important for downsamped volumes to preserve the informative features while reducing the spatial resolution of the input. The creation of volumes that do not fully capture the structural and feature appearances can have a negative impact on the performance. An example of such loss in detail can be seen in Figure 3. Average pooling decreases the resolution of activations uniformly. Instead, using the exponent of the Dice-Sørensen Coefficient ($e$DSCWPool) can improve on the activation preservation by exponentially weighting kernel values based on their similarity to their regional mean, while ensuring non-zero weights are assigned.

### B. Smooth Approximated Maximum Pooling

Complementary to the smooth approximated average within a kernel region, we discuss the formulation of downsampling based on the smooth approximated maximum which has been recently introduced as *SoftPool* [9]. For clarity, and in line with the used terminology, we refer to SoftPool as exponential maximum pooling ($e$MPool).

The motivation behind the use of the exponential maximum is influenced by the cortex neural simulations [18], [26] that downsample hand-coded features. The method is based on the natural exponent ($e$), which ensures that larger activations will have a greater effect on the final output while also ensuring that a minimum weight value is assigned to the lowest activations.

The weights in exponential maximum pooling ($e$MPool) are used as non-linear transforms based on the value of the corresponding activation. Higher-valued activations will become more dominant than lower-valued ones. As the majority of pooling operations are performed over high-dimensional feature spaces, highlighting the activations with greater effect is more balanced than the selection of the maximum activation alone. In the latter case, discarding the majority of the activations presents the risk of losing important information.

The output of $e$MPool is produced through a summation of all weighted activations within the kernel region $\mathbf{R}$:

$$\underset{eM}{\widetilde{\mathbf{a}}} = \sum_{i \in \mathbf{R}} \underset{eM}{w}(\mathbf{a}_i) \cdot \mathbf{a}_i, \; \text{where} \; \underset{eM}{w}(\mathbf{a}_i) = \frac{e^{\mathbf{a}_i}}{\sum\limits_{j \in \mathbf{R}} e^{\mathbf{a}_j}} \tag{4}$$

$e$MPool produces normalized results, similarly to $e$DSCWPool. The results are based on a probability distribution that is proportional to the values of each activation with respect to the neighboring activations within the kernel region.

### C. AdaPool: Adaptive Exponential Pooling

Based on their properties, $e$DSCWPool uses the similarity of vectors $\mathbf{a}_i$ within the kernel region $\mathbf{R}$ to the mean activation

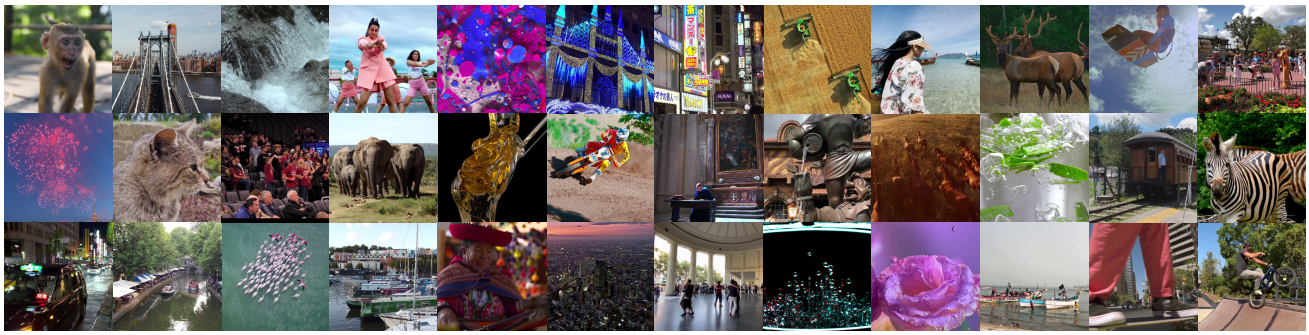


Fig. 4. **Inter4K video frame samples.** These samples show the high resolution (UHD/4K) and variation in the frames. The videos are challenging for video processing due to rapid motions and movements, complex lighting, textures and object detail.

$\bar{\mathbf{a}}$. *e*MPool, however, uses the vectors in proportion to their values, with higher-valued activations being weighted more. From Figure 3, neither of the two methods can be considered superior to the other. However, their properties can be complementary to discover the most informative features within the kernel region. With this observation, and in line with Lee et al.'s introduction of average and maximum pooling fusion strategies [5], we use a trainable weight mask $\boldsymbol{\beta}$ to create a combined volume of both smooth approximated average and smooth approximated maximum. Here, $\boldsymbol{\beta}$ is used to learn the proportion that will be used from each of the two methods within each kernel region $\mathbf{R}$. Introducing $\boldsymbol{\beta}$ as part of the network training process has the advantage of creating a generalized pooling strategy that relies on the combination of the properties of both *e*MPool and *e*DSCWPool. We formulate the method as a regionally-learned combination of the downsampled smooth approximated average ($\underset{eDSC}{\widetilde{\mathbf{a}}}$) and the smooth approximated maximum ($\underset{eM}{\widetilde{\mathbf{a}}}$):

$$\underset{ada}{\widetilde{\mathbf{a}}} \overset{(3,4)}{=\!=} \underset{eDSC}{\widetilde{\mathbf{a}}} \cdot \beta + \underset{eM}{\widetilde{\mathbf{a}}} \cdot (1-\beta) \tag{5}$$

where $\boldsymbol{\beta} \in \{0, \ldots, 1\}$ is a weight mask of the same size as the downsampled volume $\widetilde{\mathbf{a}}$ $(H' \times W')$. A visualization of adaPool appears in Figure 1. The gradients of $\boldsymbol{\beta}$ for backpropagation are calculated based on the chain rule as:

$$\frac{\partial E}{\partial \beta} = \frac{\partial E}{\partial \underset{ada}{\widetilde{\mathbf{a}}}} \frac{\partial \underset{ada}{\widetilde{\mathbf{a}}}}{\partial \beta} = \frac{\partial E}{\partial \underset{ada}{\widetilde{\mathbf{a}}}} (\underset{i}{max}\ \mathbf{a}_i - \frac{1}{|R|} \sum_{i \in \mathbf{R}} \mathbf{a}_i) \tag{6}$$

### *D. Upsampling Using adaUnPool*

Pooling condenses regional information to a single output. The majority of the sub-sampling methods do not establish a bi-directional mapping between the sub-sampled and the original input, as most tasks do not require this link. However, tasks such as semantic segmentation [27], [28], [29], super-resolution [30], [31], [32], [33] or frame interpolation [34], [35], [36], [37] significantly benefit from it. As adaPool is differentiable and uses a minimum weight value assignment, the discovered weights can be used as prior knowledge during upsampling. We refer to this upsampling operation as *adaUnPool*.

For a given pooled volume $(\widetilde{\mathbf{a}})$, we use the smooth approximated maximum $(\underset{eM}{w}(\mathbf{a_i}))$ and smooth approximated average weights $(\underset{eDSCW}{w}(\bar{\mathbf{a}}, \mathbf{a_i}))$ with learned weights mask $\boldsymbol{\beta}$. The final unpooled output $(\mathbf{a_i})$ for the $i$th kernel region ($i \in \mathbf{R}$) is computed as:

$$\mathbf{a}_i = \beta \cdot \frac{e^{\underset{DSC}{w}(\bar{\mathbf{a}}, \mathbf{a}_i)}}{\sum_{j \in \mathbf{R}} e^{\underset{DSC}{w}(\bar{\mathbf{a}}, \mathbf{a}_j)}} \cdot \mathcal{I}_A(\widetilde{\mathbf{a}}) + (1-\beta) \cdot \underset{eM}{w}(\mathbf{a}_i) \cdot \mathcal{I}_A(\widetilde{\mathbf{a}}) \tag{7}$$

where $\mathcal{I}_A(\cdot)$ interpolates by assigning the pooled volume $(\widetilde{\mathbf{a}})$ of the original kernel region at each position $i$. The method is used to inflate the volume from size $H' \times W'$ to $H \times W$.

## IV. The Inter4K Video Dataset

We introduce a novel high-resolution video dataset to benchmark upsampling methods. Inter4K is a collection of 1,000 ultra-high (4K) resolution clips with 60 frames per second (fps) sourced from YouTube. The dataset provides standardized video resolutions at ultra-high definition (UHD/4K), quad-high definition (QHD/2K), full-high definition (FHD/1080p), (standard) high definition (HD/720p), one quarter of full HD (qHD/520p) and one ninth of a full HD (nHD/360p). Available frame rates for each resolution include 60, 50, 30, 24 and 15 fps. Based on this standardization, both super-resolution and frame interpolation tests can be performed for different scaling sizes ($\times 2$, $\times 3$, and $\times 4$). In our experiments, we use Inter4K to address both tasks of frame upsampling and interpolation.

In contrast to other datasets used for video super-resolution and interpolation [38], [39], [40], [41], [42], [43], [44], Inter4K provides standardized UHD resolution at 60 fps for all videos. The dataset is divided into 800 videos for training, 100 videos for validation, and 100 videos for testing. Videos are of 5-second length (examples are shown in Figure 4) and include diverse scenes based on equipment used (e.g., professional 4K cameras, mobile phones), lighting conditions, static and moving cameras, and variations in movements, actions, and objects. We include a summary of the videos in Inter4K based on six main categories as presented in Figure 5. Categories are chosen given the primary focus of the video. The main four categories that correspond to 90% of the videos include *Urban* environments (e.g. buildings, streets, or vehicles), *Nature and animals*, *Sports and people* depicting human activities and actions, and *Demos and abstract* with demo videos for video

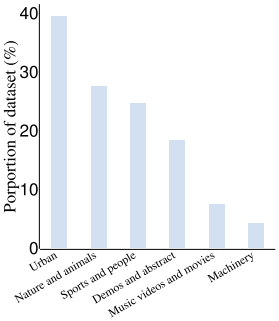



Fig. 5. **Inter4K category proportions**. Categories are selected based on broad concepts of the videos.

Fig. 6. **Inter4K video locations by continent**. Darker colors correspond to a larger number of videos.

resolution and frame rates, or videos with computer-generated abstract shapes. The last two categories are less prevalent in the dataset either due to copyright restrictions (*Music videos and movies*) or scarcity of videos (*Machinery*). In Figure 6 we present a visualization of the locations of 632 out of the 1,000 videos. These locations were found based on available geo-tags, video titles, and keywords or depictions of identifiable landmarks in the video. Both Figures 5 and 6 demonstrate the diversity of Inter4K in terms of video content and the locations where the videos were shot.

## V. Main Results

We initially evaluate the information loss caused by downsampling with various pooling methods. We compare the downsampled and original images using standard similarity measures (Section V-B). In addition, we examine the computational overhead of each pooling method (Section V-C).

We proceed by testing the performance of widely-used CNN architectures on ImageNet1K when we substitute the network's original pooling layers by *e*MPool, *e*DSCWPool and adaPool (Section V-D). We also provide comparisons between different pooling methods (Section V-E).

We present our results for object detection (Section V-F) on MS COCO [45] with RetinaNet [46] and Mask R-CNN [47] using several backbones. We additionally experiment on spatio-temporal data by focusing on action recognition in video (Section V-G).

Lastly, we present our results on image super-resolution, frame interpolation, and their combination (Section V-H).

### A. Experimental Settings

*1) Datasets:* For our image-based experiments, we use seven different datasets for quantitative evaluation of the downsampled image quality, image classification, object detection, and image super-resolution. For the assessment of image quality and similarity, we use the high-resolution DIV2K [48], Urban100 [49], Manga109 [50], and Flicker2K [48] datasets. For image classification we use ImageNet1K [51], and MS COCO [45] for image object detection. For image super-resolution we employ the Urban100, Manga109, and B100 [52] datasets. For our video-based experiments, we employ six datasets. For action recognition, we use the large-scale HACS [53] and Kinetics-700 [54] datasets, as well as the smaller UCF-101 [41] dataset. For frame interpolation, we use Vimeo90K [44] and Middlebury [38] video processing datasets, as well as our newly introduced Inter4K dataset, which is also used for the combined task of frame interpolation and super-resolution.

*2) Classification Training Scheme:* For image classification, we use a random spatial region crop of size $294 \times 294$, which is then resized to $224 \times 224$. The initial learning rate across our experiments is set to 0.1 with an SGD optimizer. We train for a total of 100 epochs with a step-wise learning rate reduction every 40 epochs. For higher numbers of epochs, no further improvements were observed. The batch size is set to 256.

For our video action recognition tests, we use a multigrid training scheme [55], with frame sizes between 4–16 and frame crops of 90–256 depending on the cycle. The average video inputs are of size $8 \times 160 \times 160$, while the batch sizes are between 64 and 2048. The size for each of the batches is counter-equal to the input size in every step in order to optimize memory use. We use the same learning rate, optimizer, learning rate schedule, and maximum number of epochs as in the image-based experiments.

*3) Object Detection Details:* We first rescale the images to ensure that the smallest side has a minimum size of at least 800 pixels [47], [56]. If after rescaling the largest side is larger than 1024 pixels, we resize the entire image so that the largest side becomes 1024 pixels. Our rescaling and resizing preserves the aspect ratio of the images. We use the pre-trained networks from the image classification task as backbones. The learning rate is set to $1e{-}5$ and we use an SGD optimizer with 0.9 momentum.

### B. Downsampling Similarity

In the first set of tests, we evaluate the information loss when using our proposed methods for downsampling. The comparisons focus on the similarity of the original inputs and downsampled outputs. Three widely used pooling kernel sizes are employed ($k = \{2, 3, 5\}$). We use three standardized evaluation metrics [58], [59]:

**Structural Similarity Index Measure (SSIM)** is calculated as the difference of two images in terms of their luminance, contrast, and a structural term. Larger SSIM values correspond to larger structural similarities.

**Peak Signal-to-Noise Ratio (PSNR)** is a quantification of the produced image's compression quality. PSNR takes into account the inverse of the Mean Squared Error (MSE) of two

TABLE I

**QUANTITATIVE RESULTS ON BENCHMARK HIGH-RES DATASETS**. BEST RESULTS FOR EACH SETTING ARE DENOTED IN **BOLD**

| | Pooling method | DIV2K [48] | | | | | | | | | Urban100 [49] | | | | | | | | | Manga109 [50] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $k=2$ | | | $k=3$ | | | $k=5$ | | | $k=2$ | | | $k=3$ | | | $k=5$ | | | $k=2$ | | | $k=3$ | | | $k=5$ | | |
| | | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS |
| | Avg | 0.714 | 51.247 | 0.204 | 0.578 | 44.704 | 0.325 | 0.417 | 29.223 | 0.494 | 0.691 | 50.380 | 0.212 | 0.563 | 41.745 | 0.352 | 0.372 | 28.270 | 0.416 | 0.695 | 54.326 | 0.189 | 0.582 | 43.657 | 0.277 | 0.396 | 29.862 | 0.374 |
| | Max | 0.685 | 49.826 | 0.229 | 0.370 | 41.944 | 0.367 | 0.358 | 22.041 | 0.524 | 0.662 | 48.266 | 0.252 | 0.528 | 40.709 | 0.405 | 0.330 | 20.654 | 0.476 | 0.671 | 50.085 | 0.210 | 0.544 | 41.128 | 0.324 | 0.324 | 22.307 | 0.413 |
| | Pow-avg | 0.419 | 35.587 | 0.323 | 0.286 | 26.329 | 0.500 | 0.178 | 16.567 | 0.657 | 0.312 | 31.911 | 0.313 | 0.219 | 24.698 | 0.512 | 0.124 | 15.659 | 0.591 | 0.381 | 29.248 | 0.313 | 0.276 | 18.874 | 0.460 | 0.160 | 9.266 | 0.553 |
| | Sum | 0.408 | 35.153 | 0.315 | 0.268 | 26.172 | 0.489 | 0.193 | 17.315 | 0.634 | 0.301 | 31.657 | 0.308 | 0.208 | 24.735 | 0.511 | 0.123 | 15.243 | 0.602 | 0.374 | 30.169 | 0.321 | 0.271 | 20.150 | 0.456 | 0.168 | 13.081 | 0.549 |
| Trainable | $L_p$ [2] | 0.686 | 49.912 | 0.213 | 0.542 | 43.083 | 0.328 | 0.347 | 25.139 | 0.502 | 0.676 | 48.508 | 0.224 | 0.534 | 39.986 | 0.365 | 0.326 | 26.365 | 0.453 | 0.675 | 51.721 | 0.206 | 0.561 | 41.824 | 0.292 | 0.367 | 27.469 | 0.407 |
| Trainable | Gate [5] | 0.689 | 50.104 | 0.211 | 0.560 | 43.437 | 0.326 | 0.353 | 25.672 | 0.497 | 0.675 | 49.769 | 0.212 | 0.537 | 40.422 | 0.363 | 0.328 | 26.731 | 0.433 | 0.679 | 51.980 | 0.198 | 0.569 | 42.127 | 0.283 | 0.374 | 27.754 | 0.393 |
| Trainable | DPP [21] | 0.702 | 50.598 | 0.206 | 0.562 | 44.076 | 0.324 | 0.396 | 27.421 | 0.498 | 0.684 | 49.947 | 0.213 | 0.551 | 40.813 | 0.357 | 0.365 | 27.136 | 0.425 | 0.691 | 52.646 | 0.191 | 0.573 | 42.794 | 0.294 | 0.386 | 28.598 | 0.386 |
| Trainable | LIP [6] | 0.711 | 50.831 | 0.203 | 0.559 | 44.432 | 0.323 | 0.401 | 28.285 | 0.492 | 0.689 | 50.266 | 0.212 | 0.558 | 41.159 | 0.354 | 0.370 | 27.849 | 0.415 | 0.689 | 53.537 | 0.185 | 0.579 | 43.018 | 0.273 | 0.391 | 29.331 | 0.373 |
| Stoch. | Stochastic [3] | 0.631 | 45.362 | 0.321 | 0.479 | 39.895 | 0.497 | 0.295 | 21.314 | 0.609 | 0.616 | 44.342 | 0.285 | 0.463 | 37.223 | 0.476 | 0.286 | 19.358 | 0.561 | 0.583 | 46.274 | 0.316 | 0.427 | 39.259 | 0.433 | 0.255 | 22.953 | 0.521 |
| Stoch. | S3 [4] | 0.609 | 44.760 | 0.318 | 0.454 | 39.326 | 0.486 | 0.280 | 20.773 | 0.615 | 0.608 | 44.239 | 0.276 | 0.459 | 36.965 | 0.463 | 0.272 | 19.645 | 0.548 | 0.576 | 46.613 | 0.305 | 0.426 | 39.866 | 0.427 | 0.232 | 23.242 | 0.521 |
| IDW (Ours) | $L_1$ | 0.724 | 51.415 | 0.218 | 0.596 | 44.739 | 0.346 | 0.418 | 29.576 | 0.511 | 0.696 | 50.723 | 0.218 | 0.573 | 41.796 | 0.367 | 0.366 | 28.205 | 0.437 | 0.712 | 54.614 | 0.195 | 0.574 | 43.756 | 0.283 | 0.395 | 30.224 | 0.389 |
| IDW (Ours) | $L_2$ | 0.726 | 51.421 | 0.217 | 0.601 | 44.753 | 0.346 | 0.421 | 29.581 | 0.504 | 0.698 | 50.731 | 0.217 | 0.575 | 41.794 | 0.368 | 0.372 | 28.211 | 0.433 | 0.711 | 54.617 | 0.191 | 0.579 | 43.762 | 0.281 | 0.404 | 30.256 | 0.379 |
| IDW (Ours) | Huber [57] $\delta=1/4$ | 0.728 | 51.465 | 0.215 | 0.611 | 44.813 | 0.336 | 0.427 | 29.736 | 0.502 | 0.702 | 50.734 | 0.216 | 0.579 | 41.862 | 0.364 | 0.383 | 28.576 | 0.429 | 0.714 | 54.620 | 0.192 | 0.584 | 43.897 | 0.278 | 0.412 | 30.445 | 0.378 |
| IDW (Ours) | Huber [57] $\delta=1/2$ | 0.730 | 51.487 | 0.214 | 0.617 | 44.924 | 0.327 | 0.429 | 29.861 | 0.497 | 0.710 | 50.742 | 0.216 | 0.581 | 41.916 | 0.359 | 0.389 | 28.674 | 0.431 | 0.721 | 54.637 | 0.190 | 0.588 | 43.936 | 0.278 | 0.421 | 30.429 | 0.376 |
| IDW (Ours) | Huber [57] $\delta=3/4$ | 0.727 | 51.459 | 0.215 | 0.606 | 44.846 | 0.330 | 0.421 | 29.728 | 0.499 | 0.705 | 50.737 | 0.215 | 0.576 | 41.874 | 0.358 | 0.384 | 28.612 | 0.428 | 0.716 | 54.626 | 0.188 | 0.581 | 43.885 | 0.277 | 0.409 | 30.432 | 0.375 |
| exponential (Ours) | *e*M | 0.729 | 51.436 | 0.204 | 0.594 | 44.747 | 0.339 | 0.421 | 29.583 | 0.498 | 0.694 | 50.687 | 0.211 | 0.578 | 41.851 | 0.352 | 0.394 | 28.326 | 0.417 | 0.704 | 54.563 | 0.189 | 0.586 | 43.782 | 0.276 | 0.403 | 30.114 | 0.373 |
| exponential (Ours) | *e*DSCW | 0.732 | 51.470 | 0.203 | 0.619 | 45.324 | 0.324 | 0.430 | 30.247 | 0.498 | 0.706 | 50.734 | 0.211 | 0.588 | 42.173 | 0.351 | 0.412 | 29.796 | 0.413 | 0.715 | 54.633 | 0.173 | 0.593 | 44.185 | 0.268 | 0.417 | 30.967 | 0.371 |
| exponential (Ours) | adapt. $\beta=1/4$ | 0.730 | 51.523 | 0.199 | 0.605 | 44.979 | 0.321 | 0.429 | 29.956 | 0.492 | 0.712 | 50.733 | 0.199 | 0.582 | 41.970 | 0.345 | 0.397 | 28.861 | 0.407 | 0.709 | 54.625 | 0.168 | 0.589 | 43.994 | 0.265 | 0.412 | 30.843 | 0.368 |
| exponential (Ours) | adapt. $\beta=1/2$ | 0.736 | 52.186 | 0.198 | 0.614 | 45.857 | 0.321 | 0.432 | 30.881 | 0.491 | 0.754 | 51.376 | 0.197 | 0.594 | 42.462 | 0.345 | 0.410 | 29.874 | 0.406 | 0.716 | 54.646 | 0.167 | 0.598 | 44.576 | 0.264 | 0.421 | 31.392 | 0.364 |
| exponential (Ours) | adapt. $\beta=3/4$ | 0.742 | 53.341 | 0.194 | 0.618 | 46.173 | 0.319 | 0.438 | 31.764 | 0.489 | 0.741 | 51.132 | 0.196 | 0.605 | 42.996 | 0.341 | 0.418 | 30.163 | 0.405 | 0.731 | 55.284 | 0.164 | 0.602 | 45.344 | 0.260 | 0.427 | 31.638 | 0.366 |
| exponential (Ours) | adapt. $\nabla\beta$ | **0.778** | **53.769** | **0.184** | **0.624** | **46.892** | **0.305** | **0.443** | **32.216** | **0.483** | **0.769** | **52.438** | **0.188** | **0.614** | **43.637** | **0.337** | **0.425** | **30.845** | **0.396** | **0.747** | **55.890** | **0.156** | **0.609** | **46.253** | **0.254** | **0.436** | **32.794** | **0.355** |

TABLE II

**LATENCY AND PIXEL SIMILARITY**. LATENCY FOR THE FORWARD AND BACKWARD PASS ON BOTH CPU AND GPU IS AVERAGED OVER ALL IMAGES IN IMAGENET1K. PIXEL SIMILARITY REPORTED ON FLICKER2K. BEST RESULTS IN **BOLD**

| Pooling | CPU (ms) | CUDA (ms) | Flicker2K [48] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $k=2$ | | | $k=3$ | | | $k=5$ | | |
| | ($\downarrow$F/$\uparrow$B) | ($\downarrow$F/$\uparrow$B) | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS | SSIM | PSNR | LPIPS |
| Avg | 9/49 | 14/76 | 0.709 | 51.786 | 0.2138 | 0.572 | 44.246 | 0.3289 | 0.408 | 28.957 | 0.5020 |
| Max | 91/152 | 195/267 | 0.674 | 47.613 | 0.2367 | 0.385 | 40.735 | 0.3781 | 0.329 | 21.368 | 0.5372 |
| Pow-avg | 74/329 | 120/433 | 0.392 | 34.319 | 0.3214 | 0.271 | 26.820 | 0.5346 | 0.163 | 15.453 | 0.6608 |
| Sum | 26/163 | 79/323 | 0.386 | 34.173 | 0.3276 | 0.265 | 26.259 | 0.5212 | 0.161 | 15.218 | 0.6573 |
| $L_p$ [2] | 116/338 | 214/422 | 0.683 | 48.617 | 0.2269 | 0.437 | 42.079 | 0.3572 | 0.341 | 24.432 | 0.5326 |
| Gate [5] | 245/339 | 327/540 | 0.687 | 49.314 | 0.2241 | 0.449 | 42.722 | 0.3453 | 0.358 | 25.687 | 0.5245 |
| DPP [21] | 427/860 | 634/1228 | 0.691 | 50.586 | 0.2155 | 0.534 | 43.608 | 0.3341 | 0.385 | 27.430 | 0.5137 |
| LIP [6] | 134/257 | 258/362 | 0.696 | 50.947 | 0.2140 | 0.548 | 43.882 | 0.3292 | 0.390 | 28.134 | 0.5034 |
| Stoch. [3] | 162/341 | 219/485 | 0.625 | 46.714 | 0.3416 | 0.474 | 38.365 | 0.4876 | 0.264 | 21.428 | 0.5724 |
| S3 [4] | 233/410 | 345/486 | 0.611 | 46.547 | 0.3205 | 0.476 | 37.706 | 0.4531 | 0.252 | 21.363 | 0.5640 |
| $L_1$ | N/A | 48/227 | 0.712 | 51.985 | 0.2283 | 0.574 | 44.814 | 0.3365 | 0.411 | 29.246 | 0.5208 |
| $L_2$ | N/A | 49/231 | 0.714 | 51.592 | 0.2276 | 0.576 | 44.832 | 0.3342 | 0.416 | 29.251 | 0.5188 |
| Huber [57] $\delta=1/4$ | N/A | 51/234 | 0.726 | 51.879 | 0.2265 | 0.583 | 44.916 | 0.3314 | 0.421 | 29.458 | 0.5162 |
| Huber [57] $\delta=1/2$ | N/A | 51/234 | 0.727 | 52.053 | 0.2214 | 0.593 | 45.231 | 0.3299 | 0.424 | 29.635 | 0.5134 |
| Huber [57] $\delta=3/4$ | N/A | 51/234 | 0.723 | 51.912 | 0.2218 | 0.584 | 45.105 | 0.3312 | 0.418 | 29.476 | 0.5127 |
| *e*M | 31/156 | 56/234 | 0.721 | 52.356 | 0.2143 | 0.587 | 44.893 | 0.3297 | 0.416 | 29.341 | 0.5026 |
| *e*DSCW | N/A | 52/249 | 0.743 | 53.293 | 0.2124 | 0.615 | 45.274 | 0.3275 | 0.421 | 29.957 | 0.4986 |
| adaptive | $N/A$ | 119/490 | **0.766** | **54.608** | **0.2045** | **0.629** | **47.139** | **0.3089** | **0.434** | **31.883** | **0.4831** |

images' channels. Higher PSNR values translate to smaller channel-wise distances between the two images.

**Learned Perceptual Image Patch Similarity (LPIPS)** is a similarity measurement between patches from two images. LPIPS compares the distances of features from the two images extracted by a deep learning backbone. Lower LPIPS values correspond to higher similarity between images.

In Tables I and II, we present the SSIM, PSNR, and LPIPS values averaged over all images in DIV2K [48], Urban100 [49], Manga109 [50], and Flicker2K [48] datasets, for different kernel sizes. IDW-based distance methods outperform non-trainable and stochastic methods. The randomized policy of stochastic methods does not seem to allow to fully capture details. Additionally, the use of exponential weighting to our IDW-based methods yields clear improvements. Both *e*MPool and *e*DSCWPool are top-performing across kernel sizes and datasets, demonstrating the benefits of exponential approximation methods for image downsampling. Finally, the combination of the two exponential methods into adaPool consistently achieves the best overall performance when the fusion parameter $\boldsymbol{\beta}$ is learned.

### C. Latency and Memory Use

Costs in terms of the memory and latency required by pooling operations are largely overlooked in literature as single operations have minor latency times and memory consumption. However, given potentially limited available resources, and the fact that operations are executed thousands of times per epoch, we advocate an evaluation of the running times and memory use. Slow or memory-intensive operations can have a detrimental effect on the performance and may become potential computational bottlenecks.

Computation overheads are reported in Table II based on the inference over CPU and GPU (CUDA) for forward ($\downarrow F$) and backward ($\uparrow B$) passes over each operation. We observe that our implementations achieve reasonable inference times on CUDA despite the additional computations in comparison to methods such as average, maximum, power average or sum pooling.

### D. Image Classification Performance on ImageNet1K

We test the assumption that a better preservation of information during downsampling with the exponential weighting method leads to an increase in image classification accuracy. Based on the results between average and max pooling and adaPool (Tables I and II), we replace the original pooling layers in ResNet [60], DenseNet [61], ResNeXt [62] and wide-ResNet [63] networks with our exponential pooling method and test their performance on ImageNet1K. Results appear in Table III. In Table IV, we summarize the results of four runs

TABLE III

PAIRWISE COMPARISONS OF TOP-1 AND TOP-5 ACCURACIES ON IMAGENET1K [51] BETWEEN ORIGINAL NETWORKS AND THEIR COUNTERPARTS WITH POOLING REPLACED BY *e*MPOOL, *e*DSCWPOOL AND ADAPOOL. ALL NETWORKS HAVE BEEN TRAINED FROM SCRATCH. BEST RESULTS IN **BOLD**. MORE DETAILS FOR THE PARAMETERS AND FLOPS ARE PROVIDED IN APPENDIX VII-E

| Model | Params (M) | GFLOPs | Original (Baseline) top-1 | Original (Baseline) top-5 | *e*MPool top-1 | *e*MPool top-5 | *e*DSCWPool top-1 | *e*DSCWPool top-5 | adaPool top-1 | adaPool top-5 |
|---|---|---|---|---|---|---|---|---|---|---|
| ResNet-18 | 11.7 | 1.83 | 69.76 | 89.08 | 71.27 (+1.51) | 90.16 (+1.08) | 70.79 (+1.03) | 89.96 (+0.88) | **71.78** (+2.02) | **90.65** (+1.57) |
| ResNet-34 | 21.8 | 3.68 | 73.30 | 91.42 | 74.67 (+1.37) | 92.30 (+0.88) | 74.36 (+1.06) | 92.15 (+0.73) | **75.43** (+2.13) | **92.87** (+1.45) |
| ResNet-50 | 25.6 | 4.14 | 76.15 | 92.87 | 77.35 (+1.17) | 93.63 (+0.76) | 77.38 (+1.23) | 93.90 (+1.03) | **78.42** (+2.27) | **94.16** (+1.29) |
| ResNet-101 | 44.5 | 7.87 | 77.37 | 93.56 | 78.32 (+0.95) | 94.21 (+0.65) | 78.58 (+1.21) | 94.42 (+0.86) | **79.59** (+2.22) | **94.88** (+1.32) |
| ResNet-152 | 60.2 | 11.61 | 78.31 | 94.06 | 79.24 (+0.92) | 94.72 (+0.66) | 79.54 (+1.23) | 94.98 (+0.92) | **80.74** (+2.43) | **95.08** (+1.02) |
| DenseNet-121 | 8.0 | 2.90 | 74.65 | 92.17 | 75.88 (+1.23) | 92.92 (+0.75) | 76.06 (+1.41) | 93.16 (+0.99) | **77.29** (+2.64) | **93.21** (+1.04) |
| DenseNet-161 | 28.7 | 7.85 | 77.65 | 93.80 | 78.72 (+0.93) | 94.41 (+0.61) | 78.77 (+1.12) | 94.53 (+0.73) | **80.10** (+2.35) | **94.87** (+1.07) |
| DenseNet-169 | 14.1 | 3.44 | 76.00 | 93.00 | 76.95 (+0.95) | 93.76 (+0.76) | 77.19 (+1.19) | 93.86 (+0.86) | **78.56** (+2.56) | **94.23** (+1.23) |
| ResNeXt-50 32x4d | 25.0 | 4.29 | 77.62 | 93.70 | 78.48 (+0.86) | 93.37 (+0.67) | 78.76 (+1.14) | 94.48 (+0.78) | **79.98** (+2.36) | **94.82** (+1.12) |
| ResNeXt-101 32x8d | 88.8 | 7.89 | 79.31 | 94.28 | 80.12 (+0.81) | 94.88 (+0.60) | 80.57 (+1.26) | 95.02 (+0.74) | **81.69** (+2.38) | **95.51** (+1.23) |
| Wide-ResNet-50 | 68.9 | 11.46 | 78.51 | 94.09 | 79.52 (+1.01) | 94.85 (+0.76) | 79.61 (+1.10) | 94.92 (+0.83) | **80.24** (+1.73) | **95.26** (+1.17) |

TABLE IV

TOP-1 ACCURACY OVER RUNS ON IMAGENET1K [51] FOR ORIGINAL NETWORKS AND THOSE WITH *e*MPOOL, *e*DSCWPOOL AND ADAPOOL. WE PERFORMED FOUR RUNS FOR EACH COMBINATION OF NETWORK AND POOLING TYPE. THE BEST RUN IS DENOTED WITH (BEST). BEST OVERALL RESULTS IN **BOLD**

| Pooling / Run / Model | Original (Baseline) 1 | 2 | 3 | (best) | *e*MPool 1 | 2 | 3 | (best) | *e*DSCWPool 1 | 2 | 3 | (best) | adaPool 1 | 2 | 3 | (best) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ResNet-18 | 69.61 | 69.73 | 69.69 | 69.76 | 71.18 | 71.04 | 71.25 | 71.27 | 70.65 | 70.78 | 70.73 | 70.79 | 71.70 | 71.74 | 71.62 | **71.78** |
| ResNet-34 | 73.26 | 73.11 | 73.24 | 73.30 | 74.66 | 74.52 | 74.31 | 74.67 | 74.25 | 74.30 | 74.28 | 74.36 | 75.35 | 75.42 | 75.37 | **75.43** |
| ResNet-50 | 76.01 | 75.97 | 76.04 | 76.15 | 77.26 | 77.24 | 77.19 | 77.35 | 77.35 | 77.26 | 77.23 | 77.38 | 78.36 | 78.38 | 78.41 | **78.42** |

over different training seeds for three models to ensure fair comparisons. The highest accuracies are denoted by (best).

Overall, we notice that networks with their pooling layers replaced by adaPool yield improved accuracy rates. We provide a further discussion per CNN architecture.

*1) ResNet [60]:* We report an average of 2.19% top-1 and 1.33% top-5 improvement on ResNet models when replacing their pooling layers with adaPool. Improvements in accuracy are also observed with replacements based on both *e*MPool and *e*DSCWPool with an average +1.17% and +1.15% top-1 accuracy, respectively. ResNet architectures include only a single pooling operation after the first convolution layer. The improvements from replacing only a single layer demonstrate the benefits of adaPool for image classification. In Table IV, we do not notice a significant divergence in accuracy over multiple runs on ResNet-18, ResNet-34, and ResNet-50 networks. On average, a replacement with adaPool can improve by +2.01% the original ResNet-18 across runs, by +2.24% on ResNet-34 and +2.38% on ResNet-50.

*2) DenseNet [61]:* DenseNets include five pooling layers. Our replacements concern the maximum pooling layer after the first convolution and the four average pooling layers between dense blocks. The average top-1 accuracy gains based on layer replacements with adaPool are between 2.35–2.64%. More modest increases are found for *e*MPool and *e*DSCWPool with +(0.93–1.23)% and +(1.12–1.41)%, respectively.

*3) ResNeXt [62]:* We achieve an average of 2.37% top-1 and 1.17% top-5 accuracy improvement with adaPool. An average gain of 1.20% and 0.76% for the top-1 and top-5 accuracies are observed with pooling layer replacement with *e*DSCWPool. For *e*MPool, these improvements are 0.83% and 0.64% for the top-1 and top-5 accuracies, respectively.

*4) Wide-Resnet-50 [62]:* On Wide-ResNet-50, we observe the best top-1 accuracy of 80.24% with a 1.73% improvement when we replace the original pooling layers with adaPool. Gains in performance are also observed for *e*MPool with +1.01% and *e*DSCWPool with +1.10%.

TABLE V

POOLING LAYER SUBSTITUTION TOP-1 ACCURACY FOR A VARIETY OF POOLING METHODS. EXPERIMENTS WERE PERFORMED ON IMAGENET1K. BEST RESULTS PER NETWORK IN **BOLD**

| Pooling | Networks: ResNet−18 | ResNet−34 | ResNet−50 | ResNeXt−50 | DenseNet−121 | InceptionV1 |
|---|---|---|---|---|---|---|
| Original (Baseline) | (Max) 69.76 | (Max) 73.30 | (Max) 76.15 | (Max) 77.62 | (Avg+Max) 74.65 | (Max) 69.78 |
| Stochastic [3] | 70.13 | 73.34 | 76.11 | 77.71 | 74.84 | 70.14 |
| S3 [4] | 70.15 | 73.56 | 76.24 | 77.82 | 74.85 | 70.17 |
| $L_p$ [5] | 70.45 | 73.74 | 76.56 | 77.86 | 74.93 | 70.32 |
| Gate [2] | 70.74 | 73.68 | 76.75 | 77.98 | 74.88 | 70.52 |
| DPP [21] | 70.86 | 74.25 | 77.09 | 78.20 | 75.37 | 70.95 |
| LIP [6] (drop-in) | 70.83 | 73.95 | 77.13 | 78.14 | 75.31 | 70.77 |
| LIP [6] (multi) | 71.42 | 74.86 | 78.19 | 79.25 | 76.64 | N/A |
| *e*MPool **(ours)** | 71.27 | 74.67 | 77.35 | 78.48 | 75.88 | 71.43 |
| *e*DSCWPool **(ours)** | 70.79 | 74.36 | 77.38 | 78.76 | 76.06 | 71.85 |
| adaPool **(ours)** | **71.78** | **75.43** | **78.42** | **79.98** | **77.29** | **72.56** |

## *E. Comparison With Alternative Pooling Methods*

We provide quantitative comparisons between different pooling methods over six different models in Table V.

TABLE VI

OBJECT DETECTION BOUNDING BOX AP RESULTS ON MS COCO TEST-DEV FOR MODELS WITH ORIGINAL BACKBONE NETWORKS AND THE SAME NETWORKS WITH POOLING LAYERS REPLACED BY OUR EXPONENTIAL POOLING LAYERS. ALL MODELS ARE PRE-TRAINED ON IMAGENET1K [51]. BEST RESULTS IN **BOLD**

| Model | Backbone | Original (Baseline) | | | | | | *e*MPool | | | | | | *e*DSCWPool | | | | | | adaPool | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
| RetinaNet [46] | ResNet-18 | 28.3 | 48.7 | 31.6 | 12.6 | 33.6 | 40.9 | 29.7 | 50.2 | 33.3 | 14.1 | 35.2 | 42.6 | 28.9 | 49.6 | 32.8 | 13.8 | 34.7 | 41.5 | **31.2** | **51.4** | **34.7** | **15.4** | **36.5** | **43.4** |
| | ResNet-34 | 31.6 | 50.8 | 33.9 | 15.1 | 36.0 | 43.6 | 32.8 | 52.1 | 35.5 | 16.2 | 37.3 | 45.0 | 32.4 | 51.4 | 34.8 | 15.9 | 36.8 | 44.7 | **33.6** | **53.4** | **36.4** | **16.9** | **38.2** | **44.7** |
| | ResNet-50 | 34.0 | 52.5 | 36.5 | 17.0 | 37.4 | 45.1 | 34.9 | 53.4 | 37.6 | 18.0 | 38.5 | 46.4 | 34.6 | 53.1 | 37.2 | 17.7 | 38.2 | 46.1 | **35.6** | **53.9** | **38.0** | **18.4** | **39.1** | **47.2** |
| | ResNet-101 | 39.1 | 59.1 | 42.3 | 21.8 | 42.7 | 50.2 | 39.8 | 59.9 | 43.3 | 22.4 | 43.5 | 51.1 | 40.1 | 60.3 | 43.7 | 22.6 | 43.9 | 51.4 | **40.8** | **61.6** | **44.8** | **23.7** | **44.8** | **52.5** |
| Mask R-CNN [47] | ResNet-34 | 32.9 | 53.6 | 32.7 | 14.5 | 35.1 | 43.2 | 34.0 | 54.8 | 34.1 | 15.7 | 36.6 | 44.6 | 33.8 | 54.1 | 33.6 | 15.3 | 36.2 | 44.0 | **35.7** | **56.9** | **36.4** | **16.8** | **38.6** | **46.5** |
| | ResNet-50 | 33.6 | 55.2 | 35.3 | 15.4 | 36.8 | 45.3 | 34.5 | 56.2 | 36.4 | 16.2 | 37.7 | 46.3 | 34.4 | 56.2 | 36.3 | 16.3 | 37.5 | 46.2 | **36.3** | **57.5** | **36.9** | **17.1** | **39.0** | **47.3** |
| | ResNet-101 | 38.2 | 60.3 | 41.7 | 20.1 | 41.1 | 50.2 | 39.0 | 61.1 | 42.6 | 20.9 | 42.0 | 51.3 | 39.5 | 61.7 | 43.1 | 21.5 | 42.8 | 51.9 | **42.4** | **62.8** | **45.1** | **24.5** | **45.6** | **52.8** |

We systematically replaced the pooling layers of the original model (baseline). For LIP, we consider both drop-in replacements, in line with the rest of our experiments, as well as multiple replacements following the LIP-ResNet and LIP-DenseNet architectures of the paper [6]. Non-adaptive *e*MPool and *e*DSCWPool still outperform stochastic methods while the obtained accuracies are similar to those of learnable methods. Across the tested architectures, adaPool outperforms other learnable and stochastic pooling methods. The largest overall margins are observed for InceptionV1 with improvements over other methods in the range of 1.61–2.78% and on DenseNet-121 (0.65–2.64%).

### *F. Object Detection Performance on MS COCO*

To investigate the merits of our proposed exponentially-weighted pooling on encapsulating relevant local information, we present results for object detection on MS COCO [45] in Table VI. We use RetinaNet [46] and Mask-RCNN [47] with several different backbone networks. We chose these two models based on their wide popularity. Overall, we observe that both *e*MPool and *e*DSCWPool come with average precision (AP) improvements of 1.00% and 0.86%, respectively. A 2.40% increase over the original models is observed for adaPool. Similar trends in AP are also visible for $AP_{50}$ and $AP_{75}$, demonstrating that adaPool does not only benefit tasks that rely primarily on general features such as classification, but also provides a performance boost for local-based feature tasks such as object detection.

### *G. Video Classification Performance*

We evaluate our pooling operators on spatio-temporal data by focusing on the task of action recognition in videos. The accurate classification and representation of space-time features stands as a major challenge in the field of video understanding [71].

The majority of space-time networks are based on the extension of 2D convolutions to 3D to include the temporal dimension. Stacks of frames are used as inputs. Similarly, the only modification in our method is the inclusion of the temporal dimension in kernel region **R**.

For our tests, we first train models from scratch on HACS [53] using the author implementations. These models are then used to initialize the weights for the Kinetics-700 and UCF-101 tests. SlowFast (SF) [69] and ir-CSN-101 [67] are the only two models that use different initialization weights, with ir-CSN-101 pre-trained on IG65M and SF on ImageNet.

TABLE VII

ACTION RECOGNITION TOP-1 AND TOP-5 ACCURACIES FOR HACS, K-700 AND UCF-101. MODELS ARE TRAINED ON HACS AND FINE-TUNED ON K-700 AND UCF-101, EXCEPT FOR IR-CSN-101 AND SF R3D-50 (SEE TEXT). N/A MEANS NO TRAINED MODEL WAS PROVIDED. BEST RESULTS IN **BOLD**

| Model | FLOPs (G) | HACS | | K-700 | | UCF-101 | |
|---|---|---|---|---|---|---|---|
| | | top-1 | top-5 | top-1 | top-5 | top-1 | top-5 |
| r3d-101 [64]** | 78.5 | 80.49 | 95.18 | 52.58 | 74.63 | 95.76 | 98.42 |
| r(2+1)d-50 [65]** | 50.0 | 81.34 | 94.51 | 49.93 | 73.40 | 93.92 | 97.84 |
| I3D [66]‡* | 55.3 | 79.95 | 94.48 | 53.01 | 69.19 | 92.45 | 97.62 |
| ir-CSN-101 [67]‡† | 17.3 | N/A | N/A | 54.66 | 73.78 | 95.13 | 97.85 |
| SRTG [68]†† | 78.7 | 81.66 | 96.37 | 56.46 | 76.82 | 97.32 | 99.56 |
| SF r3d-50 [69]‡† | 36.7 | N/A | N/A | 56.17 | 75.57 | 94.62 | 98.75 |
| $MTNet_L$ [70]†† | 17.6 | 86.62 | 96.68 | 63.31 | 84.14 | 97.38 | 99.23 |
| [65] w/ adaPool | 53.2 | 81.13 | 94.96 | 50.87 | 74.06 | 94.21 | 97.76 |
| [68] w/ adaPool | 78.7 | 84.37 | 97.84 | 58.62 | 78.56 | 98.53 | **99.86** |
| [70] w/ adaPool | 17.8 | **87.83** | **98.21** | **64.67** | **84.78** | **98.60** | 99.74 |

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

We report in Table VII the performance of three spatio-temporal CNNs with pooling layers replaced by adaPool. We observe state-of-the-art performance using $MTNet_L$ with adaPool on HACS and Kinetics-700, with 87.83% and 64.67% top-1 accuracies, respectively. This corresponds to an increase of 1.21% and 1.36% over the same networks with the original pooling layers. This also comes with negligible additional GFLOPs (+0.2). On UCF-101, we show that both $MTNet_L$ and SRTG r3d-101 with adaPool outperform the original and other top-performing models. Increases in top-1 performance are also observed for SRTG r3d-101 with +2.71% on HACS and +1.47% on Kinetics-700.

These results further demonstrate that the simple replacement of a pooling operator by adaPool consistently results in a modest but important performance gain. Even for the almost saturated performance on UCF-101, using adaPool results in a performance increase of 1.22% on $MTNet_L$.

### *H. Image Super-Resolution and Frame Interpolation Results*

In order to assess the benefits of re-using the learned adaPool weights in adaUnPool, we experiment on image super-resolution, video frame interpolation, and their combination. For each task we replace pooling layers with adaPool and the respective bilinear interpolation with adaUnPool.

TABLE VIII

IMAGE SUPER-RESOLUTION WITH ×2 AND ×4 UPSAMPLING. BEST AND SECOND BEST RESULTS IN **BOLD** AND UNDERLINED

| Scale | Model | Urban100 [44] | | | Manga109 [50] | | B100 [52] | | |
|---|---|---|---|---|---|---|---|---|---|
| | | PSNR | SSIM | LPIPS | PSNR | SSIM | PSNR | SSIM | LPIPS |
| 2x | Bicubic | 26.88 | 0.8431 | 0.383* | 30.80 | 0.9339 | 29.56 | 0.8316 | 0.396* |
| | SRCNN [72] | 29.50 | 0.8946 | N/A | 35.60 | 0.9663 | 31.36 | 0.8879 | N/A |
| | RCAN [73] | 33.34 | 0.9384 | 0.046* | 39.44 | 0.9786 | 32.41 | 0.9027 | 0.064* |
| | SAN [74] | 33.10 | 0.9370 | N/A | 39.32 | 0.9792 | 32.42 | 0.9028 | N/A |
| | HAN+ [75] | 33.53 | 0.9398 | 0.038* | 39.62 | 0.9787 | 32.41 | 0.9027 | 0.060* |
| | RCAN w/ adaP/U | 33.58 | 0.9456 | 0.036 | 39.67 | 0.9834 | 32.63 | 0.9103 | 0.057 |
| | HAN+ w/ adaP/U | **33.72** | **0.9469** | **0.027** | **39.82** | **0.9841** | **32.79** | **0.9187** | **0.051** |
| 4x | Bicubic | 23.14 | 0.6577 | 0.473 | 24.89 | 0.7866 | 25.96 | 0.6675 | 0.525 |
| | SRCNN [72] | 24.52 | 0.7221 | N/A | 27.58 | 0.8555 | 26.90 | 0.7101 | N/A |
| | RCAN [73] | 26.82 | 0.8087 | 0.098* | 31.22 | 0.9173 | 27.77 | 0.7436 | 0.121* |
| | SFTGAN [76] | 25.51 | 0.7549 | 0.177 | N/A | N/A | 27.13 | 0.7354 | 0.178 |
| | SAN [74] | 26.79 | 0.8068 | N/A | 31.18 | 0.9169 | 27.78 | 0.7436 | N/A |
| | SRGAN [77] | 25.50 | 0.7485 | 0.198 | N/A | N/A | 27.09 | 0.7360 | 0.171 |
| | HAN+ [75] | 27.02 | 0.8131 | 0.093* | 31.73 | 0.9207 | 27.85 | 0.7454 | 0.105* |
| | RCAN w/ adaP/U | 27.24 | 0.8195 | 0.089 | 31.78 | 0.9243 | **28.11** | 0.7482 | **0.093** |
| | HAN+ w/ adaP/U | **27.96** | **0.8246** | **0.066** | **32.30** | **0.9286** | 28.06 | **0.7513** | 0.095 |

TABLE IX

QUALITATIVE FRAME INTERPOLATION RESULTS ON VIMEO90K, MIDDLEBURY AND INTER4K. N/A INDICATES THAT THE RESULTS WERE NOT PROVIDED IN THE ORIGINAL WORKS. BEST RESULTS IN **BOLD**

| Model | Vimeo90K [44] | | | Middlebury [38] | | | Inter4K (4K, 30→60fps) | | |
|---|---|---|---|---|---|---|---|---|---|
| | PSNR | SSIM | LPIPS | PSNR | SSIM | LPIPS | PSNR | SSIM | LPIPS |
| DAIN [34] | 34.70 | 0.964 | 0.022 | 36.70 | 0.965 | 0.017 | 35.48 | 0.959 | 0.021 |
| CAIN [78] | 34.65 | 0.959 | 0.020 | 35.11 | 0.951 | 0.019 | 34.92 | 0.953 | 0.019 |
| BMBC [79] | 35.06 | 0.964 | 0.015 | 36.79 | 0.965 | 0.015 | 35.76 | 0.966 | 0.015 |
| XVFI [80] | 34.27 | 0.971 | N/A | N/A | N/A | N/A | 35.28 | 0.969 | 0.018 |
| CDFI [81] | 35.17 | 0.964 | 0.010 | 37.14 | 0.966 | 0.007 | 36.31 | 0.967 | 0.010 |
| [34] w/ adaP/U | 34.96 | 0.968 | 0.017 | 36.82 | 0.968 | 0.015 | 35.73 | 0.964 | 0.012 |
| [81] w/ adaP/U | **35.23** | **0.972** | **0.008** | **37.22** | **0.970** | **0.006** | **36.57** | **0.972** | **0.007** |

Our comparisons on image super-resolution are shown in Table VIII. Both RCAN [73] and HAN+ [75] perform favorably with down and up-sampling layers substituted by ada(Un)Pool. We observe that, in both cases of 2× and 4× image upsampling, our converted networks not only outperform their original implementations, but also other methods.

We demonstrate the merits of replacing all pooling and interpolation layers with ada(Un)Pool for frame interpolation in Table IX. The two converted networks, DAIN [34] and CDFI [81], produce improved results across the tested datasets. CDFI with adaPool and adaUnPool yields state-of-the-art results on both Vimeo90K and Middlebury as well as on our Inter4K for 4K video interpolation from 30 to 60 fps.

We also perform benchmarking tests on Inter4K with CDFI+ada(Un)Pool for the combined task of frame super-resolution and interpolation. Our findings are reported in Table X. Overall, we observe only slight degradation in performance on high-resolution, high-frame-rate conversions.

## VI. ABLATION STUDIES

In this section, we investigate the impact of different design choices for adaPool. We initially consider the effect of setting the $\boldsymbol{\beta}$ weight mask as trainable parameter or as constant value (Section VI-A). Additionally, we provide results on pooling layer replacements on the InceptionV3 [82] (Section VI-B), evaluate the performance over fusion and pooling method substitutions (Section VI-C), and compare against attention-based methods converted to downsampling (Section VI-D). Finally, we present qualitative visualizations of network saliency and the feature embedding space over original and adaPool-replaced models (Section VI-E). Unless otherwise specified, experiment settings follow those described in Section V-A.

TABLE X

FRAME INTERPOLATION AND SUPER-RESOLUTION WITH CDFI ON INTER4K. THE RESOLUTIONS AND FPS OF THE ORIGINAL AND PROCESSED VIDEOS ARE INDICATED IN THE SECOND COLUMN. BEST RESULTS IN **BOLD**

| Scale | Resolution and fps conversions | Measures | | |
|---|---|---|---|---|
| | | PSNR | SSIM | LPIPS |
| 2x | nHD15fps → HD30fps | 33.95 | 0.936 | 0.018 |
| | qHD24fps → FHD50fps | 33.91 | 0.928 | 0.020 |
| | HD30fps → QHD60fps | 33.87 | 0.925 | 0.021 |
| | FHD30fps → UHD60fps | 33.81 | 0.918 | 0.021 |
| 4x | nHD15fps → QHD60fps | 25.32 | 0.822 | 0.028 |
| | qHD15fps → UHD60fps | 25.38 | 0.819 | 0.031 |

TABLE XI

EFFECT OF $\boldsymbol{\beta}$ ON IMAGENET1K IMAGE CLASSIFICATION. LARGER VALUES OF $\boldsymbol{\beta}$ CORRESPOND TO STRONGER RELIANCE ON *e*DSCWPOOL WHILE SMALLER $\boldsymbol{\beta}$ VALUES PRIORITIZE *e*MPOOL. BEST RESULTS ARE IN **BOLD** WHILE SECOND BEST RESULTS ARE UNDERLINED

| Mode | $\beta$ value | ResNet-18 | | ResNet-34 | | InceptionV3 | |
|---|---|---|---|---|---|---|---|
| | | top-1 | top-5 | top-1 | top-5 | top-1 | top-5 |
| Constant | $\beta$=1/8 | 71.31 | 90.21 | 74.83 | 92.42 | 78.98 | 93.77 |
| | $\beta$=1/4 | 71.34 | 90.26 | 74.76 | 92.38 | 79.23 | 93.84 |
| | $\beta$=3/8 | 71.31 | 90.19 | 74.63 | 92.34 | 79.35 | 93.92 |
| | $\beta$=1/2 | 71.28 | 90.07 | 74.56 | 92.31 | 79.54 | 93.89 |
| | $\beta$=5/8 | 71.16 | 90.02 | 74.48 | 92.28 | 79.68 | 94.01 |
| | $\beta$=3/4 | 71.19 | 89.95 | 74.38 | 92.16 | 79.97 | 94.05 |
| | $\beta$=7/8 | 71.04 | 89.96 | 74.41 | 92.20 | 80.16 | 94.19 |
| trainable | | **71.78** | **90.65** | **75.43** | **92.87** | **81.34** | **94.57** |

### A. Effect of $\beta$ Weight Mask

In order to study how different combinations of the approximated maximum and average effect our proposed adaPool method, we present results in Table XI on ImageNet1K with several constant $\boldsymbol{\beta}$ values and study the performance gains when $\boldsymbol{\beta}$ is converted to a trainable weight mask.

Overall, the trainable setting provides the best performance across all three tested networks. The performance improvement of the trainable weight mask over the best-performing constant value becomes more apparent in complex architectures. In ResNet-18 the difference in top-1 is 0.44% while in InceptionV3 it becomes 1.18%. We provide further parameterization-based ablations in Appendix VII-D.

### B. Layer-Wise Ablation on InceptionV3

To understand the effect of adaPool at different network depths, we hierarchically ablate over pooling layers of the InceptionV3 architecture. This choice is primarily based on the Inception block's structure that includes pooling operations. This allows for a per-block evaluation of the change in the pooling operator.

TABLE XII
PROGRESSIVE LAYER SUBSTITUTION FOR INCEPTIONV3 ON IMAGENET1K. COLUMN NUMBERS REFER TO THE NUMBER OF REPLACED POOLING LAYERS, MARKED WITH ✓. BEST RESULTS IN **BOLD**

| Layer | Pooling layer substitution with adaPool | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 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 | 78.34 | 78.89 | 79.32 | 79.78 | 80.21 | 80.54 | **81.34** |
| Top-5 (%) | 93.56 | 93.77 | 93.92 | 94.05 | 94.17 | 94.26 | 94.32 | **94.57** |

TABLE XIII
TOP-1 ACCURACY OVER RUNS ON IMAGENET1K BASED ON DIFFERENT POOLING AND POOLING COMBINATION METHODS. A RESNET-18 IS USED FOR ALL EXPERIMENTS. TOP RESULTS ARE IN **BOLD** AND THE BEST RESULT PER FUSION METHOD IS UNDERLINED

| Fusion \ Pooling | avg+max | avg+*e*M | *e*DSCW+max | *e*DSC+*e*M |
|---|---|---|---|---|
| mixed | 70.37 | 70.73 | 70.65 | <u>71.08</u> |
| gate | 71.04 | 71.25 | 71.32 | <u>71.44</u> |
| adaptive **(ours)** | 71.42 | 71.56 | 71.53 | **71.78** |

From results summarized in Table XII, we observe that we can expect an average increase of 0.56% in top-1 accuracy with each additional replacement of an original pooling operation by adaPool. While the performance gains are systematic, the largest improvements are observed for replacements over the first pooling operation after the initial convolutional layer ($pool_1$) with a 0.89% jump in accuracy, and at the final Inception block ($mixed7_{b-d}$) with a 0.80% increase. We thus demonstrate that adaPool yields accuracy improvement through its adaptive weighting, regardless of the network depth and number of channels.

### C. Pooling Combinations Over Fusion Methods

We provide comparisons over additional pooling methods and fusion strategies proposed in [5]. The *mixed* pooling fusion strategy corresponds to using a single parameter to fuse the pooling methods used. This can be considered as a special case of adaptive pooling in which $|\boldsymbol{\beta}| = 1$. The *gate* fusion method uses a learned parameter to select either of the two used pooling methods. In addition to our *e*DSCW+*e*M combination, we also test fusion strategies with average/maximum pooling.

Our comparisons are shown in Table XIII. The combination of the smooth approximated average and maximum performs favorably over the different average or maximum-based combinations. We also observe that the use of a parameter mask through adaptive fusion helps to improve performance.

### D. Comparisons to Attention-Based Downsampling

The recent introduction of attention-based methods has shown great promise for a range of high-level vision tasks.

TABLE XIV
COMPARISON OF ADAPOOL TO ATTENTION-BASED DOWNSAMPLING FOR DENSENET-121 ON IMAGENET1K, WITH SE [83], CBAM [84], AND MSA [85]. BEST RESULTS ARE IN **BOLD**

| Method | top-1 | top-5 | +Params | +FLOPs |
|---|---|---|---|---|
| *Fixed approaches* | | | | |
| AvgPool (Baseline) | 74.65 | 92.17 | - | - |
| *e*M/SoftPool [9] | 75.88 | 92.92 | - | - |
| *e*DSCWPool | 76.06 | 93.16 | - | - |
| *Learned approaches* | | | | |
| AvgPool + SE [83] | 76.32 | 93.06 | +43.9K | +0.2G |
| *e*M/SoftPool + SE | 76.45 | 93.09 | +43.9K | +0.2G |
| AvgPool + CBAM [84] | 77.03 | 93.16 | +44.3K | +0.5G |
| *e*M/SoftPool + CBAM | 77.11 | 93.18 | +44.3K | +0.5G |
| AvgPool + MSA [85] | 77.38 | 93.27 | +1.4M | +2.5G |
| *e*M/SoftPool + MSA | **77.51** | **93.36** | +1.4M | +2.5G |
| adaPool **(ours)** | 77.29 | 93.21 | **+4.2K** | **+1.5M** |

We therefore also investigate the usability of three different attention-based approaches by adapting them for downsampling. We test the channel-wise Squeeze-and-Excitation (SE) [83] attention module, the locally-applied Convolutional Block Attention Module (CBAM) [84], and the Multiscale Self Attention module (MSA) [85] that uses global attention over spatially reduced **KQV** linear projections of the input. The tested modules are converted for downsampling by pooling after (SE, CBAM) or before (MSA) the attention modules.

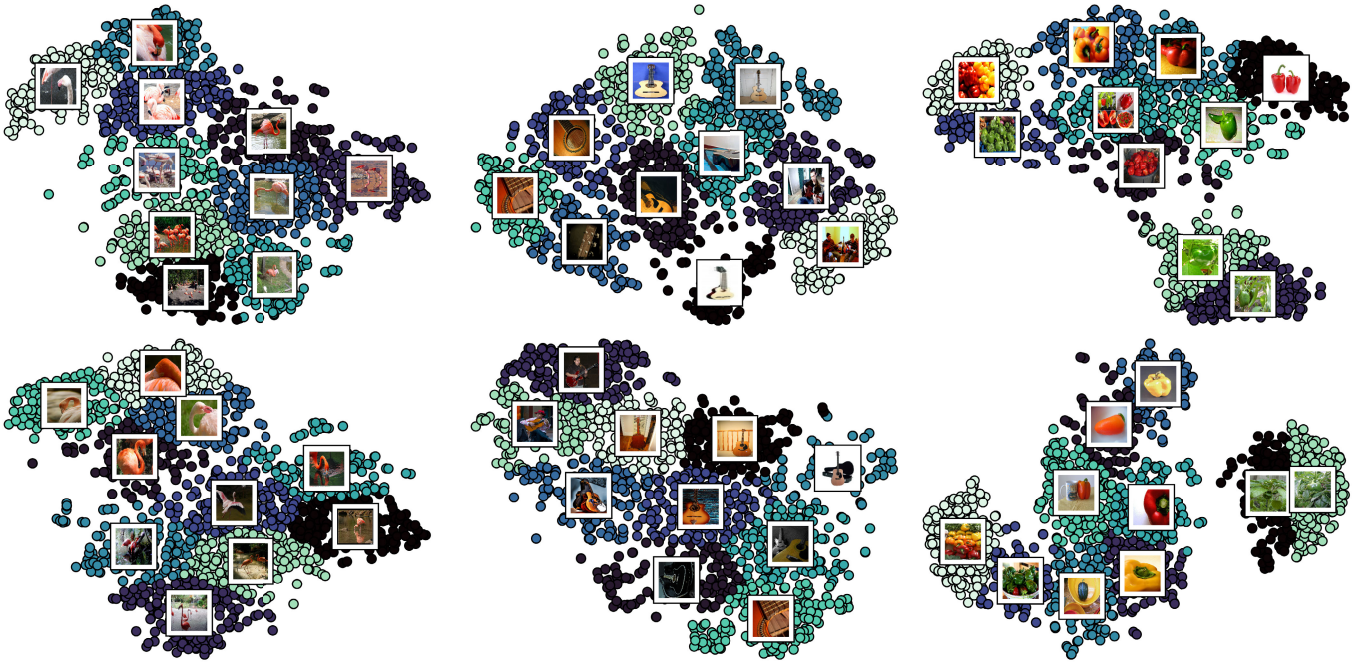
From the results presented in Table XIV, we observe that our proposed adaPool is substantially more efficient than any attention-based method with only requiring +1.5 additional MFLOPs and 4.2K parameters. AdaPool shows to perform favorably compared to SE-based and CBAM-based pooling methods while a small decrease in performance is observed in comparison to MSA with average or SoftPool. We note that the performance-to-computational complexity trade-off between adaPool and MSA-based pooling is substantial, with MSA requiring 1,600 more FLOPs than adaPool. For DenseNet-121 the computational burden with using MSA-based pooling is 30% of the total number of FLOPs used by the model.

### E. Qualitative Visualizations

To better understand the effect of adaPool in the feature extraction process, we compute saliency maps using Grad-CAM [86] to visualize the salient regions for the original and adaPool-substituted networks, shown in Figure 8. We use a fixed ResNet-50 model from Table III and sample examples from the ImageNet classes “pirate ship”, “tennis ball”, “go-cart”, “sea lion”, “convertible” and “paddle boat”.

For cases such as “go-cart” and “sea lion” where multiple objects of the class appear in the image, the adaPool-based network produces saliency maps that better fit their regions. Because details regarding the input are better preserved, the model focuses more on regions containing more descriptive features of the class, for example the sails in the “pirate ship” example or the racket and ball for “tennis ball”.

Additionally, in Figure 7 we provide t-SNE [87] visualizations for the feature embeddings of the original and adaPool-replaced InceptionV3. We follow the same recipe as in [9] and reduce the dimensionality to 50 channels with PCA. Overall,



a. Class “flamingo” b. Class “acoustic guitar” c. Class “bell pepper”

Fig. 7. **t-SNE feature embeddings for InceptionV3 with (bottom) and without (top) adaPool**. The ImageNet1K classes used are “flamingo”, “acoustic guitar” and “bell pepper.”

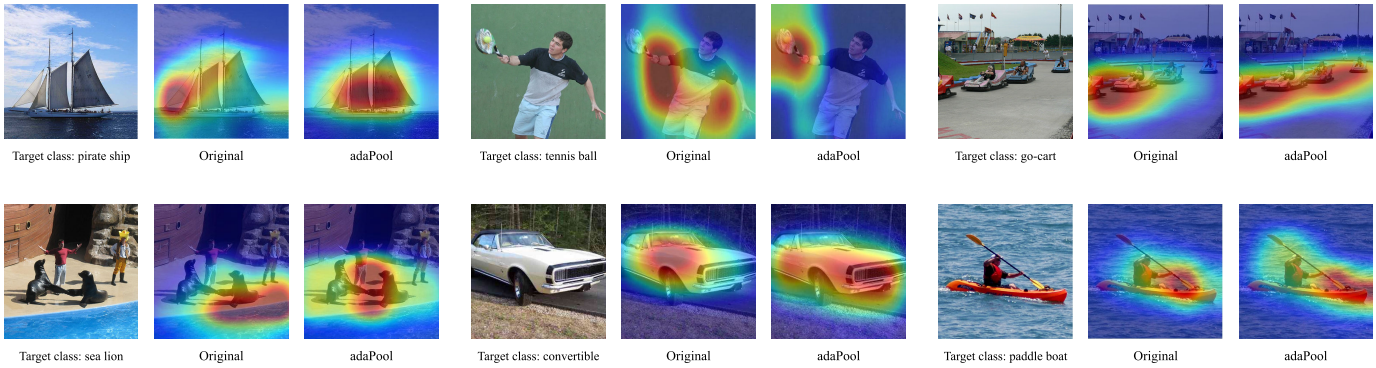



Fig. 8. **Saliency maps.** We compare maps of the visual saliency from two ResNet-50 models with the original max pooling and the proposed adaPool. Examples are sampled from the validation set of ImageNet1K. For each image we show the ground truth label.

feature embeddings for similar examples are shown to be mapped somewhat closer on the adaPool-enabled network. For example, there is a clearer distinction between the color of the peppers for the class “bell pepper” as well as a distinction between multiple or single peppers in an image.

## VII. Conclusion

In this paper, we have proposed adaPool, a pooling method for the preservation of informative features based on adaptive exponential weighting. It is a regionally-adaptive method that uses the parameterized fusion of the exponential maximum eMPool and exponential average eDSCWPool. The weights of adaPool can be used to invert the pooling operation (adaUnPool), to achieve upsampling.

We have tested our approach on image and video classification, image similarity, object detection, image and frame super-resolution tasks, as well as frame interpolation. The experiments consistently demonstrate the merits of our proposed approach when faced with various challenges such as capturing global and local information, or to consider 2D image data and 3D video data. Over all downstream tasks, and using a variety of network backbones and experiment settings, adaPool systematically outperforms any other method while computational latencies and memory use remain modest. Based on these extensive experiments, we believe adaPool is a good alternative for currently popular pooling operators.

## Appendix A

In this appendix, we provide more details on Inverse Distance Weighting (IDW) pooling (Section VII-A), a motivation for our use of the Dice-Sørensen Coefficient (DSC, Section VII-B), a comparison with other soft average methods (Section VII-C), and a description of the computational complexity of our implementation (Section VII-E).

### A. Inverse Distance Weighting Pooling

To assign a weight value, IDW relies on the measured observation distances within the region. A visual representation of this weighting process is shown in Figure 9.

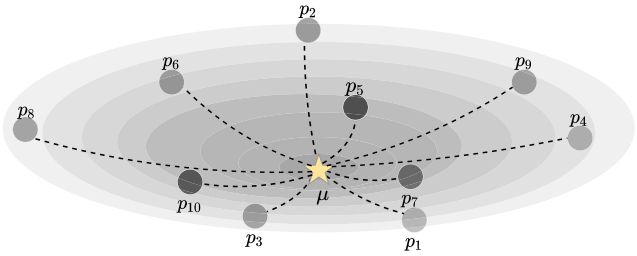



Fig. 9. **Inverse Distance Weighting**. Given multiple points $\{p_1, \ldots, p_n\}$ in a feature space and their mean ($\mu$), their weights are equal to the inverse of their distance divided by their sum.

To overcome the limitations of uniformly-weighted region averaging, we adapt IDW for pooling, which we term *IDW-Pool*. Our results in Section V use the Euclidean distance ($L_2$) between the mean and the individual activations. We also provide an overview alongside results for alternative distance functions in the following sections. In comparison to uniformly-weighted averaging, IDWPool produces normalized results with higher weights for feature activation vectors that are geometrically closer to the mean. This also applies to the calculation of the gradients, and reduces the effect of outliers, providing a better representative update rate based on feature activation relevance. In that aspect, IDWPool works differently than the common approach of averaging all activations in which the output activation is not regularized.

Although IDWPool can provide an improvement over uniformly-weighted averaging, we argue that weighted averaging based on distance is sub-optimal over multi-dimensional spaces. One of the main drawbacks of a naive IDWPool implementation is that the $L_1$ or $L_2$ distance between the feature activation vector and the average over the region are calculated based on the mean, sum or maximum per-channel pair. The resulting distance is unbounded since the pair-wise distances are also unbounded. In addition, the calculated distance is sensitive to channel pair outliers. The effect of this is visible with the pixel artifacts of the inverse distance weighting approaches in Figure 10. When using distance methods, the computed distance in certain channels can be significantly larger than in others. This creates the problem of weights that are nearing zero ($\underset{IDW}{w}(\bar{\mathbf{a}}_\mathbf{c}, \mathbf{a}_{j,c}) \rightarrow 0$).

### B. Coefficient-Based Methods

We have considered other similarity-based methods to find the relevance of two volumes of vectors [90]. Apart from the cosine similarity, the Kumar and Hassebrook Peak-to-correlation energy (PCE) [91] can be applied to vector volumes (as shown in Table XVI). We present the differences in the pooling quality based on different similarity methods in Figure 10. Considering the aforementioned shortfalls of cosine similarity, our use of DSC over PCE is primarily due to PCE's non-monotonic nature and value distribution [91].

### C. Comparison With Alternative Soft Average Methods

To evaluate the effect of different distance and similarity measures for average-approximating pooling in image classification performance, we use a ResNet-18 as backbone. We set as baseline the original ResNet-18 with maximum pooling.

TABLE XV
**DISTANCE FUNCTIONS FOR VECTORS**. ALL METHODS CAN BE APPLIED TO MULTI-DIMENSIONAL VECTOR VOLUMES

| | | |
|---|---|---|
| Manhattan ($L_1$) | $\underset{L_1}{d} = \sum_{c \in \mathbf{C}} \lvert\lvert \bar{\mathbf{a}}_c - \mathbf{a}_{i,c} \rvert\rvert$ | (8) |
| Euclidean ($L_2$) | $\underset{L_2}{d} = \sum_{c \in \mathbf{C}} \sqrt{\lvert\lvert \bar{\mathbf{a}}_c - \mathbf{a}_{i,c} \rvert\rvert^2}$ | (9) |
| Huber [57] | $\underset{Hub}{d} = \begin{cases} \frac{d_{L_1}^2}{2}, \; if \; d_{L_2} \leq \delta \\ \delta \cdot (d_{L_1} - \frac{\delta}{2}) \end{cases}$ | (10) |
| Chebyshev [88] | $\underset{L_{Che}}{d} = \underset{c \in C}{max} \; d_{L_1}$ | (11) |
| Gower [89] | $\underset{L_{Gow}}{d} = \frac{1}{C} \cdot d_{L_1}$ | (12) |

TABLE XVI
**SIMILARITY FUNCTIONS FOR VECTORS**. ALL METHODS CAN BE DIRECTLY APPLIED TO MULTI-DIMENSIONAL VECTOR VOLUMES

| | | |
|---|---|---|
| Cosine | $\underset{cos}{S} = \frac{\sum_{c \in \mathbf{C}} \bar{\mathbf{a}} \cdot \mathbf{a}_{i,c}}{\sqrt{\sum_{c \in C} \bar{\mathbf{a}}_c^2} \cdot \sqrt{\sum_{c \in C} \mathbf{a}_c^2}}$ | (13) |
| PCE | $\underset{PCE}{S} = \frac{\sum_{c \in \mathbf{C}} \bar{\mathbf{a}} \cdot \mathbf{a}_{i,c}}{\sum_{c \in C} \bar{\mathbf{a}}_c^2 + \sum_{c \in C} \mathbf{a}_c^2 - \sum_{c \in C} \bar{\mathbf{a}}_c \cdot \mathbf{a}_c}$ | (14) |
| DSC | $\underset{DSC}{S} = \sum_{c \in \mathbf{C}} \frac{2 \cdot \lvert\lvert \bar{\mathbf{a}}_c \cdot \mathbf{a}_{i,c} \rvert\rvert}{\lvert\lvert \bar{\mathbf{a}}_c \rvert\rvert^2 + \lvert\lvert \mathbf{a}_{i,c} \rvert\rvert^2}$ | (15) |

The results in Table XVII show negligible differences between distances in IDW pooling. Huber-based pooling shows small top-1 accuracy improvements, in the range of +(0.10–0.19)% over $L_1$, $L_2$ and Chebyshev distance-weighting. A slight performance reduction is observed with the Gower method. This could be because of the production of small weight values as Gower uses the $L_1$ distance divided by the number of channels (Equation 12).

Compared to distance approaches, similarity measures show a larger increase over the baseline model. This can be

TABLE XVII
IMAGENET1K CLASSIFICATION WITH DISTANCE- AND SIMILARITY-BASED POOLING ALTERNATIVES ON RESNET-18. DISTANCE-BASED METHODS ARE DENOTED BY IDW, WHILE SIMILARITY-BASED METHODS ARE DENOTED WITH SIM. BEST RESULTS IN **BOLD**

| | Method | top-1 | top-5 |
|---|---|---|---|
| | Original (Baseline) | 69.76 | 89.08 |
| IDW | $L_1$ | 69.94 (+0.18) | 89.24 (+0.16) |
| IDW | $L_2$ | 70.02 (+0.23) | 89.28 (+0.20) |
| IDW | Huber [57] $\delta = 1/4$ | 70.11 (+0.35) | 89.33 (+0.25) |
| IDW | Huber [57] $\delta = 1/2$ | 70.09 (+0.33) | 89.27 (+0.19) |
| IDW | Huber [57] $\delta = 3/4$ | 70.13 (+0.37) | 89.32 (+0.24) |
| IDW | Chedyshev | 69.96 (+0.20) | 89.20 (+0.12) |
| IDW | Gower | 69.58 (-0.18) | 88.94 (-0.14) |
| Sim. | Cosine | 70.45 (+0.69) | 89.44 (+0.36) |
| Sim. | PCE | 70.54 (+0.78) | 89.51 (+0.43) |
| Sim. | DSC | **70.66** (+0.90) | **89.77** (+0.69) |


Fig. 10. **Instances of Average Distance/Similarity Weighting Methods**. Distance kernel weights based on IDW [12] with various inverse distance functions. Similarity kernel weights based on (*e*)PCEW, (*e*)cosW and (*e*)DSCW.

attributed to the sparsity of the per-input volumes. Considering the relatively small size of the kernel ($k \times k$) and the high-dimensional spaces they are represented in, distances between points and their mean are large [92]. The Dice-Sørensen coefficient is most effective with 70.66% and 89.77% top-1 and top-5 accuracies. Increases are observed by the exponent of DSC in *e*DSCWPool shown in Table III, with 70.79% top-1 and 90.16% top-5 accuracies.

### D. Ablations Over $\beta$ Parameterization Alternatives

As adaPool introduces additional parameters. Therefore, we evaluate if the observed gains in performance are indeed due to improved information retainment or simply due to the inclusion of more parameters. We use three different $\beta$ sizes: a single $|\beta| = 1$ parameter shared across each location, our proposed mask $|\beta| = H' \times W'$ for individual parameters across each location, and a channel-wise mask $|\beta| = H' \times W' \times C$ for both location and channel-based parameters. We present results on ResNet-50 and DenseNet-161 in Table XVIII. We observe a difference between our proposed mask-based $\beta$ and the largely parameterized channel-wise $\beta$ on both models, with 1.01% in ResNet-50 and 1.28% in DenseNet-121. The results suggests that improvements in performance are not solely dependent on the inclusion of additional parameters. The channel-wise $\beta$ underperforms compared to the other non-channel-wise parameterization approaches. This suggests that the pooling approach is better suited for data with larger channel and feature dependencies. Our proposed approach introduces only a small fraction of additional parameters compared to the parameters used by most models, with +3.1K on ResNets and +4.2K on DenseNets (see Table XIX). We conclude that the observed performance improvements are strongly related to the design of adaPool instead of the additional parameters.

TABLE XVIII
ADAPOOL $\beta$ PARAMETERIZATION ALTERNATIVES ON IMAGENET1K FOR RESNET-50 AND DENSENET-121. BEST RESULTS AND SETTINGS IN **BOLD**

| Method | top-1 | Params | FLOPs |
|---|---|---|---|
| ResNet-50 | | | |
| Baseline (AvgPool) | 76.15 | 25.6M | 4.14G |
| $\beta$ single | 77.76 | +1 | +0.8M |
| $\beta$ mask (proposed) | **78.42** | +3.1K | +0.8M |
| $\beta$ channel-wise | 77.41 | +198.5K | +0.8M |
| DenseNet-121 | | | |
| Baseline (AvgPool) | 74.65 | 8.6M | 2.9G |
| $\beta$ single | 76.41 | +4 | +1.5M |
| $\beta$ mask (proposed) | **77.29** | +4.2K | +1.5M |
| $\beta$ channel-wise | 76.13 | +0.5M | +1.5M |

TABLE XIX
**PARAMETERS AND FLOPS** OVERHEAD WITH THE INCLUSION OF ADAPOOL PER FAMILY OF ARCHITECTURES

| Arch. | Params (K) | FLOPs (M) |
|---|---|---|
| ResNets | +3.1 | +0.8 |
| InceptionV3 | +3.5 | +1.3 |
| DenseNets | +4.2 | +1.5 |

### E. Computational Description

Our implementation is in CUDA and thus allows the native run on GPUs, providing inference times close to those of native methods such as average and maximum pooling. Due to the parallelization capabilities of both exponential maximum and average pooling methods, running times are close to those of average pooling with $\mathcal{O}(2)$ and $\mathcal{O}(3)$ respectively, as operations can be performed in parallel over the kernel region matrix. In contrast, max pooling has $\mathcal{O}(n)$ computational complexity, due to the sequential consideration of each input within the region in order to discover the maximum.

Both eMPool and eDSCWPool are on par with average and maximum pooling due to CUDA's memory reduction through data partitioning with tiling. In addition, both can be implemented through fused multiply-adds (FMA) that significantly improve performance on CUDA-enabled devices [93].

## ACKNOWLEDGMENT

The authors would like to thank the Netherlands Organization for Scientific Research (NWO) for their support.

## REFERENCES


[1] J. B. Estrach, A. Szlam, and Y. LeCun, "Signal recovery from pooling representations," in *Proc. ICML*, 2014, pp. 307–315.

[2] C. Gulcehre, K. Cho, R. Pascanu, and Y. Bengio, "Learned-norm pooling for deep feedforward and recurrent neural networks," in *Proc. ECML PKDD*, 2014, pp. 530–546.

[3] M. D. Zeiler and R. Fergus, "Stochastic pooling for regularization of deep convolutional neural networks," in *Proc. ICLR*, 2013, pp. 1–9.

[4] S. Zhai et al., "S3Pool: Pooling with stochastic spatial sampling," in *Proc. CVPR*, 2017, pp. 4970–4978.

[5] C.-Y. Lee, P. W. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," in *Proc. AISTATS*, 2016, pp. 464–472.

[6] Z. Gao, L. Wang, and G. Wu, "LIP: Local importance-based pooling," in *Proc. ICCV*, 2019, pp. 3355–3364.

[7] L. R. Dice, "Measures of the amount of ecologic association between species," *Ecology*, vol. 26, no. 3, pp. 297–302, 1945.

[8] T. J. Sørensen, *A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species Content and Its Application to Analyses of the Vegetation on Danish Commons*. The Danish Academy of Sciences and Letters, 1948.

[9] A. Stergiou, R. Poppe, and K. Grigorios, "Refining activation downsampling with SoftPool," in *Proc. ICCV*, 2021, pp. 10357–10366.

[10] J. Zhao and C. G. Snoek, "LiftPool: Bidirectional ConvNet pooling," in *Proc. ICLR*, 2021, pp. 1–15.

[11] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder–decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[12] D. Shepard, "A two-dimensional interpolation function for irregularly-spaced data," in *Proc. 23rd ACM Nat. Conf.*, 1968, pp. 517–524.

[13] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Proc. ECCVW*, 2004, pp. 1–22.

[14] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3360–3367.

[15] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2006, pp. 2169–2178.

[16] J. Yang, K. Yu, Y. Gong, and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1794–1801.

[17] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, pp. 994–1000.

[18] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proc. ICML*, 2010, pp. 111–118.

[19] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[20] D. Yu, H. Wang, P. Chen, and Z. Wei, "Mixed pooling for convolutional neural networks," in *Proc. RSKT*, 2014, pp. 364–375.

[21] F. Saeedan, N. Weber, M. Goesele, and S. Roth, "Detail-preserving pooling in deep networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9108–9116.

[22] R. D. Luce, "The choice axiom after twenty years," *J. Math. Psychol.*, vol. 15, no. 3, pp. 215–233, Jun. 1977.

[23] H. Akima, "A method of bivariate interpolation and smooth surface fitting for irregularly distributed data points," *ACM Trans. Math. Softw.*, vol. 4, no. 2, pp. 148–159, 1978.

[24] R. Franke, "Scattered data interpolation: Tests of some methods," *J. Math. Comput.*, vol. 38, no. 157, pp. 181–200, 1982.

[25] B. Fernando and S. Herath, "Anticipating human actions by correlating past with the future with Jaccard similarity measures," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 13224–13233.

[26] M. Riesenhuber and T. Poggio, "Hierarchical models of object recognition in cortex," *Nature Neurosci.*, vol. 2, no. 11, pp. 1019–1025, Nov. 1999.

[27] S. Jegou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers Tiramisu: Fully convolutional DenseNets for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 11–19.

[28] G. Lin, A. Milan, C. Shen, and I. Reid, "RefineNet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1925–1934.

[29] P. Wang et al., "Understanding convolution for semantic segmentation," in *Proc. WACV*, 2018, pp. 1451–1460.

[30] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, Atrous convolution, and fully connected CRFs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 834–848, Apr. 2016.

[31] W. Li, X. Tao, T. Guo, L. Qi, J. Lu, and J. Jia, "MuCAN: Multi-correspondence aggregation network for video super-resolution," in *Proc. ECCV*. Cham, Switzerland: Springer, 2020, pp. 335–351.

[32] L. Lu, W. Li, X. Tao, J. Lu, and J. Jia, "MASA-SR: Matching acceleration and spatial adaptation for reference-based image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 6368–6377.

[33] Z. Wang, J. Chen, and S. Hoi, "Deep learning for image super-resolution: A survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 10, pp. 3365–3387, Mar. 2020.

[34] W. Bao, W.-S. Lai, C. Ma, X. Zhang, Z. Gao, and M.-H. Yang, "Depth-aware video frame interpolation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 3703–3712.

[35] Y. L. Liu, Y. T. Liao, Y. Y. Lin, and Y. Y. Chuang, "Deep video frame interpolation using cyclic frame generation," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 8794–8802.

[36] S. Niklaus and F. Liu, "Context-aware synthesis for video frame interpolation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1701–1710.

[37] S. Niklaus and F. Liu, "Softmax splatting for video frame interpolation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 5437–5446.

[38] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski, "A database and evaluation methodology for optical flow," *Int. J. Comput. Vis.*, vol. 92, no. 1, pp. 1–31, Mar. 2011.

[39] C. Liu and D. Sun, "On Bayesian adaptive video super resolution," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 2, pp. 346–360, Feb. 2014.

[40] A. Mercat, M. Viitanen, and J. Vanne, "UVG dataset: 50/120 fps 4K sequences for video codec analysis and development," in *Proc. 11th ACM Multimedia Syst. Conf.*, May 2020, pp. 297–302.

[41] K. Soomro, A. R. Zamir, and M. Shah, "UCF101: A dataset of 101 human actions classes from videos in the wild," 2012, *arXiv:1212.0402*.

[42] H. Takeda, P. Milanfar, M. Protter, and M. Elad, "Super-resolution without explicit subpixel motion estimation," *IEEE Trans. Image Process.*, vol. 18, no. 9, pp. 1958–1975, Sep. 2009.

[43] P. Yi, Z. Wang, K. Jiang, J. Jiang, and J. Ma, "Progressive fusion video super-resolution network via exploiting non-local spatio–temporal correlations," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3106–3115.

[44] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman, "Video enhancement with task-oriented flow," *Int. J. Comput. Vis.*, vol. 127, no. 8, pp. 1106–1125, 2019.

[45] T.-Y. Lin et al., "Microsoft COCO: Common objects in context," in *Proc. ECCV*, 2014, pp. 740–755.

[46] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.

[47] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proc. ICCV*, 2017, pp. 2961–2969.

[48] E. Agustsson and R. Timofte, "NTIRE 2017 challenge on single image super-resolution: Dataset and study," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jul. 2017, pp. 126–135.

[49] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 5197–5206.

[50] Y. Matsui et al., "Sketch-based Manga retrieval using Manga109 dataset," *Multimedia Tools Appl.*, vol. 76, no. 20, pp. 21811–21838, 2017.

[51] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.

[52] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th IEEE Int. Conf. Comput. Vis. (ICCV)*, vol. 2, Jul. 2001, pp. 416–423.

[53] H. Zhao, A. Torralba, L. Torresani, and Z. Yan, "HACS: Human action clips and segments dataset for recognition and temporal localization," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 8668–8678.

[54] J. Carreira, E. Noland, C. Hillier, and A. Zisserman, "A short note on the kinetics-700 human action dataset," 2019, *arXiv:1907.06987*.

[55] C.-Y. Wu, R. Girshick, K. He, C. Feichtenhofer, and P. Krahenbuhl, "A multigrid method for efficiently training video models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 153–162.

[56] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2117–2125.

[57] P. J. Huber, "Robust estimation of a location parameter," in *Breakthroughs in Statistics*. New York, NY, USA: Springer, 1992, pp. 492–518.

[58] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[59] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 586–595.

[60] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[61] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 2261–2269.

[62] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 5987–5995.

[63] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. BMVC*, 2016, pp. 87.1–87.12.

[64] H. Kataoka, T. Wakamiya, K. Hara, and Y. Satoh, "Would mega-scale datasets further enhance spatiotemporal 3D CNNs?" 2020, *arXiv:2004.04968*.

[65] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, "A closer look at spatiotemporal convolutions for action recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6450–6459.

[66] J. Carreira and A. Zisserman, "Quo vadis, action recognition? A new model and the kinetics dataset," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4724–4733.

[67] D. Tran, H. Wang, M. Feiszli, and L. Torresani, "Video classification with channel-separated convolutional networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 5552–5561.

[68] A. Stergiou and R. Poppe, "Learn to cycle: Time-consistent feature discovery for action recognition," *Pattern Recognit. Lett.*, vol. 141, pp. 1–7, Jan. 2021.

[69] C. Feichtenhofer, H. Fan, J. Malik, and K. He, "SlowFast networks for video recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 6202–6211.

[70] A. Stergiou and R. Poppe, "Multi-temporal convolutions for human action recognition in videos," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2021, pp. 1–9.

[71] A. Stergiou and R. Poppe, "Analyzing human–human interactions: A survey," *Comput. Vis. Image Understand.*, vol. 188, Nov. 2019, Art. no. 102799.

[72] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Proc. ECCV*, 2014, pp. 184–199.

[73] Y. Zhang, K. Li, K. Li, L. Wang, B. Zhong, and Y. Fu, "Image super-resolution using very deep residual channel attention networks," in *Proc. ECCV*, 2018, pp. 286–301.

[74] T. Dai, J. Cai, Y. Zhang, S.-T. Xia, and L. Zhang, "Second-order attention network for single image super-resolution," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 11065–11074.

[75] B. Niu et al., "Single image super-resolution via a holistic attention network," in *Proc. ECCV*, 2020, pp. 191–207.

[76] X. Wang, K. Yu, C. Dong, and C. C. Loy, "Recovering realistic texture in image super-resolution by deep spatial feature transform," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 606–615.

[77] C. Ledig et al., "Photo-realistic single image super-resolution using a generative adversarial network," in *Proc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2017, pp. 4681–4690.

[78] M. Choi, H. Kim, B. Han, N. Xu, and K. M. Lee, "Channel attention is all you need for video frame interpolation," in *Proc. AAAI*, 2020, vol. 34, no. 7, pp. 10663–10671.

[79] J. Park, K. Ko, C. Lee, and C.-S. Kim, "BMBC: Bilateral motion estimation with bilateral cost volume for video interpolation," in *Proc. ECCV*, 2020, pp. 109–125.

[80] H. Sim, J. Oh, and M. Kim, "XVFI: eXtreme video frame interpolation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2021, pp. 14489–14498.

[81] T. Ding, L. Liang, Z. Zhu, and I. Zharkov, "CDFI: Compression-driven network design for frame interpolation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 8001–8011.

[82] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[83] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 7132–7141.

[84] S. Woo, J. Park, J. Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 3–19.

[85] Y. Li et al., "MViTv2: Improved multiscale vision transformers for classification and detection," in *Proc. CVPR*, 2022, pp. 4804–4814.

[86] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," *Int. J. Comput. Vis.*, vol. 2017, pp. 618–626, Dec. 2017.

[87] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. 11, pp. 1–27, 2008.

[88] F. Van Der Heijden, R. P. Duin, D. De Ridder, and D. M. Tax, *Classification, Parameter Estimation and State Estimation: An Engineering Approach Using MATLAB*. Hoboken, NJ, USA: Wiley, 2005.

[89] J. C. Gower, "A general coefficient of similarity and some of its properties," *Biometrics*, vol. 27, no. 4, pp. 857–871, 1971.

[90] S.-H. Cha, "Comprehensive survey on distance/similarity measures between probability density functions," *Int. J. Current Microbiol. Appl. Sci.*, vol. 1, no. 2, p. 1, 2007.

[91] B. V. Kumar and L. Hassebrook, "Performance measures for correlation filters," *Appl. Opt.*, vol. 29, no. 20, pp. 2997–3006, 1990.

[92] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[93] V. W. Lee et al., "Debunking the 100X GPU vs. CPU myth: An evaluation of throughput computing on CPU and GPU," in *Proc. ISCA*, 2010, pp. 451–460.

**Alexandros Stergiou** (Student Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science from the University of Essex and the Ph.D. degree in computer science from the Department of Information and Computing Sciences, Utrecht University, in 2021. He is currently a Research Associate with the Department of Computer Science, University of Bristol. His research interests include recognition and prediction of human actions from videos and deep learning model explainability.

**Ronald Poppe** (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Twente, The Netherlands, in 2009. He was a Visiting Researcher at the Delft University of Technology, Stanford University, and the University of Lancaster. He is currently an Associate Professor with the Department of Information and Computing Sciences, Utrecht University. His research interests include modeling of visual attention and the analysis of human (interactive) behavior from video.