Refining activation downsampling with SoftPool – Supplementary Material

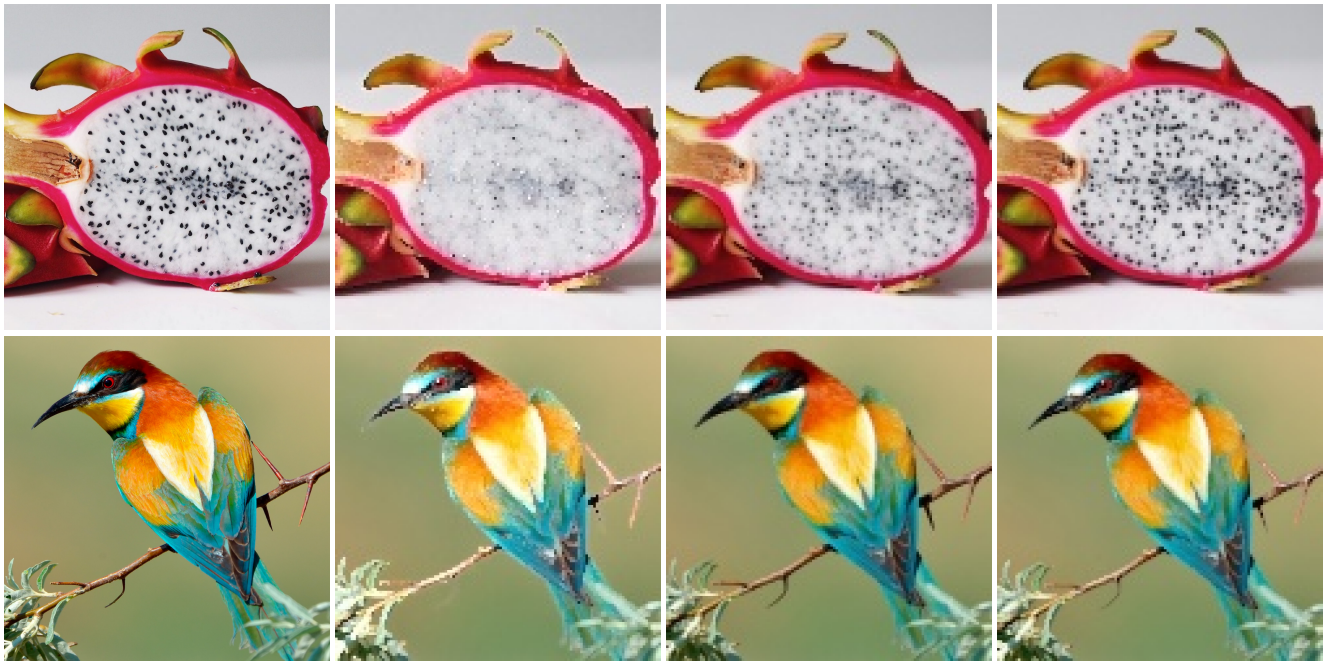


Figure S1: **High resolution pooled images**. Original images are of size 1200×1200. The downsampled images were created with $\times 3$ pooling operations. To match the sizes of the original images and make the downsampling result more visible, we used inter area interpolation to resize the pooled images. This does not create a smoothing effect between neighboring pixels that are interpolated but rather populates new pixels based on the area relation.

S1. Detail preservation

We discussed and demonstrated the feature preservation capabilities of SoftPool in Section 3.3. Here, we provide high-resolution images from examples in Figure 3 of the paper, in order to demonstrate clearer the effects of each pooling method. As it can be seen in Figure S1, SoftPool can better capture detail in high-contrasting areas. Evidence of this can be seen in the top image where the dragon fruit seeds are not always preserved in the down-sampled images. For max pooling, most of the seeds are lost. This also applies to a certain extent to average pooling. By selecting the average within a region, features with high contrast are smoothed over, which reduces their effect significantly. In contrast, SoftPool preserves such regions in the sub-sampled outputs. By including part of the low-intensity regions in the output while weighting the high-intensity regions more, it can preserve the little-contrasting pixels. A similar pattern can also be seen with low-contrasting regions such as the bird's eye in the second row where, again, max pooling will highlight the high intensity features in the output while the downsampled output becomes less similar to that of the original image. Average pooling would instead make low-contrast features much more difficult to be recognized. SoftPool provides a balance between the two methods by weighting each part of the region respectfully to its intensity value.

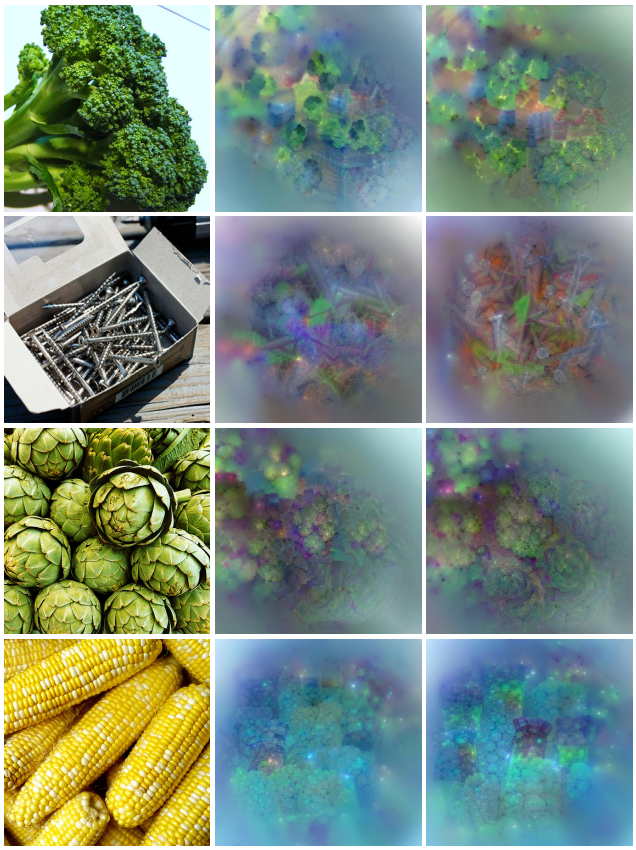


Figure S2: **Neuron activation maximization for InceptionV3 with original and SoftPool pooling layers.** We maximized the top-10 neurons [4, 13] of the final block (*Mixed7c*) in InceptionV3 [16]. Target top-10 neuron combinations for each row were selected for ImageNet1K classes "broccoli", "nails", "artichoke" and "corn".

S2. Model feature visualization

Learned feature interpretability aims at understanding the features that networks associate with each class. One technique is *activation maximization* [4] which creates a synthetic image by maximizing the activations relating to a specific neuron. Neurons could correspond to a specific class [12] or features within the feature extractor [2].

To test the representation capabilities of networks with SoftPool, we use InceptionV3 [16] as a backbone model and visualize the top-10 most informative features. To train, we initialize an image with random noise, which we use as input for each of the two tested networks. During the training process, the image is optimized with the maximization of the activations of the top-10 kernels in the final InceptionV3 block (*Mixed7c*) as objective function. The top-10 kernels are selected based on the highest average activations across all class examples. To eliminate additional noise, we use a mask-based approach similar to Wei *et al.* [18]. However, in our setting, the mask is responsible for reducing the size of the gradient vectors for regions that are further away from the image center.

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

We show the outputs in Figure S2 for four ImageNet1K classes: "broccoli", "nails", "artichoke" and "corn". We include in the left column the ImageNet1K images with the highest activations to provide representative examples. The majority of the features are fairly similar between the two models. Since SoftPool does not change the overall architecture nor the parameters, a high degree of similarity is expected. However, in cases such as the heads of the nails, we do notice a better definition of the objects. We also notice for the artichoke class that the structure of the petals and the thorns are more easy to distinguish from the network with SoftPool layers compared to the network with the original pooling layers. Although the differences remain small, by only changing the downsampling method used by the network, it can affect the robustness and improve the feature interpretability to a degree.

S3. Spatio-temporal volume pooling

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

We show in Figure S3 with four different examples the effects of spatio-temporal pooling with average pooling, max-pool, and SoftPool operations. As none of the methods is tailored towards completely alleviating the effects of encoded motion in the pooled output, they are visible in edges and regions where cross-frame motion exists. However, differences between the three methods become apparent.

We demonstrate part of these differences in Figure S3(e). Where we include two cases of zoomed-in frame regions that show some variance based on the pooling method used. In the top one, gaps in the wooden planks of the floor are significantly less distinguishable within the max-pooled frame. Consequently, in the average pooled frame region, the nails are not visible at all anymore. This effect is in line with our observations for image-based downsampling of high-contrast and low-contrast regions. In contrast, SoftPool preserves features in both cases, which allows for the extraction of representative features after pooling.

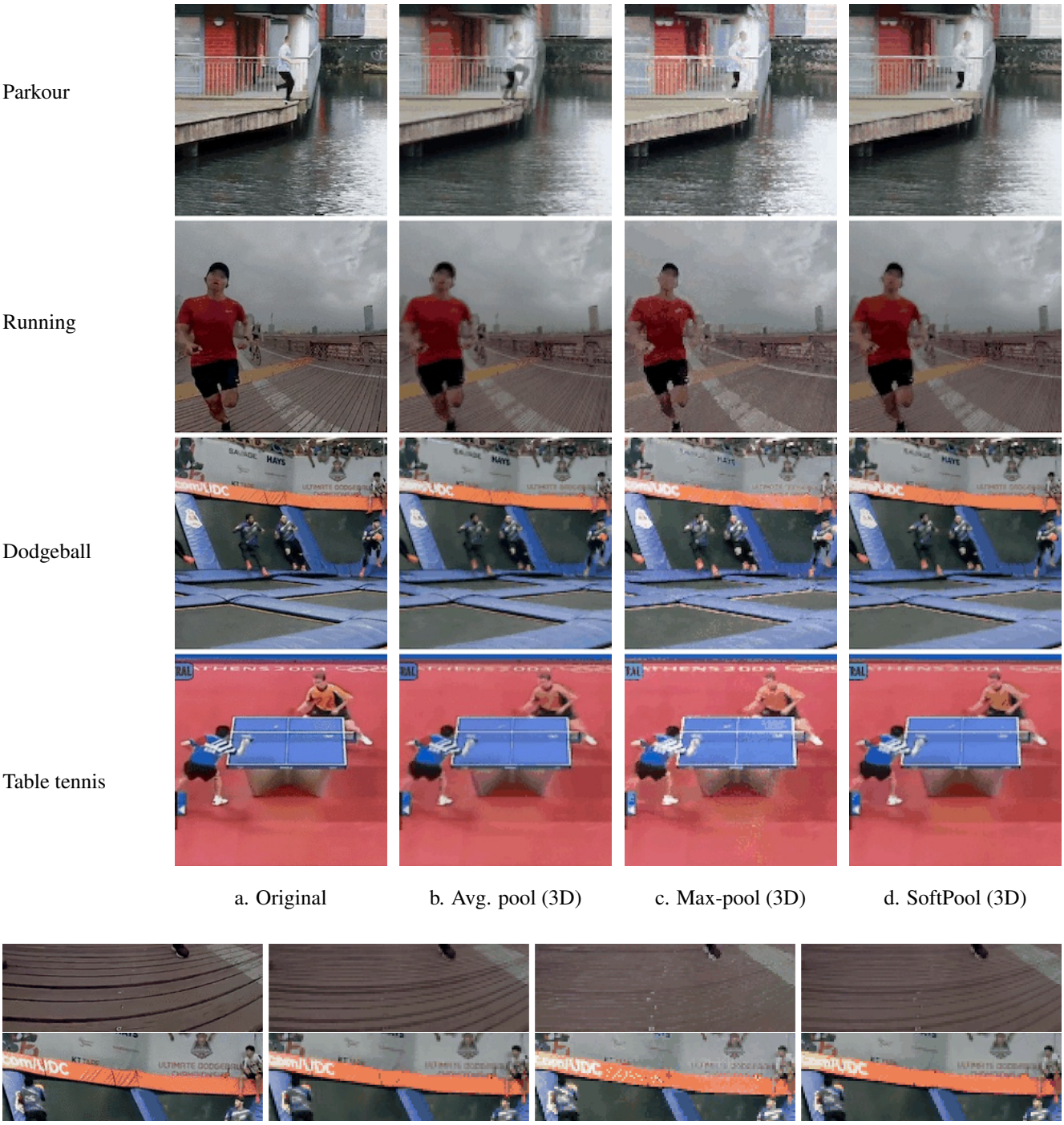


Figure S3: **Spatio-temporally downsampled videos.** Sub-sampling over spatio-temporal volumes compared to original videos (a), and downsampled with average pooling (b), maximum pooling (c) and our proposed SoftPool (d). Two zoomed-in frame regions appear in (e).

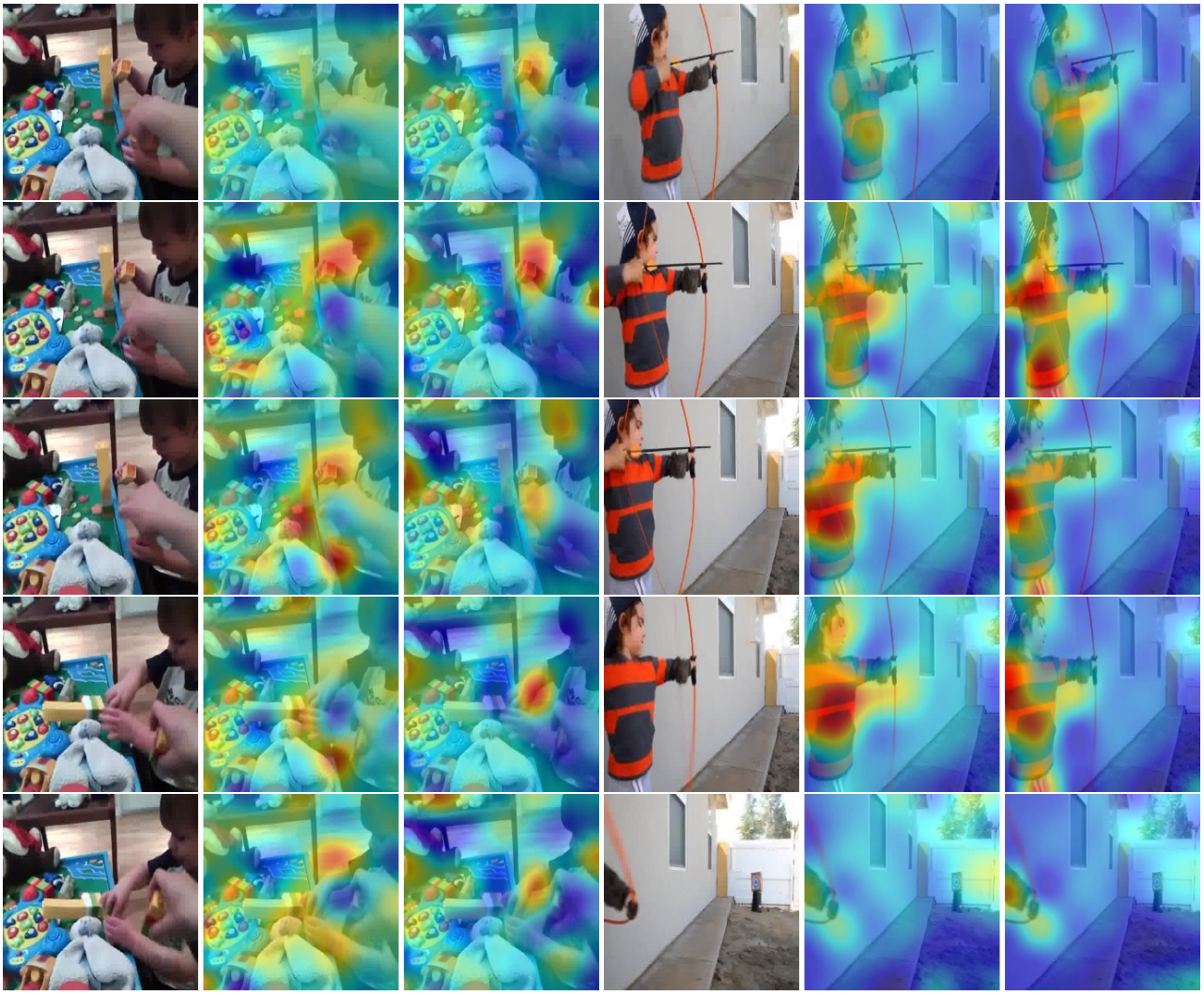


Figure S4: **Spatio-temporal saliency region visualizations for r3d-50 with and without SoftPool.** Class Feature Pyramids [14, 15] were used to generate the regional activations in the final *conv* layer of r3d-50.

S4. Time-inclusive salient regions

To better understand the use of SoftPool in 3D-CNNs, we study the spatio-temporal regions that the network finds more informative. Similar to the activation maximization visualizations for images, we use a fixed network structure as a backbone and only study the variations produced by replacing the original pooling operations with SoftPool. For the visualizations in Figure S4, we use the r3d-50 network from Table 5 in the paper. The examples are sampled from the Kinetics-700 dataset from the classes "building lego" and "archery".

Based on the examples presented in Figure S4, there are no significant differences in the salient regions. However, in multi-object scenes such as for the "building lego" class, the regional focus of the SoftPool network is shown to be a bit more distinct towards the region where there is a clear definition of the action performed (i.e. the hand with the lego brick). The case of "archery" exhibit small amounts of variations, with both either focusing on the main actor within the video.

S5. Embedding spaces visualizations

We provide t-SNE [8] visualizations of feature embeddings from an InceptionV3 model with original pooling operations and their counterparts with all pooling operations replaced by SoftPool. We use the averaged feature vectors of the final block in InceptionV3 (*Mixed7c*) with a reduced dimensionality of 50 channels produced by PCA [6] and

then perform t-SNE. We further perform k-means clustering [7] to better represent the different sub-clusters within the embedding space. In well-defined feature spaces, images in clusters that are closer together should be more similar.

The visualizations in Figures S5-S7 show distinct embeddings for the two networks. While structurally both model yield similar embeddings, for some classes the differences are more apparent. For example, class "jack-o-lantern" (Figure S6) and "sax" (Figure S7) show more compact representations when SoftPool is used.

S6. Additional runs on ImageNet1K

To test the improvements of SoftPool based on pooling layer replacements, supplementary to Table 2, we perform multiple runs over different seeds for four networks. For each of the training seeds, we train both an original network and a network with pooling layers replaced by SoftPool to ensure a fair comparison. As shown in Table S1, networks with SoftPool achieve higher top-1 classification accuracy rates. The improvements of ResNet18 range between 1.31-1.57%, 1.07-1.41% for ResNet34, 1.15-1.27% for ResNet50 and 1.46-1.75% for InceptionV1. This demonstrates that the choice in seeding plays minimal to no effect on the improvements when replacing the original network's pooling layers with SoftPool.

Model / Run	Original				SoftPool			
	1	2	3	(best)	1	2	3	(best)
ResNet18	69.61	69.73	69.69	69.76	71.18	71.04	71.25	**71.27**
ResNet34	73.26	73.11	73.24	73.30	74.66	74.52	74.31	**74.67**
ResNet50	76.01	75.97	76.04	76.15	77.26	77.24	77.19	**77.35**
InceptionV1	69.63	69.57	69.46	69.78	71.09	71.32	71.15	**71.43**

Table S1: **Top-1 Accuracy rates over runs** between the original networks and the same networks with SoftPool on ImageNet1K.

S7. Error rates and statistical significance

We further evaluate the statistical significance of the validation accuracy rates achieved in Table 2 in the context of the classification performance between the original models and models with pooling layers replaced by SoftPool. We perform a McNemar's test [3, 9], which is based on a null hypothesis (H_0) corresponding to accuracy homogeneity between the two models. It is calculated based on a contingency table holding the number of correct or incorrect class prediction instances with respect to each model. The tests studies the number of disagreements in the predictions between the two models. The Chi-Square statistic (χ^2) is then calculated based on the number of model i correct predictions against model j incorrect predictions (n_{ij}) and model j correct against model i incorrect predictions (n_{ji}). This is expressed as:

$$\chi^2 = \frac{(|n_{ij} - n_{ji}| - 1)^2}{n_{ij} + n_{ji}} \tag{S1}$$

We note that the null hypothesis (H_0) can be rejected with different significance levels based on the Chi-Square distribution table [10]. As the statistic is calculated with a single degree of freedom, χ^2 values of $\{3.84, 5.02, 6.63\}$ correspond to equivalent probabilities of $\{95\%, 97.5\%, 99.0\%\}$ that the two methods indeed differ.

Prediction distributions of each model pair are presented in Table S2 and the resulting χ^2 statistics and ρ homogeneity probabilities are presented in Table 4 in the main text. Based on the very low homogeneity probability $\ll 0.01\%$ [1, 5, 11], the differences between the original networks and the networks that have been re-trained with SoftPool cannot be attributed to statistical errors.

S8. Implementation Details

Range definitions. The exponential weighting of activations can correspond to the produced values being smaller than the type's (16, 32, 64-bit) precision level lower threshold. This can either result in a computational underflow or in a zero-valued dividend. For this reason, we include additional checks that each produced exponentially scaled activation $e^{\mathbf{a}_i}$ and their resulting weight mask $\mathbf{w}_i$, based on which their values (x) are transformed, to $x = max(0, x)$. The sum of the weights is constrained similarly, based on $x = max(x_{min}, x)$, where x_{min} is the lowest limit based on type chosen to ensure non-zero dividend. We note that these checks do need to address changes in the activation functions used. The current tested networks use ReLU activations which have lower bounds of zero. When considering other non-zero or negative-valued lower bound functions, the transformations need to be adjusted accordingly.

Computational description. As our implementation is native to CUDA-enabled devices, we are able to achieve inference times close to those of native methods such as average and maximum pooling. However, the parallelization capabilities of SoftPool allow for running times similar to those of average pooling with $O(1)$. This is based on the fact that operations can be performed through a matrix over the kernel region. This is beneficial for processes that have parallelization as a backbone (CUDA). In contrast, max pooling has complexity of at least $O(n)$, as the selection of the maximum value within the region can only be performed through the sequential consideration of each pixel within the region. In terms of the number of FLOPs, SoftPool is on par with other pooling methods such as average or max pooling because of tiling and CUDA's internal optimization during compiling. This can be traced to Valkov and Demmel's [17] comment that CUDA's C produces peak FLOP performance for Nvidia cards only

SoftPool \ Original	Correct	Incorrect	Total
Correct	31838	3795	35633
Incorrect	3011	11356	14367
Total	34849	15151	50000

(a) ResNet18

SoftPool \ Original	Correct	Incorrect	Total
Correct	34090	3246	37336
Incorrect	2940	9724	12664
Total	37030	12970	50000

(b) ResNet34

SoftPool \ Original	Correct	Incorrect	Total
Correct	35472	3203	38675
Incorrect	2603	8722	11325
Total	38075	11925	50000

(c) ResNet50

SoftPool \ Original	Correct	Incorrect	Total
Correct	36553	2609	39162
Incorrect	2272	8566	10838
Total	38825	11175	50000

(d) ResNet101

SoftPool \ Original	Correct	Incorrect	Total
Correct	37247	2373	39620
Incorrect	1908	8472	10380
Total	39155	10845	50000

(e) ResNet152

SoftPool \ Original	Correct	Incorrect	Total
Correct	34097	3836	37933
Incorrect	2253	9814	12067
Total	36350	13650	50000

(f) DenseNet121

SoftPool \ Original	Correct	Incorrect	Total
Correct	35160	4237	39397
Incorrect	3631	6972	10603
Total	38791	11209	50000

(g) DenseNet161

SoftPool \ Original	Correct	Incorrect	Total
Correct	35074	3407	38481
Incorrect	2988	8531	11519
Total	38062	11938	50000

(h) DenseNet169

SoftPool \ Original	Correct	Incorrect	Total
Correct	36628	2618	39246
Incorrect	2153	8601	10754
Total	38781	11219	50000

(i) ResNeXt50

SoftPool \ Original	Correct	Incorrect	Total
Correct	38633	2587	41220
Incorrect	1005	7775	8780
Total	39638	10362	50000

(j) ResNeXt101

SoftPool \ Original	Correct	Incorrect	Total
Correct	36638	3113	39751
Incorrect	2636	7613	10249
Total	38781	10726	50000

(k) wide-ResNet50

SoftPool \ Original	Correct	Incorrect	Total
Correct	31826	3897	35723
Incorrect	3037	11240	14277
Total	34863	15137	50000

(l) InceptionV1

SoftPool \ Original	Correct	Incorrect	Total
Correct	36341	3029	14874
Incorrect	2370	8260	10630
Total	38711	11289	50000

(m) InceptionV3

Table S2: Error summaries of original models and models with pooling replaced with SoftPool.

assuming multiply-and-accumulate (MACCs) operations, with only MACCs contributing to CUDA's FLOP count.

Testing environment. All of our tests were done with half floating point precision (float16) instead of single-point (float32) for better memory utilization during the model training phase. The batch sizes are split equally with 64 images per GPU. Our testing environment consists of an AMD Threadripper 2950X with 2400MHz RAM frequency and four Nvidia 2080 TIs.

References

[1] Robert M Craparo. Significance level. *Encyclopedia of measurement and statistics*, 3:889–891, 2007. 5

[2] Alexey Dosovitskiy and Thomas Brox. Inverting visual representations with convolutional networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4829–4837. IEEE, 2016. 2

[3] Allen L Edwards. Note on the "correction for continuity" in testing the significance of the difference between correlated proportions. *Psychometrika*, 13(3):185–187, 1948. 5

[4] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009. 2

[5] Ronald Aylmer Fisher. Statistical methods for research workers. In *Breakthroughs in statistics*, pages 66–70. Springer, 1992. 5

[6] IT Jolliffe. Principal component analysis. *Technometrics*, 45(3):276, 2003. 4

[7] Stuart Lloyd. Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2):129–137, 1982. 5

[8] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008. 4

[9] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947. 5

[10] Yoni Nazarathy. Chi-square distribution table. https://people.smp.uq.edu.au/YoniNazarathy/stat_models_B_course_spring_07/distributions/chisqtab.pdf. 5

[11] Jerzy Neyman. Outline of a theory of statistical estimation based on the classical theory of probability. *Philosophical Transactions of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 236(767):333–380, 1937. 5

[12] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013. 2

[13] Alexandros Stergiou. The mind's eye: Visualizing class-agnostic features of cnns. *arXiv preprint arXiv:2101.12447*, 2021. 2

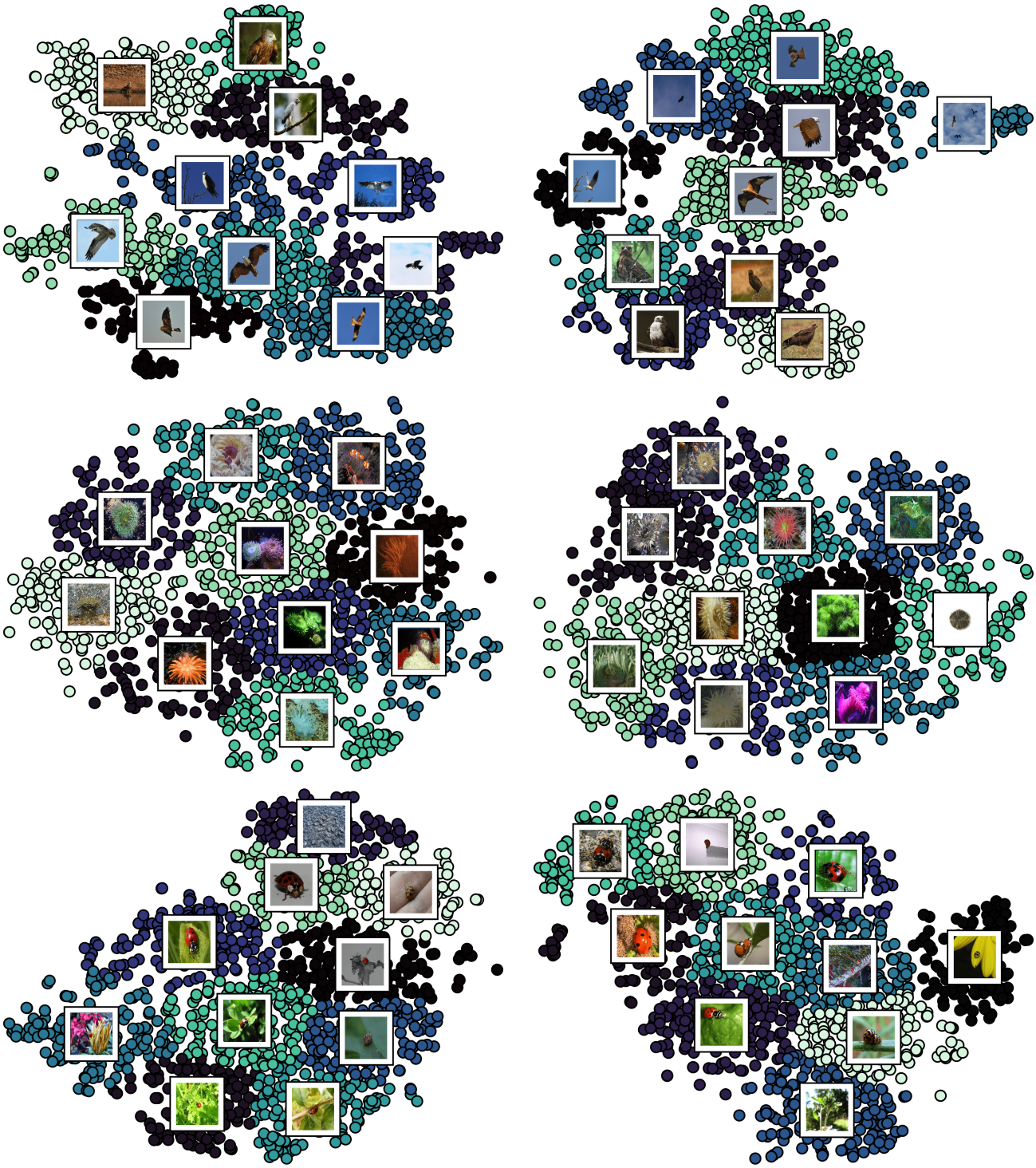
[14] Alexandros Stergiou, Georgios Kapidis, Grigorios Kalliatakis, Christos Chrysoulas, Ronald Poppe, and Remco Veltkamp. Class feature pyramids for video explanation. In *International Conference on Computer Vision Workshop (ICCVW)*, pages 4255–4264. IEEE, 2019. 4

[15] Alexandros Stergiou, Georgios Kapidis, Grigorios Kalliatakis, Christos Chrysoulas, Remco Veltkamp, and Ronald Poppe. Saliency tubes: Visual explanations for spatio-temporal convolutions. In *International Conference on Image Processing (ICIP)*, pages 1830–1834. IEEE, 2019. 4

[16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826. IEEE, 2016. 2

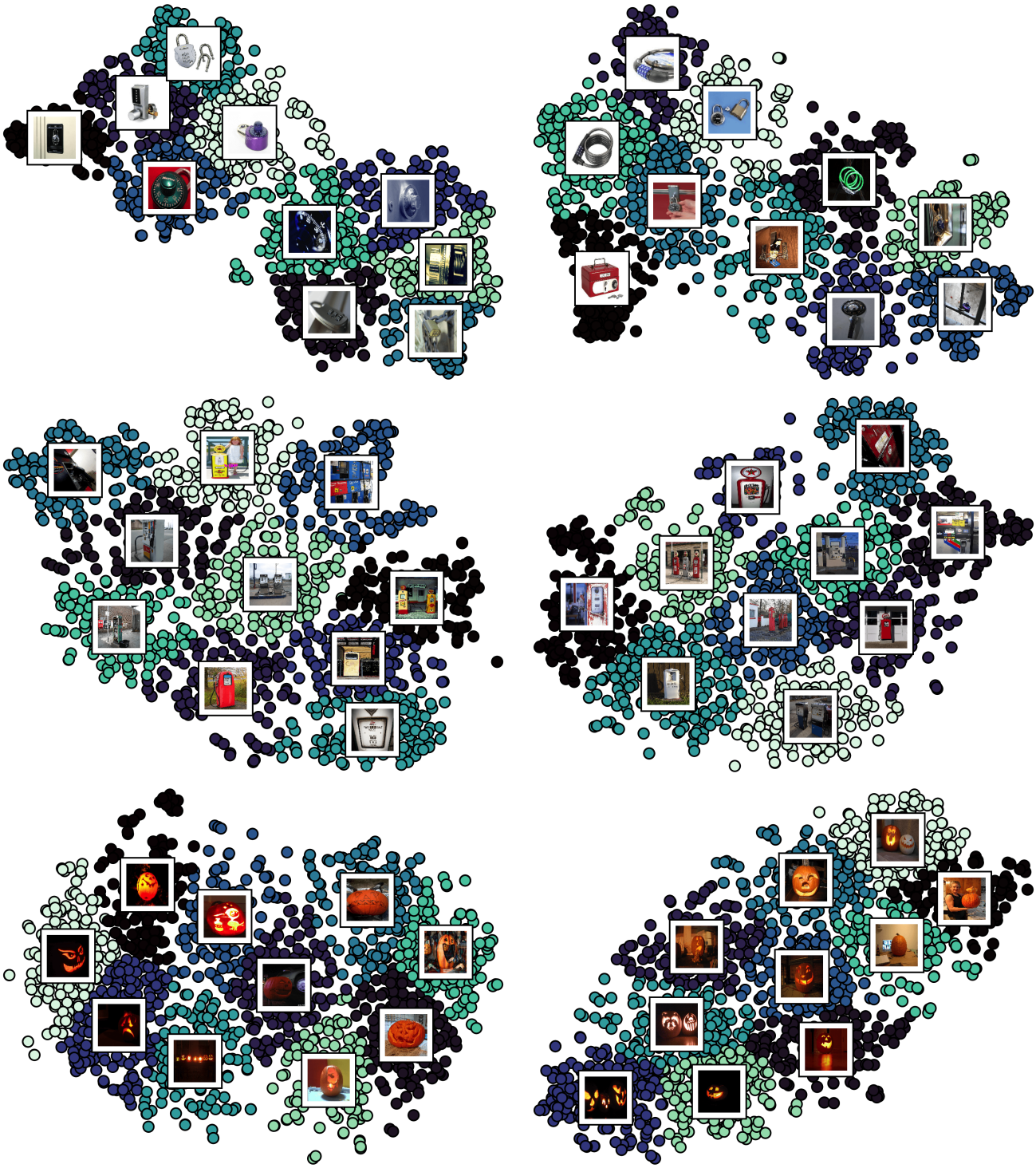
[17] Vasily Volkov and James W Demmel. Benchmarking gpus to tune dense linear algebra. In *Conference on Supercomputing*. IEEE, 2008. 5

[18] Donglai Wei, Bolei Zhou, Antonio Torrabla, and William Freeman. Understanding intra-class knowledge inside CNN. *arXiv preprint arXiv:1507.02379*, 2015. 2



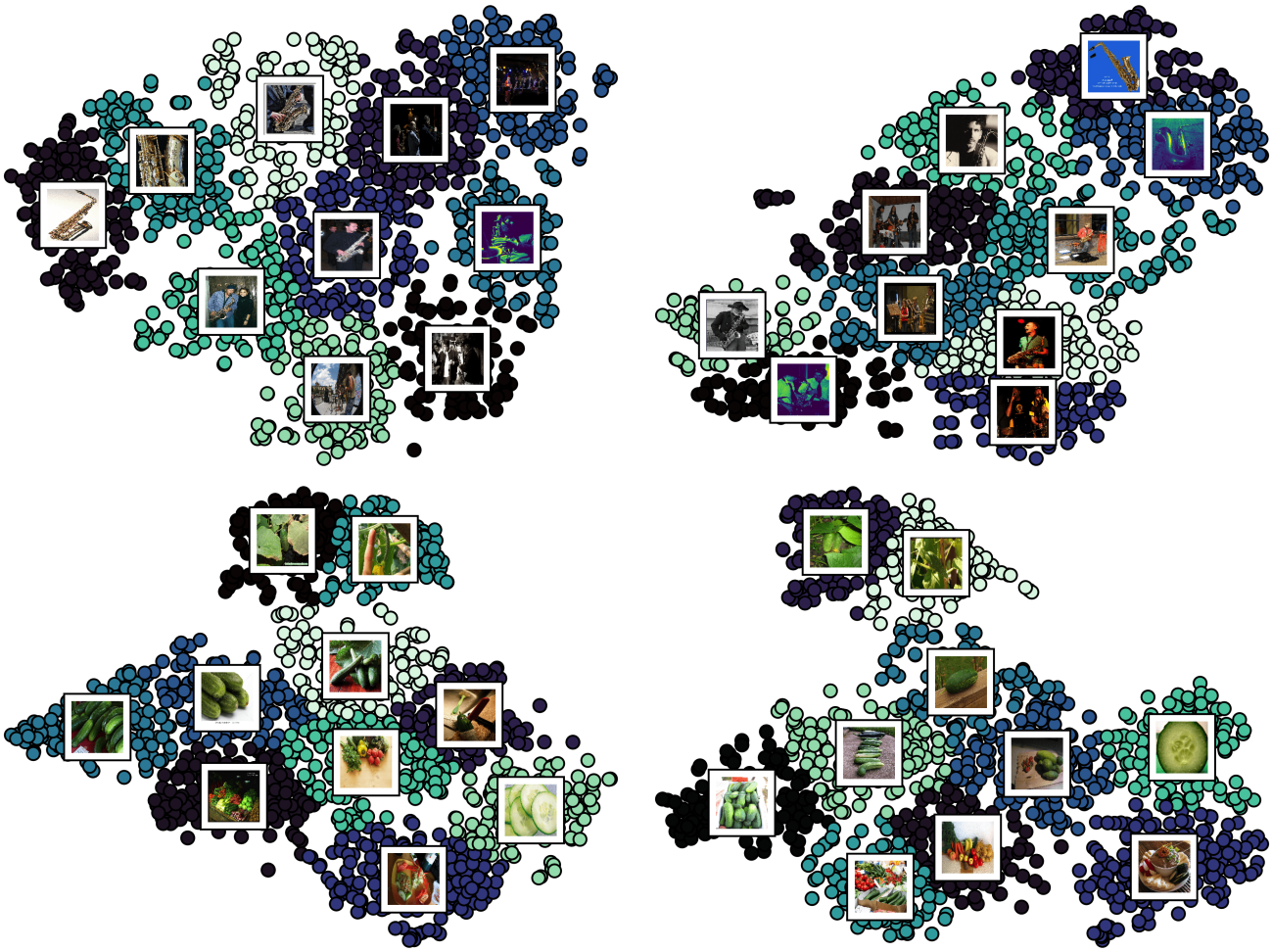
a. Original b. SoftPool

Figure S5: **t-SNE feature embeddings for InceptionV3 with and without SoftPool**. ImageNet1K classes "bald eagle", "sea anemone" and "ladybug". Cluster centers are found with k-means to better visualize the feature space. Images displayed are the closest examples for each cluster center.



a. Original b. SoftPool

Figure S6: **t-SNE feature embeddings for InceptionV3 with and without SoftPool**. ImageNet1K classes "combination lock", "gas pump" and "jack-o-lantern". Cluster centers are found with k-means to better visualize the feature space. Images displayed are the closest examples for each cluster center.



a. Original b. SoftPool

Figure S7: **t-SNE feature embeddings for InceptionV3 with and without SoftPool**. ImageNet1K classes "sax" and "zucchini". Cluster centers are found with k-means to better visualize the feature space. Images displayed are the closest examples for each cluster center.