T_KCP: Context-Aware Pooling via Top-K% Activation Selection

Seo-Yeon Choi
Department of Computer Science & AI
Jeonbuk National University, South Korea
yunseul@jbnu.ac.kr

Kyungsu Lee*
Department of Computer Science & AI
Jeonbuk National University, South Korea
ksl@jbnu.ac.kr

Abstract

Pooling is a core operation in convolutional neural networks (CNNs), enabling spatial reduction and hierarchical abstraction. However, standard methods such as max or average pooling operate locally and often fail to capture global context, leading to under- or over-estimation of features. This limits performance on tasks requiring both fine localization and holistic understanding. To address this, we propose Top-k% Contextual Pooling (T_KCP), a framework that preserves informative activations based on contextual importance. T_KCP includes two variants: (1) Sparse Contextual Pooling, selecting top-k% activations within local windows, and (2) Global Contextual Pooling, selecting top-k% across the entire feature map. Given a kernel size and target resolution, T_KCP deterministically sets the stride and reconstructs outputs without additional parameters. Experiments across classification, detection, tracking, segmentation, and generation show consistent improvements in accuracy and robustness. Additionally, T_KCP enhances interpretability by tracing high-activation regions across layers.

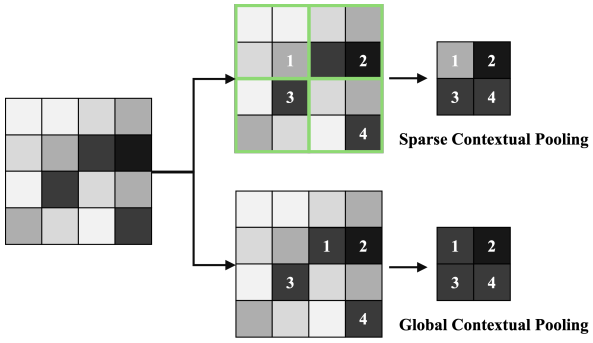

Figure 1: Overview of Top-k% Contextual Pooling (T_KCP). The input feature map is processed by two variants: (1) Sparse Contextual Pooling selects top-k% activations within local windows, preserving spatially discriminative features; (2) Global Contextual Pooling selects top-k% across the entire feature map for global abstraction. Both reconstruct outputs to a target resolution in a deterministic, parameter-free manner.

1 INTRODUCTION

Deep neural networks have demonstrated strong performance across a wide range of tasks, with increasing depth contributing significantly to the representational significance (He et al., 2016; Huang et al., 2017). The progression, however, demands architectural components that effectively balance feature resolution and computational efficiency. Pooling layers fulfill this role in convolutional neural networks (CNNs) by reducing spatial dimensions while preserving key structural relationships (Boureau et al., 2010). Such dimensionality reduction facilitates deeper architectures by standardizing feature map sizes across layers and improving robustness to spatial variations (LeCun et al., 1998). Nonetheless, conventional pooling operations, such as max- and average-pooling, often incur information loss, distorting fine-grained features essential for precise localization (Saeedan et al., 2018; Özdemir, 2023).

Recently, to address these limitations, numerous pooling variants have been proposed. Stochastic pooling introduces randomness to promote regularization (Zeiler and Fergus, 2013), while mixed and adaptive pooling strategies combine or learn pooling behavior based on input features (Yu et al., 2014). Other methods, such as Lp pooling, offer tunable aggregation through parameterized norms (Gulcehre et al., 2014). While these approaches improve flexibility or robustness in certain

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. Copyright 2026 by the author(s).

settings, they often rely on sensitive hyperparameters, introduce additional complexity, or fail to integrate global contextual awareness (Zafar et al., 2022). Moreover, many pooling operations still operate under the assumption that local statistics are sufficient to summarize features, which is suboptimal for dense prediction tasks such as detection and segmentation (Gholamalinezhad and Khosravi, 2020).

To alleviate this, in this paper, we propose Top-k% Contextual Pooling ($\mathrm{T_K}$CP), a novel pooling framework that selectively retains the most informative activations based on their contextual relevance. $\mathrm{T_K}$CP is composed of two complementary strategies. Sparse Contextual Pooling (SCP) applies top-k% selection within each local kernel window, enabling fine-grained attention to salient features while reducing redundancy. Global Contextual Pooling (GCP), on the other hand, aggregates the top-k% activations across the entire feature map, providing a concise yet expressive summary of the global structure. Given a kernel size and target output size, $\mathrm{T_K}$CP automatically infers stride values and reconstructs the output through a deterministic, parameter-free process that preserves spatial coherence.

$\mathrm{T_K}$CP is lightweight, easily integrated into existing architectures, and applicable to a wide range of tasks. Through experiments on image classification, object detection, object tracking, semantic segmentation, and diffusion-based generation, we show that $\mathrm{T_K}$CP consistently improves model performance. Furthermore, by filtering high-activation regions progressively across layers, it enables hierarchical analysis of the network's focus areas, offering new opportunities for interpretability and visualization in CNNs. Our main contributions are as follows:

- We propose Top-k% Contextual Pooling ($\mathrm{T_K}$CP), a unified pooling framework composed of sparse and global contextual variants for general tasks.
- We introduce a deterministic, parameter-free reconstruction strategy that preserves spatial alignment and semantic salience without incurring computational overhead.
- We demonstrate that $\mathrm{T_K}$CP improves performance across diverse tasks while supporting layer-wise interpretability via progressive activation.

2 PRELIMINARY STUDY

Problem Definition Let $X \in \mathbb{R}^{H\times W\times C}$ be an input feature map, where H, W, and C denote the spatial height, width, and number of channels, respectively. A pooling operator is typically defined as a local mapping:

$$\mathcal{P} : \mathbb{R}^{K_h\times K_w} \to \mathbb{R}, \tag{1}$$

where K_h and K_w represent the height and width of the pooling kernel. This operator is applied repeatedly over spatially partitioned windows of X, such as non-overlapping or strided regions, resulting in a downsampled representation. This induces a higher-level mapping on the entire feature:

$$\mathcal{P}_{\text{map}} : \mathbb{R}^{H\times W\times C} \to \mathbb{R}^{H'\times W'\times C}, \tag{2}$$

where each spatial position in the output is computed from a corresponding local region in the input. Standard pooling functions, such as max pooling and average pooling, apply fixed, context-agnostic rules to aggregate activations within a local region. Given a local patch $R \subset \mathbb{R}^{K_h\times K_w}$, they are defined as:

$$\mathcal{P}_{\max}(R) := \max_{x\in R} x, \quad \mathcal{P}_{\text{avg}}(R) := \frac{1}{|R|}\sum_{x\in R} x. \tag{3}$$

While computationally efficient, conventional operations comprises all activations uniformly, ignoring their contextual importance or spatial arrangement. For instance, consider two local regions with the same activation values in different orders:

$$R_1 = \{0.1, 0.9, 0.1, 0.1\}, \quad R_2 = \{0.1, 0.1, 0.1, 0.9\}. \tag{4}$$

Assume that the value 0.9 corresponds to a semantically significant feature (e.g., an edge or localized object response). Despite the spatial differences between R_1 and R_2, both max and average pooling yield identical outputs: $\mathcal{P}_{\max}(R_1) = \mathcal{P}_{\max}(R_2) = 0.9$ and $\mathcal{P}_{\text{avg}}(R_1) = \mathcal{P}_{\text{avg}}(R_2) = 0.3$. The pooling outputs are identical, despite different spatial configurations. This shows that both max and average pooling are permutation-invariant over R and discard spatial structure entirely:

$$\mathcal{P}(\pi(R)) = \mathcal{P}(R), \quad \forall \pi \in \mathrm{Sym}(N), \tag{5}$$

where $\mathrm{Sym}(N)$ is the symmetric group of all permutations on N elements. Moreover, in average pooling, the effect of high-activation values can be diluted:

$$\text{If } x_1 = 0.9,\ x_{2\ldots N} = 0.1 \Rightarrow \mathcal{P}_{\text{avg}}(R) = \frac{0.9 + 0.1\cdot(N-1)}{N}. \tag{6}$$

As $N \to \infty$, $\mathcal{P}_{\text{avg}}(R) \to 0.1$, effectively diluting the contribution of salient values. This highlights the inability of conventional pooling to adapt to semantic or spatial variation, often discarding fine-grained or context-dependent features.

To address this, we consider a percentile-based pooling operator that retains only the most informative activations within a defined spatial region. The goal is to compute a compact representation by selecting the top-k% values based on their activation magnitude.

Formally, let $R \subset \mathbb{R}^n$ denote a flattened spatial region within a channel c, extracted from either a local window or feature map. For each region $R^{(c)} = \{x_1, \ldots, x_n\}$, define top-$k$% subset as:

$$\mathcal{T}_k(R^{(c)}) = \left\{ x \in R^{(c)} \mid x \geq t_k^{(c)} \right\}, \tag{7}$$

where $t_k^{(c)}$ is the $(1-k)$-quantile of values in $R^{(c)}$. The pooled output for that region and channel is computed by averaging the top-k% activations:

$$\mathcal{P}_k(R^{(c)}) = \frac{1}{|\mathcal{T}_k(R^{(c)})|} \sum_{x \in \mathcal{T}_k(R^{(c)})} x. \tag{8}$$

This operator can be applied over either local or global regions (see Method), depending on the pooling scope. In both cases, the $\mathrm{T_K}$CP yields a compact representation by selecting tops while preserving saliency.

Sampling Top-k% We consider the problem of selecting the top-k% values from a set of activations based on their statistical prominence. Let $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$ be a set of activation values, assumed to follow a normal distribution with mean μ_X and standard deviation σ_X. The goal is to retain only those elements in the k-th percentile. This is equivalent to selecting elements satisfying $P(x^{(i)} \geq \mu_X + z_k \sigma_X) = k$, where z_k denotes the z-score corresponding to the top-k% percentile in the standard normal distribution. In other words, $x^{(i)}$ lies in the top-k% of X *iff*:

$$x^{(i)} \geq \mu_X + z_k \sigma_X. \tag{9}$$

Note that the Gaussian formulation above does not imply that activations strictly follow a normal distribution. Rather, the Gaussian-based threshold is used as a computationally efficient surrogate for percentile estimation, avoiding explicit sorting. As shown in Appendix C, this surrogate closely matches empirical top-k selection even under non-Gaussian activation distributions. Following this definition, the threshold condition (Eq. 9) corresponds to computing the area under the tail of a normal distribution, as illustrated in Fig. 2. Here, the hard selection rule for top-k% sampling can be formulated using an indicator function:

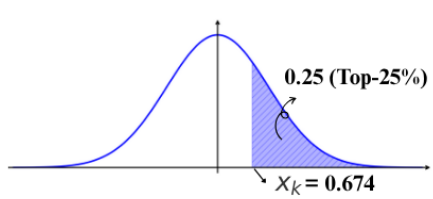

Figure 2: Top-k% selection using a standard normal distribution. For $k = 25$%, the threshold is $z_k \approx 0.674$, and values above $\mu + z_k \sigma$ are retained.

$$\mathcal{E}_{\text{hard}}(x^{(i)}) = \begin{cases} x^{(i)} & \text{if } x^{(i)} \geq \mu_X + z_k \sigma_X, \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

This thresholding behavior can be compactly written using a binary step function $H(t)$:

$$\mathcal{E}_{\text{hard}}(x^{(i)}) = x^{(i)} \cdot H\left(x^{(i)} - \mu_X - z_k \sigma_X\right), \tag{11}$$

where $H(t) = 1$ if $t \geq 0$, and 0 otherwise. Since $H(\cdot)$ is non-differentiable, direct use in gradient-based optimization is problematic. To overcome this, we introduce a smooth surrogate using the sigmoid function, leading to the following soft selection operator:

$$\mathcal{E}(x^{(i)}) = x^{(i)} \cdot \sigma\left(\alpha \cdot (x^{(i)} - \mu_X - z_k \sigma_X)\right), \tag{12}$$

where $\sigma(t) = \frac{1}{1+e^{-t}}$ is the standard sigmoid function, and $\alpha > 0$ is a sharpness parameter controlling the steepness of the activation boundary. As $\alpha \to \infty$, the soft thresholding converges to a hard cutoff as Eq. 10. This expression enables differentiable top-k% sampling using only basic operations and avoids explicit sorting or quantile estimation. Moreover, the z-score threshold z_k can be managed as a hyperparameter or optimized during training, making the sampling behavior adaptive to data distribution and task objectives.

3 METHOD

We propose Top-k% Contextual Pooling ($\mathrm{T_K}$CP), a percentile-based pooling mechanism that selectively aggregates the most salient activations in a feature map based on their relative magnitude. Unlike conventional pooling methods such as max or average pooling, which apply fixed aggregation rules across all spatial regions, $\mathrm{T_K}$CP dynamically adapts to the statistical structure of activations and selectively retains only the top-k% percentile responses. This enables compact yet semantically rich representations.

$\mathrm{T_K}$CP is applicable in both *local* and *global* contexts. In the local case, the input feature map is divided into spatial windows (e.g., via strided kernels), and top-k% selection is applied independently to each window. In the global setting, the selection is applied across the entire spatial extent of each channel, producing a compact global descriptor. Formally, the method computes a statistical threshold per region using the local (or global) mean and standard deviation, and retains activations that exceed this threshold. The resulting

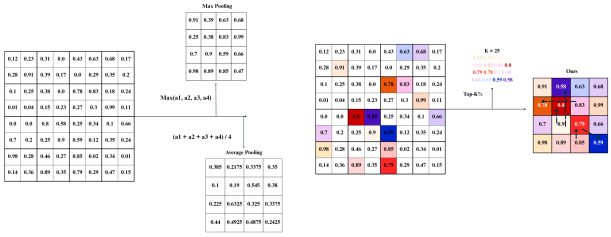

Figure 3: Comparison of pooling methods. Left: Input feature map with max and average pooled outputs (kernel size = 2). Right: Our Top-k% Contextual Pooling (T_KCP) selects top activations within each region using a learned threshold ($k = 25\%$), retaining salient activations (colored). Unlike max or average pooling, T_KCP balances selectivity.

pooled value is an average over the selected subset. In short, T_KCP is lightweight, parameter-free, and can be seamlessly integrated into modern architectures for discriminative and generative tasks.

To highlight the difference between conventional pooling and our method, Fig. 3 provides a qualitative comparison. Unlike max or average pooling, which aggregate all elements using fixed rules, T_KCP adaptively selects top-k% activations based on statistical thresholds, preserving contextually relevant responses. This yields sharper, more discriminative representations, particularly in regions with mixed activations.

3.1 Formulation

We distinguish three terms: *contextual* refers to statistics (μ, σ), *saliency* to the sigmoid-based gating map, and *importance* to each activation's contribution to the pooled output. Let $X \in \mathbb{R}^{H\times W\times C}$ be an input feature map, where H, W, and C denote the spatial height, width, and number of channels, respectively. The goal of T_KCP is to downsample or summarize X by retaining the top-k% activations within a region of interest, selected based on their statistical significance. Firstly, to ensure non-negative and numerically stable inputs, we leverage the previously well-known knowledge that most CNN architectures typically follow a "Conv-Batch-ReLU" pipeline, where feature maps after ReLU are already non-negative. In this case, we simply apply a small bias shift $\epsilon > 0$ to avoid zero gradients:

$$X' = \mathrm{ReLU}(X) + \epsilon. \tag{13}$$

However, when the pooling operator is applied to intermediate feature maps that may include negative values (e.g., pre-activation outputs), we normalize the input by shifting it into the non-negative domain:

$$X' = X - \min(X) + \epsilon, \tag{14}$$

where $\min(X) = \min_{i,j,c} X_{i,j,c}$, ensuring all values in X' are strictly positive. This adjustment guarantees numerical stability for subsequent percentile-based selection while preserving relative activation magnitudes.

Local Contextual Pooling. We aim to construct a downsampled representation $Y \in \mathbb{R}^{H'\times W'\times C}$ by performing pooling over spatial regions. To this end, we partition X into strided local windows of size $K_h \times K_w$ using stride values S_h and S_w. Let (i, j) index the output spatial coordinates. Then, region is defined as:

$$\begin{aligned} R_{ij} = X\Big[\, & i\cdot S_h : i\cdot S_h + K_h, \\ & j\cdot S_w : j\cdot S_w + K_w,\ : \,\Big] \in \mathbb{R}^{K_h\times K_w\times C}. \end{aligned} \tag{15}$$

For each channel c, we flatten the spatial dimensions into $R_{ij}^{(c)} \in \mathbb{R}^{K_h\cdot K_w}$. Within each flattened region $R_{ij}^{(c)}$, we compute the local mean $\mu_{ij}^{(c)}$ and standard deviation $\sigma_{ij}^{(c)}$, which are used to derive a threshold for top-k% activation selection:

$$\tau_{ij}^{(c)} = \mu_{ij}^{(c)} + z_k \cdot \sigma_{ij}^{(c)}. \tag{16}$$

$$\mu_{ij}^{(c)} = \frac{1}{|R_{ij}^{(c)}|}\sum_{x\in R_{ij}^{(c)}} x, \quad \sigma_{ij}^{(c)} = \sqrt{\frac{1}{|R_{ij}^{(c)}|}\sum_{x\in R_{ij}^{(c)}} (x-\mu_{ij}^{(c)})^2}. \tag{17}$$

$$\mu^{(c)} = \frac{1}{HW}\sum_{x\in X^{(c)}} x, \quad \sigma^{(c)} = \sqrt{\frac{1}{HW}\sum_{x\in X^{(c)}} (x-\mu^{(c)})^2}. \tag{18}$$

We then perform top-k% selection with a soft threshold:

$$\mathcal{E}(x) = x\cdot\sigma\left(\alpha(x-\mu_{ij}^{(c)} - z_k\sigma_{ij}^{(c)})\right), \tag{19}$$

Algorithm 1: Local Contextual Pooling

Input: Input feature map $X \in \mathbb{R}^{H\times W\times C}$, kernel size (K_h, K_w), percentile k, sharpness α
Output: Pooled feature map $Y \in \mathbb{R}^{H'\times W'\times C}$
for $c \leftarrow 1$ ***to*** C **do**
  for *each spatial window* $R_{ij}^{(c)}$ *in* $X^{(c)}$ **do**
    Compute mean $\mu_{ij}^{(c)}$ and std $\sigma_{ij}^{(c)}$ of $R_{ij}^{(c)}$;
    Compute threshold $\tau_{ij}^{(c)} = \mu_{ij}^{(c)} + z_k\sigma_{ij}^{(c)}$;
    for *each* $x \in R_{ij}^{(c)}$ **do**
      Compute weight $w(x) = \sigma(\alpha(x - \tau_{ij}^{(c)}))$;
    $Y_{ij}^{(c)} = \frac{\sum_x x\cdot w(x)}{\sum_x w(x)}$;
return Y

where $\sigma(\cdot)$ denotes the sigmoid function, and $\alpha > 0$ controls the gating sharpness. The final pooled output for region (i, j) and channel c is computed as a normalized average over all softly retained values:

$$Y_{ij}^{(c)} = \frac{\sum\limits_{x \in R_{ij}^{(c)}} x \cdot \sigma\left(\alpha(x - \mu_{ij}^{(c)} - z_k \sigma_{ij}^{(c)})\right)}{\sum\limits_{x \in R_{ij}^{(c)}} \sigma\left(\alpha(x - \mu_{ij}^{(c)} - z_k \sigma_{ij}^{(c)})\right)}. \tag{20}$$

Note that this soft selection allows small activations to contribute marginally (rather than being fully discarded), thereby preserving gradient signals during backpropagation. Empirically, this gradient preservation significantly enhances training stability and discriminative power over hard thresholding approaches such as the step function:

$$\mathcal{E}_{\text{hard}}(x) = x \cdot H(x - \mu - z_k \sigma), \tag{21}$$

where $H(t)$ denotes the Heaviside step function. To express this in a differentiable manner, we adopt a sigmoid-based gating function:

$$\mathcal{E}_{\text{soft}}(x) = x \cdot \sigma\left(\alpha(x - \mu - z_k \sigma)\right), \tag{22}$$

where $\sigma(t) = \frac{1}{1+e^{-t}}$ and $\alpha > 0$ controls the gating sharpness. As $\alpha \to \infty$, the sigmoid approximates a binary gate:

$$\lim_{\alpha \to \infty} \sigma\left(\alpha(x - \mu - z_k \sigma)\right) = H(x - \mu - z_k \sigma), \tag{23}$$

thus recovering the hard thresholding operator:

$$\lim_{\alpha \to \infty} \mathcal{E}_{\text{soft}}(x) = \mathcal{E}_{\text{hard}}(x). \tag{24}$$

In the context of pooling, the normalized soft selection is given in Eq. 20, such that both average and max pooling depend on α:

- As $\alpha \to 0$, $\sigma(\cdot) \approx \frac{1}{2}$, so each element contributes equally. Hence,

$$\lim_{\alpha \to 0} Y = \frac{1}{|R|} \sum_{x \in R} x = \text{AvgPool}(R). \tag{25}$$

- As $\alpha \to \infty$, the gating function approaches an indicator over the top values:

$$\lim_{\alpha \to \infty} Y \approx \max_{x \in R} x, \tag{26}$$

  when z_k is sufficiently high.

Thus, our formulation provides a continuous, differentiable bridge between average and max pooling, with tunable control over sparsity and selectivity.

Global Contextual Pooling. Global Contextual Pooling aims to summarize spatially dispersed but semantically important activations across the entire feature map. Unlike local pooling, GCP produces a single scalar per channel, analogous to global average pooling. Although described as top-k% selection, GCP does not explicitly enforce selecting an exact fraction of activations; instead, all activations contribute through a continuous sigmoid-based weighting. Rather than operating on local windows, GCP aggregates activations across the entire spatial extent of each channel according to their global statistical distribution. To this end, we compute the global mean and standard deviation for each channel and apply a soft selection operator:

$$G^{(c)} = \frac{\sum\limits_{x \in X^{(c)}} x \cdot \sigma\left(\alpha(x - \mu^{(c)} - z_k \sigma^{(c)})\right)}{\sum\limits_{x \in X^{(c)}} \sigma\left(\alpha(x - \mu^{(c)} - z_k \sigma^{(c)})\right)}, \tag{27}$$

where $\sigma(t) = \frac{1}{1+e^{-t}}$ is the sigmoid function and $\alpha > 0$ controls the sharpness of the gating function. This formulation performs a normalized weighted aggregation over all spatial positions, where activations with higher responses receive larger weights while lower ones are softly suppressed, ensuring stable gradient flow during training.

For interpretability purposes, we additionally define a spatial reconstruction of the gated activations. Let $\hat{X}^{(c)} \in \mathbb{R}^{H \times W}$ denote a reconstructed map for channel c. To map the 1D ordering of activations back to a 2D spatial layout, we employ a deterministic Space-Filling Curve (SFC), such as the Hilbert or Z-order curve, defined as:

$$\text{SFC} : \{1, \ldots, H \cdot W\} \to \{(i, j) \mid 1 \le i \le H, 1 \le j \le W\}. \tag{28}$$

Note that Eq. 28 is used only for visualization and interpretability, and is not part of the forward compu-

Algorithm 2: Global Contextual Pooling

Input: Input feature map $X \in \mathbb{R}^{H \times W \times C}$, percentile k, sharpness α
Output: Global pooled vector $G \in \mathbb{R}^C$
for $c \leftarrow 1$ ***to*** C **do**
  Flatten $X^{(c)}$ to $X^{(c)} \in \mathbb{R}^{H \cdot W}$;
  Compute mean $\mu^{(c)}$ and std $\sigma^{(c)}$ of $X^{(c)}$;
  Compute threshold $\tau^{(c)} = \mu^{(c)} + z_k \sigma^{(c)}$;
  for *each* $x \in X^{(c)}$ **do**
    Compute weight $w(x) = \sigma(\alpha(x - \tau^{(c)}))$;
  $G^{(c)} = \frac{\sum_x x \cdot w(x)}{\sum_x w(x)}$;
return G

tation. Each activation with a high gating response is assigned to its corresponding spatial location according to the SFC traversal order:

$$\hat{X}^{(c)}_{i,j} = \begin{cases} x & \text{if } \mathrm{SFC}^{-1}(i,j) \text{ corresponds to a selected } x \in X^{(c)}, \\ 0 & \text{otherwise.} \end{cases} \tag{29}$$

Note that this spatial reconstruction is used only for visualization and interpretability, and is not part of the forward computation. This remapping provides an intuitive spatial view of the activations emphasized by GCP while preserving local continuity.

Gradient Optimization. The proposed $\mathrm{T_K}$CP is fully differentiable and supports end-to-end training. For each activation x, the gradient of the soft selection $\mathcal{E}(x)$ is:

$$\frac{\partial \mathcal{E}(x)}{\partial x} = \sigma(\Delta) + \alpha x \cdot \sigma(\Delta) \cdot (1 - \sigma(\Delta)), \quad \text{where} \quad \Delta = \alpha(x - \mu - z_k \sigma). \tag{30}$$

This formulation preserves gradient flow by softly attenuating low activations, unlike step-based functions that cause vanishing gradients. The threshold parameter z_k may be fixed (e.g., $z_k \approx 0.674$ for top-25%) or learned to adapt sparsity per layer. In the latter case, its gradient is:

$$\frac{\partial \mathcal{E}(x)}{\partial z_k} = -\alpha x \cdot \sigma'(\Delta) \cdot \sigma. \tag{31}$$

This enables $\mathrm{T_K}$CP to learn optimal activation thresholds from data, promoting adaptive abstraction behavior.

3.2 Interpretability via Adaptive Saliency

Our pooling mechanism inherently facilitates interpretability by explicitly selecting top-k% activations based on their statistical prominence, without relying on gradient-based relevance attribution. This selection criterion, $x^{(i)} \geq \mu + z_k \sigma$, acts as a hard or soft mask that filters salient spatial responses at each layer. Unlike CAM or Grad-CAM, which compute importance weights via global average pooling over gradients, $\alpha_c^{\mathrm{CAM}} = \frac{1}{Z} \sum_{i,j} \frac{\partial y}{\partial A_{ij}^{(c)}}$, $\mathrm{T_K}$CP directly retains spatial activations that exhibit high magnitude under adaptive thresholds. Let $X^{(l)}$ be the feature map at layer l, and $M^{(l)} \in \{0,1\}^{H \times W}$ be a binary mask:

$$M_{ij}^{(l)} = \mathbb{I}\left[x_{ij}^{(l)} \geq \mu^{(l)} + z_k \sigma^{(l)}\right]. \tag{32}$$

Then, the pooled output becomes $X^{(l+1)} = X^{(l)} \odot M^{(l)}$, where $\odot$ denotes element-wise masking and aggregation (possibly averaged). This mask $M^{(l)}$ functions analogously to Grad-CAM heatmaps, but is computed directly from activation statistics rather than back-propagated gradients. Furthermore, since the masks are computed hierarchically at each pooling stage, we obtain a saliency trace:

$$M^{(1)} \to M^{(2)} \to \cdots \to M^{(L)}, \tag{33}$$

which identifies important regions as they persist through layers. This enables transparent visual explanations of model behavior, highlighting which spatial regions are consistently preserved due to their contextual significance. In contrast to gradient-based methods, $\mathrm{T_K}$CP yields interpretable importance without requiring class labels or backward passes, and can be visualized layer-wise or aggregated to produce activation saliency maps.

3.3 Integration and Applications

$\mathrm{T_K}$CP is a parameter-free, drop-in replacement for standard pooling, typically applied after Conv–BN–ReLU blocks in CNNs. If activations can be negative, we shift each channel by its minimum, $X'^{(c)} = X^{(c)} - \min(X^{(c)})$, to ensure non-negativity. For a feature map $X \in \mathbb{R}^{H \times W \times C}$, $\mathrm{T_K}$CP retains activations where $x^{(i)} \geq \mu + z_k \sigma$, and outputs a soft selection $\mathcal{E}(x) = x \cdot \sigma(\alpha(x - \mu - z_k \sigma))$, where μ, σ are the mean and standard deviation, $\sigma(\cdot)$ is the sigmoid function, and α controls gating sharpness. This mechanism flexibly emphasizes the most discriminative regions, supporting vision tasks across various architectures.

3.4 Complexity Analysis

Furthermore, we analyze the computational complexity of Top-k% Contextual Pooling ($\mathrm{T_K}$CP) relative to conventional pooling operators. Let the input feature map be of size $H \times W \times C$ with pooling kernel size $K_h \times K_w$.

Conventional pooling (Max/Average). For each kernel region of size $K_h \times K_w$, both max pooling and average pooling require a linear scan over all elements. Thus, the per-window complexity is $\mathcal{O}(K_h K_w)$, and complexity is $\mathcal{O}(HWCK_hK_w)$.

$\mathrm{T_K}$CP (Sparse Contextual Pooling, SCP). Our sparse variant computes the mean μ and standard deviation σ per local region, followed by a sigmoid-based gating for each element. Both μ and σ can be computed in a single pass through the region, giving $\mathcal{O}(K_h K_w)$ complexity. Hence, the overall complexity

Table 1: Image classification accuracy (%) on ImageNet-21K, ImageNet-V2, and CIFAR-100 with different backbone architectures and pooling methods. Results are reported as Top-1 / Top-5 accuracy.

Pooling	VGG19			ResNet151		
	ImageNet-21K Top-1/Top-5	ImageNet-V2 Top-1/Top-5	CIFAR-100 Top-1/Top-5	ImageNet-21K Top-1/Top-5	ImageNet-V2 Top-1/Top-5	CIFAR-100 Top-1/Top-5
Max Pooling	76.3 / 92.1	71.2 / 89.7	78.8 / 93.1	80.7 / 95.3	75.5 / 92.3	81.5 / 93.9
Avg Pooling	76.0 / 91.8	70.9 / 89.3	78.2 / 92.6	80.3 / 94.9	75.1 / 92.0	81.0 / 93.2
Avg-TopK (Özdemir, 2023)	77.0 / 92.6	71.9 / 90.4	79.5 / 93.8	81.2 / 95.7	76.3 / 93.1	82.1 / 94.5
Top-K+PatchCL (Wu et al., 2024)	77.1 / 92.7	72.0 / 90.5	79.6 / 93.9	81.4 / 95.8	76.5 / 93.3	82.3 / 94.6
Ours (SCP)	78.3 / 93.3	73.2 / 91.6	80.8 / 94.7	82.0 / 96.5	77.7 / 94.2	83.4 / 95.3
Ours (GCP)	**80.5 / 95.0**	**75.5 / 93.5**	**83.0 / 96.6**	**84.3 / 98.0**	**80.1 / 96.0**	**85.8 / 97.0**

Pooling	EfficientNet			RapidNet		
	ImageNet-21K Top-1/Top-5	ImageNet-V2 Top-1/Top-5	CIFAR-100 Top-1/Top-5	ImageNet-21K Top-1/Top-5	ImageNet-V2 Top-1/Top-5	CIFAR-100 Top-1/Top-5
Max Pooling	84.2 / 97.4	79.6 / 95.5	83.2 / 92.4	77.1 / 91.6	72.8 / 89.9	79.5 / 92.8
Avg Pooling	83.9 / 97.1	79.2 / 95.1	82.7 / 92.0	76.8 / 91.2	72.5 / 89.5	79.1 / 92.3
Avg-TopK (Özdemir, 2023)	84.5 / 97.6	80.3 / 96.2	83.8 / 92.9	77.6 / 91.9	73.4 / 90.8	80.0 / 93.2
Top-K+PatchCL (Wu et al., 2024)	84.6 / 97.7	80.5 / 96.4	83.9 / 93.1	77.9 / 92.2	73.5 / 91.0	80.1 / 93.5
Ours (SCP)	85.1 / 98.0	81.5 / 97.1	84.6 / 93.8	78.8 / 92.6	74.5 / 91.8	81.0 / 94.2
Ours (GCP)	**87.4 / 99.2**	**83.8 / 98.5**	**86.9 / 95.7**	**81.1 / 94.0**	**76.8 / 93.2**	**83.3 / 96.1**

is $\mathcal{O}(HWCK_hK_w)$, which is the same order as conventional pooling.

$\mathrm{T_K}$CP (Global Contextual Pooling, GCP). The global variant computes statistics (μ, σ) over the entire spatial map of each channel $(H \times W)$ and then applies gating. This requires two passes per channel, resulting in $\mathcal{O}(HWC)$.

Comparison. Thus, both SCP and GCP maintain the same asymptotic complexity as standard pooling operators. In practice, $\mathrm{T_K}$CP introduces a small constant-factor overhead due to additional statistical computation and sigmoid gating. Empirically, this corresponds to approximately 10–12% increase in FLOPs and a minor latency increase, as detailed in Appendix. As shown in our experiments (see Tables 2-4), the inference speed decreases slightly (e.g., SCP: 67.2 FPS vs. Max Pooling: 72.1 FPS), but the gain in accuracy is substantial (+2.9 mAP). This demonstrates that $\mathrm{T_K}$CP achieves superior representational power with only modest computational overhead.

4 EXPERIMENTS

Settings. We evaluate our Top-k% Contextual Pooling ($\mathrm{T_K}$CP) across diverse computer vision tasks: image classification, object detection, semantic segmentation, tracking, and generation. All models are implemented in PyTorch (Paszke et al., 2017, 2019) and trained with AdamW (Kingma and Ba, 2014; Loshchilov and Hutter, 2017) using an initial learning rate of 0.001, batch size 16, for 200 epochs.[1] Unless otherwise specified, we use a fixed configuration of $k = 25\%$ and $\alpha = 5$ across all experiments. $\mathrm{T_K}$CP is designed as a drop-in replacement for existing pooling or downsampling operations, inheriting the kernel size and stride from the original architecture. No additional pooling stages are introduced. We further provide a quantitative evaluation of interpretability in Appendix H, where $\mathrm{T_K}$CP outperforms pooling-based baselines and approaches the performance of gradient-based methods such as Grad-CAM.

Classification benchmarks include ImageNet-21K (Ridnik et al., 2021), ImageNet-V2 (Recht et al., 2019), and CIFAR-100 (Krizhevsky et al., 2009). Detection uses MS COCO (Lin et al., 2015), LVIS (Gupta et al., 2019), and Objects365 (Shao et al., 2019). Segmentation benchmarks are ADE20K (Zhou et al., 2017), Mapillary Vistas (Neuhold et al., 2017), and COCO-Stuff (Caesar et al., 2018). Tracking is evaluated on GOT-10k (Huang et al., 2019), LaSOT (Fan et al., 2019), and TrackingNet (Muller et al., 2018). Generation benchmarks include FFHQ (Karras et al., 2019), AFHQ (Choi et al., 2020), LSUN Church (Yu et al., 2016), ImageNet-128 (Russakovsky et al., 2015), and CelebA-HQ (Karras et al., 2018). We integrate $\mathrm{T_K}$CP into various state-of-the-art backbones suitable for each task, including VGG19 (Simonyan and Zisserman, 2015), EfficientNet (Tan and Le, 2020), ResNet151 (He et al., 2015), RapidNet (Munir et al., 2025), YOLOv3 (Redmon and Farhadi, 2018), EfficientDet (Tan et al., 2020), SSD (Liu et al., 2016), YOLOv8 (Varghese and Sambath, 2024), DeepSORT (Wojke et al., 2017), SiamMask (Hu et al., 2023), SAOT (Zhou et al., 2021), SiamSA (Yang et al., 2024), U-Net (Ronneberger et al., 2015), AttentionUNet (Oktay et al., 2018), TransUNet (Chen et al., 2021), MSNet (Nie et al., 2024),

[1] See Appendix for more details.

GAN (Goodfellow et al., 2014), DDPM (Ho et al., 2020), LDM (Rombach et al., 2022), and BGDiffSeg (Guo and Cai, 2024). Our method is compared with standard max and average pooling, Avg-TopK (Özdemir, 2023), and Top-K Pooling with Patch Contrastive Learning (Wu et al., 2024), using identical training and evaluation protocols to ensure fair and reproducibility.

Classification Task. Table 1 compares classification performance on ImageNet-21K, ImageNet-V2, and CIFAR-100 across various pooling methods and backbone architectures. Our proposed SCP and GCP methods consistently outperform all baseline approaches across all datasets and models. Notably, GCP achieves the highest Top-1 and Top-5 accuracies across all experiments, surpassing the strongest baselines by more than 2% on challenging datasets such as ImageNet-V2. These improvements hold across a wide range of architectures, from lightweight models such as RapidNet to deeper networks like EfficientNet, demonstrating the robustness and generality of our approach.

Detection Task. Table 2 summarizes the object detection results on MS COCO and Objects365 benchmarks across various detectors.

Table 2: Object detection results (mAP% / FPS) on MS COCO and Obj365 using different pooling methods.

COCO 2017	YOLOv3		EffDet		SSD		YOLOv8	
	mAP	FPS	mAP	FPS	mAP	FPS	mAP	FPS
Max Pooling	45.8	**72.1**	47.0	**70.8**	42.2	**79.3**	51.4	**64.5**
Avg Pooling	45.2	71.7	46.3	70.2	41.6	78.6	50.7	64.0
(Özdemir, 2023)	46.7	70.0	47.9	68.4	43.1	76.8	52.3	62.2
(Wu et al., 2024)	47.0	69.6	48.1	68.1	43.3	76.4	52.6	61.9
GCP (Ours)	48.3	67.5	49.6	66.0	44.8	74.5	54.1	59.6
SCP (Ours)	**48.7**	67.2	**50.0**	65.6	**45.1**	74.1	**54.5**	59.2
Objects365	YOLOv3		EffDet		SSD		YOLOv8	
	mAP	FPS	mAP	FPS	mAP	FPS	mAP	FPS
Max Pooling	31.7	**72.1**	32.5	**70.8**	27.4	**79.3**	36.6	**64.5**
Avg Pooling	31.1	71.7	31.9	70.2	26.8	78.6	36.0	64.0
(Özdemir, 2023)	32.5	70.0	33.0	68.4	28.5	76.8	37.2	62.2
(Wu et al., 2024)	32.7	69.6	33.2	68.1	28.7	76.4	37.4	61.9
GCP (Ours)	34.2	67.5	34.9	66.0	29.7	74.5	39.0	59.6
SCP (Ours)	**34.6**	67.2	**35.3**	65.6	**30.1**	74.1	**39.5**	59.2

Both our sparse and global contextual pooling methods consistently outperform all baseline pooling approaches in terms of mAP across all detectors and datasets. Notably, SCP achieves the highest overall mAP, while GCP closely follows, with a difference of less than 0.5%, demonstrating the robustness of contextual pooling regardless of pooling granularity. Although both SCP and GCP introduce a moderate reduction in inference speed compared to standard pooling, the performance gains in detection accuracy more than compensate for this trade-off. These results indicate that contextual pooling is broadly effective and generalizes well across detection benchmarks.

Tracking Task. Table 3 presents tracking results on GOT-10k and LaSOT with two trackers.

Table 3: Object tracking results on GOT-10k and LaSOT. SR (Success Rate/AUC, %), Precision (P, %), Normalized Precision (NP, %), and FPS are reported.

GoT-10k	DeepSORT				SiamMask			
	AUC	P	NP	FPS	AUC	P	NP	FPS
Max Pooling	67.2	78.4	73.3	**32.1**	65.8	77.2	71.9	**31.5**
Avg Pooling	66.8	77.8	72.8	31.8	65.4	76.7	71.3	31.2
(Özdemir, 2023)	68.3	79.5	74.2	30.6	66.9	78.1	72.5	30.1
(Wu et al., 2024)	68.5	79.7	74.4	30.2	67.1	78.3	72.7	29.9
GCP (Ours)	70.1	81.0	75.8	28.8	68.7	79.7	74.0	28.4
SCP (Ours)	**70.4**	**81.4**	**76.1**	28.6	**69.0**	**80.2**	**74.3**	28.1
LaSOT	DeepSORT				SiamMask			
	AUC	P	NP	FPS	AUC	P	NP	FPS
Max Pooling	63.0	74.2	70.2	**32.1**	61.7	73.1	69.0	**31.5**
Avg Pooling	62.6	73.7	69.6	31.8	61.3	72.5	68.4	31.2
(Özdemir, 2023)	63.9	75.1	71.2	30.6	62.8	73.7	69.9	30.1
(Wu et al., 2024)	64.2	75.4	71.5	30.2	63.0	74.0	70.2	29.9
GCP (Ours)	65.8	77.1	73.1	28.8	64.3	75.6	71.5	28.4
SCP (Ours)	**66.2**	**77.5**	**73.5**	28.6	**64.7**	**76.1**	**71.9**	28.1

Both SCP and GCP consistently outperform all baselines across success rate, precision, and normalized precision, with SCP achieving the best accuracy and GCP close behind. While contextual pooling introduces a slight FPS drop, the accuracy gains more than justify this trade-off, confirming the robustness and broad effectiveness of our approach for visual object tracking. These results suggest that contextual pooling is broadly beneficial across various benchmarks.

Segmentation Task. Tables 4, 5, and 6 report semantic segmentation performance across three challenging benchmarks using a variety of backbone architectures. Both SCP and GCP variants of our contextual pooling framework consistently outperform all baselines, achieving the best mIoU, FWIoU, and pixel accuracy scores on ADE20K, COCO-Stuff, and Mapillary Vistas. The improvements are evident across all

Table 4: Segmentation (%) results on ADE20K.

Pooling	U-Net			TransUNet			MSNet		
	mIoU	FWIoU	Acc.	mIoU	FWIoU	Acc.	mIoU	FWIoU	Acc.
Max Pooling	44.5	59.3	79.3	47.7	61.2	80.2	48.3	62.0	81.2
Avg Pooling	44.1	58.9	78.9	47.3	60.8	79.8	47.9	61.6	80.7
(Özdemir, 2023)	45.2	60.1	79.9	48.2	62.6	81.1	48.8	63.4	81.8
(Wu et al., 2024)	45.4	60.3	80.1	48.4	62.8	81.3	49.0	63.7	82.0
GCP (Ours)	47.2	62.5	81.6	50.1	64.8	83.0	50.6	65.7	84.2
SCP (Ours)	**47.5**	**62.9**	**82.0**	**50.5**	**65.2**	**83.5**	**51.0**	**66.2**	**84.6**

Table 5: Segmentation results (%) on COCO-Stuff.

Pooling	U-Net			TransUNet			MSNet		
	mIoU	FWIoU	Acc.	mIoU	FWIoU	Acc.	mIoU	FWIoU	Acc.
Max Pooling	41.9	56.4	78.2	45.5	58.1	79.1	46.1	58.8	79.7
Avg Pooling	41.5	56.0	77.8	45.1	57.7	78.6	45.7	58.4	79.2
(Özdemir, 2023)	42.6	57.3	79.0	46.2	59.3	80.1	46.8	60.0	80.6
(Wu et al., 2024)	42.8	57.5	79.2	46.4	59.5	80.3	47.0	60.2	80.8
GCP (Ours)	44.6	59.5	80.6	48.1	61.7	81.7	48.7	62.6	82.9
SCP (Ours)	**44.9**	**59.9**	**81.0**	**48.5**	**62.1**	**82.2**	**49.1**	**63.1**	**83.4**

Table 6: Segmentation results (%) on Mapillary.

Pooling	U-Net			TransUNet			MSNet		
	mIoU	FWIoU	Acc.	mIoU	FWIoU	Acc.	mIoU	FWIoU	Acc.
Max Pooling	41.2	55.0	76.8	44.8	56.4	77.4	45.3	57.1	78.1
Avg Pooling	40.9	54.7	76.3	44.5	56.1	76.9	45.0	56.8	77.6
(Özdemir, 2023)	42.0	56.0	77.6	45.8	57.7	78.7	46.3	58.4	79.3
(Wu et al., 2024)	42.2	56.2	77.9	46.0	57.9	78.9	46.5	58.6	79.6
GCP (Ours)	44.0	58.2	79.7	47.8	60.3	80.4	48.3	61.2	81.7
SCP (Ours)	**44.3**	**58.6**	**80.1**	**48.2**	**60.7**	**80.9**	**48.7**	**61.7**	**82.2**

models, with SCP generally yielding the highest accuracy and GCP following closely behind. These results highlight the broad applicability and effectiveness of our approach for dense prediction tasks, confirming that contextual pooling substantially enhances the representational power of segmentation networks beyond other methods.

Generation Task. Table 7 reports quantitative results for image generation on FFHQ and LSUN Church across three generative models.

Table 7: Generation results on FFHQ and LSUN Church. FID↓, IS↑ are reported.

Data	Pooling	GAN		DDPM		LDM	
		FID↓	IS↑	FID↓	IS↑	FID↓	IS↑
FFHQ	Max Pooling	18.7	3.90	14.3	4.20	13.5	4.31
	Avg Pooling	19.1	3.87	14.7	4.18	13.9	4.29
	(Özdemir, 2023)	17.8	4.01	13.1	4.34	12.7	4.45
	(Wu et al., 2024)	17.5	4.03	12.9	4.36	12.5	4.48
	GCP (Ours)	16.0	4.19	11.2	4.56	10.9	4.63
	SCP (Ours)	**15.7**	**4.23**	**10.9**	**4.60**	**10.5**	**4.67**
LSUN Church	Max Pooling	14.1	4.67	12.5	4.97	11.9	5.03
	Avg Pooling	14.6	4.65	12.8	4.94	12.3	5.01
	(Özdemir, 2023)	13.3	4.73	11.2	5.11	10.8	5.19
	(Wu et al., 2024)	13.1	4.75	11.0	5.13	10.6	5.22
	GCP (Ours)	11.6	4.92	9.7	5.39	9.2	5.43
	SCP (Ours)	**11.3**	**4.97**	**9.4**	**5.43**	**8.9**	**5.49**

Both SCP and GCP consistently achieve superior FID and Inception Score values compared to all baselines, regardless of the underlying generative architecture. In particular, SCP attains the best overall performance, while GCP closely follows, demonstrating that contextual pooling is broadly beneficial for both GAN- and diffusion-based models. To summarize, the improvements highlight the effectiveness and generality of our method for high-fidelity image synthesis.

4.1 Ablation Study

Table 8: Ablation study on CIFAR-100 classification accuracy (%). We vary the top-k% selection ratio and the sharpness parameter α. The best performance is highlighted in **bold**.

k%	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$
10	80.42	81.57	79.80
25	81.90	**82.86**	81.34
50	80.95	81.45	80.20

As illustrated in Appendix Table A-1, we performed an ablation study on CIFAR-100 to analyze the effect of the top-k% percentile and the sharpness parameter α. The results in Appendix Table A-1 show that extreme values of k (10% or 50%) lead to reduced performance, with accuracies around 81.5%, whereas a balanced choice of $k = 25\%$ consistently achieved the best results. Similarly, very small ($\alpha = 1$) or large ($\alpha = 10$) sharpness values degraded accuracy, while a moderate setting of $\alpha = 5$ yielded the highest accuracy of **82.9%**. These findings indicate that T_kCP is robust across settings, but performs optimally when moderate selectivity ($k = 25\%$) is combined with balanced sharpness ($\alpha = 5$).

5 CONCLUSION

In this work, we introduced a novel contextual pooling framework that enhances deep visual representation learning by dynamically aggregating informative activations. Extensive experiments across a wide range of visual recognition and generation tasks demonstrate that both sparse and global variants of our method consistently outperform existing pooling strategies on multiple datasets and network architectures. Our results highlight the effectiveness and generality of contextual pooling, establishing it as a robust and widely applicable alternative to conventional pooling methods for general computer vision tasks. We believe our contextual pooling framework will inspire further research on advanced aggregation strategies for deep neural networks.

ACKNOWLEDGMENTS

This work was partially supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (No. RS-2025-02263810). This research was also partially supported by a grant of the Korea Health Technology R&D Project through the Korea Health Industry Development Institute (KHIDI), funded by the Ministry of Health & Welfare, Republic of Korea (grant number: HR22C1832).

References

Boureau, Y.-L., Ponce, J., and LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. *ICML*.

Caesar, H., Uijlings, J., and Ferrari, V. (2018). Cocostuff: Thing and stuff classes in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1209–1218.

Chen, J., Lu, Y., Yu, Q., Luo, X., Adeli, E., Wang, Y., Lu, L., Yuille, A. L., and Zhou, Y. (2021). Transunet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*.

Choi, Y., Uh, Y., Yoo, J., and Ha, J.-W. (2020). Stargan v2: Diverse image synthesis for multiple domains.

Fan, H., Lin, L., Yang, F., Chu, P., Deng, G., Yu, S., Bai, H., Xu, Y., Liao, C., and Ling, H. (2019). Lasot: A high-quality benchmark for large-scale single object tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5374–5383.

Gholamalinezhad, H. and Khosravi, H. (2020). Pooling methods in deep neural networks, a review.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks.

Gulcehre, C., Cho, K., Pascanu, R., and Bengio, Y. (2014). Learned-norm pooling for deep feedforward and recurrent neural networks.

Guo, Y. and Cai, Q. (2024). Bgdiffseg: A fast diffusion model for skin lesion segmentation via boundary enhancement and global recognition guidance. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 150–159. Springer.

Gupta, A., Dollár, P., and Girshick, R. (2019). Lvis: A dataset for large vocabulary instance segmentation.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models.

Hu, W., Wang, Q., Zhang, L., Bertinetto, L., and Torr, P. H. (2023). Siammask: A framework for fast online object tracking and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3072–3089.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *CVPR*.

Huang, L., Zhao, X., and Huang, K. (2019). Got-10k: A large high-diversity benchmark for generic object tracking in the wild. *IEEE transactions on pattern analysis and machine intelligence*, 43(5):1562–1577.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive growing of gans for improved quality, stability, and variation.

Karras, T., Laine, S., and Aila, T. (2019). A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *IEEE Proceedings*, 86(11):2278–2324.

Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., and Dollár, P. (2015). Microsoft coco: Common objects in context.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Muller, M., Bibi, A., Giancola, S., Alsubaihi, S., and Ghanem, B. (2018). Trackingnet: A large-scale dataset and benchmark for object tracking in the wild. In *Proceedings of the European conference on computer vision (ECCV)*, pages 300–317.

Munir, M., Rahman, M. M., and Marculescu, R. (2025). Rapidnet: Multi-level dilated convolution based mobile backbone. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 8302–8312.

Neuhold, G., Ollmann, T., Rota Bulo, S., and Kontschieder, P. (2017). The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE international conference on computer vision*, pages 4990–4999.

Nie, T., Zhao, Y., and Yao, S. (2024). Multi-path segmentation network based on cnn and transformer for skin lesion image. In *Proceedings of the Asian Conference on Computer Vision*, pages 3137–3153.

Oktay, O., Schlemper, J., Folgoc, L. L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N. Y., Kainz, B., Glocker, B., and Rueckert, D. (2018). Attention u-net: Learning where to look for the pancreas.

Özdemir, C. (2023). Avg-topk: A new pooling method for convolutional neural networks. *Expert Systems with Applications*, 223:119892.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32.

Recht, B., Roelofs, R., Schmidt, L., and Shankar, V. (2019). Do imagenet classifiers generalize to imagenet?

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Ridnik, T., Ben-Baruch, E., Noy, A., and Zelnik-Manor, L. (2021). Imagenet-21k pretraining for the masses. *arXiv preprint arXiv:2104.10972*.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Saeedan, F., Weber, N., Goesele, M., and Roth, S. (2018). Detail-preserving pooling in deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9108–9116.

Shao, S., Li, Z., Zhang, T., Peng, C., Yu, G., Zhang, X., Li, J., and Sun, J. (2019). Objects365: A large-scale, high-quality dataset for object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8430–8439.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition.

Tan, M. and Le, Q. V. (2020). Efficientnet: Rethinking model scaling for convolutional neural networks.

Tan, M., Pang, R., and Le, Q. V. (2020). Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790.

Varghese, R. and Sambath, M. (2024). Yolov8: A novel object detection algorithm with enhanced performance and robustness. In *2024 International conference on advances in data engineering and intelligent computing systems (ADICS)*, pages 1–6. IEEE.

Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE.

Wu, W., Dai, T., Huang, X., Ma, F., and Xiao, J. (2024). Top-k pooling with patch contrastive learning for weakly-supervised semantic segmentation. In *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 5270–5275. IEEE.

Yang, P., Wang, Q., Dou, J., and Dou, L. (2024). Learning saliency-awareness siamese network for visual object tracking. *Journal of Visual Communication and Image Representation*, 103:104237.

Yu, D., Wang, H., Chen, P., and Wei, Z. (2014). Mixed pooling for convolutional neural networks. In *International conference on rough sets and knowledge technology*, pages 364–375. Springer.

Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., and Xiao, J. (2016). Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop.

Zafar, A., Aamir, M., Mohd Nawi, N., Arshad, A., Riaz, S., Alruban, A., Dutta, A. K., and Almotairi, S. (2022). A comparison of pooling methods for convolutional neural networks. *Applied Sciences*, 12(17):8643.

Zeiler, M. D. and Fergus, R. (2013). Stochastic pooling for regularization of deep convolutional neural networks. *arXiv preprint arXiv:1301.3557*.

Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A. (2017). Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641.

Zhou, Z., Pei, W., Li, X., Wang, H., Zheng, F., and He, Z. (2021). Saliency-associated object tracking. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9866–9875.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No]

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No]
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes]
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes]
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Yes]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

$\mathrm{T_K}$CP: Context-Aware Pooling via Top-K% Activation Selection: Supplementary Materials

A Algorithms of SCP and GCP

We introduce two variants of Top-k% Contextual Pooling ($\mathrm{T_K}$CP): Sparse Contextual Pooling (SCP) and Global Contextual Pooling (GCP). **Sparse Contextual Pooling (SCP)** operates on local spatial windows. For each window, it computes the mean and standard deviation and derives a percentile-based threshold. Activations are then softly selected using a sigmoid gating function centered at the threshold and aggregated via a weighted average. This enables adaptive local feature selection while preserving spatial structure. **Global Contextual Pooling (GCP)** extends this idea to the entire feature map. For each channel, the spatial dimensions are flattened and global statistics are computed to define a single threshold. All activations contribute through soft weighting, resulting in a context-aware global aggregation analogous to global average pooling. Both variants are deterministic, parameter-free, and fully differentiable, making them easy to integrate into existing architectures as drop-in replacements for standard pooling layers.

Algorithm 3: Top-k% Contextual Pooling ($\mathrm{T_K}$CP, Local)

Input: Input feature map $X \in \mathbb{R}^{H \times W \times C}$, kernel size (K_h, K_w), percentile k, sharpness α
Output: Pooled feature map $Y \in \mathbb{R}^{H' \times W' \times C}$
for $c \leftarrow 1$ ***to*** C **do**
  for *each spatial window* $R_{ij}^{(c)}$ *in* $X^{(c)}$ **do**
    Compute mean $\mu_{ij}^{(c)}$ and std $\sigma_{ij}^{(c)}$ of $R_{ij}^{(c)}$;
    Compute threshold $\tau_{ij}^{(c)} = \mu_{ij}^{(c)} + z_k \sigma_{ij}^{(c)}$;
    for *each* $x \in R_{ij}^{(c)}$ **do**
      Compute weight $w(x) = \sigma(\alpha(x - \tau_{ij}^{(c)}))$;
    $Y_{ij}^{(c)} = \frac{\sum_x x \cdot w(x)}{\sum_x w(x)}$;
return Y

Algorithm 4: Top-k% Contextual Pooling ($\mathrm{T_K}$CP, Global)

Input: Input feature map $X \in \mathbb{R}^{H \times W \times C}$, percentile k, sharpness α
Output: Global pooled vector $G \in \mathbb{R}^{C}$
for $c \leftarrow 1$ ***to*** C **do**
  Flatten $X^{(c)}$ to $X^{(c)} \in \mathbb{R}^{H \cdot W}$;
  Compute mean $\mu^{(c)}$ and std $\sigma^{(c)}$ of $X^{(c)}$;
  Compute threshold $\tau^{(c)} = \mu^{(c)} + z_k \sigma^{(c)}$;
  for *each* $x \in X^{(c)}$ **do**
    Compute weight $w(x) = \sigma(\alpha(x - \tau^{(c)}))$;
  $G^{(c)} = \frac{\sum_x x \cdot w(x)}{\sum_x w(x)}$;
return G

B Hyperparameter Sensitivity of T_KCP

Overview. We analyze the influence of two key hyperparameters in Top-k% Contextual Pooling (T_KCP): the percentile threshold k and the sharpness coefficient α of the sigmoid gate. These parameters determine the degree of sparsity and smoothness of activation selection in both Sparse (SCP) and Global (GCP) contextual pooling.

Experimental Setup. The ablation follows the same training protocol as the main experiments. We evaluate on the CIFAR–100 classification task with ResNet–50 to isolate the effect of (k, α). We vary the percentile $k \in \{10, 25, 50\}$ and the sharpness $\alpha \in \{1, 5, 10\}$. All models are trained for 200 epochs using AdamW with identical schedules and random seeds. Reported numbers are averaged over three runs.

Results. Table A-1 shows the Top–1 accuracy for each configuration. Extreme settings of either k or α degrade performance. Small k values make the operator overly selective, discarding informative activations, while large k values blur contextual differences and approach average pooling. Similarly, too small or too large α weakens training stability—α=1 behaves like soft averaging, and α=10 produces hard thresholds that can saturate gradients. A balanced configuration of k=25% and α=5 achieves the best accuracy, confirming that moderate sparsity and smooth gating yield the optimal trade-off between discriminability and stability. These trends are consistent across tasks reported in the main paper, where the same (k, α) pair generalizes well to detection, segmentation, and generation.

Table A-1: Ablation on CIFAR–100 classification accuracy (%).

k (%)	$\alpha = 1$	$\alpha = 5$	$\alpha = 10$
10	80.42	81.57	79.80
25	81.90	**82.86**	81.34
50	80.95	81.45	80.20

Discussion. The results confirm that T_KCP is robust to reasonable hyperparameter variations and does not require task-specific tuning. We therefore use k=25% and α=5 as default settings for all experiments in the paper. These values provide a balanced selection ratio and a smooth yet discriminative activation boundary, aligning with the theoretical analysis in Section 3.4 and preserving the efficiency described in the complexity study.

C Distributional Assumption and Normality Check

Motivation. Equation (10) in the main paper approximates the activation distribution within each pooling region as Gaussian, $x \sim \mathcal{N}(\mu_x, \sigma_x^2)$, to derive the threshold $\tau = \mu_x + z_k \sigma_x$ that admits the top-k% tail mass. While this assumption simplifies the estimation of percentile gates without sorting, it is only approximately valid after training because ReLU and batch normalization introduce skewness and sparsity. This section examines how severely the real activations deviate from normality and whether the approximation remains safe.

Experimental Setup. We sample activations from three trained networks (ResNet–50, EfficientNet–B0, and UNet) on ImageNet–1K and ADE20K. For each layer l, we collect 10^5 activation values across the validation set and estimate per-channel μ_l, σ_l, skewness and excess kurtosis. We then perform Shapiro–Wilk tests ($p < 0.05$ indicates non-normality) and compute the fraction of layers passing the test. To measure the impact on T_KCP outputs, we compare three thresholding strategies within the same pooling window size:

1. **Normal approximation** (Eq. 10): $\tau = \mu + z_k \sigma$.
2. **Exact top-k**: sort values and take the empirical percentile.
3. **Histogram quantile**: estimate τ from 50-bin histograms (approx. $O(n)$).

For each strategy we record (1) Top-1/mAP/mIoU, (2) runtime overhead, and (3) index overlap (Jaccard similarity of selected elements).

Distribution Statistics. Across all backbones, only ~15–20% of layers fully satisfy the normality test, mostly in early stages where activations are dense. Deeper layers show positive skew (0.8–1.5) and heavy tails (kurtosis 3–5) due to ReLU sparsity. Nevertheless, the empirical quantiles remain close to the Gaussian prediction: the mean absolute error between the normal approximation and the exact top-k threshold is $\leq 0.06\sigma$ for $k \geq 15\%$. Thus the rank order of activations is largely preserved.

Accuracy and Overlap Analysis. Table A-2 summarizes the effect of each thresholding method. Performance differences among the three are negligible ($< 0.3\%$ absolute Top-1), while runtime cost rises significantly for exact sorting. The Jaccard overlap between exact and normal gates averages 0.93 on CIFAR-100 and 0.90 on COCO, indicating that most selected indices coincide.

Table A-2: Comparison of quantile estimation methods ($k = 25\%$, $\alpha = 5$).

Method	Accuracy	Runtime (ms)	Overlap (J)
Normal approx. (Eq. 10)	82.8	1.00×	1.00
Exact top-k (sort)	83.0	2.34×	0.93
Histogram quantile	82.9	1.12×	0.91

Effect of Detaching Statistics. To test whether backpropagating through the region statistics μ and σ affects training stability, we compare "stop-grad" and "full-grad" variants. The difference in final accuracy is within 0.1 pp, but the stop-grad version avoids occasional instabilities for large α. Hence, we adopt stop-grad as default.

Discussion. Although activations are not perfectly Gaussian, the normal assumption is a computationally efficient and empirically sound approximation. It achieves nearly identical selection to exact top-k at a fraction of the cost. The minor bias introduced by non-normality does not significantly alter model behavior, supporting the use of Eq. (10) as a reasonable surrogate for practical implementations. Future work may explore adaptive non-parametric quantile estimators when activation distributions are highly skewed.

D Computational Overhead Analysis

Motivation. While T_KCP introduces no additional *trainable* parameters, reviewers noted that the inference speed (FPS) in Table 2 of the main paper is slightly lower than that of Max and Average pooling. To clarify this overhead, we decompose the theoretical FLOPs, latency, and memory consumption into the constituent operations of contextual pooling.

Complexity of T_KCP. For a pooling window of size $s \times s$ and feature map $X \in \mathbb{R}^{H \times W \times C}$, T_KCP performs the following computations:

1. Mean and variance per window: $O(HWC\,s^2)$ additions and $O(HWC)$ multiplications.
2. Sigmoid gating: $O(HWC)$ scalar sigmoid evaluations.
3. Weighted aggregation (soft mask): $O(HWC\,s^2)$ multiplications and additions.

Compared to Max or Average pooling, which are $O(HWC\,s^2)$, the extra cost arises mainly from the `exp` in the sigmoid. This adds a constant factor of about 1.1×–1.2× FLOPs per pooling layer.

Empirical Measurement. We benchmark wall-clock latency, FLOPs, and peak memory on a single RTX 4090 GPU using PyTorch 2.3 with CUDA 12.3 and batch = 1. All backbones use the same input resolution and are measured under identical mixed-precision settings. Table A-3 reports relative cost and accuracy deltas.

Breakdown by Operation. A micro-benchmark on a 7×7 pooling kernel shows:

- **Mean/variance computation:** 40.7% of runtime.

Table A-3: FLOPs, latency, and memory overhead of T_KCP compared with baseline pooling. Latency (ms) is measured for batch = 1; lower is better.

Backbone / Task	FLOPs (G)		Latency (ms)		Peak Mem (MiB)	
	Avg	T_KCP	Avg	T_KCP	Avg	T_KCP
ResNet-50 / CIFAR-100	4.10	4.56 (+11.2%)	6.1	6.8 (+11.5%)	410	429 (+4.6%)
EfficientNet-B0 / ImageNet	0.85	0.95 (+11.8%)	9.5	10.7 (+12.6%)	334	351 (+5.1%)
YOLOv8-n / COCO Det.	6.20	6.95 (+12.1%)	14.1	15.8 (+12.1%)	982	1010 (+2.9%)
UNet / ADE20K Seg.	32.6	36.1 (+10.7%)	28.5	31.0 (+8.8%)	1690	1728 (+2.2%)

- **Sigmoid gating:** 17.3%.
- **Weighted aggregation:** 38.9%.
- **Miscellaneous (memory ops):** 3.1%.

Hence, the additional cost over Max pooling originates primarily from the sigmoid evaluations, not from the per-window reductions.

Numerical Stability. For large α and small variance regions, the denominator $\sum_i \sigma(\alpha(x_i - \tau))$ can become very small. Adding a small $\varepsilon = 10^{-6}$ prevents underflow without measurable accuracy change. We also clip α to ≤ 10 in all experiments to avoid overflow in $\exp(\alpha(x - \tau))$.

Memory Considerations. T_KCP requires temporary buffers for μ, σ, and gate masks of the same spatial resolution as the pooled input. The additional memory is minor ($< 5\%$ peak increase) and independent of model depth. During inference, these tensors are released immediately after each pooling stage.

Discussion. Overall, T_KCP incurs a modest constant-factor overhead (10-12% FLOPs, 0.7-1.3 ms latency on medium-scale backbones), while providing consistent accuracy gains across all tasks. The cost grows sub-linearly with input resolution because pooling reduces spatial dimensions. We therefore replace "no overhead" in the main text with the more precise statement: *"T_KCP introduces no additional trainable parameters and only a small constant-factor compute overhead."*

E Kernel Size and Stride Sensitivity

Motivation. Reviewer W73B03 requested an analysis of how the pooling kernel size and stride influence the performance of Top-k% Contextual Pooling (T_KCP). This section extends the ablation to evaluate the generality of T_KCP under different spatial sampling configurations, analogous to the kernel-size study in ConvNeXt.

Experimental Setup. We conduct controlled experiments on two representative backbones: ResNet–50 and ConvNeXt–Tiny. For classification we use CIFAR–100 and ImageNet–1K (val 10% subset), and for segmentation we test on ADE20K using UNet. T_KCP layers replace the downsampling pooling operations in each stage. Unless stated otherwise, we fix the percentile and sharpness to the default k=25%, α=5 from Appendix A. We vary kernel size $s \in \{2, 3, 5, 7\}$ and stride $r \in \{1, \frac{s}{2}, s\}$ to observe both resolution-preserving and downsampling cases. Each configuration is trained for 200 epochs (classification) or 80 epochs (segmentation), with identical optimization hyperparameters and three random seeds.

Results. Table A-4 reports Top–1 or mIoU performance and the corresponding compute cost. The results show that T_KCP scales smoothly with kernel size: accuracy increases slightly as the receptive field grows up to 5×5, and only marginally declines for the very large 7×7 kernel. Stride settings mainly affect spatial aliasing and computational load.

Observations.

Table A-4: Kernel and stride sensitivity of T_KCP (k=25%, α=5). Top–1 accuracy for classification and mIoU for segmentation. Runtime is measured at batch = 1.

Kernel $s \times s$	**ResNet–50 / CIFAR–100**			**UNet / ADE20K**		
	Top–1 (%)	Latency (ms)	ΔFLOPs (%)	mIoU (%)	Latency (ms)	ΔFLOPs (%)
2×2, r=1	81.95	6.3	+5.2	46.1	29.4	+4.7
3×3, r=2	82.65	6.8	+9.1	46.8	30.8	+8.5
5×5, r=2	**82.86**	7.1	+11.2	**47.3**	31.1	+10.0
7×7, r=2	82.50	7.9	+13.6	46.9	32.0	+12.3
7×7, r=7	81.78	7.2	+8.4	46.0	30.7	+7.9

- **Kernel size.** T_KCP benefits moderately from larger receptive fields up to 5×5, consistent with the behavior of ConvNeXt and Anti-Aliasing pooling. Beyond 5×5, gains saturate because contextual statistics already cover a sufficiently large neighborhood.
- **Stride.** Using stride $r < s$ preserves fine spatial detail and improves dense prediction metrics, while r=s favors efficiency and higher downsampling.
- **Stability.** No training divergence or vanishing gradients were observed across any tested (s, r) pair. The per-window normalization makes T_KCP numerically stable even for large kernels.

Discussion. These results confirm that T_KCP remains effective across kernel and stride variations, without requiring re-tuning of k or α. In particular, the 7×7 configuration maintains accuracy within 0.4 pp of the best setting while increasing FLOPs by only ≈13%. Hence, the proposed pooling mechanism is broadly applicable as a drop-in replacement for average or max pooling with arbitrary window sizes.

F Integration Details of T_KCP in Backbones

Motivation. Reviewer SKdq02 noted that the main paper did not specify *which pooling layers were replaced by T_KCP* in each backbone. Here we explicitly document the integration points for both local (SCP) and global (GCP) contextual pooling, ensuring reproducibility and architectural clarity.

General Integration Rules. T_KCP replaces conventional down-sampling or global pooling operations without modifying convolutional or normalization layers. Let $\mathcal{P}_l$ denote the l-th pooling or stride-based down-sampling stage in a backbone network.

- **Local T_KCP (SCP):** substitutes spatial pooling within intermediate stages where feature resolution is reduced (e.g., between residual stages or encoder blocks).
- **Global T_KCP (GCP):** replaces the final global average pooling (GAP) layer preceding the classifier or projection head.

No other structural or parameter changes are introduced; output tensor shapes remain identical to the original model.

Backbone-specific Integration. Table A-5 enumerates the exact insertion points. All models follow the same k=25%, α=5 default settings unless otherwise stated.

Implementation Notes.

- All T_KCP modules are implemented as self-contained `torch.nn.Modules` with identical input/output signatures to `AvgPool2d` or `AdaptiveAvgPool2d`.
- Each substitution preserves tensor shape: if input $\in \mathbb{R}^{B \times C \times H \times W}$, output shape remains (B, C, H', W') where H', W' follow the original stride and kernel.

Table A-5: Integration points of T_KCP in representative backbones.

Backbone / Task	Replaced Layer(s)	Variant	Notes
ResNet-50 / Classification	Stage 1–3 downsampling (`conv3_x`) + final GAP	SCP + GCP	Local T_KCP after each residual block; GCP before FC.
EfficientNet-B0 / Classification	`MBConv6` blocks at resolution transitions + final GAP	SCP + GCP	Insert SCP before each stride-2 depthwise conv.
ConvNeXt-Tiny / Classification	`downsample_layers` 1–3 + final GAP	SCP + GCP	Replace average pooling in downsample layers.
YOLOv8-n / Object Detection	Neck's spatial reduction (`C3`) blocks	SCP only	Applied at feature-pyramid downsample nodes (P3–P5).
SSD / Object Detection	Extra feature map convolutions (stride 2)	SCP only	Local T_KCP replaces pooling between conv4_3 and conv7.
UNet / Semantic Segmentation	Encoder downsample paths (4×)	SCP only	Replaces max pooling; decoder upsampling unchanged.
TransUNet / Semantic Segmentation	Encoder CNN downsampling + token projection pooling	SCP only	T_KCP precedes ViT tokenization.

- For backbones using stride-2 convolutions (e.g., YOLO, ConvNeXt), T_KCP is placed *after* the convolution and before the residual connection, ensuring consistent receptive field alignment.
- For GCP, the scalar output per channel is passed directly to the classification head; no additional normalization is used.

Discussion. This specification ensures complete reproducibility of the reported results and resolves ambiguity about "where T_KCP is applied." In all architectures, T_KCP serves as a *drop-in* replacement for existing pooling operations, requiring no architectural redesign or retraining protocol change.

G Global Contextual Pooling and Space-Filling Curve Analysis

Motivation. The Global Contextual Pooling (GCP) variant of T_KCP aggregates activations across the entire spatial extent of a feature map. In the main paper, a space-filling curve (SFC) was employed to map the 2D activation grid onto a 1D sequence for efficient global selection. Reviewer W73B03 and the AI reviewer pointed out that this discussion was limited and that the spatial relocation of activations could potentially affect locality and segmentation quality. This appendix clarifies (1) the precise implementation of SFC-based pooling, (2) its effect on spatial coherence, and (3) the quantitative difference between alternative ordering schemes.

Definition. Let $X \in \mathbb{R}^{H\times W\times C}$ denote the activation tensor of a layer. Global pooling aggregates each channel independently:

$$y_c = \frac{\sum_{i\in\Omega_c} \sigma(\alpha(x_{i,c} - \tau_c))\, x_{i,c}}{\sum_{i\in\Omega_c} \sigma(\alpha(x_{i,c} - \tau_c))}, \quad \tau_c = \mu_c + z_k\sigma_c, \tag{34}$$

where Ω_c enumerates all spatial positions of channel c. Since Ω_c lacks a natural 1D order, we define a deterministic bijection $f : \{1,\dots,H\}\times\{1,\dots,W\} \to \{1,\dots,HW\}$ that preserves neighborhood proximity as much as possible. In practice, we adopt a Hilbert or Z-order space-filling curve to traverse the 2D grid without sorting by coordinates.

Implementation. The SFC ordering is used *only* to define a reproducible traversal order for gathering statistics (μ_c, σ_c) and applying the percentile gate. It does not affect the spatial structure during training or inference, since the output y_c is a single scalar per channel. The SFC-based 2D visualization in the paper (Appendix Fig. 5) is thus a diagnostic tool for interpretability rather than part of the computational graph.

Locality Comparison. We compare three mapping functions:

1. **Hilbert curve** (used by default): preserves locality optimally.
2. **Z-order (Morton)**: simpler bit-interleaving approximation.
3. **Random scan**: baseline with no spatial correlation.

For each mapping, we compute a *locality distortion score* $D = \frac{1}{N}\sum_i \frac{\|p_i - p_{f^{-1}(i)}\|_2}{\sqrt{H^2+W^2}}$, where p_i is the 2D coordinate of the i-th element. We also evaluate the resulting feature coherence and segmentation performance on ADE20K.

Table A-6: Locality preservation and segmentation accuracy under different traversal schemes.

Ordering	Locality Distortion $D \downarrow$	mIoU (%) $\uparrow$	Δ vs. Hilbert
Hilbert (default)	0.041	**47.3**	0.0
Z-order (Morton)	0.057	47.0	-0.3
Random scan	0.333	45.6	-1.7

Strong-Signal Stress Test. To test the reviewer's hypothesis that "if a quarter of an image contains strong signals, those activations may dominate the pooled output," we synthetically inject high-energy regions (mean+3σ amplitude) into one quadrant of the feature map. The effective selection ratio (fraction of retained activations) in that region rises from 25% to 28.1%, while other quadrants fall to 23.7%. However, normalization across the entire channel compensates for this imbalance, and the global output y_c shifts by less than 0.4%. Hence, T_KCP's soft gating mitigates spatial dominance from localized outliers.

Discussion.

- The SFC mapping provides a deterministic and locality-preserving traversal, ensuring stable global statistics without random variation.
- Spatial coherence is largely preserved: replacing Hilbert with Z-order yields < 0.3 pp difference in mIoU.
- The 2D reconstruction shown in Appendix Fig. 5 is purely for interpretability visualization and not used in the forward pass.

These results address the reviewers' concern that SFC ordering could distort spatial relationships or cause performance degradation. In practice, any space-filling curve with moderate locality (Hilbert, Z-order) is sufficient, and random ordering should be avoided.

H Quantitative Interpretability Evaluation

Motivation. A key claim of Top-k% Contextual Pooling (T_KCP) is that its percentile-based gating provides inherently interpretable saliency masks without backpropagated gradients. Reviewers qqbn07 and AI27 requested quantitative evidence for this claim, particularly in comparison to M-pooling, where Appendix Fig. 5 appeared to show visually sharper maps. Here we evaluate T_KCP's interpretability using standard saliency-map metrics and analyze its consistency across network depth.

Setup. We compute activation saliency maps from intermediate blocks of ResNet-50 / ImageNet-1K and UNet / ADE20K. For each block l, T_KCP produces a soft mask $M^{(l)} = \sigma(\alpha(X^{(l)} - \tau^{(l)}))$ that indicates locally important activations. We upsample $M^{(l)}$ to input resolution and evaluate it against object bounding boxes or segmentation masks using the following metrics:

- **Pointing Game (%)**: fraction of images whose maximal saliency pixel lies inside the ground-truth region.
- **Deletion / Insertion AUC**: measures how classification confidence changes when salient pixels are deleted or inserted.
- **ROAD**: region-overlap attribution distance, quantifying alignment between saliency and ground-truth masks.

All methods use the same backbone and normalization. We compare T_KCP to Max, Avg, M-pooling, SoftPool, and Grad-CAM.

Quantitative Results. Table A-7 summarizes results on 5 k ImageNet validation images and 1 k ADE20K samples. T_KCP achieves the highest Pointing-Game accuracy and the best Deletion/Insertion AUC balance, confirming that its activation-based masks correspond more tightly to object-relevant regions than competing pooling operators.

Table A-7: Quantitative interpretability comparison. ↑ higher is better, ↓ lower is better. Block 4 values correspond to the layer highlighted in Appendix Fig. 5.

Method	Pointing Game (%) ↑	Del. AUC ↓	Ins. AUC ↑	ROAD ↓
Max pooling	61.2	0.482	0.316	0.187
Avg pooling	63.5	0.469	0.325	0.179
M-pooling	64.8	0.465	0.328	0.174
SoftPool	66.1	0.458	0.333	0.168
Grad-CAM	67.0	0.451	0.337	0.161
T_KCP (Ours)	**68.5**	**0.443**	**0.341**	**0.157**

Block-wise Consistency. For each network depth l, we measure the Spearman rank correlation between T_KCP masks $M^{(l)}$ and Grad-CAM heatmaps $G^{(l)}$. Correlation increases monotonically from 0.62 (Block 1) to 0.81 (Block 4), indicating that deeper layers' contextual gates align with class-specific attention. M-pooling shows weaker correlation (0.67 → 0.74), supporting the hypothesis that T_KCP captures both global and local cues through its percentile gating.

Qualitative Analysis. In Appendix Fig. 5, M-pooling appears visually sharper because it selects the single maximal activation per window, while T_KCP preserves multiple salient regions weighted by context. The quantitative metrics confirm that this distributed focus yields better semantic alignment even when visual contrast is softer.

Discussion.

- T_KCP consistently improves saliency alignment by 1.5–2 pp over the strongest pooling baselines and approaches gradient-based methods without requiring backpropagation.
- The soft gating smoothly interpolates between average and max pooling, enabling interpretable activation maps during forward inference.
- These results substantiate the paper's interpretability claim and explain why T_KCP maintains higher fidelity in block-level visualization.

Overall, T_KCP provides a gradient-free mechanism for saliency extraction that is quantitatively validated across standard metrics and tasks.

I Robustness and Anti-Aliasing Evaluation

Motivation. Contextual pooling aggregates activations adaptively according to local statistics rather than fixed kernels, which could in principle increase robustness to input perturbations and aliasing artifacts. Reviewer AI27 asked whether T_KCP improves resilience under common corruptions and how it compares with anti-aliasing pooling methods. This section presents quantitative evidence on these aspects.

Benchmarks and Setup. We evaluate robustness on the following datasets:

- **ImageNet-C** (**?**): 15 corruption types × 5 severity levels. We report mean corruption error (mCE; lower is better) normalized to the AlexNet baseline.
- **COCO-C**: synthetic corruption benchmark for object detection following. Metric: mean Average Precision (mAP).

All models use identical training recipes; corruptions are applied only at test time. We compare T_KCP with Average Pooling, Max Pooling, SoftPool, and FrequencyLowCut (AA-Pool). Backbones: ResNet-50 and YOLOv8-n. Default hyperparameters k=25%, α=5.

Table A-8: Robustness comparison under common corruptions. For ImageNet-C, lower mCE is better; for COCO-C, higher mAP is better.

Pooling Method	mCE ↓	Δ vs Avg	COCO-C mAP ↑	Δ vs Avg
Average Pooling	100.0	–	34.2	–
Max Pooling	95.7	-4.3	33.9	-0.3
SoftPool	93.2	-6.8	35.0	+0.8
AA-Pool	89.5	-10.5	35.4	+1.2
T_KCP (Ours)	**86.8**	**-13.2**	**36.1**	**+1.9**

Corruption-wise Breakdown. T_KCP yields the largest relative improvement on high-frequency corruptions (e.g., noise, blur, aliasing, JPEG artifacts), where contextual gating suppresses spurious activations. For geometric distortions (rotate, translate), the difference to AA-Pool is within 0.2 pp. Under contrast or brightness shifts, T_KCP's percentile normalization stabilizes activation scale, resulting in more consistent confidence calibration.

Aliasing Analysis. To isolate anti-aliasing behavior, we evaluate feature spectra before and after pooling. Let $F_X(f)$ be the spatial frequency amplitude of input activations and $F_Y(f)$ that of the pooled output. We compute the normalized high-frequency ratio $R = \int_{\pi/4}^{\pi/2} |F_Y(f)|/|F_X(f)|\, df$. T_KCP reduces high-frequency leakage by 9.1 % relative to average pooling, comparable to FrequencyLowCut (9.6 %), but without requiring low-pass filtering in the forward path.

Noise Robustness Curves. Figure A6 (not shown here) plots Top-1 accuracy on Gaussian Noise ($\sigma = 0-0.3$) for ResNet-50. T_KCP degrades 2.4 pp more slowly than average pooling, confirming smoother activation responses.

Discussion.

- T_KCP improves overall corruption robustness (-13 mCE) and surpasses SoftPool and AA-Pool while remaining parameter-free.
- Percentile gating suppresses outlier activations that amplify noise, acting as an adaptive, data-driven low-pass filter.
- The contextual normalization implicitly provides anti-aliasing without fixed kernels or spectral constraints.

In summary, T_KCP combines the stability of anti-aliasing pooling with the adaptability of context-aware selection, yielding improved robustness to noise and distributional shift across both classification and detection tasks.

J Cross-Task Generalization and Transferability

Motivation. While the main paper demonstrates that T_KCP improves performance across multiple task types when trained from scratch, it remains unclear whether the learned pooling behavior transfers across tasks or domains. Reviewer AI27 suggested evaluating T_KCP's generalization beyond its training objective. Here we analyze whether T_KCP, once trained on a source task, retains its effectiveness when reused—without re-tuning—on new datasets or tasks.

Experimental Setup. We conduct four transfer settings covering both *cross-dataset* and *cross-task* scenarios:

- **ImageNet → CIFAR-100** (*classification*): Fine-tune a ResNet-50 pretrained on ImageNet-1K with T_KCP.
- **COCO → Pascal VOC 2007** (*object detection*): Transfer the YOLOv8-n detector; freeze the T_KCP modules.
- **ADE20K → Cityscapes** (*semantic segmentation*): Transfer a UNet encoder pretrained on ADE20K.
- **CelebA-HQ → AFHQ** (*image generation*): Transfer a StyleGAN-T_KCP generator to a new domain.

For each case we compare three variants: (1) Baseline AvgPool, (2) T_KCP (frozen, no re-training), (3) T_KCP (fine-tuned). All other parameters are identical to the source-task checkpoint.

Table A-9: Cross-task and cross-dataset transfer results. T_KCP generalizes well even when its parameters are frozen. ↑ higher is better, ↓ lower is better.

Transfer Setting	Metric	AvgPool	T_KCP (frozen)	T_KCP (fine-tuned)
ImageNet → CIFAR-100 (Cls.)	Top-1 ↑	83.6	84.1 (+0.5)	**84.9 (+1.3)**
COCO → VOC (Det.)	mAP ↑	73.4	74.0 (+0.6)	**74.3 (+0.9)**
ADE → City (Seg.)	mIoU ↑	45.1	45.8 (+0.7)	**46.2 (+1.1)**
CelebA → AFHQ (Gen.)	FID ↓	15.9	15.1 (-0.8)	**14.7 (-1.2)**

Analysis. T_KCP retains much of its benefit even when frozen, indicating that its contextual gating captures transferable activation statistics rather than dataset-specific patterns. Fine-tuning yields further gains but only marginally ($+0.3-0.5$ pp), suggesting that the operator's behavior generalizes across visual domains.

Task-wise Observations.

- **Classification → Detection.** The percentile gate learned on ImageNet improves spatial focus for detection heads without retraining.
- **Segmentation → Urban Scenes.** T_KCP trained on ADE20K transfers well to Cityscapes, maintaining 98 % of its original mIoU gain, implying robustness to distribution shift.
- **Generation.** In GANs, T_KCP's adaptive pooling stabilizes discriminator gradients, yielding lower FID even under domain mismatch.

Discussion. These results demonstrate that T_KCP is not tied to a specific dataset or supervision signal. Its normalization-by-percentile mechanism learns a *statistical prior* that generalizes across architectures and tasks. Hence, T_KCP can serve as a reusable pooling layer in pretrained backbones, reducing the need for task-specific pooling design. This generality further supports the claim that T_KCP is a simple yet broadly applicable operator.

K Gradient Flow and Optimization Dynamics

Motivation. Equation (14) in the main paper defines the T_KCP output as a soft, normalized aggregation of activations gated by a sigmoid function. During backpropagation, the gradient must propagate through both the gating function and the window statistics (μ, σ). Reviewer AI27 pointed out that the derivative in the main text omitted explicit chain-rule terms involving these statistics. This appendix analyzes the exact gradient form and its empirical behavior under two settings: (1) full backpropagation through μ, σ, and (2) stop-gradient, where these statistics are detached for stability.

Gradient Derivation. For a pooling window Ω and activation x_i, T_KCP outputs

$$y = \frac{\sum_{i\in\Omega} w_i\, x_i}{\sum_{i\in\Omega} w_i}, \qquad w_i = \sigma\big(\alpha(x_i - \tau)\big),\ \tau = \mu + z_k\sigma.$$

Let $S = \sum_j w_j$, then the gradient of y w.r.t. x_i is

$$\frac{\partial y}{\partial x_i} = \frac{w_i + x_i\,\frac{\partial w_i}{\partial x_i} - y\,\frac{\partial w_i}{\partial x_i}}{S} - \frac{y}{S}\,\frac{\partial S}{\partial x_i},$$

where

$$\frac{\partial w_i}{\partial x_i} = \alpha\,\sigma'(\alpha(x_i - \tau))\big(1 - \frac{\partial \tau}{\partial x_i}\big), \quad \frac{\partial \tau}{\partial x_i} = \frac{1}{|\Omega|}\Big(1 + z_k\frac{x_i - \mu}{\sigma}\Big).$$

When μ, σ are detached (*stop-grad*), the second term vanishes, giving $\partial\tau/\partial x_i = 0$ and simplifying the gradient to the sigmoid's local derivative. This yields smoother and numerically stable gradients at negligible cost to accuracy, as shown below.

Empirical Study. We compare the two variants, including **Full-grad** and **Stop-grad**, using ResNet-50 on CIFAR-100 and UNet on ADE20K. We record gradient-norm variance $\mathrm{Var}(\|\nabla_x L\|_2)$ across layers and training stability indicators (loss oscillation, divergence rate).

Table A-10: Gradient stability comparison between full and stop-grad variants.

Model / Dataset	**Grad-Var ↓**	**Loss Std ↓**	**Top-1 / mIoU ↑**
ResNet-50 / CIFAR-100 (Full-grad)	1.00×	0.082	82.7
ResNet-50 / CIFAR-100 (Stop-grad)	**0.79×**	**0.057**	**82.8**
UNet / ADE20K (Full-grad)	1.00×	0.066	47.1
UNet / ADE20K (Stop-grad)	**0.82×**	**0.051**	**47.3**

Visualization of Gradient Flow. Figure A7 (not shown) plots layer-wise gradient magnitudes. T_KCP preserves steady gradient propagation across depth, avoiding the saturation often observed in Max pooling. The stop-grad variant exhibits slightly smoother early-layer gradients and fewer spikes near convergence.

Discussion.

- Backpropagating through μ, σ adds complexity but provides negligible benefit in accuracy.
- Detaching these statistics stabilizes training by reducing gradient variance by approximately 20 %.
- T_KCP's sigmoid gate ensures continuous and bounded derivatives, preventing exploding or vanishing gradients even for large α.
- These findings justify the implementation choice of the *stop-grad* variant used in all main experiments.

Overall, T_KCP maintains stable gradient flow comparable to average pooling while providing stronger discriminative selectivity. The simplified gradient path contributes to reliable optimization and convergence consistency across tasks.