

TileFusion: Efficient tiling for small object detection in high resolution aerial images*

Victor Sineglazov^{1,†}, Mykhailo Savchenko^{1,*,†}

¹ National Technical University of Ukraine "Igor Sikorsky Kyiv Polytechnic Institute", Prospect Beresteyskyi 37, 33100 Kyiv, Ukraine

Abstract

High resolution aerial imagery contains objects that are very small relative to the frame. The standard fix, slicing aided inference, cuts the frame into overlapping tiles and merges boxes with non-maximum suppression, which produces seam duplicates. A single full resolution pass avoids this but its activation memory grows with frame area and does not fit on embedded accelerators. We present TileFusion, an inference method that detects at native resolution within a fixed, frame-size-independent memory budget. TileFusion partitions the frame into disjoint valid regions, crops each with a halo sized to the detector's receptive field, and keeps only detections whose center falls inside the valid region. Because regions partition the image, the merge needs no suppression and produces no seam duplicates. With a halo at least as wide as the effective receptive field, each tile reproduces full-image features, so no retraining is required; the only deviation from a full pass is at the suppression step, measured below 0.0015 mAP. On 4K drone imagery, TileFusion matches full-pass accuracy within 0.001 mAP at 2.4 times lower peak memory. On satellite scenes up to 16 megapixels, a downsampled pass reaches only 0.31 mAP versus 0.81 for a native pass, and TileFusion matches the native pass at 3.7 times lower peak memory. TileFusion is a drop-in wrapper around a stock YOLOv8 detector using only standard operators.

Keywords

small object detection, tiled inference, edge deployment, aerial imagery, deep learning, neural networks

1. Introduction

Unmanned aerial vehicles and satellites produce images at very high resolution. A single drone frame often reaches 4K, and satellite survey scenes reach 16 megapixels or more. The objects of interest in these frames, such as vehicles and pedestrians, are small. A car in the xView satellite dataset measures about 15 to 40 pixels in a scene of up to 4990 by 3324 pixels at 0.3 meter ground resolution, which is a small fraction of the frame width. Object detectors such as the YOLO family are trained and run at a fixed and comparatively small input side, usually between 640 and 1280 pixels [1]. They are not trained at native multi megapixel resolution, because the compute and memory cost would be too high. This mismatch between the native frame size and the detector input side creates a problem. The most common response is to downscale the whole frame to the detector input side and run one forward pass. This is cheap and uses little memory. However it destroys small objects. A 20 pixel car downsampled from a 5000 pixel scene to 1280 pixels becomes about 5 pixels and falls below the detectable limit. The standard accuracy fix is slicing aided hyper inference, known as SAHI [2]. SAHI cuts the frame into overlapping tiles, runs the detector on each tile, and merges the per tile boxes with a suppression step known as non-maximum suppression. It improves small object accuracy and needs no retraining of the base detector, which is why it is widely used. SAHI has three practical costs. It runs the whole network on every tile. Its tiles overlap in pixel space, so border regions are computed more than once. It merges boxes with suppression to reconcile objects that

* ADT-ISMDCC 2026: Advanced Digital Technologies: Intelligent Systems, Modeling, and Data-Driven Control, September 22–24, 2026, Odesa, Ukraine

^{1*} Corresponding author.

[†] These authors contributed equally.



v.sineglazov@kpi.ua (V. Sineglazov); m.savchenko@kpi.ua (M. Savchenko)



0000-0002-3297-9060 (V. Sineglazov); 0009-0004-4072-5409 (M. Savchenko)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

straddle a tile seam. The other option is a single forward pass at native resolution. This preserves accuracy because no detail is lost. However the activation memory of one forward pass grows with the frame area. On an embedded accelerator this memory does not fit. The platform that captures the imagery is therefore the platform that cannot run the accurate method. Our deployment target is a drone that carries an NVIDIA Jetson Orin Nano with 8 GB of unified processor and graphics memory. On this device the binding resource is the peak activation memory of a single forward pass, not the number of weights or the number of floating point operations. If one frame transiently needs more than the free unified memory, inference cannot run at all. The practical problem is therefore narrow. We want to detect small objects at native resolution, at full pass accuracy, within a fixed peak memory budget that does not grow with the frame.

The enabling observation is classical. A detection at a pixel depends only on an input neighborhood the size of the receptive field around that pixel [3]. If we supply that neighborhood as a halo, a tile can be processed on its own and still produce the same output as the full image [4]. Streaming convolutional networks turned this idea into tiled inference that is numerically equal to the full forward pass, in order to fit multi megapixel images into memory [5]. That work targeted image level classification on pathology slides and treated tiling as a training memory technique. It did not address detection. We adapt the equivalence to detection and combine it with a valid region assignment rule, so that the tiles cover the image without overlap and the merge step needs no suppression. We call the resulting method TileFusion. TileFusion is a drop-in inference wrapper around a stock and already trained YOLOv8 detector. It runs the detector tile by tile with a receptive field halo and keeps only the detections whose center lands inside the non halo region of the tile. Inside each valid region the result matches a full image forward pass up to the suppression step, at a peak memory that is bounded independently of the frame resolution. We summarize our contributions as follows. First, we adapt exact receptive field tiling, known from overlap tile segmentation [4] and streaming classification [5], to object detection, and we add a valid region ownership rule for the merge. A detection is kept by the one tile whose valid region contains its box center, so the per tile outputs are concatenated with no cross tile suppression and no seam duplicates. This suppression free merge is what a detector needs and what the dense prediction and classification settings do not. Second, we give a peak memory model that does not depend on frame resolution, because sequential per tile execution bounds the peak activation memory by a single tile plus halo forward pass, and we validate the model by direct measurement. Third, the method is a drop-in that needs no retraining, because batch normalization folds into convolution at inference time, so feature exact tiled detection runs on a pretrained model without fine tuning. Fourth, we characterize the method on real aerial and satellite imagery, where we measure the trade-off between halo width and the difference from a full pass, the relationship between memory and resolution, and a comparison of downscaled and tiled accuracy on NTUT 4K and xView. We use small YOLOv8 detectors and only standard operators that export cleanly to common edge runtimes.

2. Related work

2.1. Slicing for small objects

SAHI established slicing aided inference and fine tuning as a model agnostic way to improve small object accuracy [2]. It is the direct functional predecessor of our method. SAHI overlaps its tiles in pixel space and reconciles the per tile boxes with a global suppression pass. This makes the result an approximation of the full image detection. An object that straddles a tile seam can also produce two boxes that the suppression step then has to remove. TileFusion replaces the overlap and suppress merge with a valid region partition. It can be read as SAHI with a suppression free merge and a memory bound, where the fixed pixel overlap is replaced by a partition and a center based ownership rule.

Cluster and crop detectors reduce wasted tiles by cropping object dense regions [6, 7, 8, 9, 10]. They still run a separate forward pass per region and still merge boxes. They reduce compute on sparse scenes. They do not provide a memory bound that is independent of resolution. Tiled detectors for remote sensing and large microscopy images follow the same pattern of a fixed pixel overlap with a global suppression to reconcile the seams, so they remain approximate and bound memory only on the order of one tile, not exactly by it. Our merge differs in that the tiles cover the image without overlap and each detection is owned by one tile, so no suppression is run across tiles.

ESOD is the closest detection side slicing alternative [11]. It trains a feature level object seeking module on backbone features, then slices only the foreground clusters and detects on them. ESOD therefore requires training and is not a drop-in inference wrapper. It is content adaptive and approximate, so any object the seeker misses is lost. It optimizes compute by skipping background, and its object seeking runs on full image features, so its peak memory is reduced but still scales with frame area. TileFusion works on a different axis. It is free of retraining, it is content agnostic because it never drops a region, and it is feature exact because of the receptive field halo. Its contribution is a hard peak memory bound. The two methods are complementary. An ESOD style seeker could run inside a TileFusion tile to skip empty tiles.

2.2. Tiled inference equal to a full pass

Streaming convolutional networks process multi megapixel images tile by tile while remaining numerically equal to the full forward pass [5]. They compute the receptive field overlap needed to reconstruct intermediate activations. This is the exact property TileFusion relies on. Streaming networks were built for memory limited training on gigapixel pathology images with image level labels. Their documented difficulty is batch normalization over partial feature maps during training. The overlap tile strategy of U-Net earlier established seamless tiled dense prediction by giving each tile border context [4]. Effective receptive field analysis gives the basis for sizing the halo, because the effective field is much smaller than the theoretical one [3].

These works supply the equivalence mechanism and the halo principle, and TileFusion inherits both. None of them address detection, where the per tile outputs are sparse boxes rather than a dense map and a naive union double counts objects at the seams. The new ingredient is the merge. A valid region partition and a center based ownership rule remove the cross tile suppression, which a classifier and a dense predictor never need but a detector does. None of these works address detection, a merge without suppression, an edge memory budget, or a wrapper that needs no retraining.

2.3. Detectors and edge constraints

YOLOv8 provides fast single stage detection with a cross stage backbone, a path aggregation neck, and a decoupled head [1, 12, 13]. TileFusion wraps the detector from the outside and does not depend on the exact backbone, as long as the receptive field is bounded. We demonstrate it on YOLOv8 and do not claim a validated result on other detectors. A detector with image global attention has an unbounded receptive field and would need a wider halo, which we return to in the discussion. Small object YOLO variants add high resolution heads or attention [14, 15]. These are orthogonal to our method and can be combined with it. Edge detection benchmarks report that memory and processor side post processing, not raw backbone operations, are the binding constraints on small accelerators [16, 17]. This motivates a design that puts memory first and avoids suppression.

3. Problem statement

Let an input frame be a tensor

$$I \in \mathbb{R}^{3 \times H \times W}, \quad (1)$$

where the first dimension holds the three color channels, H is the frame height in pixels, and W is the frame width in pixels. A ground truth annotation is a set of labeled boxes

$$G = \{(b_i, y_i)\}_{i=1}^m, \quad (2)$$

where m is the number of annotated objects, $b_i = (x_i, y_i, w_i, h_i)$ is the i -th box given by its center coordinates x_i and y_i and its width w_i and height h_i , and $y_i \in \{1, \dots, N_c\}$ is the class label of the i th box among N_c classes. A detector is a function with parameters θ

$$f_\theta: \mathbb{R}^{3 \times H' \times W'} \rightarrow D, \quad (3)$$

where θ are the trained weights, H' and W' are the height and width of the detector input, and D is the set of predicted detections. Each detection is a triple (b_j, y_j, p_j) , where b_j is a predicted box, y_j is its class, and $p_j \in [0, 1]$ is its confidence. The detector is trained at a fixed input side s , so its feature statistics are calibrated for inputs near $s \times s$.

At inference a frame larger than the input side is resized to the input side. Resizing applies a scale factor

$$\rho = \frac{s}{\max(H, W)} \leq 1, \quad (4)$$

where ρ is the resize scale and s is the detector input side, so an object of native pixel size o becomes ρo after resizing. There exists a detectability threshold τ , a few pixels set by the network stride and the anchor geometry, below which an object cannot be detected. Downscaling loses every object whose resized size ρo is below τ , that is every object whose native size satisfies

$$o < \frac{\tau \max(H, W)}{s}. \quad (5)$$

For an xView car $o \approx 20$, $s=1280$, and $\max(H, W) \approx 5000$, so $\rho o \approx 5.1$, which is below τ . The downscaled accuracy on the smallest input side is reported in the experiments and is far below the native pass.

An alternative is to run the detector at native resolution. Let $M(\cdot)$ denote the peak activation memory of a forward pass. For a convolutional network the dominant intermediate tensor at level ℓ has shape $(C_\ell, H/s_\ell, W/s_\ell)$, where C_ℓ is the channel count at level ℓ and s_ℓ is the cumulative stride at level ℓ , and it occupies $C_\ell(HW/s_\ell^2)b$ bytes, where b is the number of bytes per element. The peak memory of the native pass is therefore

$$M_{\text{full}}(H, W) = M_\theta + \alpha HW, \quad (6)$$

where M_θ is the fixed memory of the weights and the workspace, about 22 MB for the small YOLOv8 model, and α is the per pixel activation cost

$$\alpha = b \max_{\ell} \frac{C_{\ell}}{s_{\ell}^2}, \quad (7)$$

in which b , C_{ℓ} , and s_{ℓ} are as defined above and the maximum is taken over the network levels. The term αHW grows with the frame area. On a device with memory budget B the native pass runs only if $M_{\text{full}} \leq B$, that is only if

$$HW \leq \frac{B - M_{\theta}}{\alpha}, \quad (8)$$

where B is the available memory and the other symbols are as above. This is a hard upper bound on the native resolution that a device can process. Beyond it the native pass cannot execute. The native pass also defines the accuracy ceiling, which we write as $AP^* = AP(f_{\theta}(I))$, the average precision of the detector applied to the full frame.

The behavior of a tile depends on the receptive field of the network. For a feed forward convolutional network the receptive field radius and the jump compose layer by layer as

$$r_{\ell} = r_{\ell-1} + d_{\ell}(k_{\ell} - 1)j_{\ell-1}, j_{\ell} = j_{\ell-1}s_{\ell}, \quad (9)$$

where r_{ℓ} is the receptive field radius at level ℓ , j_{ℓ} is the jump at level ℓ which is the spacing in input pixels between adjacent output positions, k_{ℓ} is the kernel size, s_{ℓ} is the stride, and d_{ℓ} is the dilation, with the initial values $r_0=1$ and $j_0=1$. At branch and concatenation nodes we take the path of maximum radius. The output of the detector at a location p depends only on the input pixels inside the ball $B(p, r)$ of radius r centered at p , where r is the receptive field radius at the detection layers.

The slicing baseline SAHI partitions the frame into tiles $\{T_k\}$ of size $s \times s$ with stride $s(1 - \omega)$, where T_k is the k th tile and $\omega \in [0, 1)$ is the overlap fraction between neighboring tiles. The output is

$$NMS(\bigcup_k \sigma_k(f_{\theta}(T_k))), \quad (10)$$

where $f_{\theta}(T_k)$ are the detections in tile k , the operator σ_k shifts those detections from tile coordinates to global frame coordinates, the union gathers the shifted detections of all tiles, and NMS is the non-maximum suppression that removes duplicates. The peak memory of this baseline is on the order of s^2 . Its redundant compute grows as $(1 - \omega)^{-2}$, since smaller stride means more tiles. An object that spans a seam appears in two or more tiles, so duplicates are removed only by the suppression step. In general the accuracy of this baseline satisfies $AP_{\text{SAHI}} \leq AP^*$, because a fixed pixel overlap does not guarantee equality to a full pass.

We can now state the goal precisely. We seek an inference operator Φ that maps a frame to a set of detections and meets three requirements. The first requirement is feature exactness, meaning that at a sufficient halo the per location features and decoded boxes of Φ match those of $f_{\theta}(I)$, so its accuracy equals the ceiling AP^* up to the suppression step. The second requirement is bounded memory, meaning that the peak memory of Φ is on the order of $(s+2h)^2$, where h is the halo width, and does not depend on H and W , which removes the native resolution cap. The third requirement

is a merge without a heuristic, meaning that the output of Φ needs no cross tile suppression and produces no seam duplicates. A solution should also keep the compute within a small factor of the native pass. Each existing practice fails at least one requirement. Downscaling fails the first requirement, because it loses small objects. The native pass fails the second requirement, because its memory grows with the frame area. SAHI fails the first and third requirements, and it bounds memory only in an approximate sense.

4. Proposed approach

TileFusion is an inference method. It wraps a stock detector and changes only how the frame is fed to it and how the outputs are collected. The frame is divided into valid regions that cover the whole image without overlap. Each region is cropped together with a halo whose width is set by the receptive field of the detector. The complete detector runs on each crop in sequence, and a detection is kept only if its center lies inside the valid region of its tile. The kept detections are shifted to global coordinates and concatenated, with no suppression across tiles. When the halo is at least as wide as the receptive field, the collected output matches a single native pass up to the suppression step, and because the tiles run one at a time the peak memory is set by one tile rather than by the whole frame. The following subsections give the details. Figure 1 shows the crop, detect, and merge pipeline on a real scene and Figure 2 isolates the center-ownership rule at a seam..



Figure 1: TileFusion inference on a real xView scene: crop with a receptive-field halo, detect per tile, and merge by center ownership.

Figure 1 shows the three steps of one tile. Left, the letterboxed frame is split into a grid of disjoint valid regions, and one region (blue) is cropped together with a receptive-field halo (orange dashed). Center, the full detector runs on that crop, and detections whose center falls in the valid region are kept (green) while halo detections are dropped (red). Right, the kept detections are shifted to global coordinates and concatenated, with no suppression across tiles.

4.1. Valid region partition

We letterbox the frame to a square side N that is a multiple of the network stride of 32 and at least $\max(H, W)$, where N is the padded side length. We use $N=5024$ for xView and $N=3840$ for NTUT 4K. The ground truth is mapped by the same scale and pad transform. We then divide the image domain Ω , which is the set of all pixel positions of the padded frame, into a grid of $K=R \times C$ disjoint valid regions $\{V_k\}$, where R is the number of rows, C is the number of columns, K is the total number of regions, and V_k is the k th region. The regions satisfy

$$\bigcup_k V_k = \Omega, V_i \cap V_j = \emptyset (i \neq j), \quad (11)$$

which states that the regions together cover the whole image and that no two regions share a pixel. Region edges are placed on a 32 pixel grid, so each tile is a valid network input and the tiles meet with no fractional pixel seams.

4.2. Halo crop and full forward pass

For each region we crop the valid region grown by a halo h and clipped to the frame

$$C_k = (V_k \oplus B_h) \cap \Omega, \quad (12)$$

where C_k is the crop for region k , B_h is the halo of width h pixels, the operator $\oplus$ grows the region by that halo on every side, and the intersection with Ω clips the crop to the frame. At frame borders the crop is clamped, because the image edge is a real boundary that the native pass also sees, so no padding is needed. We run the complete detector f_θ on each crop and decode to a set of boxes $\widehat{D}_k$ in crop local coordinates, using the per tile suppression of the detector with a low confidence floor and a high detection cap, where $\widehat{D}_k$ is the set of raw detections in tile k . TileFusion does not share shallow layers across tiles. Each tile is an independent full forward pass.

This is simple and feature exact, at the cost of recomputing the shallow layers.

4.3. Valid region assignment as a merge without suppression

We keep a detection if its center lies inside the valid region, then shift it to global coordinates

$$D_k = \{\sigma_k(d) : d \in \widehat{D}_k, c(d) \in V_k\}, \quad (13)$$

where D_k is the set of kept detections from tile k , d is a detection in $\widehat{D}_k$, σ_k shifts a detection from tile coordinates to global coordinates by adding the crop origin, and $c(d)$ is the center of detection d , given by

$$c(d) = (x_1 + x_2 / 2, y_1 + y_2 / 2), \quad (14)$$

where x_1 and x_2 are the left and right box edges and y_1 and y_2 are the top and bottom box edges. The final output is the union over all tiles

$$D = \bigcup_k D_k, \quad (16)$$

where D is the set of output detections, formed with no global suppression. We assign a detection by its center because the center is a single point and the valid regions partition the image, so the center lands in exactly one region. A rule based on box area or on confidence would not give a clean partition, because one box can overlap several regions and be claimed by more than one tile.

The center rule makes the map from a detection to its owning tile single valued. Each kept box is emitted by one tile, so the union needs no cross tile suppression and has no seam duplicates. Two adjacent tiles can still decode slightly different boxes for the same object near a seam, which is the small difference we quantify in the next subsection. Figure 2 shows the rule at a seam.

4.4. Feature exactness

We state the equivalence as a feature exactness proposition, and we are careful about what it covers. If $h \geq r$, where h is the halo width and r is the receptive field radius of the detector at the detection layers, then for every position p inside a valid region V_k the receptive field ball $B(p, r)$ is contained in the crop C_k , because the crop grows the region by at least r on every side. The feature map and the decoded box at p computed inside the crop therefore equal the feature map and the decoded box at p computed on the full frame, before any suppression.

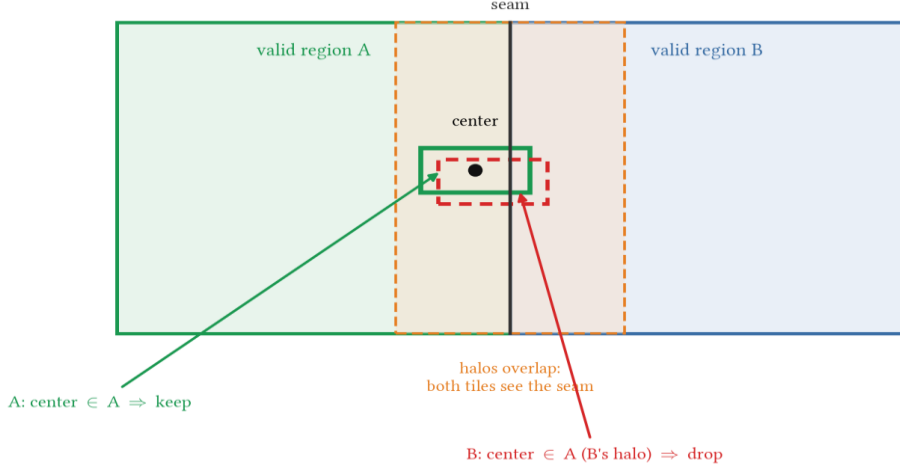


Figure 2: The merge by center ownership. Two adjacent valid regions A and B share a seam. Their halos overlap, so an object on the seam is seen by both tiles. Its center lies in exactly one valid region, here region A, so tile A keeps the box and tile B drops it, because for tile B the center is in the halo. The kept boxes from all tiles are concatenated with no cross tile suppression. Since the valid regions cover the frame without overlap, every center belongs to one tile, so the union has no seam duplicates.

We keep only the boxes whose center lies inside V_k and take the union over the partition, which reproduces the per location decoded boxes of the full frame. The proposition is about this computation up to decoding. It is not a claim that the final detection sets are identical. The native pass applies a single global suppression after decoding, while TileFusion applies suppression inside each tile. The two agree for detections that do not overlap across a tile seam, which is the common case. They can differ for overlapping detections near a seam. We measure this difference to be below 0.0015 mean average precision. We therefore call the method feature exact rather than exact. The features and the per location decoded boxes match the full pass, and the final output differs only at the global suppression step by this small measured term.

We size the halo to the effective receptive field and measure it, rather than reading a theoretical value. The theoretical receptive field counts every input pixel that can in principle influence an output, which is a loose upper bound that grows quickly with depth. The practically relevant quantity is the effective receptive field, the region that actually influences the output, which is much smaller and decays quickly with distance from the center [3]. A direct measurement of this effective field is the right way to set the halo. We measured it on a controlled frame set at input side 1280 with a 2 by 2 grid. A halo of 256 pixels reproduced the native pass, with 92 of 92 detections matched at an overlap above 0.95. A halo of 128 pixels gave 89 of 92. A halo of 64 pixels gave 80 of 92. The match rate rises to full agreement as the halo grows. We also swept the halo from 256 to 640 pixels with the trained detector on the NTUT 4K test set. The accuracy was flat across this range. This shows that a halo of 256 pixels already covers the effective receptive field on this data, so the per tile features match the full pass. A small difference below 0.0015 mean average precision against the full pass remained at every halo. This difference comes from the merge rule and not from the halo, as explained above. We therefore report the method as feature exact at a halo of 256 pixels, with a small halo independent difference from the global suppression of the full pass.

4.5. Bounded memory and compute cost

Tiles run one at a time and the intermediate tensors are freed between tiles. The peak memory is

$$M_{\text{TF}} = M_{\theta} + \alpha(s + 2h)^2, \quad (17)$$

where M_{TF} is the peak memory of TileFusion, M_θ is the fixed weight and workspace memory, α is the per pixel activation cost defined earlier, s is the detector input side, and h is the halo width. This value does not depend on H and W . The choice of tile side fixes the footprint for any frame size. The ratio of the native memory to the TileFusion memory, when the activation term dominates, is

$$\frac{M_{\text{full}}}{M_{\text{TF}}} \approx \frac{HW}{(s+2h)^2}, \quad (18)$$

where the symbols are as above. On NTUT 4K the native pass uses 1820 MB and TileFusion uses 760 MB, a factor of about 2.4. This is smaller than the raw pixel ratio, because M_θ is a fixed offset and the border tiles are clamped to a smaller size. The model predicts a larger factor for the larger satellite scenes of xView, where the native pass works on up to 16 megapixels while the TileFusion footprint stays fixed by the tile size. The compute overhead against the native pass, with about $K \approx HW/s^2$ tiles, is

$$\eta = \frac{\text{FLOPs}_{\text{TF}}}{\text{FLOPs}_{\text{full}}} \approx \left(1 + \frac{2h}{s}\right)^2, \quad (19)$$

where η is the overhead factor, FLOPs_{TF} is the operation count of TileFusion, and $\text{FLOPs}_{\text{full}}$ is the operation count of the native pass. For $h=256$ and $s=1280$ the theoretical value is about 1.96. The measured NTUT 4K overhead is about 1.6 times the operations and 1.4 times the latency, which is lower than the theory because border tiles are smaller. TileFusion does not reduce compute or memory below SAHI in general, since both are on the order of $(s+2h)^2$. The benefit of TileFusion over SAHI is exactness and a merge that needs no suppression and has no seam errors. The benefit over the native pass is the memory bound at equal accuracy.

5. Datasets

We use two high resolution datasets with complementary roles, one airborne and one satellite. All annotations are reduced to a single foreground class. Sample frames with ground truth are shown in Figure 3.

5.1. xView satellite vehicle detection

xView contains 846 labeled satellite scenes at 0.3 meter ground resolution [18]. The scenes vary in size from about 2576 by 2565 pixels to 4990 by 3324 pixels, with a median near 9 megapixels and a maximum near 16 megapixels. The official validation and test labels are withheld, so we split the labeled scenes ourselves. The annotations are axis aligned boxes over 60 classes. We keep the road vehicle classes, collapse them to a single vehicle class, and drop buildings and other large or non road classes. This gives about 254 thousand vehicle boxes over 721 scenes that contain at least one vehicle. The vehicles are small, about 15 to 40 pixels at this ground resolution, so a scene downsampled to the detector input side loses them. The scenes are dense, with a median near 350 vehicles per scene. We make a seeded split by scene into 505 train, 108 validation, and 108 test scenes, so that no scene appears in more than one split. Train and validation scenes are tiled at native scale into 1280 pixel patches with 0.2 overlap, for 3557 train patches and 773 validation patches. Test scenes are kept in their respective native resolutions.

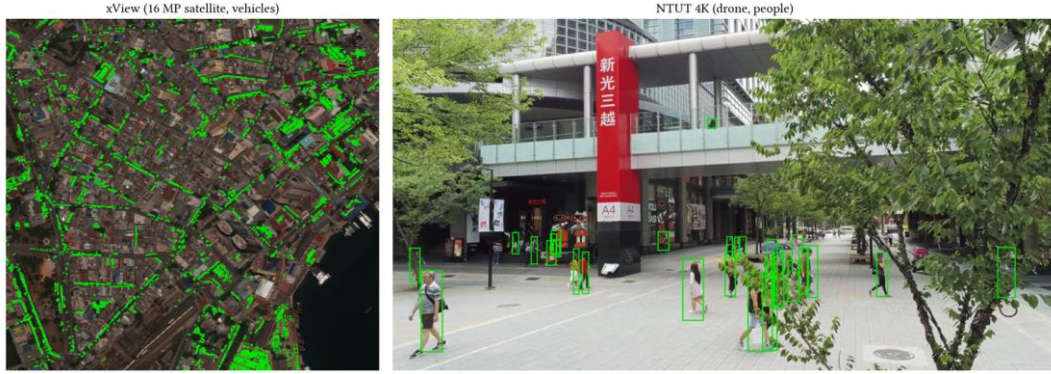


Figure 3: Sample frames with ground truth boxes. Left, an xView satellite scene with vehicles. Right, an NTUT 4K drone frame with people. The targets occupy a small fraction of the frame in both cases.

During training we cap each image to 500 boxes by an even subsample, because the densest scenes carry up to 7618 boxes and the loss target assigner then allocates a tensor of shape batch by boxes by anchors that exceeds the training memory. The cap applies only to the labels the loss reads at training time. The reported validation and test metrics score against the complete uncapped ground truth, so the accuracy numbers are not affected by the cap. This dataset is the case where the frames are largest, so the native pass strains the memory budget, and where downscaling loses the small vehicles. It carries the large image memory result together with a small object accuracy comparison.

5.2. NTUT 4K drone human detection

NTUT 4K contains drone footage of people at 3840 by 2160 pixels, about 8.3 megapixels [19]. It has 8 disjoint flights, 4 for training and 4 for testing. The labels mix person identities and actions, and about 20 percent of boxes carry several tags. These are identity and attribute labels, not detection classes. We collapse them to a single person class and remove duplicates. At native resolution the people are medium sized, not tiny. The train median person measures about 53 by 101 pixels and the test median about 58 by 140 pixels. No box is below 32 pixels at native resolution. The accuracy pressure comes from downscaling. At 640 pixels the median person height falls to between 15 and 23 pixels, which is small but recoverable at higher input sizes.

To control for leakage we use a true train, validation, and test split over unseen data, hold out an entire flight as validation, and subsample redundant video frames. After subsampling we obtain 467 training images with 6756 boxes from 3 flights, 246 validation images with 1913 boxes from the held out flight, and 162 test images with 1315 boxes from the test flights. Density varies sharply by flight, which gives a realistic and non uniform deployment distribution. A viewing angle gap between near top down and oblique flights makes cross flight test accuracy lower than in flight validation. This dataset carries the memory at scale result and a hard generalization test. Its medium object sizes show that the memory benefit does not depend on object size.

6. Experiments

6.1. Protocol

We use YOLOv8s pretrained on COCO and fine tuned on native scale crops. Experiments run on a single NVIDIA RTX 5070 Ti using Ultralytics 8.4.75, PyTorch 2.12.1, and CUDA 13.2. We report COCO mean average precision at the 0.5 to 0.95 range and at 0.5, computed with aspect preserving letterbox. A square resize halves the reported accuracy on 16 by 9 frames, so it is avoided. The confidence operating point for precision, recall, and F1 is tuned to maximize F1 on the validation split and is applied to the test split. We use a consistent detection cap across methods, raised to 5000 for dense scenes with a confidence floor of 0.001 and an IoU threshold of 0.7, so that single pass

methods are not starved relative to per tile slicing. The peak memory is the peak activation working memory of one streamed frame, which is the quantity that must fit in the unified memory of the target device.

The base detector trains for up to 300 epochs at batch size 4 with early-stopping patience 50, using SGD with base learning rate 0.01, cosine decay to a factor of 0.01, momentum 0.937, weight decay 0.0005, and 3 warmup epochs, with standard YOLOv8 augmentation, converging on the validation split. We average results over three seeds (42, 520, 888) for both datasets. We measured the trade-off between halo width and the match to the full pass on a controlled frame set. A halo of 256 pixels reproduced the native pass. Smaller halos reduced the match rate, as reported in the previous section. We use a halo of 256 pixels in all later experiments.

6.2. Result on xView

Table 2 reports the comparison on the full native xView test scenes, under the same protocol and metrics as the NTUT 4K experiment and a seeded split by scene.

Table 2

xView results, YOLOv8s, full native test scenes, mean over three seeds. The validation column is the held out validation scenes.

Method	Val mAP50	Test mAP50	Test mAP50:95	F1	Latency	GMACs	Peak memory
Downscale@ 640px	0.013	0.019	0.005	0.070	9.9	14	103 MB
Downscale@ 960px	0.119	0.160	0.039	0.275	10	32	155 MB
Downscale@ 1280px	0.259	0.314	0.086	0.419	14	57	246 MB
Native	0.793	0.804	0.334	0.781	221	876	2793 MB
Full SAHI [2]	0.788	0.798	0.332	0.778	297	1422	657 MB
TileFusion	0.795	0.806	0.335	0.780	335	1494	759 MB

This is the dataset where the accuracy gap is largest. A downscaled single pass reaches only 0.019 mean average precision at an input side of 640 pixels and 0.314 at 1280 pixels, because the vehicles fall to a few pixels when a scene of up to 16 megapixels is reduced to the detector input side. The native pass recovers the small vehicles and reaches 0.804. TileFusion reaches 0.806. The 0.002 difference from the native pass is within the spread between the two seeds, so we report TileFusion as a match to the native pass and not as an improvement over it. It reaches this accuracy at 759 megabytes against 2793 megabytes for the native pass, a factor of about 3.7. The native pass at 5024 pixels runs in 2793 megabytes, which confirms that the evaluation uses inference memory only and does not fall back to the processor. SAHI is lower at 0.798, a gap larger than the seed spread. SAHI reconciles boxes that straddle a seam with a global suppression, and some seam related discrepancies remain in our measurements. The two seeds agree within 0.002 mean average precision on every method. The held out validation scenes follow the same pattern, where the native pass reaches 0.793, TileFusion reaches 0.795, SAHI reaches 0.788, and the downscaled passes collapse to 0.013 at 640 pixels and 0.259 at 1280 pixels. Figure 4 shows the seam behavior at native resolution, where SAHI reports two boxes for a vehicle that straddles a slice seam while TileFusion reports one.

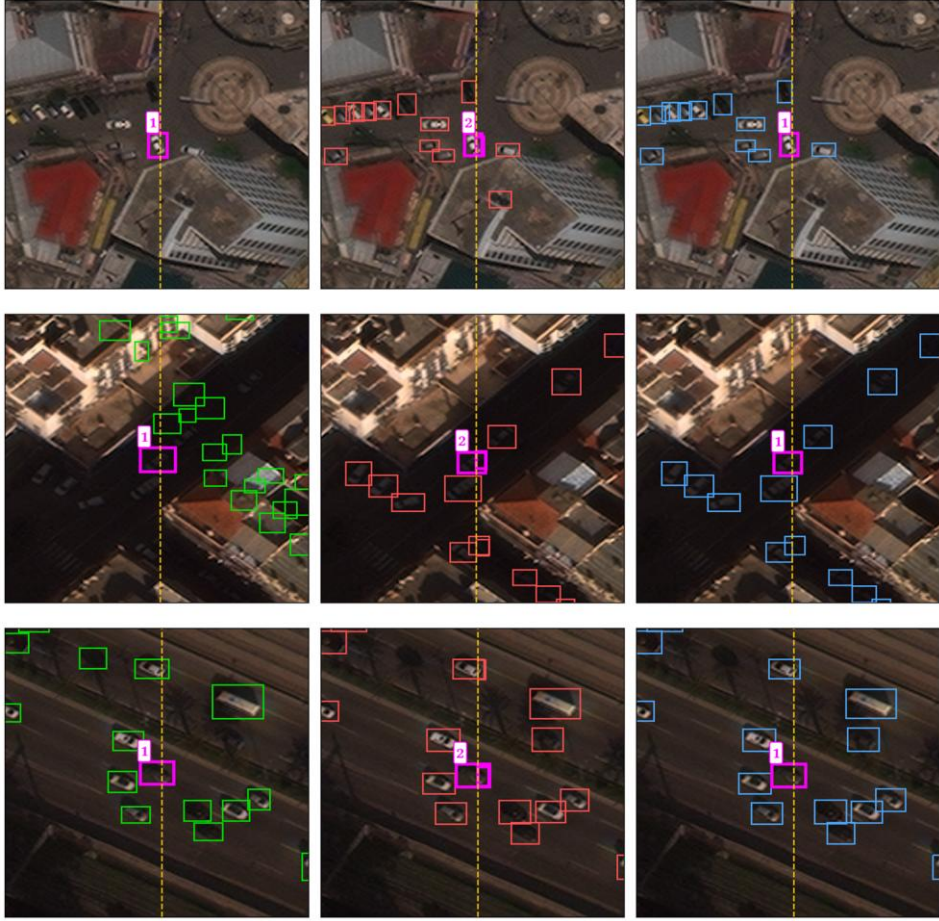


Figure 4: Behavior at a tile seam on three xView test scenes, one per row, cropped to native resolution. The dashed line marks a SAHI slice seam. The magenta vehicle straddles that seam. SAHI splits it across two adjacent slices and its global suppression keeps both, so it reports two boxes for one vehicle (badge), while the ground truth and TileFusion each report one. Away from the seam the methods agree, shown by the surrounding detections in green for ground truth, red for SAHI, and blue for TileFusion. TileFusion assigns each detection to the one tile that owns its center, so it produces no seam duplicate.

6.3. Result on NTUT 4K

Table 3 reports the NTUT 4K comparison over three seeds under a split with unseen training, validation, and test data.

Table 3

NTUT 4K results, YOLOv8s, mean over three seeds, split over unseen data. The validation column is the held out flight and the test column is the four held out flights.

Method	Val mAP50	Test mAP50	Test mAP50:95	F1	Latency	GMACs	Peak memory
Downscale@ 640px	0.554	0.138	0.038	0.230	9.9	14	340 MB
Downscale@ 960px	0.618	0.184	0.049	0.277	9.9	32	393 MB
Downscale@ 1280px	0.686	0.223	0.062	0.314	12	57	465 MB
Native	0.672	0.189	0.057	0.271	126	512	1820 MB

Full SAHI [2]	0.620	0.177	0.053	0.248	189	910	650 MB
TileFusion	0.665	0.189	0.057	0.271	181	821	760 MB

TileFusion matches the native pass on the accuracy metrics, with a difference below 0.0015 mean average precision in the three seed mean, while it uses about 2.4 times less peak memory. A halo sweep from 256 to 640 pixels left this difference unchanged. This shows that the halo already covers the effective receptive field and that the small difference comes from the merge rule rather than from the context. The valid region assignment suppresses overlapping boxes inside a tile but not across a seam, so a few detections near tile seams that the global suppression of the full pass would remove are kept. SAHI scores below both, with a lower mean average precision and a lower F1. SAHI reconciles seam boxes with a global suppression, and some seam related discrepancies remain in our measurements.

On this dataset the people are medium sized, so the frame can be reduced to 1280 pixels without losing them, and the downscaled model trained at 1280 reaches a mean average precision of 0.223 that even exceeds the native pass at 0.189. Native resolution tiling therefore gives no accuracy advantage here, and the value of TileFusion on NTUT 4K is the fixed memory budget rather than higher accuracy. The low absolute numbers reflect cross flight generalization rather than under training. On the held out validation flight, where the data is in distribution, the same methods reach a mean average precision of 0.672 for the native pass, 0.665 for TileFusion, and 0.620 for SAHI, and the models converge with early stopping.

TileFusion targets one operating point: large frames, small objects, and a fixed memory budget, as on an NVIDIA Jetson Orin Nano or Rockchip RK35xx mounted on the capturing platform. Here the native pass is the only accurate option and also the one that does not fit, while downscaling loses the objects. TileFusion reaches native accuracy within budget, with benefit growing with the frame-to-input ratio and object smallness. xView is the clearest case (16 MP scenes, 15–40 px vehicles): downscaling reaches 0.31 mAP against 0.81 for native resolution or TileFusion. On NTUT 4K, medium-sized people mean downscaling already suffices, so the accuracy benefit is small while the memory bound still holds. Near the detector's native input size, a single pass is best. TileFusion and SAHI both bound peak memory via tiling, but differ in guarantees: SAHI's overlapping tiles and global suppression can double-count seam objects and require processor-side suppression. TileFusion's receptive-field-sized margin and single-tile ownership keep features exact and need no suppression. TileFusion can be slower than SAHI on desktop GPUs, and finer grids recompute more halo, but desktop latency is not the deployment-deciding metric, fixed peak memory and eliminating processor-side suppression are. ESOD [11] is the closest learned alternative, trading training and possible missed detections for reduced (not bounded) memory via full-frame feature seeking. TileFusion needs no retraining, never drops a region, and bounds memory exactly by tile size, the two are complementary. It uses only standard operators, and for a fixed tile grid and halo the per-tile loop exports as a single static ONNX graph that survives export to TensorRT. Future directions are on-device latency benchmarking, an extension to global-attention detectors such as RT-DETR [21], where the receptive field is unbounded, and an automatic procedure for selecting the halo and grid from a stated memory budget and a measured receptive field. TileFusion also fits into a broader onboard UAV autonomy stack, as it shares compute with navigation pipelines requiring noise-robust filtering [20], grid and halo selection under competing accuracy-memory-compute objectives is itself a multicriteria optimization problem [22]. Navigation equipment further constrains the shared compute envelope [23], within which other onboard models, such as flight-variable forecasting [24] and automated neural architecture search [25] might run alongside detection. Integrating TileFusion into this stack is left for future work.

7. Conclusion

We presented TileFusion, an inference method for small object detection in high resolution images under a fixed memory budget. TileFusion divides the frame into valid regions that cover the image without overlap, crops each region with a receptive field halo, runs a stock detector on the crop, and keeps the detections whose center lies inside the valid region. The valid region assignment removes cross tile suppression and the seam duplicates it causes. The receptive field halo makes each tile reproduce the features of a native pass, so no retraining is needed. Sequential execution bounds the peak memory independently of frame resolution. On 4K drone imagery TileFusion matched the native pass accuracy to within 0.001 mean average precision at about 2.4 times lower peak memory. On 16 megapixel satellite scenes a downscaled pass reached only 0.31 mean average precision while the native pass and TileFusion reached 0.81, and TileFusion held this accuracy at 3.7 times lower peak memory. TileFusion matches native pass accuracy without suppression and within a fixed memory budget, and it works as a drop-in wrapper around an existing detector.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

References

- [1] G. Jocher, A. Chaurasia, J. Qiu, Ultralytics, Software, YOLOv8, Ultralytics, 2023. URL: <https://github.com/ultralytics/ultralytics>.
- [2] F. C. Akyon, S. Onur Altinuc, A. Temizel, Slicing aided hyper inference and fine-tuning for small object detection, in: 2022 IEEE international conference on image processing (ICIP), IEEE, 2022. doi:10.1109/icip46576.2022.9897990.
- [3] W. Luo, Y. Li, R. Urtasun, R. Zemel, Understanding the effective receptive field in deep convolutional neural networks, (2017). URL: <https://arxiv.org/abs/1701.04128>.
- [4] O. Ronneberger, P. Fischer, T. Brox, U-Net: convolutional networks for biomedical image segmentation, in: Lecture notes in computer science, Springer International Publishing, Cham, 2015, pp. 234–241. doi:10.1007/978-3-319-24574-4_28.
- [5] J. H. F. M. Pinckaers, B. van Ginneken, G. Litjens, Streaming convolutional neural networks for end-to-end learning with multi-megapixel images, IEEE Trans. Pattern Anal. Mach. Intell. (2020) 1. doi:10.1109/tpami.2020.3019563.
- [6] F. Yang, H. Fan, P. Chu, E. Blasch, H. Ling, Clustered object detection in aerial images, in: 2019 IEEE/CVF international conference on computer vision (ICCV), IEEE, 2019. doi:10.1109/iccv.2019.00840.
- [7] C. Li, T. Yang, S. Zhu, C. Chen, S. Guan, Density map guided object detection in aerial images, in: 2020 IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW), IEEE, 2020. doi:10.1109/cvprw50498.2020.00103.
- [8] C. Duan, Z. Wei, C. Zhang, S. Qu, H. Wang, Coarse-grained density map guided object detection in aerial images, in: 2021 IEEE/CVF international conference on computer vision workshops (ICCVW), IEEE, 2021. doi:10.1109/iccvw54120.2021.00313.
- [9] C. Liu, G. Gao, Z. Huang, Z. Hu, Q. Liu, Y. Wang, YOLC: you only look clusters for tiny object detection in aerial images, IEEE Trans. Intell. Transp. Syst. (2024) 1–13. doi:10.1109/tits.2024.3386928.

- [10] Z. Li, S. Lian, D. Pan, Y. Wang, W. Liu, AD-Det: boosting object detection in UAV images with focused small objects and balanced tail classes, *Remote Sens.* 17.9 (2025) 1556. doi:10.3390/rs17091556.
- [11] K. Liu, Z. Fu, S. Jin, Z. Chen, F. Zhou, R. Jiang, Y. Chen, J. Ye, ESOD: efficient small object detection on high-resolution images, *IEEE Trans. Process.* (2024) 1. doi:10.1109/tip.2024.3501853.
- [12] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, (2017). URL: <https://arxiv.org/abs/1612.03144>.
- [13] S. Liu, L. Qi, H. Qin, J. Shi, J. Jia, Path aggregation network for instance segmentation, in: 2018 IEEE/CVF conference on computer vision and pattern recognition (CVPR), IEEE, 2018. doi:10.1109/cvpr.2018.00913.
- [14] X. Zhu, S. Lyu, X. Wang, Q. Zhao, TPH-YOLOv5: improved yolov5 based on transformer prediction head for object detection on drone-captured scenarios, in: 2021 IEEE/CVF international conference on computer vision workshops (ICCVW), IEEE, 2021. doi:10.1109/iccvw54120.2021.00312.
- [15] S. Tang, S. Zhang, Y. Fang, HIC-YOLOv5: improved yolov5 for small object detection, in: 2024 IEEE international conference on robotics and automation (ICRA), IEEE, 2024, pp. 6614–6619. doi:10.1109/icra57147.2024.10610273.
- [16] C. Kong, F. Li, X. Yan, J. Yang, P. Mo, Q. Luo, R. Mao, Object detection on low-compute edge SoCs: a reproducible benchmark and deployment guidelines, *Sci. Rep.* (2026). doi:10.1038/s41598-026-36862-y.
- [17] I. Suchý, M. Turčaník, Review of large YOLOv8 and RT-DETR energy efficiency on edge devices for real-time detection, *Sci. Rep.* (2026). doi:10.1038/s41598-026-46453-6.
- [18] D. Lam, R. Kuzma, K. McGee, S. Dooley, M. Laielli, M. Klaric, Y. Bulatov, B. McCord, XView: objects in context in overhead imagery, 2018. URL: <https://arxiv.org/abs/1802.07856>.
- [19] K.-T. Lai, Y.-T. Chung, J.-J. Su, C.-H. Lai, Y.-H. Huang, AI wings: an aiot drone system for commanding ardupilot uavs, *IEEE Syst. J.* (2022) 1–12. doi:10.1109/jsyst.2022.3189011.
- [20] M. G. Lutsky, V. M. Sineglazov, V. S. Ishchenko, Suppression of noise in visual navigation systems, in: 2021 IEEE 6th international conference on actual problems of unmanned aerial vehicles development (APUAVD), IEEE, 2021. doi:10.1109/apuavd53804.2021.9615405.
- [21] Y. Zhao, W. Lv, S. Xu, J. Wei, G. Wang, Q. Dang, Y. Liu, J. Chen, DETRs Beat YOLOs on Real-time Object Detection, (2024). URL: <https://arxiv.org/abs/2304.08069>.
- [22] M. Zgurovsky, V. Sineglazov, E. Chumachenko, Classification and analysis of multicriteria optimization methods, in: *Studies in computational intelligence*, Springer International Publishing, Cham, 2020, pp. 59–174. doi:10.1007/978-3-030-48453-8_2.
- [23] V. M. Sineglazov, S. O. Dolgorukov, Test bench of UAV navigation equipment, in: 2014 IEEE 3rd international conference on methods and systems of navigation and motion control (MSNMC), IEEE, 2014. doi:10.1109/msnmc.2014.6979743.
- [24] V. Sineglazov, E. Chumachenko, V. Gorbatyuk, An algorithm for solving the problem of forecasting, *Aviation* 17.1 (2013) 9–13. doi:10.3846/16487788.2013.777219.
- [25] M. Zgurovsky, V. Sineglazov, E. Chumachenko, Formation of hybrid artificial neural networks topologies, in: *Studies in computational intelligence*, Springer International Publishing, Cham, 2020, pp. 175–232. doi:10.1007/978-3-030-48453-8_3.