Learning to Zoom and Unzoom

Chittesh Thavamani[1] Mengtian Li[†][1] Francesco Ferroni[‡][2] Deva Ramanan[1]
[1]Carnegie Mellon University [2]Argo AI
tchittesh@gmail.org mengtial@alumni.cmu.edu fferroni@nvidia.com deva@cs.cmu.edu

Abstract

Many perception systems in mobile computing, autonomous navigation, and AR/VR face strict compute constraints that are particularly challenging for high-resolution input images. Previous works propose nonuniform downsamplers that "learn to zoom" on salient image regions, reducing compute while retaining task-relevant image information. However, for tasks with spatial labels (such as 2D/3D object detection and semantic segmentation), such distortions may harm performance. In this work (LZU), we "learn to zoom" in on the input image, compute spatial features, and then "unzoom" to revert any deformations. To enable efficient and differentiable unzooming, we approximate the zooming warp with a piecewise bilinear mapping that is invertible. LZU can be applied to any task with 2D spatial input and any model with 2D spatial features, and we demonstrate this versatility by evaluating on a variety of tasks and datasets: object detection *on Argoverse-HD,* semantic segmentation *on Cityscapes, and* monocular 3D object detection *on nuScenes. Interestingly, we observe boosts in performance even when high-resolution sensor data is unavailable, implying that LZU can be used to "learn to upsample" as well. Code and additional visuals are available at* https://tchittesh.github.io/lzu/.

1. Introduction

In many applications, the performance of perception systems is bottlenecked by strict inference-time constraints. This can be due to limited compute (as in mobile computing), a need for strong real-time performance (as in autonomous vehicles), or both (as in augmented/virtual reality). These constraints are particularly crippling for settings with high-resolution sensor data. Even with optimizations like model compression [4] and quantization [23], it is common practice to downsample inputs during inference.

However, running inference at a lower resolution undeniably destroys information. While some information loss is unavoidable, the usual solution of uniform downsampling assumes that each pixel is equally informative towards the task at hand. To rectify this assumption, Recasens *et al.* [20] propose Learning to Zoom (LZ), a nonuniform downsampler that samples more densely at salient (task-relevant) image regions. They demonstrate superior performance relative to uniform downsampling on human gaze estimation and fine-grained image classification. However, this formulation warps the input image and thus requires labels to be invariant to such deformations.

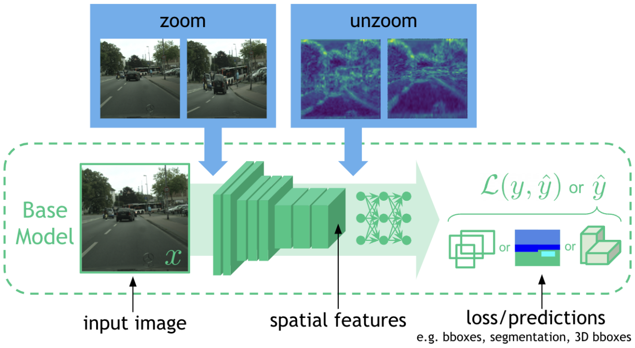
zoom
unzoom
Base Model
x
$\mathcal{L}(y, \hat{y})$ or $\hat{y}$
or
or
input image
spatial features
loss/predictions
e.g. bboxes, segmentation, 3D bboxes

Figure 1. LZU is characterized by "zooming" the input image, computing spatial features, then "unzooming" to revert spatial deformations. LZU can be applied to any task and model that makes use of internal 2D features to process 2D inputs. We show visual examples of output tasks including 2D detection, semantic segmentation, and 3D detection from RGB images.

Adapting LZ downsampling to tasks with spatial labels is trickier, but has been accomplished in followup works for semantic segmentation (LDS [11]) and 2D object detection (FOVEA [22]). LDS [11] does not unzoom during learning, and so defines losses in the warped space. This necessitates additional regularization that may not apply to non-pixel-dense tasks like detection. FOVEA [22] *does* unzoom bounding boxes for 2D detection, but uses a special purpose solution that avoids computing an inverse, making it inapplicable to pixel-dense tasks like semantic segmentation. Despite these otherwise elegant solutions, there doesn't seem to be a general task-agnostic solution for intelligent downsampling.

[†]Now at Waymo.
[‡]Now at Nvidia.

Our primary contribution is a general framework in which we zoom in on an input image, process the zoomed image, and then *un*zoom the output back with an inverse warp. Learning to Zoom and Unzoom (LZU) can be applied to *any* network that uses 2D spatial features to process 2D spatial inputs (Figure 1) *with no adjustments to the network or loss*. To unzoom, we approximate the zooming warp with a piecewise bilinear mapping. This allows efficient and differentiable computation of the forward and inverse warps.

To demonstrate the generality of LZU, we demonstrate performance a variety of tasks: *object detection* with RetinaNet [17] on Argoverse-HD [14], *semantic segmentation* with PSPNet [29] on Cityscapes [7], and *monocular 3D detection* with FCOS3D [26] on nuScenes [2]. In our experiments, to maintain favorable accuracy-latency tradeoffs, we use cheap sources of saliency (as in [22]) when determining where to zoom. On each task, LZU increases performance over uniform downsampling and prior works with minimal additional latency.

Interestingly, for both 2D and 3D object detection, we also see performance boosts even when processing low resolution input data. While prior works focus on performance improvements via intelligent downsampling [20, 22], our results show that LZU can also improve performance by intelligently *up*sampling (suggesting that current networks struggle to remain scale invariant for small objects, a well-known observation in the detection community [18]).

2. Related Work

We split related work into two sections. The first discusses the broad class of methods aiming to improve efficiency by paying "attention" to specific image regions. The second delves into works like LZU that accomplish this by differentiably resampling the input image.

2.1. Spatial Attentional Processing

By design, convolutional neural networks pay equal "attention" (perform the same computations) to all regions of the image. In many cases, this is suboptimal, and much work has gone into developing attentional methods that resolve this inefficiency.

One such method is Dynamic Convolutions [24], which uses sparse convolutions to selectively compute outputs at only the salient regions. Similarly, gated convolutions are used in [12,28]. Notably, these methods implement "hard" attention in that the saliency is binary, and non-salient regions are ignored completely.

Deformable Convolutions [8, 30] provides a softer implementation of spatial attention by learning per pixel offsets when applying convolutions, allowing each output pixel to attend adaptively to pixels in the input image. SegBlocks [25] also provides a softer attention mechanism by splitting the image into blocks and training a lightweight reinforcement learning policy to determine whether each block should be processed at a high or low resolution. This is akin to our method, which also has variable resolution, albeit in a more continuous manner. Our method is also generalizable to tasks in which it's infeasible to "stitch" together outputs from different blocks of the image (e.g. in detection where an object can span multiple blocks).

2.2. Spatial Attention via Differentiable Image Resampling

Spatial Transformer Networks [10] introduces a differentiable method to resample an image. They originally propose this to invert changes in appearance due to viewpoint, thereby enforcing better pose invariance.

Learning to Zoom (LZ) [20] later adapts this resampling operation to "zoom" on salient image regions, acting as a spatial attention mechanism. Their key contribution is a transformation parameterized by a saliency map such that regions with higher saliency are more densely sampled. However, this deforms the image, requiring the task to have non-spatial labels.

Followup works [11, 19, 22] adapt LZ downsampling to detection and semantic segmentation. For object detection, FOVEA [22] exploits the fact that image resampling is implemented via an inverse mapping to map predicted bounding boxes back into the original image space. This allows all processing to be done in the downsampled space and the final bounding box regression loss to be computed in the original space. However, when there are intermediate losses, as is the case with two-stage detectors containing region proposal networks (RPNs) [21], this requires more complex modifications to the usual delta loss formulation, due to the irreversibility of the inverse mapping. For semantic segmentation, Jin *et al*. [11] apply LZ downsampling to both the input image and the ground truth and computes the loss in the downsampled space. This is elegant and model-agnostic but leads to misalignment between the training objective and the desired evaluation metric. In the extreme case, the model learns degenerate warps that sample "easy" parts of the image to reduce the training loss. To address this, they introduce additional regularization on the downsampler. Independently, [19] handcraft an energy minimization formulation to sample more densely at semantic boundaries.

In terms of warping and unwarping, the closest approach to ours is Dense Transformer Networks [13], which also inverts deformations introduced by nonuniform resampling. However, their warping formulation is not saliency-based, which makes it hard to work with spatial or temporal priors and also makes it time-consuming to produce the warping parameters. Additionally, they only show results for semantic segmentation, whereas we show that our formulation generalizes across spatial vision tasks.

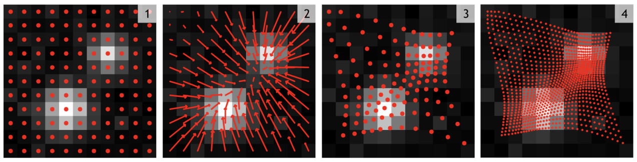

Figure 2. Illustration of $\mathcal{T}_{\mathrm{LZ}}$ [20]. Suppose we have a saliency map $\mathbf{S} \in \mathbb{R}^{h\times w}$ (visualized in the background) and want a warped image of size $H' \times W'$. (1) We start with a uniform grid of sample locations $\mathrm{Grid}(h,w)$. (2) Grid points are "attracted" to nearby areas with high saliency. (3) Applying this "force" yields $\mathcal{T}_{\mathrm{LZ}}[\mathrm{Grid}(h,w)]$. (4) Bilinear upsampling yields $\widetilde{\mathcal{T}}_{\mathrm{LZ}}[\mathrm{Grid}(H',W')]$.

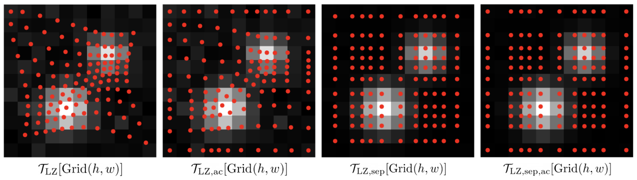
$\mathcal{T}_{\mathrm{LZ}}[\mathrm{Grid}(h,w)]$ $\mathcal{T}_{\mathrm{LZ,ac}}[\mathrm{Grid}(h,w)]$ $\mathcal{T}_{\mathrm{LZ,sep}}[\mathrm{Grid}(h,w)]$ $\mathcal{T}_{\mathrm{LZ,sep,ac}}[\mathrm{Grid}(h,w)]$

Figure 3. Examples of the anti-cropping (ac) and separable (sep) variants of $\mathcal{T}_{\mathrm{LZ}}$ from [22].

3. Background

Since our method is a generalization of previous works [11, 20, 22], we include this section as a condensed explanation of prerequisite formulations critical to understanding LZU.

3.1. Image Resampling

Suppose we want to resample an input image $\mathbf{I}(\mathbf{x})$ to produce an output image $\mathbf{I}'(\mathbf{x})$, both indexed by spatial coordinates $\mathbf{x} \in [0,1]^2$. Resampling is typically implemented via an *inverse* map $\mathcal{T} : [0,1]^2 \to [0,1]^2$ from output to input coordinates [1]. For each output coordinate, the inverse map computes the source location from which to "steal" the pixel value, i.e. $\mathbf{I}'(\mathbf{x}) = \mathbf{I}(\mathcal{T}(\mathbf{x}))$. In practice, we are often given a discretized input image $\mathbf{I} \in \mathbb{R}^{H\times W\times C}$ and are interested in computing a discretized output $\mathbf{I}' \in \mathbb{R}^{H'\times W'\times C}$. To do so, we compute $\mathbf{I}'(\mathbf{x})$ at grid points $\mathbf{x} \in \mathrm{Grid}(H',W')$, where $\mathrm{Grid}(H,W) := \mathrm{Grid}(H) \times \mathrm{Grid}(W)$ and $\mathrm{Grid}(D) := \{\frac{d-1}{D-1} : d \in [D]\}$. However, $\mathcal{T}(\mathbf{x})$ may return non-integer pixel locations at which the exact value of $\mathbf{I}$ is unknown. In such cases, we use bilinear interpolation to compute $\mathbf{I}(\mathcal{T}(\mathbf{x}))$. As proven in [10], such image resampling is differentiable with respect to $\mathcal{T}$ and $\mathbf{I}$.

3.2. Saliency-Guided Downsampling

When using nonuniform downsampling for information retention, it is useful to parameterize $\mathcal{T}$ with a saliency map $\mathbf{S}(\mathbf{x})$ representing the desired sample rate at each spatial location $\mathbf{x} \in [0,1]^2$ [20]. Recasens *et al.* [20] go on to approximate this behavior by having each sample coordinate $\mathcal{T}(\mathbf{x})$ be "attracted" to nearby areas $\mathbf{x}'$ with high saliency $\mathbf{S}(\mathbf{x}')$ downweighted according to a distance kernel $k(\mathbf{x},\mathbf{x}')$, as illustrated in Figure 2. Concretely, $\mathcal{T}_{\mathrm{LZ}}(\mathbf{x}) = (\mathcal{T}_{\mathrm{LZ},x}(\mathbf{x}), \mathcal{T}_{\mathrm{LZ},y}(\mathbf{x}))$, where

$$\mathcal{T}_{\mathrm{LZ},x}(\mathbf{x}) = \frac{\int_{\mathbf{x}'} \mathbf{S}(\mathbf{x}')k(\mathbf{x},\mathbf{x}')\mathbf{x}'_x \, d\mathbf{x}'}{\int_{\mathbf{x}'} \mathbf{S}(\mathbf{x}')k(\mathbf{x},\mathbf{x}') \, d\mathbf{x}'}, \tag{1}$$

$$\mathcal{T}_{\mathrm{LZ},y}(\mathbf{x}) = \frac{\int_{\mathbf{x}'} \mathbf{S}(\mathbf{x}')k(\mathbf{x},\mathbf{x}')\mathbf{x}'_y \, d\mathbf{x}'}{\int_{\mathbf{x}'} \mathbf{S}(\mathbf{x}')k(\mathbf{x},\mathbf{x}') \, d\mathbf{x}'}. \tag{2}$$

[22] proposes *anti-cropping* and *separable* variants of this downsampler. The anti-cropping variant $\mathcal{T}_{\mathrm{LZ,ac}}$ prevents the resampling operation from cropping the image. The separable variant marginalizes the saliency map $\mathbf{S}(\mathbf{x})$ into two 1D saliency maps $\mathbf{S}_x(x)$ and $\mathbf{S}_y(y)$, and replaces the kernel $k(\mathbf{x},\mathbf{x}')$ with a two 1D kernels k_x and k_y (although generally $k_x = k_y$). Then, $\mathcal{T}_{\mathrm{LZ,sep}}(\mathbf{x}) = (\mathcal{T}_{\mathrm{LZ,sep,x}}(\mathbf{x}_x), \mathcal{T}_{\mathrm{LZ,sep,y}}(\mathbf{x}_y))$ where

$$\mathcal{T}_{\mathrm{LZ,sep,x}}(x) = \frac{\int_{x'} \mathbf{S}_x(x')k_x(x,x')x' \, dx'}{\int_{x'} \mathbf{S}_x(x')k_x(x,x') \, dx'}, \tag{3}$$

$$\mathcal{T}_{\mathrm{LZ,sep,y}}(y) = \frac{\int_{y'} \mathbf{S}_y(y')k_y(y,y')y' \, dy'}{\int_{y'} \mathbf{S}_y(y')k_y(y,y') \, dy'}. \tag{4}$$

This preserves axis-alignment of rectangles, which is crucial to object detection where bounding boxes are specified via corners. We refer to the above method and all variants as *LZ downsamplers*, after the pioneering work "Learning to Zoom" [20]. Examples of each variant are shown in Figure 3.

4. Method

We begin by discussing our general technique for warp inversion. Then, we discuss the LZU framework and how we apply warp inversion to efficiently "unzoom".

4.1. Efficient, Differentiable Warp Inversion

Suppose we have a continuous map $\mathcal{T} : [0,1]^2 \to [0,1]^2$. Our primary technical innovation is an efficient and differentiable approximation of $\mathcal{T}^{-1}$, even in cases where $\mathcal{T}$ has no closed-form inverse.

Since $\mathcal{T}$ is potentially difficult to invert, we first approximate it as $\widetilde{\mathcal{T}}$, a piecewise tiling of simpler invertible transforms (illustrated in Figure 4). Formally,

$$\widetilde{\mathcal{T}} = \bigcup_{\substack{i\in[h-1]\\ j\in[w-1]}} \widetilde{\mathcal{T}}_{ij}, \tag{5}$$

where the ij-th tile $\widetilde{\mathcal{T}}_{ij}$ is any bijective map from the rectangle formed by corners $R_{ij} = \{\frac{i-1}{h-1}, \frac{i}{h-1}\} \times \{\frac{j-1}{w-1}, \frac{j}{w-1}\}$

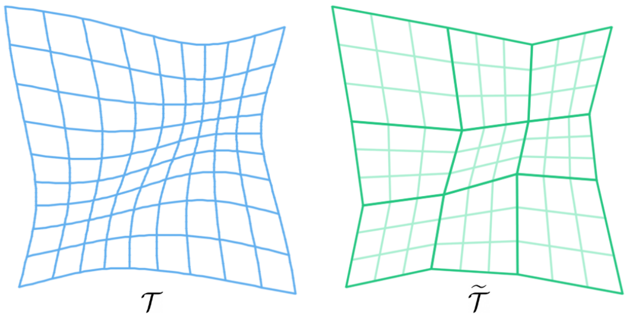

Figure 4. Given a warp $\mathcal{T}$, we construct an approximation $\widetilde{\mathcal{T}}$ designed for efficient inversion. As illustrated, $\widetilde{\mathcal{T}}$ is a piecewise tiling of simpler invertible maps. This allows us to approximate the inverse $\widetilde{\mathcal{T}}^{-1}$, even when $\mathcal{T}^{-1}$ lacks a closed form.

to quadrilateral $\mathcal{T}[R_{ij}]$. For our purposes, we choose bilinear maps as our tile function, although homographies could work just as well. Then, so long as $\widetilde{\mathcal{T}}$ is injective (if each of the tiles $\widetilde{\mathcal{T}}_{ij}$ is nondegenerate and no two tiles overlap), we are guaranteed a well-defined left inverse $\widetilde{\mathcal{T}}^{-1} : [0,1]^2 \to [0,1]^2$ given by

$$\widetilde{\mathcal{T}}^{-1}(\mathbf{x}) = \begin{cases} \widetilde{\mathcal{T}}_{ij}^{-1}(\mathbf{x}) & \text{if } \mathbf{x} \in \text{Range}(\widetilde{\mathcal{T}}_{ij}) \\ 0 & \text{else} \end{cases}. \quad (6)$$

Equation 6 is **efficient** to compute, since determining if $\mathbf{x} \in \text{Range}(\widetilde{\mathcal{T}}_{ij})$ simply involves checking if $\mathbf{x}$ is in the quadrilateral $\mathcal{T}[R_{ij}]$ and computing the inverse $\widetilde{\mathcal{T}}_{ij}^{-1}$ of a bilinear map amounts to solving a quadratic equation [27]. This efficiency is crucial to maintaining favorable accuracy-latency tradeoffs. $\widetilde{\mathcal{T}}^{-1}$ is guaranteed to be **differentiable** with respect to $\mathcal{T}$, since for each $\mathbf{x} \in \widetilde{\mathcal{T}}[R(i,j)]$, the inverse bilinear map can be written as a quadratic function of the four corners of tile ij (see Appendix A.1 for exact expression). This allows gradients to flow back into $\mathcal{T}$, letting us learn the parameters of the warp.

In the case of LZ warps, $\mathcal{T}_{\text{LZ}}$ has no closed form inverse to the best of our knowledge. Because $\mathcal{T}_{\text{LZ}}[\text{Grid}(h,w)]$ has no foldovers [20], $\widetilde{\mathcal{T}}_{\text{LZ}}$ must be injective, implying its inverse $\widetilde{\mathcal{T}}_{\text{LZ}}^{-1}$ is well-defined.

When applying an LZ warp, saliency can be learned (with trainable parameters) or unlearned (with fixed parameters), and fixed (invariant across frames) or adaptive (changes every frame). Adaptive saliency maps require efficient warp inversion since a different warp must be applied on every input. Learned saliency maps require differentiability. We note that fixed unlearned saliency maps do not technically require efficiency or differentiability, and most of our current results show that such saliency maps are already quite effective, outperforming prior work. We posit that LZU would shine even more in the learned adaptive setting, where it could make use of temporal priming for top-down saliency.

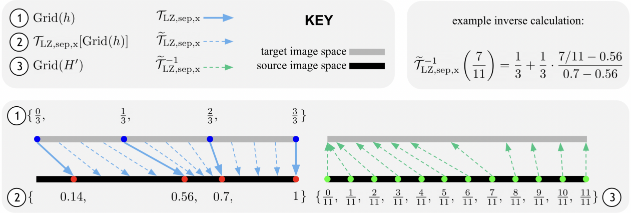
1 Grid(h)
2 $\mathcal{T}_{\text{LZ,sep,x}}[\text{Grid}(h)]$
3 Grid(H')
$\mathcal{T}_{\text{LZ,sep,x}}$
$\widetilde{\mathcal{T}}_{\text{LZ,sep,x}}$
$\widetilde{\mathcal{T}}^{-1}_{\text{LZ,sep,x}}$
KEY
target image space
source image space
example inverse calculation:
$\widetilde{\mathcal{T}}^{-1}_{\text{LZ,sep,x}}\left(\frac{7}{11}\right) = \frac{1}{3} + \frac{1}{3} \cdot \frac{7/11 - 0.56}{0.7 - 0.56}$
1 $\{\frac{0}{3}, \frac{1}{3}, \frac{2}{3}, \frac{3}{3}\}$
2 $\{0.14, 0.56, 0.7, 1\}$
$\{\frac{0}{11}, \frac{1}{11}, \frac{2}{11}, \frac{3}{11}, \frac{4}{11}, \frac{5}{11}, \frac{6}{11}, \frac{7}{11}, \frac{8}{11}, \frac{9}{11}, \frac{10}{11}, \frac{11}{11}\}$ 3

Figure 5. Inverting each axis of a separable warp. LZU first evaluates the forward warp $\mathcal{T}_{\text{LZ,sep,x}}$ (solid blue arrows) at a uniform grid of target locations (blue points). The resulting source locations are shown as red points. LZU then approximates the warp in between these samples via a *linear* transform; this piecewise linear map is $\widetilde{\mathcal{T}}_{\text{LZ,sep,x}}$ (dotted blue arrows). To evaluate the inverse $\widetilde{\mathcal{T}}^{-1}_{\text{LZ,sep,x}}$ (dotted green arrows), we must determine for each green point which red points it falls between and invert the corresponding linear transform. An example is shown in the top-right.

4.2. Learning to Zoom and Unzoom

In the Learning to Zoom and Unzoom (LZU) framework, we use existing LZ downsamplers (see Section 3.2) to "zoom" in on the input image, compute spatial features, and then use our warp inversion formulation to "unzoom" and revert any deformations in the feature map, as shown in Figure 1. This framework is applicable to all tasks with 2D spatial input and all models with some intermediate 2D spatial representation.

Notice that a poorly approximated inverse warp $\widetilde{\mathcal{T}}^{-1}$ would lead to misaligned features and a drop in performance. As a result, we use the approximate forward warp $\widetilde{\mathcal{T}}$ instead of the true forward warp $\mathcal{T}$, so that the composition of forward and inverse warps is *actually* the identity function. See Appendix A.3 for a discussion of the associated tradeoff.

To maintain favorable accuracy-latency tradeoffs, we make several optimizations to our forward and inverse warps. As done in previous works [11, 20, 22], for the forward warp or "zoom," instead of computing $\mathcal{T}_{\text{LZ}}[\text{Grid}(H', W')]$, we compute $\mathcal{T}_{\text{LZ}}[\text{Grid}(h,w)]$ for smaller $h \ll H'$ and $w \ll W'$ and bilinearly upsample this to get $\widetilde{\mathcal{T}}_{\text{LZ}}[\text{Grid}(H', W')]$. This also reduces the complexity of computing the inverse, by reducing the number of cases in our piecewise bilinear map from $H' \cdot W'$ to $h \cdot w$.

We explore efficient implementations of both separable and nonseparable warp inversion, but we find experimentally that nonseparable warps perform no better than separable warps for a strictly higher latency cost, so we use separable warps for our experiments. Details for efficiently inverting nonseparable warps are given in Appendix A.2. For separable warps $\mathcal{T}_{\text{LZ,sep}}$, we invert each axis separately and take the Cartesian Product:

$$\widetilde{\mathcal{T}}^{-1}_{\text{LZ,sep}}[\text{Grid}(H', W')] = \quad (7)$$
$$\widetilde{\mathcal{T}}^{-1}_{\text{LZ,sep,x}}[\text{Grid}(H')] \times \widetilde{\mathcal{T}}^{-1}_{\text{LZ,sep,y}}[\text{Grid}(W')].$$

This further reduces our problem from inverting a piecewise bilinear map with $h \cdot w$ pieces to inverting two piecewise *linear* maps with h and w pieces each. Figure 5 visualizes how to invert each axis.

When unwarping after feature pyramid networks (FPNs) [16], we may have to evaluate the inverse $\widetilde{\mathcal{T}}_{\text{LZ}}^{-1}$ at multiple resolutions $\text{Grid}(H', W')$, $\text{Grid}(H'/2, W'/2)$, etc. In practice, we evaluate $\widetilde{\mathcal{T}}_{\text{LZ}}^{-1}[\text{Grid}(H', W')]$ and then approximate the inverse at lower resolutions via bilinear downsampling. This is surprisingly effective (see Appendix A.3) and leads to no observable loss in performance.

Finally, as introduced in [22], we can also use a fixed warp to exploit dataset-wide spatial priors, such as how objects are concentrated around the horizon in many autonomous driving datasets. This allows us to cache forward and inverse warps, greatly reducing additional latency.

5. Experiments

First, we compare LZU to naive uniform downsampling and previous works on the tasks of 2D object detection and semantic segmentation. We include ablations to evaluate the effectiveness of training techniques and explore the upsampling regime. Then, we evaluate LZU on monocular 3D object detection, a task which no previous works have applied "zooming" to. We perform all timing experiments with a batch size of 1 on a single RTX 2080 Ti GPU. Figure 6 contains qualitative results and analysis across all tasks. Full implementation details and hyperparameters are given in Appendix A.6.

5.1. 2D Object Detection

For 2D object detection, we evaluate LZU using RetinaNet [17] (with a ResNet-50 backbone [9] and FPN [16]) on Argoverse-HD [14], an object detection dataset for autonomous driving with high resolution 1920 × 1200 videos. For our baselines, we compare to uniform downsampling and FOVEA [22], a previous work that applies LZ downsampling to detection by unwarping bounding boxes. We keep the same hyperparameters and setup as in FOVEA [22]. Experiments are run at 0.25x, 0.5x, 0.75x, and 1x scales, to measure the accuracy-latency tradeoff.

Our LZU models "unzoom" the feature map at each level after the FPN [16]. We adopt the low-cost saliency generators introduced in [22] — a "fixed" saliency map exploiting dataset-wide spatial priors, and an "adaptive" saliency map exploiting temporal priors by zooming in on detections from the previous frame. When training the adaptive version, we simulate previous detections by jittering the ground truth for the first two epochs. For the last epoch, we jitter *detections* on the current frame to better simulate previous detections; we call this "cascaded" saliency. To determine saliency hyperparameters, we run grid search at 0.5x scale on splits of the training set (details in Appendix A.4, A.6). We use a learning rate of 0.01 and keep all other training settings identical to the baseline. Latency is measured by timing only the additional operations (the "zoom" and "unzoom") and adding it to the baseline. This is done to mitigate the impact

Scale	Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	Lat (ms)
0.25x	Uniform	10.5	18.0	9.9	0.3	5.2	38.6	**23.3**
0.25x	LZU, fixed	**12.4**	22.6	11.2	1.0	**7.1**	**39.2**	23.6
0.25x	LZU, adaptive	12.3	**22.8**	**11.3**	**1.4**	6.6	38.0	26.4
0.5x	Uniform	22.6	38.7	21.7	3.7	22.1	**53.1**	**36.0**
0.5x	FOVEA [22]	24.9	40.3	**25.3**	**7.1**	**27.7**	50.6	37.9
0.5x	LZU, fixed	25.2	42.1	24.8	5.5	26.7	51.8	36.4
0.5x	LZU, adaptive	**25.3**	**43.0**	24.6	6.1	25.9	52.6	39.3
0.5x	LZU, adaptive w/o cascade sal.	22.8	39.3	22.3	5.1	22.7	48.9	39.3
0.75x	Uniform	29.5	48.4	29.6	9.1	32.4	**55.1**	**62.9**
0.75x	LZU, fixed	**30.8**	**50.4**	**31.8**	**10.9**	**33.5**	54.1	63.5
0.75x	LZU, adaptive	26.5	44.6	26.7	8.3	28.7	48.7	66.3
1x	Uniform	31.9	51.5	33.1	11.4	35.9	54.5	**98.3**
1x	LZU, fixed	**32.6**	**52.8**	**34.0**	**13.2**	36.0	**54.7**	99.3
1x	LZU, adaptive	32.0	52.4	33.1	12.5	**36.3**	52.9	102.0

Table 1. 2D object detection results of RetinaNet [17] on Argoverse-HD [14]. Fixed LZU uses a dataset-wide spatial prior, and adaptive LZU uses a temporal prior based on previous frame detections. LZU consistently outperforms the uniform downsampling baseline and prior work across all scales, with additional latency less than 4ms. We hypothesize that the drop in AP_L is because objects that are already large benefit less from zooming. Still, this drawback is offset by larger improvements on small and medium objects.

2D Object Detection

Uniform Resampling					LZU Resampling				
	From					From			
To	0.25x	0.5x	0.75x	1x	To	0.25x	0.5x	0.75x	1x
0.25x	10.5	10.5	10.5	10.5	0.25x	**11.7**	**12.4**	**12.4**	**12.4**
0.5x	17.0	22.6	22.6	22.6	0.5x	**20.9**	**24.8**	**24.8**	**25.2**
0.75x	**23.5**	28.5	29.5	29.5	0.75x	22.5	**29.4**	**30.0**	**30.8**
1x	13.5	28.4	30.9	31.9	1x	**22.1**	**30.7**	**31.2**	**32.6**

Monocular 3D Object Detection

Uniform Resampling					LZU Resampling				
	From					From			
To	0.25x	0.5x	0.75x	1x	To	0.25x	0.5x	0.75x	1x
0.25x	21.8	21.8	21.8	21.8	0.25x	**22.5**	**23.5**	**23.4**	**23.4**
0.5x	25.4	27.5	27.5	27.5	0.5x	**27.0**	**29.2**	**29.1**	**29.3**
0.75x	27.6	30.3	30.5	30.5	0.75x	**29.0**	**31.6**	**31.6**	**31.8**
1x	28.4	30.7	31.1	31.2	1x	**30.1**	**32.5**	**32.7**	**32.6**

Table 2. 2D and 3D object detection results in the upsampling and downsampling regimes, using the "Uniform" and "LZU, fixed" models from Tables 1 and 5. LZU is surprisingly effective even in the upsampling regime! This demonstrates that simply allocating more pixels to small objects (without retaining extra information) can help performance, suggesting that detectors still struggle with scale invariance for small objects.

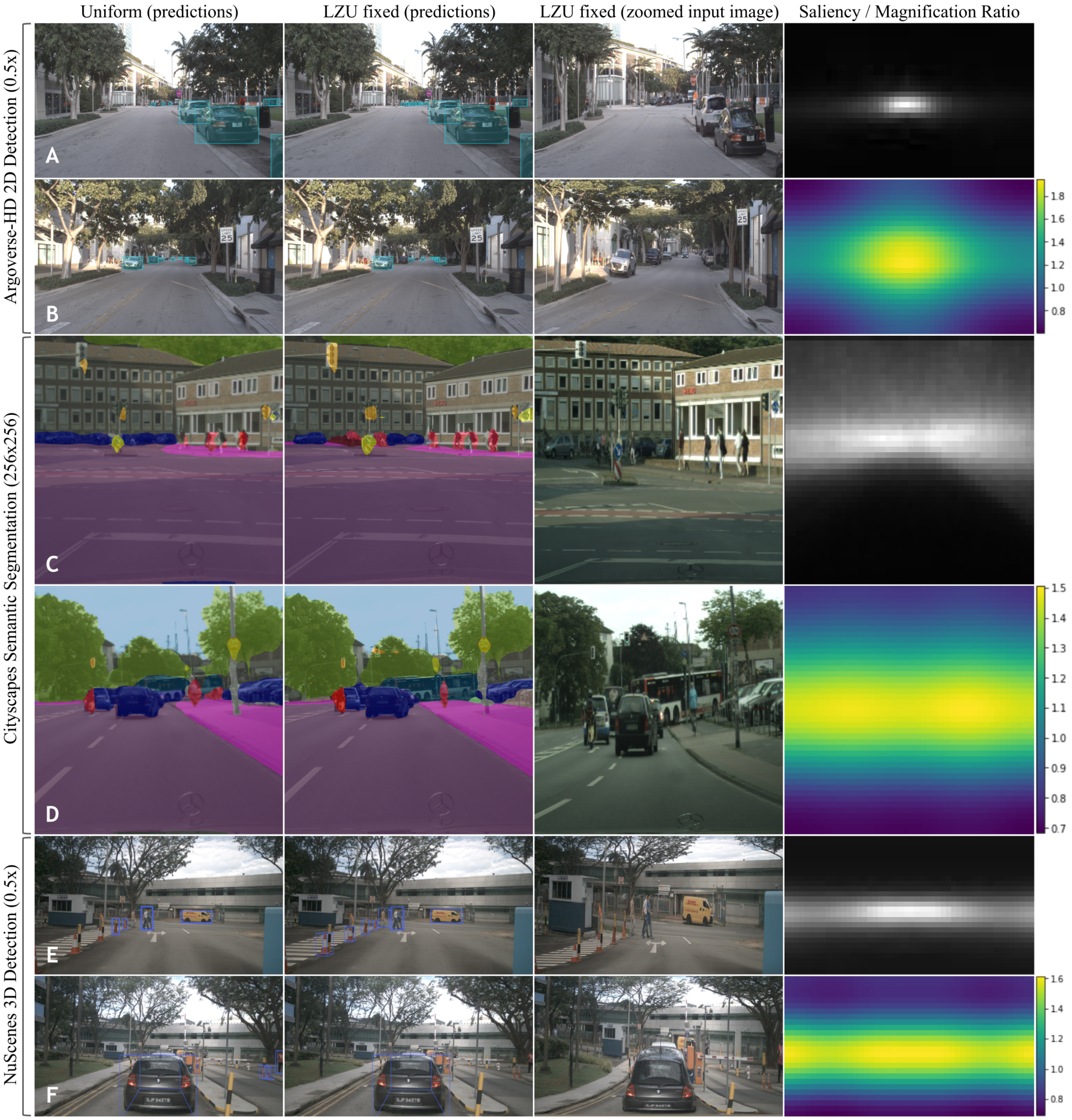
Uniform (predictions)
LZU fixed (predictions)
LZU fixed (zoomed input image)
Saliency / Magnification Ratio
Argoverse-HD 2D Detection (0.5x)
Cityscapes Semantic Segmentation (256x256)
NuScenes 3D Detection (0.5x)
A
B
C
D
E
F
1.8
1.6
1.4
1.2
1.0
0.8
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
1.6
1.4
1.2
1.0
0.8

Figure 6. Examples of the success and failure cases of LZU. Rows A and E show examples where zooming in on the horizon helps the detector pick up smaller objects. On the other hand, sometimes zooming leads to false negatives, such as the black car in Row B and objects near the edge in Row F. For segmentation, LZU consistently improves quality near the center of the image. The last column shows the saliency map used in each case and the resulting spatial magnification ratios. For the Argoverse-HD [14] dataset, the magnification ratio at the center is nearly 2x, meaning the "zoom" is preserving nearly all information in that region, at the cost of information at the corners.

of variance in the latency of the backbone and detector head.

Results are given in Table 1. We outperform both uniform downsampling and FOVEA in all but one case, while incurring an additional latency of less than 4ms. The one exception is adaptive LZU at 0.75x, which is evidence that our adaptive saliency hyperparameters, chosen at 0.5x scale, struggle to generalize to other resolutions. We also confirm that using cascaded saliency to train adaptive LZU is crucial. Although adaptive LZU outperforms fixed LZU at 0.5x scale, plotting the accuracy-latency curves (Figure 7) reveals that fixed LZU is Pareto optimal at all points.

Finally, we explore how LZU performs in the *upsampling*

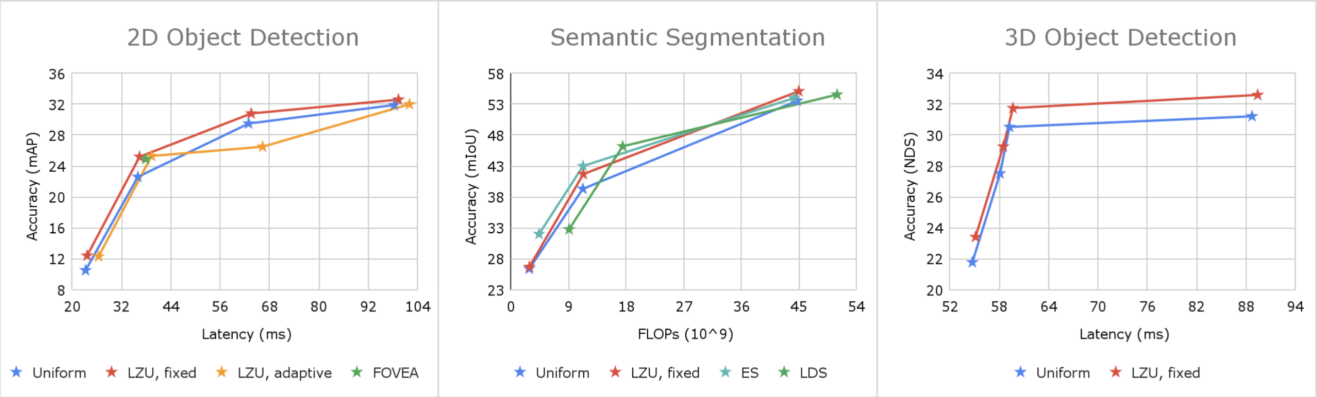

Figure 7. Plotting the accuracy-latency/FLOPs tradeoffs reveals the Pareto optimal methods for each task. Fixed LZU is Pareto optimal for both 2D and 3D object detection, outperforming uniform downsampling and FOVEA [22]. For semantic segmentation, we use FLOPs in lieu of latency to enable fair comparisons (ES [19] only reports FLOPS and LDS [11] has an unoptimized implementation). Although LDS boasts large improvements in raw accuracy at each scale, it also incurs a greater cost due to its expensive saliency generator. Overall, the Pareto frontier for segmentation is very competitive, with ES dominating at 64 × 64, LDS at 128 × 128, and LZU at 256 × 256.

Crop			IOU																			Latency
Size	Method	mIOU	road	swalk	build.	wall	fence	pole	tlight	sign	veg.	terr.	sky	person	rider	car	truck	bus	train	mbike	bike	(ms)
64	Uniform	26.4	**93.9**	35.6	68.6	3.5	**2.9**	**0.5**	**0.0**	**0.1**	72.5	21.1	**76.0**	26.8	0.9	**57.1**	8.9	**16.9**	**8.0**	**0.0**	8.2	**15.5**
64	LZU, fixed	**26.7**	93.4	**36.1**	**68.9**	**5.8**	2.3	0.4	**0.0**	0.0	**72.6**	**23.4**	75.9	**29.4**	**1.2**	56.7	**15.0**	10.2	4.1	**0.0**	**11.7**	16.9
128	Uniform	39.3	96.3	54.0	78.4	**15.0**	7.9	**8.1**	8.5	16.6	81.2	34.4	**86.7**	42.9	13.8	74.4	**22.9**	41.6	24.4	10.2	29.6	**16.1**
128	LZU, fixed	**41.7**	**96.4**	**55.2**	**78.7**	12.7	**13.4**	**8.1**	**11.4**	**19.0**	**81.7**	**39.0**	86.5	**45.7**	**17.9**	**76.8**	21.9	**48.2**	**31.7**	**11.6**	**36.3**	18.0
256	Uniform	53.6	97.5	64.0	84.7	20.0	19.0	**22.1**	34.8	41.6	**87.0**	41.9	**91.2**	59.3	33.7	84.1	39.2	62.9	**57.9**	27.7	49.1	**19.1**
256	LZU, fixed	**55.1**	**97.7**	**67.0**	**84.9**	**24.4**	**24.4**	21.3	**35.2**	**42.9**	**87.0**	**44.5**	90.7	**61.5**	**35.7**	**85.7**	**40.8**	**67.9**	52.8	**29.3**	**53.4**	21.2
512	Uniform	63.8	**98.3**	73.3	**88.8**	29.2	34.3	**40.6**	54.4	61.6	**90.7**	**47.7**	**94.0**	**72.7**	**50.6**	89.1	45.6	72.1	59.1	44.5	64.9	**32.3**
512	LZU, fixed	**64.2**	**98.3**	**73.4**	88.6	**30.0**	**35.7**	38.8	**56.0**	**63.8**	90.4	47.0	93.4	72.4	43.9	**90.1**	**50.5**	**76.4**	**59.6**	**45.4**	**65.3**	34.4

Table 3. Full semantic segmentation results of PSPNet [29] on Cityscapes [7]. At each resolution, LZU outperforms uniform downsampling.

regime. We reuse the same models trained in our previous experiments, testing them with different pre-resampling resolutions. Results are shown in Table 2. In this regime, LZU consistently outperforms uniform downsampling, even though information retention is no longer a factor.

5.2. Semantic Segmentation

For our semantic segmentation experiments, we compare to previous works ES [19] and LDS [11], so we adopt their setup. We test the PSPNet [29] model (with a ResNet-50 backbone [9] and FPN [16]) on Cityscapes [7]. Cityscapes is an urban scene dataset with high resolution 1024 × 2048 images and 19 classes. We perform our experiments at several image scales (64×64, 128×128, 256×256, and 512×512), taken by resizing a centered square crop of the input image. Our simple baseline trains and tests PSPNet with uniform downsampling. To reduce overfitting, we allot 500 images from the official training set into a mini-validation split. We train our model on the remaining training images and evaluate at 10 equally spaced intervals on the mini-validation split. We choose the best performing model and evaluate it on the official validation set.

For our LZU model, we unzoom spatial features after the FPN and use a fixed saliency map. Inspired by the idea of zooming on semantic boundaries [19], we generate our fixed saliency by averaging the ground truth semantic boundaries over the train set. Notably, our saliency hyperparameters are chosen qualitatively (for producing a reasonably strong warp) and tested one-shot.

We report our full results in Table 3 and compare to previous works in Table 4. Since our baseline results are slightly different than reported in previous works [11, 19], we compare results using a percent change relative to the corresponding baseline. We find increased performance over the baseline at all scales, and at 256 × 256, we beat both previous works with only 2.3ms of additional latency. Plotting the accuracy-FLOPs tradeoff (Figure 7) reveals that the large improvements of LDS [11] at 64×64 and 128×128 input scales come at significant cost in FLOPs. In actuality, ES [19] is Pareto optimal at 64×64 and 128×128, LDS [11]

Method	Downsampled Resolution 64 × 64	 128 × 128	 256 × 256
Uniform (theirs)	29	40	54
Uniform (ours)	26.4	39.3	53.6
ES [19]	32 (+10.3%)	43 (+7.5%)	54 (+0.0%)
LDS [11]	36 (**+24.1%**)	47 (**+17.5%**)	55 (+1.9%)
LZU, fixed	26.7 (+1.1%)	41.7 (+6.1%)	55.1 (**+2.9%**)

Table 4. Semantic segmentation results of PSPNet [29] on Cityscapes [7], in mIOU. Due to differing implementation, the performance of our baseline varies from reported values, so we report relative improvements. At 256 × 256, we outperform prior works. At 64 × 64 and 128 × 128, LZU performs worse than prior work, perhaps because "unzooming" features at such small scales is more destructive. We posit the performance losses from such aggressive downsampling factors (across all methods) may be too impractical for deployment, and so focus on the 256 × 256 regime.

Scale	Method	NDS	mAP	mATE	mASE	mAOE	mAVE	mAAE	Lat (ms)
0.25x	Uniform	21.8	11.4	**96.7**	32.6	90.1	**125.0**	**19.8**	**54.7**
0.25x	LZU, fixed	**23.4**	**13.1**	96.8	**31.9**	**82.7**	129.4	20.0	55.1
0.5x	Uniform	27.5	17.5	90.1	28.8	75.5	131.6	17.8	**58.1**
0.5x	LZU, fixed	**29.3**	**20.1**	**88.9**	**28.3**	**73.9**	**130.6**	**16.7**	58.5
0.75x	Uniform	30.5	21.0	87.3	27.9	**67.0**	**132.8**	17.5	**59.2**
0.75x	LZU, fixed	**31.8**	**22.4**	**83.8**	**27.5**	67.2	134.6	**15.9**	59.7
1x	Uniform	31.2	22.4	**84.2**	**27.4**	70.9	**129.6**	**17.4**	**88.7**
1x	LZU, fixed	**32.6**	**24.8**	84.6	27.5	**68.2**	131.6	18.3	89.4

Table 5. 3D object detection results of FCOS3D [26] on nuScenes [2]. Higher NDS and mAP is better, and lower is better on all other metrics. Intuitively, size is an important cue for depth, and image deformations would stifle this signal. Suprisingly, this is *not* the case. LZU improves upon the uniform downsampling baseline at all scales with less than 1ms of additional latency. Notably, LZU at 0.75x scale even outperforms uniform downsampling at 1x.

at 128 × 128, and LZU at 256 × 256. We hypothesize that further improvements might be possible using an adaptive, learned formulation for saliency.

5.3. Monocular 3D Object Detection

Finally, we evaluate LZU on monocular 3D object detection. To the best of our knowledge, no previous work has applied LZ downsampling to this task. The closest existing solution, FOVEA [22], cannot be extended to 3D object detection, because 3D bounding boxes are amodal and cannot be unwarped in the same manner as 2D bounding boxes. For our base model, we use FCOS3D [26], a fully convolutional model, with a ResNet-50 backbone [9] and FPN [16]. For our dataset, we use nuScenes [2], an autonomous driving dataset with multi-view 1600 × 900 RGB images for 1000 scenes and 3D bounding box annotations for 10 object classes. As is standard practice, we use the nuScenes Detection Score (NDS) metric, which is a combination of the usual mAP and measures of translation error (mATE), scale error (mASE), orientation error (mAOE), velocity error (mAVE), and attribute error (mAAE). We run experiments at 0.25x, 0.5x, 0.75x, and 1x scales and test against a uniform downsampling baseline. We train for 12 epochs with a batch size of 16 with default parameters as in MMDetection3D [5].

For our LZU model, again we unzoom post-FPN features and use a fixed saliency map. Inspired by FOVEA [22], our fixed saliency is generated by using kernel density estimation on the set of projected bounding boxes in the image space. We reuse the same saliency hyperparameters from 2D detection. All other training settings are identical to the baseline.

Results are given in Table 5. LZU performs consistently better than uniform downsampling, with less than 1ms of additional latency. Specifically, LZU improves mAP and the aggregate metric NDS, with mixed results on mATE, mASE, mAOE, mAVE, and mAAE. Since the latter five metrics are computed on only *true positives*, this demonstrates that LZU increases overall recall, while maintaining about equal performance on true positives. Plotting the accuracy-latency curves (Figure 7) shows that LZU is Pareto optimal. We also repeat the same upsampling experiments as performed in 2D object detection. Results, shown in Table 2, reaffirm the viability of LZU in the upsampling regime.

6. Conclusion

We propose LZU, a simple attentional framework consisting of "zooming" in on the input image, computing spatial features, and "unzooming" to invert any deformations. To unzoom, we approximate the forward warp as a piecewise bilinear mapping and invert each piece. LZU is highly general and can be applied to any task with 2D spatial input and any model with 2D spatial features. We demonstrate the versatility of LZU empirically on a variety of tasks and datasets, including monocular 3D detection which has never been done before. We also show that LZU may even be used when high-resolution sensor data is unavailable. For future work, we can consider alternatives to the "unzoom" formulation that are perhaps less destructive than simple resampling of features.

Broader impact. Our work focuses on increasing the efficiency and accuracy of flagship vision tasks (detection, segmentation, 3D understanding) with high-resolution imagery. We share the same potential harms of the underlying tasks, but our approach may increase privacy concerns as identifiable information may be easier to decode at higher resolutions (e.g., facial identities or license plates). Because our approach is agnostic to the underlying model, it is reproducible with minimal changes to existing codebases.

Acknowledgements: This work was supported by the CMU Argo AI Center for Autonomous Vehicle Research.

References

[1] Thaddeus Beier and Shawn Neely. Feature-based image metamorphosis. *ACM SIGGRAPH computer graphics*, 26(2):35–42, 1992. 3

[2] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 2, 8, 14

[3] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 13

[4] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017. 1

[5] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. `https://github.com/open-mmlab/mmdetection3d`, 2020. 8, 13, 14

[6] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. `https://github.com/open-mmlab/mmsegmentation`, 2020. 13

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016. 2, 7, 8, 12, 13

[8] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 2

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5, 7, 8

[10] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015. 2, 3

[11] Chen Jin, Ryutaro Tanno, Thomy Mertzanidou, Eleftheria Panagiotaki, and Daniel C Alexander. Learning to downsample for segmentation of ultra-high resolution images. *arXiv preprint arXiv:2109.11071*, 2021. 1, 2, 3, 4, 7, 8

[12] Shu Kong and Charless Fowlkes. Pixel-wise attentional gating for scene parsing. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1024–1033. IEEE, 2019. 2

[13] Jun Li, Yongjun Chen, Lei Cai, Ian Davidson, and Shuiwang Ji. Dense transformer networks. *arXiv preprint arXiv:1705.08881*, 2017. 2

[14] Mengtian Li, Yu-Xiong Wang, and Deva Ramanan. Towards streaming perception. In *European Conference on Computer Vision*, pages 473–488. Springer, 2020. 2, 5, 6, 11, 13

[15] Mengtian Li, Ersin Yumer, and Deva Ramanan. Budgeted training: Rethinking deep neural network training under resource constraints. *arXiv preprint arXiv:1905.04753*, 2019. 13

[16] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 5, 7, 8, 12

[17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 2, 5

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2

[19] Dmitrii Marin, Zijian He, Peter Vajda, Priyam Chatterjee, Sam Tsai, Fei Yang, and Yuri Boykov. Efficient segmentation: Learning downsampling near semantic boundaries. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2131–2141, 2019. 2, 7, 8

[20] Adria Recasens, Petr Kellnhofer, Simon Stent, Wojciech Matusik, and Antonio Torralba. Learning to zoom: a saliency-based sampling layer for neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 51–66, 2018. 1, 2, 3, 4

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 2

[22] Chittesh Thavamani, Mengtian Li, Nicolas Cebron, and Deva Ramanan. Fovea: Foveated image magnification for autonomous navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15539–15548, 2021. 1, 2, 3, 4, 5, 7, 8, 13

[23] Vincent Vanhoucke, Andrew Senior, and Mark Z. Mao. Improving the speed of neural networks on cpus. In *Deep Learning and Unsupervised Feature Learning Workshop, NIPS 2011*, 2011. 1

[24] Thomas Verelst and Tinne Tuytelaars. Dynamic convolutions: Exploiting spatial sparsity for faster inference. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 2320–2329, 2020. 2

[25] Thomas Verelst and Tinne Tuytelaars. Segblocks: Block-based dynamic resolution networks for real-time segmentation. *arXiv preprint arXiv:2011.12025*, 2020. 2

[26] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Fcos3d: Fully convolutional one-stage monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 913–922, 2021. 2, 8

[27] George Wolberg. *Digital image warping*, volume 10662. IEEE computer society press Los Alamitos, CA, 1990. 4, 11

[28] Zhenda Xie, Zheng Zhang, Xizhou Zhu, Gao Huang, and Stephen Lin. Spatially adaptive inference with stochastic feature sampling and interpolation. In *European conference on computer vision*, pages 531–548. Springer, 2020. 2

[29] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017. 2, 7, 8

[30] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9308–9316, 2019. 2