Improve Model Inference Cost with Image Gridding

Shreyas Krishnaswamy [1] Lisa Dunlap [1] Lingjiao Chen [2] Matei Zaharia [2] James Zou [2] Joseph E. Gonzalez [1]

Abstract

The success of AI has spurred the rise of Machine Learning as a Service (MLaaS), where companies develop, maintain, and serve general-purpose models such as object detectors and image classifiers for users that pay a fixed rate per inference. As more organizations rely on AI, the MLaaS market is set to expand, necessitating cost optimization for these services. We explore how a simple yet effective method of increasing model efficiency, aggregating multiple images into a grid before inference, can significantly reduce the required number of inferences for processing a batch of images with varying drops in accuracy. Experiments on open-source and commercial models show that image gridding reduces inferences by 50%, while maintaining low impact on mean average precision (mAP) over the Pascal VOC object detection task.

1. Introduction

Recently, large models such as DINO and ChatGPT have shown remarkable progress on a wide range of tasks in computer vision and natural language processing. However, developing, training, and serving these models requires significant up-front investment in hardware, engineering, and data. As a consequence, large organizations have taken a lead in developing these technologies and are increasingly offering Inference-as-a-Service to provide smaller organizations access to these large-scale pre-trained models.

For inference services to be cost-effective on hardware accelerators, requests must be processed in batches. Batching inputs improves arithmetic intensity (Williams et al., 2009) and increases hardware throughput (Crankshaw et al., 2017). However, input-batching also means that inference services must transform their inputs into fixed-size structures that can be batched together, effectively limiting input size and, therefore, resolution. This presents a unique opportunity for inference service users to reduce their number of prediction requests – and as a consequence their costs – by cleverly packaging multiple inputs into a single request.

[1]University of California, Berkeley [2]Stanford University. Correspondence to: Shreyas Krishnaswamy <shrekris@berkeley.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

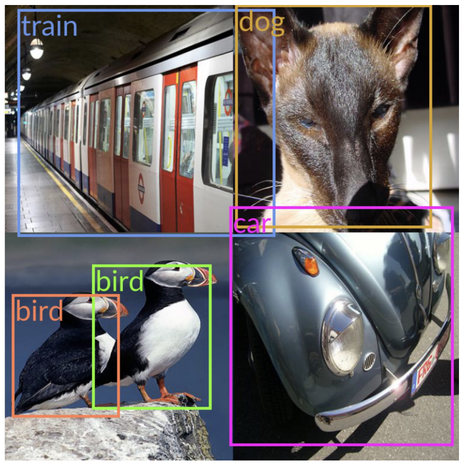

Figure 1: Object detection on a 2x2 image grid (Everingham et al., 2010) using Faster R-CNN. Compressing multiple images in one query substantially reduces the number of queries and thus the query cost.

To reduce inference cost, this paper introduces *image gridding*. Contrary to traditional batching, which stacks images in a tensor for computational efficiency, image gridding spatially arranges images into a grid and submits the grid for inference. For example, Figure 1 shows 4 images packed together into a single image grid. The resulting bounding boxes and labels were detected via a single inference.

Image gridding is particularly effective for many common and simple computer vision tasks. For complex tasks that require high accuracy, such as semantic segmentation, high-res images generally produce better results since they provide more information to the model. However, many common tasks (such as detecting if a person is in a frame) can tolerate low-res inputs without significant accuracy loss. For these tasks, sending high-res inputs wastes compute since (1) they likely get downscaled for batching and (2) they don't need high resolution. Instead, service users can downscale multiple images themselves and pack them into one larger image to send as a single prediction request.

The primary tradeoff of image gridding is balancing effi-

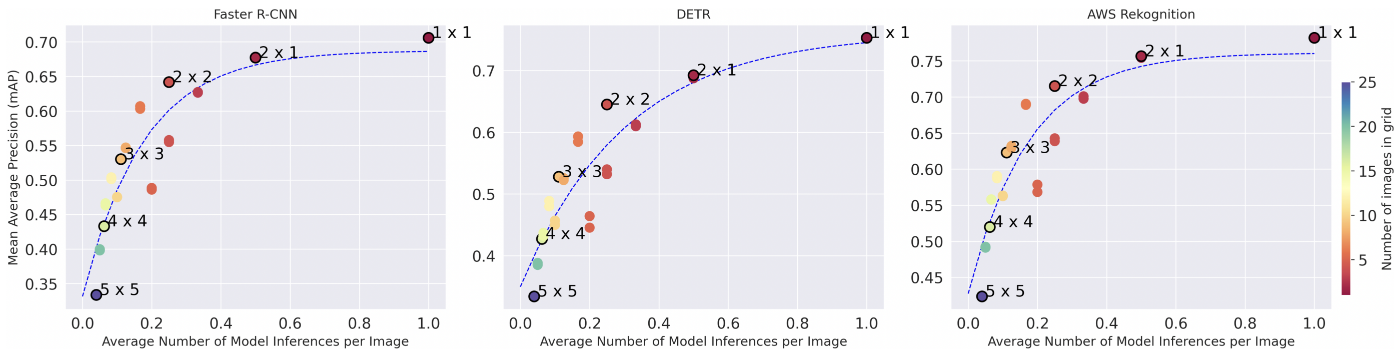

Figure 2: Scatterplot of image grid mAP@0.5 scores on Pascal (Everingham et al., 2010) using different models. Each dot is a grid configuration used to profile the dataset. Sample configurations are labeled. Logistic trend lines are overlaid.

ciency against accuracy. More images per grid means lower model accuracy as more objects are missed or misclassified. Fortunately, large cost savings can still be achieved with little drop in accuracy. Across open source and commercial models, we observe that gridding can reduce inferences by 50% for object detection tasks on the Pascal VOC datasets with as little as 0.028 reduction in mAP.

2. Image Gridding

MLaaS providers (gcp)(aws)(azu) typically charge per inference per task requested. For example, suppose a user requests face detection and object detection for a single image. The MLaaS provider charges once for face detection and once for object detection. Note that they typically do not charge for the *number* of faces or objects in the image. No matter the number detected, the user pays the same amount. This aligns with the input costs for running ML models: energy costs and environmental effects scale with the number of inferences, not the number of detections.

Method. We sketch the image gridding process for object detection, with extensions for other tasks in the appendix.

Given a set of images, each image must be scaled to fixed size $h_{img} \times w_{img}$ and then combined into a grid. The grid must be submitted to the model. Object detection models return bounding boxes and labels. Since they treat the overall grid as one image, box coordinates are relative to the grid, not an individual image. Each box must be mapped to its image by converting the grid-based coordinates to image coordinates. Given a box's grid-based coordinates $(x_{g,min}, y_{g,min}, x_{g,max}, y_{g,max})$ and image size $h_{img} \times w_{img}$:

$$I_r = \left\lfloor \frac{y_{g,min} + y_{g,max}}{2 \cdot h_{img}} \right\rfloor \qquad I_c = \left\lfloor \frac{x_{g,min} + x_{g,max}}{2 \cdot w_{img}} \right\rfloor$$

$$x_{img} = x_g - (I_c \cdot w_{img}) \qquad y_{img} = y_g - (I_r \cdot h_{img})$$

I_r and I_c denote the image row and column index, respectively, and x_{img} and y_{img} are the image coordinates for the bounding box. This formulation maps each bounding box to the image containing its midpoint.

3. Results

We profiled image gridding performance on Pascal VOC 2012 (Everingham et al., 2010) across three models: DETR (Carion et al., 2020), Faster-RCNN (Ren et al., 2015) with ResNet 50, and AWS Rekognition (aws). We used open source implementations of Faster-RCNN (rcn) and DETR (det). AWS Rekognition (aws) offers a closed source vision model that charges per inference.

The scatterplots in Figure 2 show the cost-accuracy tradeoffs across different grid settings. Each dot is a rectangular grid configuration used to profile the dataset. The 1x1 point (where each grid contains a single image) is the baseline. The 2x1 point (where two images are stacked vertically) enables the user to spend 0.5 inferences per image– and reduce inference cost by 50%– while maintaining relatively high mAP. mAP falls more sharply starting at 2x2 grids and drops extensively for higher grid sizes, such as a 5x5. Intuitively, this makes sense since the downscaling imposed on high grid sizes results in missed or misclassified objects. Numeric results can be found in the appendix.

4. Conclusion

In this work, we introduce image gridding, a technique to pack multiple images into a single inference for tasks such as object detection. This improves model efficiency and reduces the cost of commercial vision models offered as a service. Empirical results demonstrate that image gridding can achieve significant cost savings with low accuracy loss. Interesting areas of future work include extending image gridding to more vision tasks, exploring algorithms to mitigate the accuracy loss, and mirroring this approach to ML problems outside vision such as NLP.

References

Aws rekognition. https://aws.amazon.com/rekognition/. Accessed: 2023-05-10.

Azure cognitive service for vision. https://azure.microsoft.com/en-us/products/cognitive-services/vision-services. Accessed: 2023-05-10.

Detr huggingface documentation. https://huggingface.co/docs/transformers/model_doc/detr. Accessed: 2023-05-10.

Google cloud vision api. https://cloud.google.com/vision. Accessed: 2023-05-10.

Pytorch faster r-cnn models. https://pytorch.org/vision/0.13/models.html. Accessed: 2023-05-10.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. End-to-end object detection with transformers. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I*, pp. 213–229, Berlin, Heidelberg, 2020. Springer-Verlag. ISBN 978-3-030-58451-1. doi: 10.1007/978-3-030-58452-8_13. URL https://doi.org/10.1007/978-3-030-58452-8_13.

Crankshaw, D., Wang, X., Zhou, G., Franklin, M. J., Gonzalez, J. E., and Stoica, I. Clipper: A Low-Latency online prediction serving system. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, pp. 613–627, Boston, MA, March 2017. USENIX Association. ISBN 978-1-931971-37-9. URL https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/crankshaw.

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88 (2):303–338, June 2010.

Ren, S., He, K., Girshick, R. B., and Sun, J. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL http://arxiv.org/abs/1506.01497.

Williams, S., Waterman, A., and Patterson, D. Roofline: An insightful visual performance model for multi-core architectures. *Commun. ACM*, 52(4):65–76, apr 2009. ISSN 0001-0782. doi: 10.1145/1498765.1498785. URL https://doi.org/10.1145/1498765.1498785.

A. Image Gridding for Other Tasks

In this section, we sketch the image gridding method for two more tasks: classification and safe search.

Classification: Classification can be performed by running an object detection model on the grid, assigning each bounding box to an image, and aggregating the bounding boxes' labels for each image. Generally, object detection inferences are more expensive than simple image classification; however, the gains from image gridding may outweigh the added cost, depending on the particular model and dataset.

Safe Search: Safe search models provide a likelihood that the image is unsafe. Since pure object detection models are rarely trained to detect unsafe images, users can adopt a divide-and-conquer strategy. If an image grid is determined unsafe, it likely contains at least one unsafe image. The grid can then be divided into two subgrids, and each subgrid can be rerun. This recurses until the specific unsafe images are found. If the number of unsafe images in the dataset is relatively small, this approach can save inferences by quickly accepting entire groups of images that lack unsafe material.

B. Scatterplot Tables

This section contains data tables for the scatterplots in Figure 2.

Faster R-CNN

Grid Size	Inferences per Image	% Inf. Saved	mAP	Δ mAP
1x1	1	0%	0.706	0
1x2	0.5	50.0%	0.678	-0.028
1x3	0.33	66.7%	0.626	-0.080
1x4	0.25	75.0%	0.555	-0.151
1x5	0.2	80.0%	0.486	-0.220
2x1	0.5	50.0%	0.677	-0.029
2x2	0.25	25.0%	0.642	-0.064
2x3	0.17	83.3%	0.607	-0.099
2x4	0.13	87.5%	0.547	-0.159
2x5	0.1	90.0%	0.475	-0.231
3x1	0.33	66.7%	0.627	-0.079
3x2	0.17	83.3%	0.603	-0.103
3x3	0.11	88.9%	0.530	-0.176
3x4	0.08	91.7%	0.504	-0.202
3x5	0.07	93.3%	0.466	-0.240
4x1	0.25	75.0%	0.558	-0.148
4x2	0.13	87.5%	0.546	-0.160
4x3	0.08	91.7%	0.501	-0.205
4x4	0.06	93.8%	0.433	-0.273
4x5	0.05	95.0%	0.400	-0.306
5x1	0.2	80.0%	0.488	-0.488
5x2	0.1	90.0%	0.475	-0.231
5x3	0.07	93.3%	0.463	-0.243
5x4	0.05	95.0%	0.398	-0.308
5x5	0.04	96.0%	0.334	-0.372

The Δ mAP values are the change in mAP compared to the 1x1.

DETR

Grid Size	Inferences per Image	% Inf. Saved	mAP	Δ mAP
1x1	1	0%	0.753	0
1x2	0.5	50.0%	0.687	-0.066
1x3	0.33	66.7%	0.613	-0.14
1x4	0.25	75.0%	0.540	-0.213
1x5	0.2	80.0%	0.464	-0.289
2x1	0.5	50.0%	0.692	-0.061
2x2	0.25	25.0%	0.645	-0.108
2x3	0.17	83.3%	0.593	-0.16
2x4	0.13	87.5%	0.526	-0.227
2x5	0.1	90.0%	0.457	-0.296
3x1	0.33	66.7%	0.610	-0.143
3x2	0.17	83.3%	0.584	-0.169
3x3	0.11	88.9%	0.528	-0.225
3x4	0.08	91.7%	0.489	-0.264
3x5	0.07	93.3%	0.437	-0.316
4x1	0.25	75.0%	0.532	-0.221
4x2	0.13	87.5%	0.522	-0.231
4x3	0.08	91.7%	0.480	-0.273
4x4	0.06	93.8%	0.427	-0.326
4x5	0.05	95.0%	0.388	-0.365
5x1	0.2	80.0%	0.445	-0.308
5x2	0.1	90.0%	0.450	-0.303
5x3	0.07	93.3%	0.432	-0.321
5x4	0.05	95.0%	0.385	-0.368
5x5	0.04	96.0%	0.334	-0.419

The Δ mAP values are the change in mAP compared to the 1x1.

AWS Rekognition

Grid Size	Inferences per Image	% Inf. Saved	mAP	Δ mAP
1x1	1	0%	0.782	0
1x2	0.5	50.0%	0.754	-0.028
1x3	0.33	66.7%	0.698	-0.084
1x4	0.25	75.0%	0.639	-0.143
1x5	0.2	80.0%	0.568	-0.214
2x1	0.5	50.0%	0.756	-0.026
2x2	0.25	25.0%	0.715	-0.067
2x3	0.17	83.3%	0.690	-0.092
2x4	0.13	87.5%	0.632	-0.15
2x5	0.1	90.0%	0.562	-0.22
3x1	0.33	66.7%	0.701	-0.081
3x2	0.17	83.3%	0.689	-0.093
3x3	0.11	88.9%	0.623	-0.159
3x4	0.08	91.7%	0.590	-0.192
3x5	0.07	93.3%	0.558	-0.224
4x1	0.25	75.0%	0.643	-0.139
4x2	0.13	87.5%	0.629	-0.153
4x3	0.08	91.7%	0.587	-0.195
4x4	0.06	93.8%	0.520	-0.262
4x5	0.05	95.0%	0.492	-0.29
5x1	0.2	80.0%	0.578	-0.204
5x2	0.1	90.0%	0.563	-0.219
5x3	0.07	93.3%	0.558	-0.224
5x4	0.05	95.0%	0.491	-0.291
5x5	0.04	96.0%	0.423	-0.359

The Δ mAP values are the change in mAP compared to the 1x1.