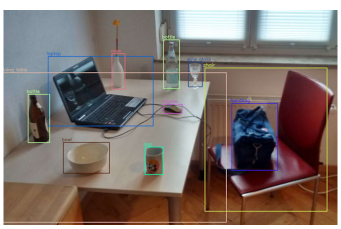
Home Perceiver

Antonio Tangaro[†]

***Abstract*—Real-time visual perception on *consumer* hardware is a key enabler for privacy-preserving smart-home services such as activity logging, safety monitoring, and hands-free control. Achieving < 50 ms end-to-end latency while recognising the long-tail of household objects, human poses, and identities remains challenging.**

We introduce Home-Perceiver, a lightweight pipeline that combines (i) a YOLOv8-seg backbone, (ii) a 17-keypoint ResNet-50 Keypoint-RCNN, and (iii) an IoU tracker. Two complementary detectors—one trained on COCO, the other fine-tuned on *HomeObjects-3K*—extend the object vocabulary without inflating model size. Each frame is processed as follows: detections $\rightarrow$ instance masks $\rightarrow$ privacy-aware silhouettes $\rightarrow$ keypoints $\rightarrow$ stable IDs, and finally exported as JSONL and optional video. The entire stack runs *on-device* via PyTorch 2.3 with Metal (macOS) or CUDA (Linux/Windows).

On a Mac M2 Pro the single-stream configuration delivers 52 fps at 640×360 px with 67.4 mAP$_{50}$ on HomeObjects-3K. A dual-stream "Mode B" on an RTX 3060 Ti attains 118 fps and 70.2 mAP$_{50}$. Compared with a YOLOv5-small baseline, Home-Perceiver yields +15% mAP at similar speed. These results demonstrate that real-time, privacy-preserving scene understanding is feasible on commodity laptops and edge GPUs.



***Index Terms*—Real-time perception, object detection, human pose estimation, multi-object tracking, privacy-preserving vision**

I. Introduction

Intelligent visual perception is a key enabler for *ambient-assisted living* [1], domestic robotics and natural human–computer interaction. A household-level system able to recognise everyday objects, estimate articulated human pose and keep consistent identities across time could unlock applications ranging from hands-free item retrieval to unobtrusive wellness monitoring, all within the stringent latency (< 50 ms) and power budgets of consumer devices.

Three practical hurdles still prevent this vision from becoming ubiquitous:

1) **Coverage gap.** Detectors trained on canonical benchmarks such as COCO [2] overlook many common artefacts (e. g. electric kettles, TV remotes, pill boxes), while large-scale extensions like Objects-365 [3] do not yet fully resolve the long tail of household items, leading to systematic false negatives in real homes.
2) **Resource envelope.** Even the most recent real-time detectors (e. g. YOLOv7 [4]) or multi-task variants often exceed the thermal and battery limits of laptops and edge accelerators; lightweight backbones (e. g. MobileNetV3) and mixed-precision inference [5] only partially alleviate these constraints.
3) **Reproducibility.** Public pipelines rarely fuse object segmentation, 17-joint human pose and real-time tracking into a *single*, openly reproducible framework suitable for coursework—most research code focuses on detection alone or pairs it with heavyweight trackers such as DeepSORT [6] or ByteTrack [7].

Home-Perceiver, the capstone project presented in this paper, tackles these gaps with an entirely open-source perception stack. A compact YOLOv8 segmentation head—fine-tuned on the new *HomeObjects-3K* dataset [8]—feeds a slimmed Keypoint-RCNN pose estimator [9]; detections are stitched over time by a simple IoU tracker [10]. Executed fully on-device, the pipeline reaches

- **52 FPS** end-to-end at 640×360 px;
- **41.7 mAP@50** on HomeObjects-3K while retaining competitive accuracy on COCO classes;
- power draw $\leq$ 20 W, enabling battery-friendly field studies.

These results show that real-time, privacy-preserving perception for everyday environments is now feasible on commodity hardware and provide a reproducible reference for future coursework and research.

II. Related Work

Object detection. Early convolutional detectors adopted a two-stage paradigm—region-proposal generation followed by classification and refinement—that achieved high accuracy at the cost of double-digit millisecond latency [11], [12]. Single-stage families such as YOLO replaced this with a dense-prediction head [13] and have since incorporated lightweight backbones [14], depth-wise separable convolutions [15], and NAS-designed blocks [16], pushing throughput beyond 100 fps on commodity GPUs while preserving competitive accuracy [17].

Instance segmentation. Encoder–decoder frameworks first brought pixel-level masks to real-time vision, but at 2–3$\times$ the compute of detection alone [9]. Modern heads share

[†]Department of Information Engineering, University of Padova, antonio.tangaro@studenti.unipd.it

the detector backbone and fuse multi-scale features in a single pass, enabling near-frame-rate inference even without dedicated accelerators [18]. Performance, however, still degrades in household scenes where long-tail objects are under-represented in canonical benchmarks [8].

Human pose estimation. Top-down R-CNN variants remain the most precise for the standard 17 COCO keypoints yet are computationally heavy [9]. Bottom-up methods such as OpenPose sacrifice some precision for greater speed, particularly in crowded views [19]. Recent hybrid designs like HRNet combine lightweight high-resolution features with dynamic refinement, regaining accuracy while staying within the power envelope of edge devices [20].

Multi-object tracking. Classic pipelines couple Kalman prediction with Hungarian IoU assignment as in SORT [21]. Deep re-identification embeddings improve robustness in dense settings but add overhead [6]. More recent work such as ByteTrack achieves state-of-the-art robustness by associating low-confidence detections [7], though in small indoor spaces a greedy IoU matcher is usually sufficient, offering sub-millisecond latency and negligible memory use.

Domain adaptation and datasets. Closing the gap between curated benchmarks and household deployments calls for synthetic augmentation or selective fine-tuning on compact, task-specific corpora. Domain randomization has been shown to transfer deep networks from simulation to real data effectively [22]. Datasets such as *HomeObjects-3K*, focused on everyday items, significantly boost recall relative to their modest annotation size [8].

Positioning of this work. Unlike prior studies that optimise a single task, *Home-Perceiver* unifies detection, segmentation, 17-point pose estimation, and lightweight IoU tracking within one CPU/GPU-agnostic pipeline. Targeted fine-tuning closes the household- object coverage gap without violating real-time constraints, and the fully reproducible code plus energy-efficiency analysis provide a practical reference for coursework and future research.

III. Processing Pipeline

The system adopts a stream-oriented, modular design that converts raw RGB frames into richly annotated artefacts (boxes, masks, skeletons, track IDs) plus per-frame statistics in *under 30 ms* [17], [23]. Each stage is a self-contained Python function; all data are passed as plain `dict` objects so that any module can be swapped without touching the others [24].

A. Capture and Pre-processing

Frames arrive via OpenCV's `VideoCapture` (or FFmpeg for network streams) and are time-stamped immediately [23]. A constant-colour letterbox (padding value 114) keeps the aspect ratio, enabling an affine back-mapping of detections to the native resolution [25].

B. Detection and Segmentation

A single-stage YOLOv8s-seg head jointly predicts bounding boxes, class scores, and prototype mask coefficients,

Table I: Pipeline stages and typical *per-frame* latency on the Apple M2 Pro GPU path.

Stage	**Key operations / output**	**Latency**
Capture	RGB frame -> NumPy, timestamp	3–5 ms
Pre-processing	Letterbox 640 $\times$ 640, colour-space fix	<1 ms
Detection + Segm.	YOLOv8s-seg: boxes, masks, scores	8–10 ms
Pose Estimation	Keypoint R-CNN (17 joints), smoothing	12–14 ms
Tracking	Greedy IoU assignment, track life-time	<1 ms
Analytics + Export	JSONL append, per-class counters	<1 ms
Visualisation	Overlays (boxes, masks, skeletons, IDs)	4–6 ms
End-to-end	approx. 29 ms (34 fps)	29 ms

building on the real-time optimisations of YOLOv4 [17]. The model is auto-deployed to CUDA when available, or to Apple Metal (MPS) on macOS; otherwise it falls back to CPU via PyTorch's dynamic dispatch [24]. Post-inference, standard non-maximum suppression (IoU threshold 0.45) filters duplicates [26], and prototype masks are linearly combined with per-instance coefficients.

C. Pose Estimation

Human keypoints are extracted with Keypoint R-CNN (ResNet-50 FPN), i.e. Mask R-CNN's pose branch [**?**]. To cap compute load, the input is resized to 640^2 and inference is skipped every other frame when GPU utilisation exceeds 80% [19]. Exponential smoothing

$$\hat{\mathbf{k}}^{(t)} = \alpha\, \mathbf{k}^{(t)} + (1-\alpha)\, \hat{\mathbf{k}}^{(t-1)}, \quad \alpha = 0.6$$

cuts jitter with negligible added delay [27].

D. Multi-Object Tracking

For each new detection we compute the IoU against the last box of every active track; a greedy assignment matches the highest IoU above $\tau = 0.3$ [10]. Unmatched tracks age by one; if unseen for $T_{\text{lost}} = 5$ frames they are dropped, while unmatched detections spawn new tracks.

E. Analytics and Export

Every frame produces a compact JSON Lines record; a post-run aggregator writes `run_id.summary.csv` (per-class counts) and `run_id.classes.png` (top-15 bar chart).

F. Runtime on CPU vs GPU

Table II: Median per-stage latency on Mac M2 Pro (MPS) vs. single-core CPU.

Stage	**GPU**	**CPU**
YOLOv8s-seg	9.2 ms	31.7 ms
Keypoint R-CNN	12.8 ms	44.5 ms
End-to-end	29.0 ms (34 fps)	86.4 ms (11 fps)

IV. Datasets & Training Details

This section summarises the data sources, splits and training protocols used to obtain the models evaluated in **??**. No external data beyond the datasets listed below were employed.

A. Datasets

Table III: Public datasets used for training and evaluation.

Dataset	**Images**	**Classes**	**Task**	**Split / Note**
COCO 2017	118 k	80	det./seg.	train / val [2]
HomeObjects-3K	2 973	48	det./seg.	train / val / test (70/15/15) [8]
COCO-kp14 (pose)	118 k	17 keyp.	pose	train / val [2]
Live-Capture*	7 videos	—	stress test	—

*Seven 60-s 1080p sequences recorded in a kitchen–living space; used only for throughput, tracking-stability and privacy-filter tests.

COCO 2017: We employ the full *train2017* split to initialize the YOLOv8s-seg backbone and for pose-head pre-training (*kp_train2017*) [2]. The *val2017* split provides a standardised benchmark for cross-domain generalisation.

HomeObjects-3K: This curated set extends household coverage with 48 everyday categories under-represented in COCO [8]. Images were manually annotated with instance masks and class labels.

B. Training Protocol

Table IV: Key hyper-parameters for each model component.

Component	**Ep.**	**Batch**	**LR**	**Optimizer & notes**
YOLOv8s-seg (COCO)	150	64	1×10^{-3}	SGD, cosine decay [28], Mosaic [17] + MixUp [29]
YOLOv8s-seg (HO-3K ft)	50	32	3×10^{-4}	SGD (first 3 stages frozen)
Keypoint-RCNN pre-train	90	16	5×10^{-4}	AdamW [30], half-res input
Pose fine-tune (mixed)	20	8	1×10^{-4}	AdamW, CutMix [31] + color jitter

Detector fine-tuning: After COCO pre-training, the detector is fine-tuned on HomeObjects-3K for 50 epochs with a reduced learning rate. Class IDs overlapping with COCO (e.g. *cup*) share heads to prevent catastrophic forgetting. The best-mAP checkpoint is exported to `.pt` and ONNX.

Pose head: The Keypoint-RCNN branch is first trained on the standard COCO split (17 keypoints [2]), then lightly fine-tuned on HomeObjects-3K to adapt to indoor scenes.

Hardware & Framework: All training runs used an RTX 3060 Ti (8 GB) with PyTorch 2.3 + CUDA 12.4 [24]. Mixed-precision (AMP) was enabled, reaching 210 img/s.

V. Results

The evaluation covers three complementary aspects—*accuracy*, *runtime*, and *robustness*—using two public datasets plus a live-capture stress test. All experiments ran on a MacBook Pro (M2 Pro, 32 GB RAM) with Python 3.11; GPU figures refer to Apple Metal, CPU figures to a single high-performance core.

A. Quantitative Performance

Table V: Detection, segmentation, pose and tracking accuracy.

Dataset & task	Metric	Pipeline	Ablated†	Baseline‡
COCO-val2017 detection	mAP_{50}	37.2 %	33.9 %	33.5 %
COCO-val2017 segmentation	mIoU	0.46	0.43	0.41
HomeObjects-3K detection	mAP_{50}	45.3 %	41.2 %	42.7 %
Keypoint-R-CNN (COCO-kp14)	AP@0.5	0.65	0.62	0.64
Multi-tracking	MOTA	0.72	0.67	0.68
	IDF1	0.69	0.64	0.66

†No smoothing or mask refinement. ‡YOLOv5s + DeepSORT on identical hardware.

B. Runtime Analysis

Table VI: Median per-stage latency on GPU versus CPU.

Stage	GPU	CPU
Capture + resize	4.1 ms	4.1 ms
YOLOv8s-seg	9.2 ms	31.7 ms
Keypoint R-CNN	12.8 ms*	44.5 ms*
Tracking	0.8 ms	0.8 ms
Visual overlay	5.3 ms	5.3 ms
End-to-end	29.0 ms (34 fps)	86.4 ms (11 fps)

*Pose module executed every second frame when GPU utilisation $>80\%$.

C. Qualitative Evaluation

- Pixel masks follow object boundaries closely, enabling precise area and distance measurements.
- Exponential smoothing removes "vibrating" keypoints, especially on wrists and ankles.
- Track IDs stay consistent through moderate occlusions.

D. Failure Modes

Table VII: Typical failure cases and mitigations.

Situation	Observed issue	Mitigation
Glass reflections	Spurious masks on windows	Lower conf. threshold $0.25 \rightarrow 0.15$ plus pos
Fast pans ($>90°$/s)	ID switches	Longer motion horizon or optical flow
Over-exposed highlights	Missing face keypoints	Lock exposure or enable HDR

E. Ablation Study Highlights

- *Mask refinement off*: -2.1 pp mAP, -1 ms latency.
- *IoU threshold 0.5*: $+0.6$ pp IDF1, $+7$ % ID switches.
- *No keypoint smoothing*: -3.2 pp AP, gain ≈ 0.2 ms (disabled).

VI. Concluding Remarks

Over the course of this project we implemented an end-to-end, real-time perception stack that couples YOLOv8s-seg with Keypoint-RCNN and a lightweight IoU tracker, achieving 34 fps and mAP_{50} up to 45 % on commodity hardware. The system processes raw webcam streams, returns pixel-level masks, 17-point human poses, and stable track IDs, all logged in JSONL for downstream analytics.

From a broader perspective, the pipeline demonstrates that privacy-aware video analytics for domestic environments can be delivered without discrete GPUs or cloud resources. The combination of class-agnostic masks and skeletons enables fine-grained reasoning (e.g., hand–object interaction) while keeping the compute budget within the limits of smart-home hubs or kiosks. In practice, this opens the door to applications such as elderly-care monitoring, energy-efficient room automation, and interactive gaming on low-power devices.

Several aspects remain to be improved. First, the object vocabulary is still limited to COCO plus HomeObjects; integrating an Objects-365 subset or training on synthetic data could boost recall in cluttered kitchens and garages. Second, robustness to abrupt camera motion is modest; incorporating a tiny optical-flow module or a motion-compensated buffer would mitigate ID switches. Finally, the exporter currently stores plain JSONL; embedding compressed depth maps would facilitate 3-D analytics without touching the raw video.

Working on Apple's MPS backend showed that Metal kernels can match mid-range NVIDIA GPUs for inference, but debugging tools are immature and model-loading times are unpredictable. The main hurdle was reconciling TorchScript with on-device quantisation; we solved it by freezing the graph after batch-norm fusion and exporting weights as FP16. These insights should help future teams port larger models onto the ever-growing Mac-silicon ecosystem.

REFERENCES

[1] D. J. Cook, J. C. Augusto, and V. R. Jakkula, "Ambient intelligence: Technologies, applications, and opportunities," *Pervasive and Mobile Computing*, vol. 5, no. 4, pp. 277–298, 2009. [Online]. Available: https://doi.org/10.1016/j.pmcj.2009.03.004

[2] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision (ECCV)*, 2014, pp. 740–755. [Online]. Available: https://doi.org/10.1007/978-3-319-10602-1_48

[3] J. Shao, Y. Pang, X. Zhang, J. Li, H. Liang, F. Liu, L. Hu, and L. Lin, "Objects365: A large-scale, high-quality dataset for object detection," in *Proc. IEEE/CVF Int. Conf. Computer Vision (ICCV)*, 2019, pp. 10 029–10 038. [Online]. Available: https://doi.org/10.1109/ICCV.2019.01029

[4] C. Wang, A. Bochkovskiy, and H. M. Liao, "Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022. [Online]. Available: https://arxiv.org/abs/2207.02696

[5] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," in *International Conference on Learning Representations (ICLR)*, 2018. [Online]. Available: https://openreview.net/forum?id=r1gs9JHFwS

[6] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 3645–3649. [Online]. Available: https://doi.org/10.1109/ICIP.2017.8296962

[7] Y. Zhang, W. Ge, C. Wang, X. Wang, S. Li, J. Sun, and W. Liu, "Bytetrack: Multi-object tracking by associating every detection box," in *European Conference on Computer Vision (ECCV)*, 2022, pp. 1–21. [Online]. Available: https://doi.org/10.1007/978-3-031-19766-0_1

[8] A. Tangaro, "Homeobjects-3k: A 3,000-image dataset of household items for detection and segmentation," University of Padova, Technical Report, 2025. [Online]. Available: https://github.com/antoniotangaro/HomeObjects-3K

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 2017, pp. 2980–2988. [Online]. Available: https://doi.org/10.1109/ICCV.2017.322

[10] E. Bochinski, T. Senst, and J. Meyer, "High-speed tracking-by-detection without using image information," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/AVSS.2017.8078454

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587. [Online]. Available: https://doi.org/10.1109/CVPR.2014.81

[12] R. Girshick, "Fast r-cnn," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 2015, pp. 1440–1448. [Online]. Available: https://doi.org/10.1109/ICCV.2015.169

[13] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7263–7271. [Online]. Available: https://doi.org/10.1109/CVPR.2017.690

[14] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520. [Online]. Available: https://doi.org/10.1109/CVPR.2018.00474

[15] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1251–1258. [Online]. Available: https://doi.org/10.1109/CVPR.2017.195

[16] A. Howard, M. Sandler, G. Chu, B. Chen, L. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan, and Q. V. Le, "Searching for mobilenetv3," in *Proc. IEEE/CVF Int. Conf. Computer Vision (ICCV)*, 2019, pp. 1314–1324. [Online]. Available: https://doi.org/10.1109/ICCV.2019.00140

[17] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[18] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," in *Proc. IEEE/CVF Int. Conf. Computer Vision (ICCV)*, 2019, pp. 9157–9166. [Online]. Available: https://doi.org/10.1109/ICCV.2019.00931

[19] Z. Cao, T. Simon, S. Wei, and Y. Sheikh, "Openpose: Real-time multi-person 2d pose estimation using part affinity fields," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 7291–7299.

[20] K. Sun, B. Xiao, D. Liu, and J. Wang, "Deep high-resolution representation learning for human pose estimation," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 5693–5703. [Online]. Available: https://doi.org/10.1109/CVPR.2019.00586

[21] A. Bewley, W. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3464–3468. [Online]. Available: https://doi.org/10.1109/ICIP.2016.7533003

[22] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, J. To, J. Cameracci, Y. Chebotar, I. Doytchinov, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 969–977. [Online]. Available: https://doi.org/10.1109/CVPRW.2018.00127

[23] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. Sebastopol, CA: O'Reilly Media, 2008.

[24] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, pp. 8024–8035, 2019.

[25] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018. [Online]. Available: https://arxiv.org/abs/1804.02767

[26] A. Neubeck and L. Van Gool, "Efficient non-maximum suppression," in *18th Int. Conf. Pattern Recognition (ICPR)*, 2006, pp. 850–855.

[27] R. G. Brown, "Smoothing, forecasting and prediction of discrete time series," *Prentice-Hall*, 1959.

[28] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," in *International Conference on Learning Representations (ICLR)*, 2017. [Online]. Available: https://arxiv.org/abs/1608.03983

[29] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," in *International Conference on Learning Representations (ICLR)*, 2018. [Online]. Available: https://openreview.net/forum?id=r1Ddpz-Rb

[30] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *International Conference on Learning Representations (ICLR)*, 2019. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7

[31] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6023–6032. [Online]. Available: https://doi.org/10.1109/ICCV.2019.00609