Supplementary Material: Rethinking Compact (<1M) Vision Models: Balancing Accuracy and Speed through Multi-Path Atrous Convolutions

Christos Kyrkou
KIOS Research and Innovation Center of Excellence
University of Cyprus
1 Panepistimiou Avenue, Nicosia, Cyprus
kyrkou.christos@ucy.ac.cy

Layer	Output Size	Dilated Kernel	Output Filters	Stride
Input image	224×224			
Stem	112×112	3	24	2
MPAC 1	112×112	3, 5	48	1
MPAC 2	56×56	3, 5	64	$2^{\dagger}$
MPAC 3	28×28	3, 5	96	$2^{\dagger}$
MPAC 4	28×28	3, 5	256	1
MPAC 5	14×14	3, 5	384	$2^{\dagger}$
MPAC 6	14×14	3, 5	384	1
MPAC 7	14×14	3, 5	416	1
Conv + BatchNorm	14×14	1	# of Classes	1
Global Pooling	1×1		# of Classes	
Softmax	1×1		# of Classes	

† Stride for image size of 32×32 is 1.

Table 1. LiteSpeed Model Structure

S1. Network Architecture Details

The detailed architecture of the sub-1M instantiated LiteSpeed Network is shown in Table 1. Overall, it is comprised of a stem that reduces the initial image resolution by a factor of 2. This is followed by a repeated instantiation of the main block with different number of channels and configurations. Finally, the classification head uses a fully convolutional approach and global average pooling to reduce the feature map to a vector with dimensions equal to the number of classes. For the Cifar-10 dataset the input image is downscaled only once to avoid removing much of the image information.

The same architecture can be repurposed for dense tasks such as detection by properly adjusting striding and extracting features from multiple scales (e.g., MPAC5, MPAC6, and MPAC7). These are then fed to a Path Aggregation Network style neck and a decoupled detection head at each scale.

S2. Theoretical Results

S2.1. Complexity Analysis

The theoretical computational complexity of the convolutional operations for the MPAC block, without the depth expand/reduce, which is optional, are as follows. First, for the two depthwise dilated convolution blocks the operations are given by:

$$MACs_{DW} = k \cdot k \cdot h \cdot w \cdot c_{in} \cdot 2 \tag{1}$$

while the computational complexity of the mixing and expansion convolution after the concatenation is given by:

$$MACs_{EXP} = 2 \cdot c_{in} \cdot h \cdot w \cdot c_{out} \tag{2}$$

where c_{in} are the input channel dimensions of the feature map, c_{out} are the channel dimensions of the output feature map, k is the kernel size modulated by the dilated convolution, remains as constant with a value of 3, and h, w are the spatial dimensions of the feature map. Here, no striding is assumed and no bias is applied. It is important to note that while MACs/FLOPs serve as indicators of computational complexity and are provided for completeness, they may not reflect actual performance as latency on hardware can follow different trends [3].

First the condition for which the inverted residual block (IRB) $MACs_{IRB}$ becomes larger than the MACs of the MPAC block $MACs_{MPAC}$ is examined. Given that the two modules would have the same input feature maps and channel input and output dimensions the impact of the expansion factor would be the main amplifier of operations within the inverted residual block. Thus the inverted residual block expansion factor r is solved to determine for which values the inequality holds. The main focus is on the operations of the learnable parameters with the assumption that stride is 1 and also batch size of 1. The derivation is as follows.

$$MACs_{IRB} > MACs_{MPAC}$$

Substituting the equations for the MAC operations for each module.

$$\begin{aligned} & c_{in} \cdot h \cdot w \cdot (r \cdot c_{in}) + \\ & k^2 \cdot h \cdot w \cdot c_{in} \cdot r + \\ & c_{in} \cdot r \cdot h \cdot w \cdot c_{out} \\ & > k^2 \cdot h \cdot w \cdot c_{in} \cdot 2 + \\ & 2 \cdot c_{in} \cdot h \cdot w \cdot c_{out}. \end{aligned} \tag{3}$$

After eliminating $w \cdot h$.

$$\begin{aligned} & c_{in} \cdot (r \cdot c_{in}) + k \cdot k \cdot c_{in} \cdot r + \\ & c_{in} \cdot r \cdot c_{out} \\ & > k \cdot k \cdot c_{in} \cdot 2 + \\ & 2 \cdot c_{in} \cdot c_{out} \end{aligned} \tag{4}$$

Solving for r.

$$r > \frac{2 \cdot (k^2 \cdot c_{in} + c_{in} \cdot c_{out})}{c_{in}^2 + k^2 \cdot c_{in} + c_{in} \cdot c_{out}} \tag{5}$$

This analysis reveals the value that ratio r needs to have for the $MACs_{IRB}$ to be larger than $MACs_{MPAC}$, and that is dependent on the number of output channels, input channels, and kernel size. Trends can be identified by examining how the ratio r changes with respect to the input channels c_{in} and output channels c_{out}

- When $c_{out} >> c_{in}$ (High Output Channels):
  The term c_{out} will dominate in the numerator because it approximately grows twice as much. Hence, as c_{out} increases the value of r will be around 2 hence even for moderate values $MACs_{IRB}$ will be larger than $MACs_{MPAC}$.
- When $c_{in} >> c_{out}$ (High Input Channels):
  The c_{in} term in the deinator will dominate due to its quadratic increase. Therefore r decreases as c_{in} grows, reducing the limit for which $MACs_{IRB}$ is higher than $MACs_{MPAC}$.
- When $c_{in} \simeq c_{out}$ (Balanced Input and Output Channels):
  In the case of balanced input and output channels the numinator becomes larger than the denominator meaning that for values larger than 1 the $MACs_{IRB}$ will be higher.

Concluding, for the values of r typically found in the literature it is anticipated that the theoretical complexity of the inverted residual block will be higher. Together with the accuracy improvements presented in the main manuscript, these results position the MPAC block as a promising alternative.

Figure 1. Edge AI Devices: NVIDIA Jetson Nano (left) - NVIDIA Jetson AGX Orin (right)

S2.2. Sampling Efficiency advantage

To quantify the computational efficiency of the proposed MPAC block, a simple sampling-efficiency measure is defined as $E = \frac{S}{MACs}$, where S is the number of distinct input s sampled by all branches and MACs is the block computational cost.

For a two-branch MPAC combining 2 atrous depthwise convolutions with dilation rates 1 and 2, $S_{MPAC} = 17$ while a standard 3×3 inverted residual block (IRB) samples $S_{IRB} = 9$ positions.

Assuming input/output channels $c_{in} = c_{out}$ and an IRB expansion factor r, the following is obtained:

$$\frac{E_{MPAC}}{E_{IRB}} = \frac{17 \cdot r \cdot (2 \cdot c_{in}^2 + k^2 \cdot c_{in})}{18 \cdot (c_{in}^2 + k^2 \cdot c_{in})} \tag{6}$$

In general it is observed that the higher the expansion factor r the more computation is spent per specific location, limiting the available budget for additional context. When $r = 1$ the ratio depends only on the channel/input parameters and remains larger than 1.

For example for $c_{in} = c_{out} = 64$ gets approximately $1.8\times$ higher efficiency for $r = 1$ and approximately $7\times$ for $r = 4$.

Thus, MPAC offers substantially more spatial coverage per MAC than the IRB, and the relative advantage increases with larger expansion ratios. This metric provides a simplified view into why allocating compute to a larger context can lead to the observed empirical gains.

S3. Considered Edge-AI Devices

Lightweight models are intended to be deployed on resource-constrained devices and often operate in real-time. Such devices have specialized low-power GPUs that can accelerate deep-learning models. Some examples include the Jetson Nano and Jetson AGX Orin (shown in Fig. 1), used in this study, which differ in processing power and memory, as well as the underlying architecture. Such devices can be

Platform	GPU	Memory
Jetson Nano	128-core Maxwell	4 GB 64-bit LPDDR4, 25.6 GB/s
Jetson AGX Orin	2048-core Ampere	64GB 256-bit LPDDR5 204.8GB/s

Table 2. System Specifications

deployed in autonomous systems or used as ground stations, enabling computer vision in edge AI scenarios. Details of the targeted devices are shown in Table 2. By evaluating the models' efficiency and effectiveness on each device, valuable insights can be provided into how models operate under varying capabilities. The selected platforms reflect common devices used in a wide range of Edge AI applications.

S4. Dataset and Training Details

S4.1. Datasets

To perform an investigation of the architectural landscape and assess the effectiveness of the proposed model, experiments are conducted using a benchmark suite of three distinct classification datasets. They share characteristics of edge AI applications such as varying image resolution, few number of classes, and a few thousand images in total. The Cifar-10 dataset (referred to as C10), a widely used benchmark, [6] consists of 32×32 RGB images, containing 50,000 images for training and 10,000 images for testing, equally distributed across ten classes. The Intel Image Classification (referred to as IIC) dataset contains around $14k$ images in the train split, and $3k$ in test split, of size 150×150 distributed under 6 categories (building, forest, glacier, mountain, sea, and street). [1] The IIC dataset offers a distinct set of image categories tailored for specific outdoor and natural scenes, enabling targeted evaluation and provides insights into how models would perform in real-world applications. Lastly, the ImageNet-16 [7] dataset (referred to as ImgN16) is another subset of the renowned ImageNet dataset. It features a more compact classification task with 16 distinct classes. Notably, ImgN16 maintains the original image resolutions, which are typically set at 224×224 pixels for standard ImageNet evaluations. Collectively, these datasets provide a comprehensive evaluation set, encompassing a wide range of class counts and image resolutions, thereby enabling a thorough assessment of the proposed approach.

S4.2. Training Details

The models are trained for 200 epochs for Cifar-10 and 100 epochs for the other two datasets since they have less samples. The optimizer is AdamW. An initial learning rate of $1e-2$ is used with a cosine scheduler for decaying the learning rate with a minimum learning rate of $1e-5$. The only hyperparameter search performed across all models was to use a lower starting learning rate of $1e-3$, as the learning rate significantly impacts model performance. The best result achieved by each model is presented which for some models was achieved with the smaller learning rate. A weight decay of $5e-2$ is used for Cifar-10 and ImageNet-16, and $1e-4$ for IIC. Only simple augmentations are applied in order to better assess the learning capabilities of each. These are color jittering, horizontal flipping, random cropping, as well as small rotation, translation and scaling. All models are trained from scratch on each dataset. The loss to train all networks is shown in Eq. 7, and incorporates the Cross Entropy loss and the Cosine similarity loss as proposed in [1].

$$Loss = -\lambda_1 \sum_{i=1}^{N_C} y_i \log(p_i) + \lambda_2 (1 - \frac{\mathbf{Y} \cdot \mathbf{P}}{\|\mathbf{Y}\| \cdot \|\mathbf{P}\|}) \quad (7)$$

where p_i,y_i are the predicted and ground truth class probabilities respectively, while $\mathbf{Y}$ $\mathbf{P}$ are vectors with all the ground truth and predicted class probabilities respectively. The λ_1 and λ_2 parameters are weights for each loss component, which are both set to 1 in the experiments.

For detection tasks the binary cross entropy loss is used for classification and Complete IoU for bounding box regression. Both well established and typical for detection tasks.

S5. Targeted Application Evaluation

LS-Net's suitability for practical real-time edge AI applications is assessed by conducting targeted evaluations on two additional datasets as well as on the task of object detection.

S5.1. Camera Scene Detection

As a more targeted evaluation the Camera Scene Detection Dataset (CamSDD) is utilized for dataset [5] containing more than 11K images belonging to 30 camera scene categories. The image resolution is 384×576 and frame-rate results are taken on the NVIDIA Orin Platform. Note, that again no aspect of the network was further optimized for this dataset, and also used the base architecture as is. Again the previously selected best performing models across the other datasets as used for the evaluation. From the results in Fig. 4 LS-Net appears to have the best balance, achieving both a high FPS the maximum accuracy (the only model to reach over 90%). ResNet has the lowest FPS with a comparable accuracy to other models.

This figure suggests that LS-Net which has an unchanged architecture from the other experiments is able to provide high accuracies on an additional dataset at higher resolution with much higher FPS.

[1] available on Kaggle https://www.kaggle.com/datasets/puneet6060/intel-image-classification

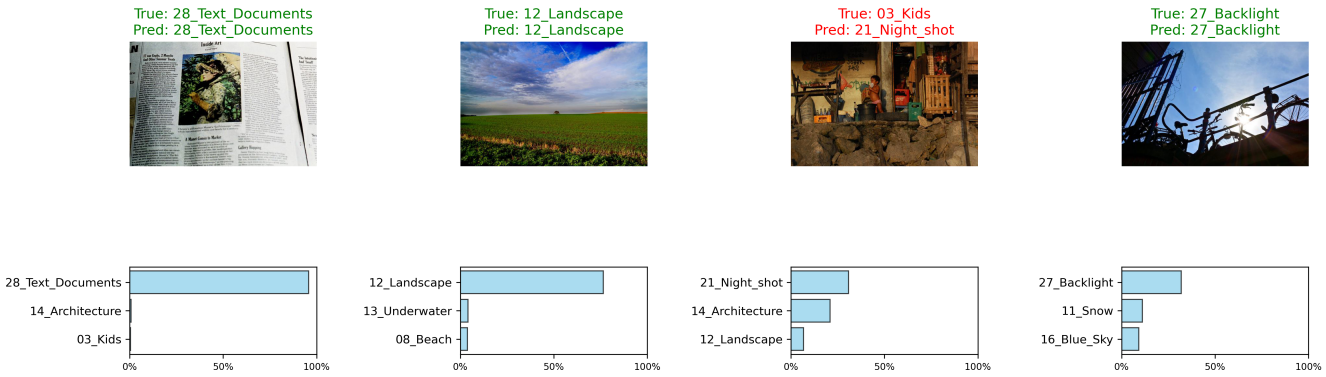

Figure 2. Image Results for CSD dataset

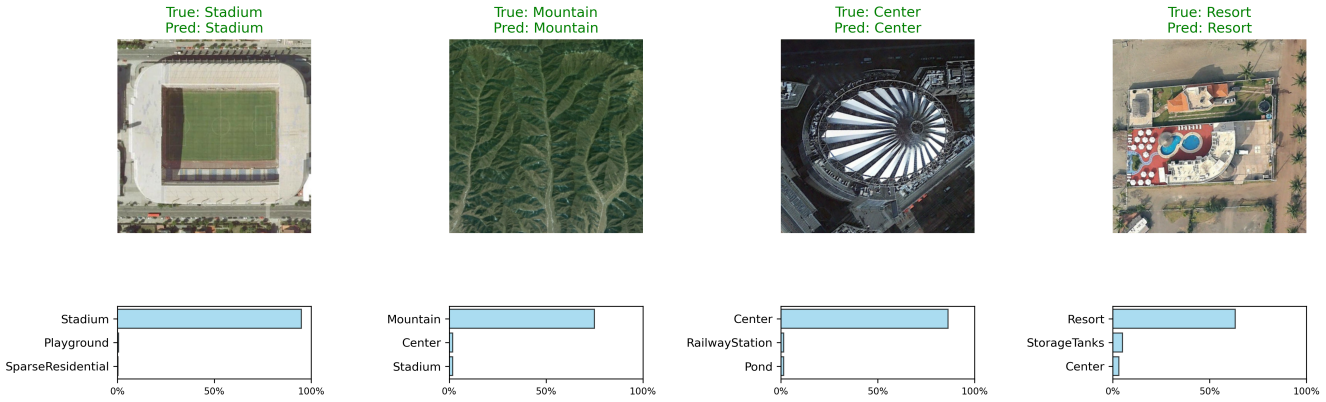

Figure 3. Image Results for AID dataset

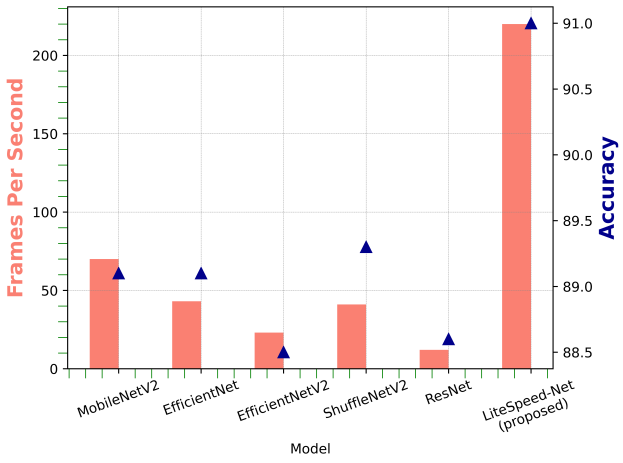

Figure 4. Frames per second (left y-axis) and Accuracy Results (right y-axis) on the CamSDD dataset.

S5.2. Aerial Image Detection

The Aerial Image Dataset (AID) is a high-resolution benchmark dataset for aerial and remote sensing image classification. It contains 10,000 RGB images across 30 diverse scene categories (e.g., airports, forests, residential areas). Sourced from Google Earth, it represents various geographic regions with varying spatial resolutions.

Table. 3 shows the accuracy achieved by each model. Again the LS-Net demonstrated the highest accuracy in this diverse dataset. Among the evaluated models, LS-Net achieves the highest accuracy of 93.5%, surpassing established models such as MobileNetV2 (93.0%), ShuffleNetV2 (92.7%), and ResNet (91.5%). While EfficientNet models lag behind. Notice again that the ranking has changed compared to other experiments highlighting the inconsistency among existing models. The improved performance of LS-Net on this aerial dataset can be attributed to the use of dilated convolutions that capture much richer multi-scale feature information, which is beneficial for this types of images at varying altitudes.

S6. Robustness

Table 4 shows the robustness accuracy (%) under three types of perturbations 1) Additive Gaussian noise to test stability to random pixel-level noise ($mean = 0.0, std = 0.1$), 2) Fast

<1M Param Models	Acc-AID (%) ↑
MobileNetV2 [9]	93.0
EfficientNet [12]	88.0
EfficientNetV2 [11]	90.5
ShuffleNetV2 [8]	92.7
ResNet [4]	91.5
***LiteSpeed-Net** (proposed)*	**93.5**

Table 3. Performance of select best Sub-1M models on the AID dataset.

<1M Param Models	Gaussian (%) ↑	FGSM (%) ↑	PGD (%) ↑
MobileNetV2 [9]	52.2	**33.2**	24.0
EfficientNetV2 [11]	50.5	20.3	0.3
ShuffleNetV2 [8]	52.6	32.6	8.8
ResNet [4]	31.0	8.8	0.8
***LiteSpeed-Net** (proposed)*	**54.6**	26.3	**24.3**

Table 4. Performance of scaled down versions of best models under three types of perturbations.

Gradient Sign Method (FGSM) to test one-step adversarial robustness (with $\epsilon = 0.03$), and 3) Projected Gradient Descent (PGD) to test iterative, stronger adversarial robustness (with $\epsilon = 0.03$, $\alpha = 0.007$ and 20 iterations). LiteSpeed-Net demonstrates balanced and consistent robustness which is highest under Gaussian noise, competitive under FGSM, and strongest under PGD. MobileNetV2, while slightly more resistant to FGSM, shows sharper degradation under iterative attacks—indicating less stable robustness. It is observed that the proposed LS-Net demonstrates the best overall balance meaning highest Gaussian robustness and PGD resistance, even if FGSM is not the absolute best. LS-Net's balance between FGSM and PGD performance suggests more stable gradient behavior and greater resilience. Meanwhile resistance to Gaussian noise indicates the effectiveness of the contextualized spatial features. It is further observed that as the attack strength increases while accuracy drops the performance is more reliable than other models. EfficientNetV2 and ResNet perform poorly under adversarial conditions, suggesting their architectural characteristics do not translate to adversarial resilience at small parameter budgets. Among 1M-parameter models, LiteSpeed-Net is the most robust overall, as it sustains high accuracy across both stochastic (Gaussian) and adversarial (PGD) perturbations. It appears to trade a modest drop under FGSM for improved resistance to stronger, iterative attack, suggesting a more advantageous robustness profile.

S7. Additional Architecture Ablations

Ablations are conducted to test the impact of the architecture. The same architecture is instantiated as shown in Table 1 but instead with only a single path for which dilation=1, and instead reallocate the channels so that it has a wider convolution layer but overall similar parameter count at $957K$. Models are trained on Cifar10 and ImageNet16 classification datasets with exactly the same hyperparameters, and resulting performance is 91.2% and 85.2% for each dataset respectively. This is a drop from the MPAC-based architecture (-0.7% and -1.3% for each dataset respectively) positioning it even below some other models, demonstrating the critical importance of the proposed block in learning effective and diverse spatial representations.

Further ablations for dilations and fusion method are also conducted. Higher dilations rates of 3 and 4 lead to reduced performance 91.1% for $d = 3$ and 91.7% for $d = 4$ on C10 and 84.5% for $d = 3$ and 85.6% for $d = 4$ on ImageNet16 while being up to 10% slower in terms of FPS. Substituting concatenation with addition for fusion in MPAC and maintaining 1M parameters leads to 91.7% on C10 and 84.8% on ImageNet16 while achieving 198 FPS.

S8. Impact of Knowledge Distillation

To test if architectural gains persist under strong training such as knowledge distillation, a ResNet18 is distilled (imagenet weights and 11M params) into LS-Net and MobileNetV2 students for the Cifar-10 dataset. LS-Net improved by +2.3% (vs. +1.6% for MobileNetV2). LS-Net performance still persists indicating that it maintains benefits with strong training techniques while providing faster inference speed.

S9. Transferability to other codebases and settings

Based on reviewer feedback a Cifar-10 specific codebase [2] was utilized to evaluate the performance of LS-Net under different training regimes. The repository contains reference implementations to many networks some of which are under 1 million parameters. Specifically, the codebase features small ResNet and MobileNetV2 variants which are trained on standard Cifar-10 settings, whereas this work uses stronger augmentations and different learning rates. LS-Net was ported to the reference codebase and trained with the same settings as the other models. The retrained reference implementations for different models was as follows: ResNet20 92.6%, MobileNetV2 92.9%, and ResNet56

[2] https://github.com/chenyaofo/pytorch-cifar-models

1M Param Backbones	Vehicles	Animals	Construction	Person	Home Objects
	mAP@0.5:0.95 (%) ↑				
MobileNetV2 [9]	22.7	21.5	36.5	36.1	26.8
EfficientNet [12]	24.8	27.7	36.6	28.9	19.8
EfficientNetV2 [11]	23.8	20.4	35.2	29.5	28.1
ShuffleNetV2 [8]	23.7	23.6	34.6	33.3	27.0
ResNet [4]	23.8	18.9	36.0	30.6	26.1
***LiteSpeed-Net** (proposed)*	**25.3**	**33.8**	**40.0**	**41.2**	**30.3**

Table 5. Object Detection mAP@0.5:0.95 Results

Detection Models	Vehicles	Animals	Construction	Person	Home Objects
	mAP@0.5 (%) ↑				
YOLOv11n [10]	**40.9**	59.8	77.9	**77.5**	53.9
***LiteSpeed-Net** (proposed)*	38.5	**60.5**	**81.5**	75.4	**54.4**

Table 6. Object Detection mAP@0.5 Results for LS-Net and YOLO11

94.0%. In the same codebase LS-Net achieved 93.4%, while the 100k LS-Net variant achieved 90.1% which was also competitive. While ResNet56 performs slightly better it severly lags behind when evaluated for frame-rate at the standard 224×224 image size with 66 FPS. ResNet20 which is much smaller performs at 174 FPS which is lower than the 220 FPS achieved by LS-Net. As discussed in the limitations section this highlights further the premise of this work that while individual models and varying settings can produce higher accuracies the main benefit of using LS-Net is in the optimal accuracy-speed trade-offs and consistent performance.

S10. Other devices

In order to evaluate transferability beyond GPU devices the top performing models on C10 and ImgN16 datasets were selected and tested on the ARM CPU of Jetson Orin. LS-Net achieves 6.2 FPS while MobileNetV2, ResNet, and EfficientNetV2 achieve 4.6, 6.1, and 3.3 FPS respectively, indicating that LS-Net is competitive even if not optimized for CPU.

S11. Additional Object Detection Results

Further to the results in the main manuscript additional qualitative and quantitative results are presented herein. In Table 5 results for the mAP@0.5:0.95 metric are presented. This metric is more affected by the neck and head which were not optimized. Results show that LS-Net still performs better under this more strict regime.

Some example results for object detection are presented in this section. The model instantiated for object detection consistently finds home objects (in Fig. 5) such as sofas, chairs, tables, lamps, windows, potted plants; with generally well-aligned bounding boxes and no signs of extreme overconfidence, since its scores stay in a moderate range between $0.3 - 0.7$.

It also handles the problem of person and person-like object detection well (in Fig. 6), with good bounding box alignment around real persons while marking mannequins, statues, and similar figures as "person-like" mostly correctly. This indicates that the model is able to utilize additional information (material and context) beyond the silhouette of the human form.

Overall, it performs steadily in different settings such as indoor or with high inter-class similarity, with reasonably calibrated confidence scores and strong alignment quality.

S12. Comparison with YOLO for object detection

Additional comparisons are made with specialized object detection models such as YOLO11 (specifically the nano variant) [10] with its results in Table 6. While the LS-Net model is used for object detection, there is room for further improving its performance in detection tasks by improving head and neck architecture and selection of multi-scale feature maps. Even without improved detection components LS-Net performs competitively with YOLO11 on these datasets.

Performance is also evaluated further considering an edge AI drone application using the visdrone dataset [2]. Under typical training settings (excluding mosaic which makes small objects even smaller) and when training from scratch for 200 epochs for both models LS-Net demonstrated a performance of 35.5% while YOLO11 a performance of 31.3% for mAP@0.5, and YOLO26 31.4%. LS-Net also demonstrates improvements in FPS on the Jetson Orin with 43 FPS compared to 38 FPS for YOLO11 and 30 FPS for YOLO26 on 640×640 images.

References

[1] Bjorn Barz and Joachim Denzler. Deep learning on small datasets without pre-training using cosine loss. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1371–1380, 2020.

[2] Yaru Cao, Zhijian He, Lujia Wang, Wenguan Wang, Yixuan Yuan, Dingwen Zhang, Jinglin Zhang, Pengfei Zhu, Luc Van Gool, Junwei Han, et al. Visdrone-det2021: The vision meets drone object detection challenge results. In *Proceedings of the IEEE/CVF International conference on computer vision*, pages 2847–2854, 2021.

[3] Jierun Chen, Shiu-hong Kao, Hao He, Weipeng Zhuo, Song Wen, Chul-Ho Lee, and S-H Gary Chan. Run, don't walk: chasing higher flops for faster neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12021–12031, 2023.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

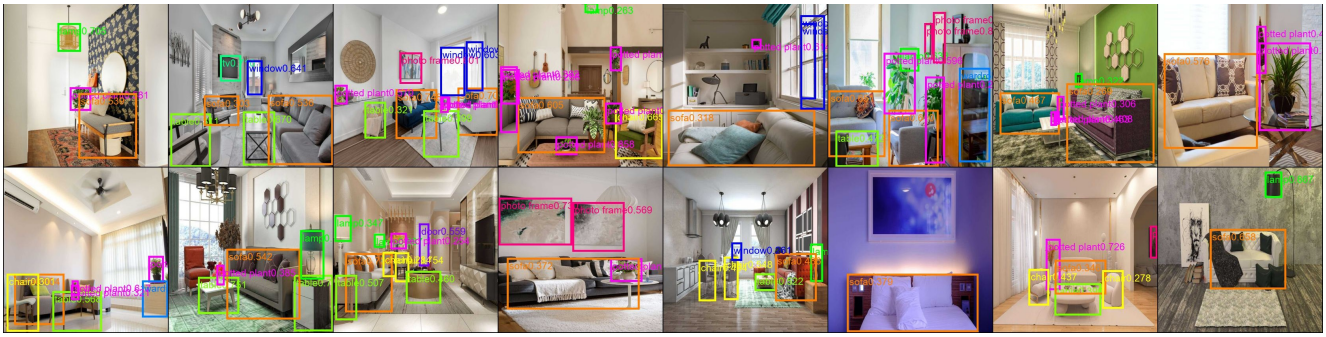

Figure 5. Detection Results for the Home Object Detection Dataset

Figure 6. Detection Results for the Person Detection Dataset

[5] Andrey Ignatov, Grigory Malivenko, and Radu Timofte. Fast and accurate quantized camera scene detection on smartphones, mobile ai 2021 challenge: Report. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 2558–2568, 2021.

[6] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

[7] Christos Kyrkou. Toward efficient convolutional neural networks with structured ternary patterns. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–8, 2024.

[8] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.

[9] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[10] Ranjan Sapkota, Marco Flores-Calero, Rizwan Qureshi, Chetan Badgujar, Upesh Nepal, Alwin Poulose, Peter Zeno, Uday Bhanu Prakash Vaddevolu, Sheheryar Khan, Maged Shoman, et al. Yolo advances to its genesis: A decadal and comprehensive review of the you only look once (yolo) series. *Artificial Intelligence Review*, 58(9):274, 2025.

[11] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR, 2021.

[12] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.