MUXConv: Information Multiplexing in Convolutional Neural Networks (Supplementary Material)

Zhichao Lu Kalyanmoy Deb Vishnu Naresh Boddeti
Michigan State University
{luzhicha, kdeb, vishnu}@msu.edu

In this supplementary material we include (1) MUXNet hyperparameter search space in Section 1, (2) computational complexity of MUXNet and comparison to a combination of $1 \times 1 + 3 \times 3$ in Section 2, (3) effectiveness of MUXNet as a backbone semantic segmentation in Section 3.1, and (4) evaluation of generalization and robustness properties of MUXNet in Section 3.2. Finally Fig. 7 shows some qualitative object detection results on PASCAL VOC 2007, and Fig. 8 shows gradCam results on the ChestX-Ray14 dataset.

1. Search Space

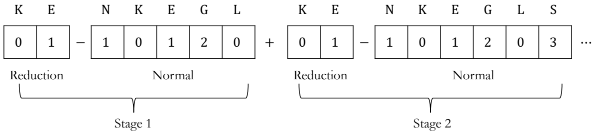


Figure 1: Search Space Encoding: Each stage is encoded as an integer string. Genetic operations are performed on such encoding. See Table 1 for full details on the options.

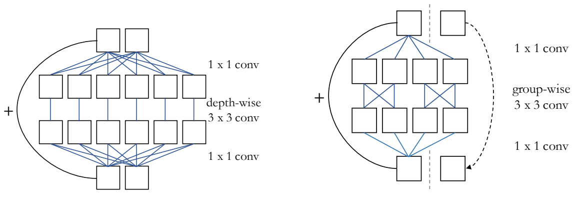
To encode the hyperparameter settings for a model, we first divide the model architectures into four stages, based on the spatial resolution of each layer's output feature map. In each stage spatial resolution does not change. The first layer in each stage reduces the feature map size by half. For each stage, we search for kernel size (K) and expansion ratio (E). In addition, from second layer in each stage, we search for # of repetitions (N), # of input channels to compute convolution (G), leave-out ratio in channel multiplexing (L) and the spatial multiplexing setting (S) (see Fig. 1). Table 1 summarizes the hyperparameters and available options for each stage. The obtained hyperparameters for our MUXNets are visualized in Figure 2. The total volume of the search space is approximately 14^{12}.

2. Computational Complexity

	Hyperparameter	Notation	Options	Stages
Normal Blocks	Kernel size	K	{3, 5}	{1, 2, 3, 4}
	Expansion rate	E	{4, 6}	{1, 2, 3, 4}
	Group factor	G	{1, 2, 4}	{1, 2, 3, 4}
	Repetitions	N	{0, 1, 2, 3}	{1, 2, 3, 4}
	Leave-out ratio	L	{0.0, 0.25, 0.5}	{1, 2, 3, 4}
	Spatial Mux	S	{0, [-1, 0, 0], [0, 0, 1], [1, 0, 1], [-1, 0, 0, 1]}	{2, 3}
Reduction Blocks	Kernel size	K	{3, [3, 5, 7], [3, 5, 7, 9]}	{1, 2, 3, 4}
	Expansion rate	E	{4, 6}	{1, 2, 3, 4}

Table 1: Hyperparameter search space summary. The searched hyperparameters depend on both the block type—i.e. normal or reduction block, and the stages. In case of spatial mutiplexing, option "-1" means subpixel multiplexing, "1" means superpixel multiplexing, and "0" means no spatial multiplexing. For instance, "[-1, 0, 1]" means applying subpixel to 1/3 of the input channels, superpixel to another 1/3 of the input channels, and the remaining 1/3 are processed at the original resolution. And we only apply spatial mutiplexing in stages two and three. For the kernel size options in case of reduction blocks, we allow multiple parallel kernels to down-sample the resolution, for example, "[3, 5, 7]" means three parallel convolutions with kernel size of 3, 5, and 7.

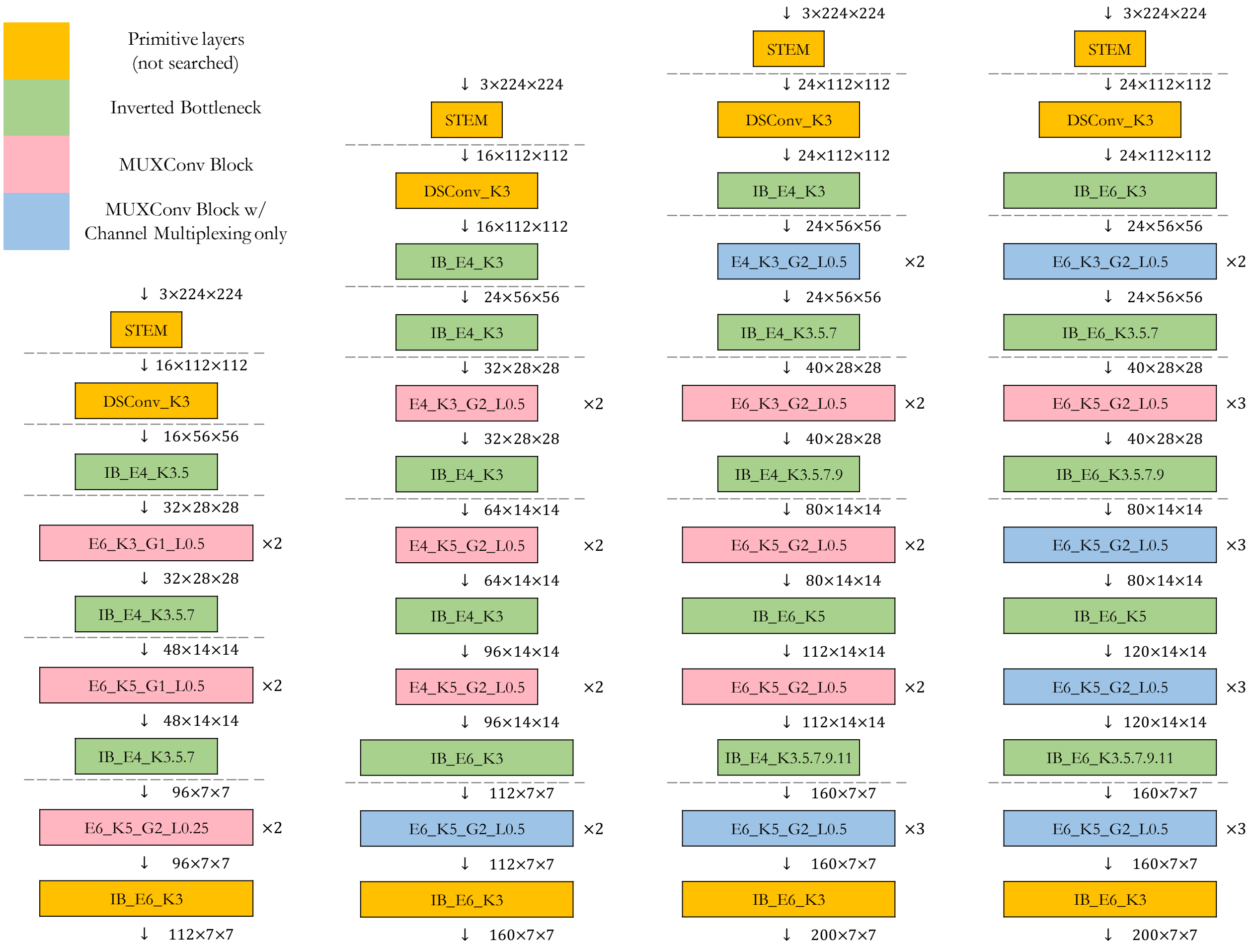
In this section, we analytically compare the computational complexity of our MUXConv block (Figure 3b) with the widely-used MobileNet block [6]. For simplicity, we ignore the computation induced by the normalization and activation layers and we assume that for both blocks the number of input and output channels is the same i.e., C channels.



(a) Mobilenet block [6] **(b)** MUXConv block

Figure 3: The visualization of the Mobilenet block (a) and our MUXConv block (b).

The Mobilenet block consist of a 1×1 convolution to expand the input channels, followed by a 3×3 depth-wise separable convolution and another 1×1 convolution to compress the channels (see Figure 3a). We use E to denote expansion rate. Then the total number of parameters and



(a) MUXNet-xs **(b)** MUXNet-s **(c)** MUXNet-m **(d)** MUXNet-l

Figure 2: The architectures of MUXNet-xs/s/m/l in Table 1 (main paper). All architectures share the same hyperparameter settings (except # of output channels) for the blocks colored in yellow and they are fixed manually. The Dash lines indicate down-sampling points and we divide the architectures into four main stages. We use E, K, G, and L to denote expansion rate, kernel size, number of channels per group and leave-out ratio, respectively. Blocks colored in green use the inverted bottleneck structure proposed in [6]. Blocks colored in pink use both spatial and channel multiplexing and blocks colored in blue only use channel multiplexing.

floating point operations are:

$$\text{Params} = \underbrace{C \cdot EC}_{1\,\times\,1\text{ conv}} + \underbrace{EC \cdot 3 \cdot 3}_{3\,\times\,3\text{ d.w. conv}} + \underbrace{EC \cdot C}_{1\,\times\,1\text{ conv}}$$

$$\text{FLOPs} = H \cdot W \cdot \left(\underbrace{C \cdot EC \cdot}_{1\,\times\,1\text{ conv}} + \underbrace{EC \cdot 3 \cdot 3}_{3\,\times\,3\text{ d.w. conv}} + \underbrace{EC \cdot C}_{1\,\times\,1\text{ conv}} \right)$$

On the other hand, our MUXConv block first select a subset of the input channels to be processed, and the remaining portion is directly propagated to the output. We use L to denote the ratio of the leave-out un-processed channels. Then we use a 1×1 convolution to expand, followed by a group-wise convolution [9] and another 1×1 convolution to compress (see Figure 3b). And we use G to denote the group factor, which indicates the # of input channels used for computing each output channel. For instance, setting G equal to 1 is equivalent as using a depth-wise separable convolution. The resulting number of parameters and the floating point operations associated with our MUXConv block is:

$$\hat{C} = (1 - L) \cdot C$$

$$\text{Params} = \underbrace{\hat{C} \cdot E\hat{C}}_{1\,\times\,1\text{ conv}} + \underbrace{G \cdot E\hat{C} \cdot 3 \cdot 3}_{3\,\times\,3\text{ group conv}} + \underbrace{E\hat{C} \cdot \hat{C}}_{1\,\times\,1\text{ conv}}$$

$$\text{FLOPs} = H \cdot W \cdot \left(\underbrace{\hat{C} \cdot E\hat{C} \cdot}_{1\,\times\,1\text{ conv}} + \underbrace{G \cdot E\hat{C} \cdot 3 \cdot 3}_{3\,\times\,3\text{ group conv}} + \underbrace{E\hat{C} \cdot \hat{C}}_{1\,\times\,1\text{ conv}} \right)$$

Figure 4 provides an visual comparison showing the ratio of the number of parameters between our MUXConv block and Mobilenet block as the group factor (G) and leave-out ratio (L) vary. The choice of G and L hyperparameters we consider in our search space (see Table 1) corresponds to computational complexity that is less than the Mobilenet block (ratio ≤ 1, i.e. red color in Fig.4).

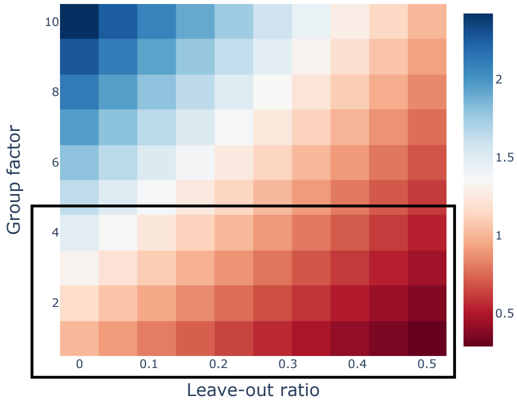


Figure 4: Ratio of #Params between our MUXConv block and Mobilenet block [6]. The search space that we consider for these two hyperparameters is highlighted by a black box.

Network	#MAdds	#Params	mIoU (%)	Acc (%)
ResNet18 [2] + C1	1.8B	11.7M	33.82	76.05
MobileNetV2 [6] + C1	0.3B	3.5M	34.84	75.75
MUXNet-m + C1	**0.2B**	**3.4M**	**32.42**	**75.00**
ResNet18 + PPM	1.8B	11.7M	38.00	78.64
MobileNetV2 + PPM	0.3B	3.5M	35.76	77.77
MUXNet-m + PPM	**0.2B**	**3.4M**	**35.80**	**76.33**

Table 2: ADE20K [12] Semantic Segmentation Results. Since networks in each section use the same segmentation head, we report the *#MAdds* and *#Params* on the backbone models only. *mIoU* is the mean IoU and *Acc* is the pixel accuracy. *C1* use one convolution module as segmentation head and *PPM* use the Pyramid Pooling Module from [10].

3. Additional Experiments

3.1. Semantic Segmentation

We further evaluate the effectiveness of our models as backbones for the task of mobile semantic segmentation. We compare MUXNet-m with both MobileNetV2 [6] and ResNet18 [2] on ADE20K [12] benchmark. Additionally, we also compare two different segmentation heads. The first one, referred as *C1*, only uses one convolution module. And the other one, Pyramid Pooling Module (*PPM*), was proposed in [10]. All models are trained under the same setup: we use SGD optimizer with initial learning rate 0.02, momentum 0.9, weight decay 1e-4 for 20 epochs. Table 2 reports the mean IoU (mIoU) and pixel accuracy on the ADE20K validation set. MUXNet-m performs comparably with MobileNetV2 when paired with PPM, while being 1.5× more efficient in MAdds. We also provide qualitative visualization of semantic segmentation examples in Figure 5.

3.2. Generalization and Robustness

To further evaluate the generalization performance of our proposed models, we compare on a recently proposed benchmark dataset, ImageNetV2 [5], complementary to the original ImageNet 2012. We use the *MatchedFrequency* version of the ImageNet-V2. Figure 6a reports the top-5 accuracy comparison between our MUXNets and a wide-range of previous models. Even though there is a significant accuracy drop of 8% to 10% on ImageNet-V2 across models, the relative rank-order of accuracy on the original ImageNet validation set translates well to the new ImageNet-V2. And our MUXNet performs competitively on ImageNet-V2 as compared to other mobile models, such as ShuffleNetV2 [4], MobileNetV2 [6] and MnasNet-A1 [7].

The vulnerability to small changes in query images has always been a concern for designing better models. Hendrycks and Dietterich [3] recently introduced a new dataset, ImageNet-C, by applying commonly observable corruptions (e.g., noise, weather, compression, etc.) to the clean images from the original ImageNet dataset. The new dataset contains images perturbed by 19 different types of corruption at five different levels of severity. And we leverage this dataset to evaluate the robustness of our proposed models. Figure 6b compares Top-5 accuracy between our MUXNet-m and four other representative models, designed both manual and automatically. MUXNet-m performs favourably on ImageNet-C, achieving better accuracy on 18 out of 19 corruption types.

References

[1] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jan 2015. 5

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[3] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations (ICLR)*, 2019. 3, 4

[4] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *European Conference on Computer Vision (ECCV)*, 2018. 3

[5] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? *arXiv preprint arXiv:1902.10811*, 2019. 3, 4

[6] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 3

[7] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[8] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 5

[9] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep

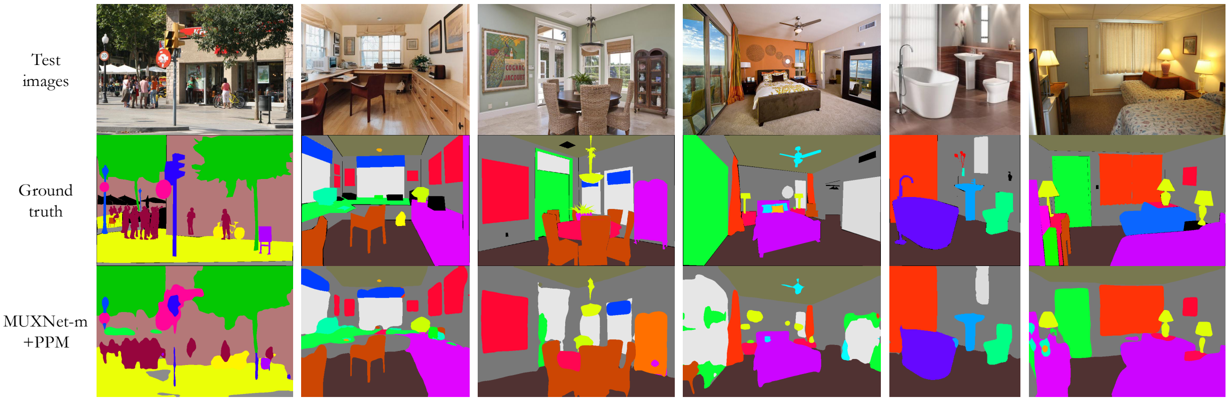
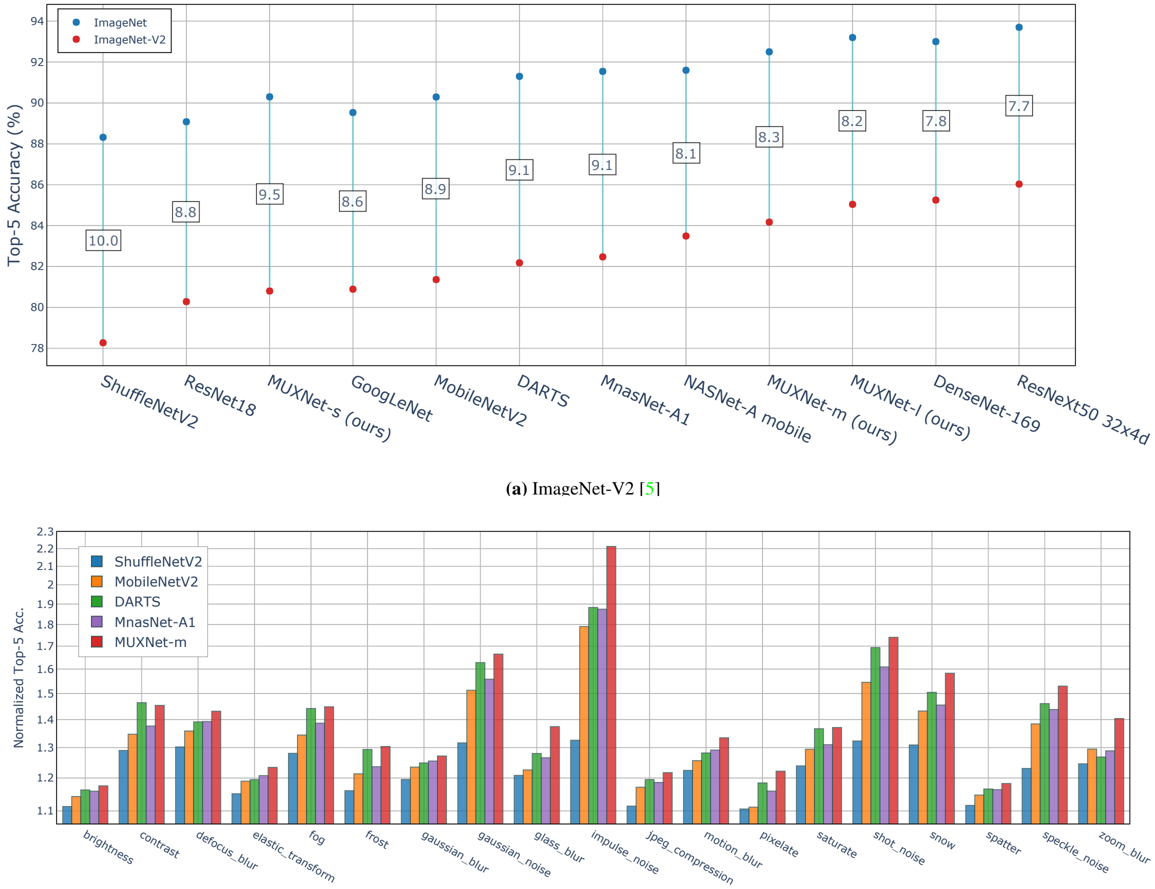


Figure 5: Examples from ADE20K validation set showing the ground truth (2nd row) and the scene parsing result (3rd row) from MUXNet-m. Color encoding of semantic categories is available from here.



(a) ImageNet-V2 [5]

(b) ImageNet-C [3]

Figure 6: (a) Generalization performance on ImageNet-V2 (MatchedFrequency) [5]. Numbers in the boxes indicate the drop in accuracy. (b) Robustness performance on ImageNet-C [3], which consist of ImageNet validation images corrupted by 19 commonly observable corruptions. Following the original paper that proposed ImageNet-C, we normalized the top-5 accuracy by AlexNet's Top-5 accuracy. DARTS is from the author's public Github repository. All other compared models are from Pytorch repository `https://pytorch.org/docs/stable/torchvision/models.html`.

neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[10] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In

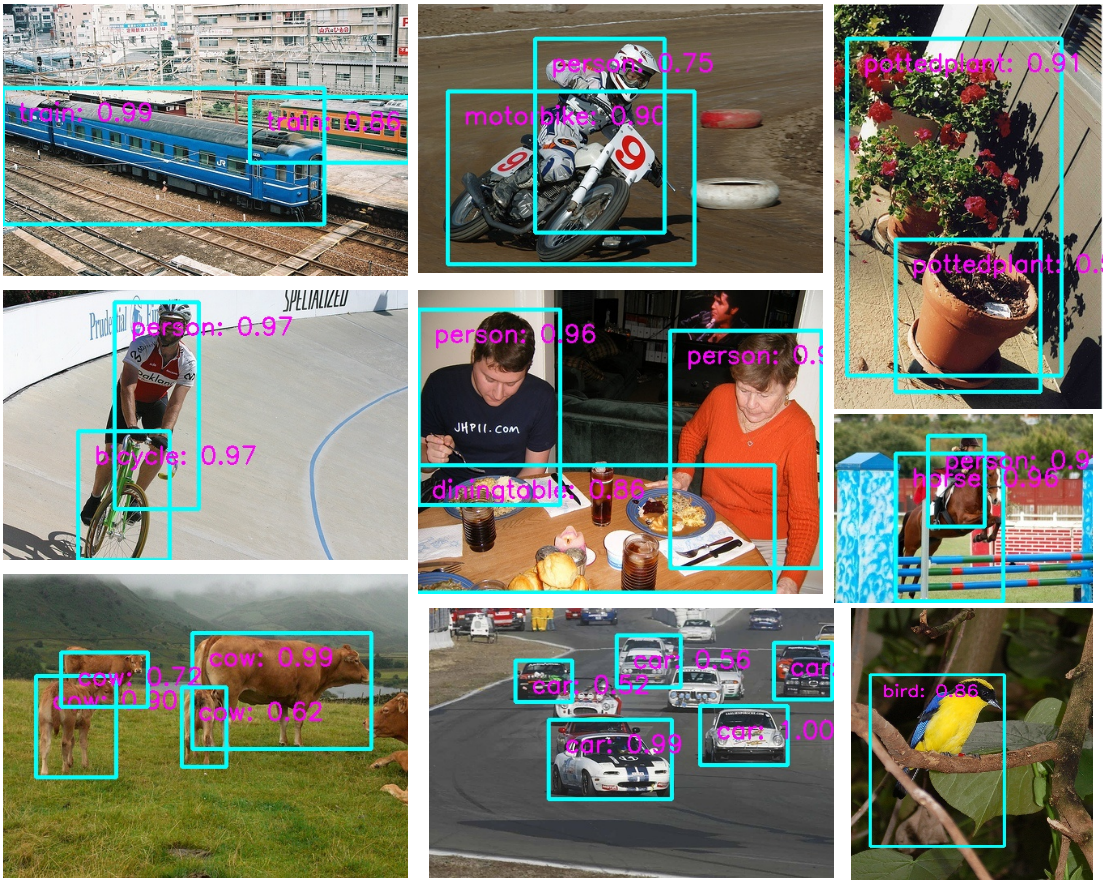


Figure 7: Examples visualizing the detection performance of MUXNet-m on PASCAL VOC 2007 [1].

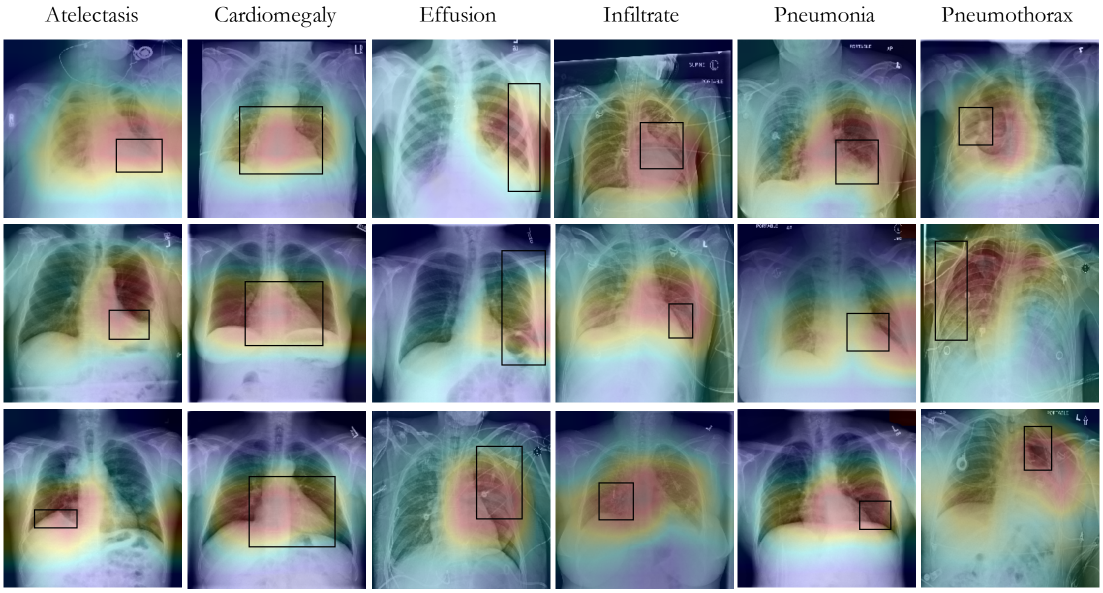


Figure 8: Examples of class activation map [11] of MUXNet-m on ChestX-Ray14 [8], highlighting the class-specific discriminative regions. The ground truth bounding boxes are plotted over the heatmaps.

IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017. 3

[11] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 5

[12] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision*, 127(3):302–321, 2019. 3