

# Mobile-URSONet: an Embeddable Neural Network for Onboard Spacecraft Pose Estimation

Julien Posso\*, Guy Bois\*, Yvon Savaria†

Department of \*Computer Engineering and †Electrical Engineering, École Polytechnique de Montréal  
Montréal (QC), Canada

Email: {firstname.lastname}@polymtl.ca

**Abstract**—Spacecraft pose estimation is an essential computer vision application that can improve the autonomy of in-orbit operations. An ESA/Stanford competition brought out solutions that seem hardly compatible with the constraints imposed on spacecraft onboard computers. URSONet is among the best in the competition for its generalization capabilities but at the cost of a tremendous number of parameters and high computational complexity. In this paper, we propose Mobile-URSONet: a spacecraft pose estimation convolutional neural network with 178 times fewer parameters while degrading accuracy by no more than four times compared to URSONet.

## I. INTRODUCTION

Estimating the relative position and orientation (commonly called Pose estimation) of a known but uncooperative spacecraft from a monocular image is an essential computer vision application that allows improving the autonomy of in-orbit spacecraft operations: formation flying, autonomous docking, satellite maintenance, debris removal, etc... [1]. Debris removal is crucial for the future of low earth orbit operations as the exploitation of this orbit grows, especially with the Starlink and OneWeb constellations. A significant increase of debris will accompany these new constellations. Several research projects aim to solve this problem: RemoveDEBRIS from the Surrey Space Center, Restore-L from NASA, Phoenix program from DARPA [2], or more recently the ClearSpace-1 mission from the European Space Agency (ESA) [3].

Sharma *et al.* were the first to propose using Convolutional Neural Networks (CNNs) for Spacecraft Pose Estimation (SPE) [4]. However, the popularity of CNNs applied to SPE increased in 2019 when ESA and Stanford SLAB (Space Rendezvous Laboratory) organized a competition that brought together 48 participants. Each team proposed a solution that is based at least in part on deep neural networks, now a dominant technique [5]. The competition was based on the SPEED (Spacecraft PosE Estimation Dataset) dataset introduced earlier by Sharma *et al.* [6]. The dataset contains 12000 synthetic images of the Tango satellite to train the models; 2998 synthetic images on which the participants were ranked (synthetic test set); and 300 real images, which allow characterizing the generalization capacity of the proposed models (real test set). A post-mortem webpage dedicated to model predictions evaluation is still available as the labels of the test sets are not provided [7].

Sharma *et al.* [6], Chan *et al.* [8] and the EPFL CVLab team used a 3-step process to estimate spacecraft pose [5]. First, they use an object detection CNN to determine the region of interest and crop the input image. Then, another CNN regress keypoints. Finally, they solve Pose estimation using an off-the-shelf Perspective-n-Point (PnP) solver. Black *et al.* follow the same 3-step process, but instead of using large CNNs, they use a MobileNet-v2 as keypoint regression network. Nevertheless, their method uses a complex pipeline in which the MobileNet-v2 CNN only represents 20.4% of the inference execution time [9]. Moreover, using keypoints limits the method to known spacecraft.

Proença *et al.* proposed URSONet: a straightforward way to solve SPE by regressing position and orientation using a single CNN [1]. It allows to directly optimize ESA competition metrics [5]. They also proposed to deal with the orientation estimation as a soft classification task which significantly improves the results. The orientation is encoded as a Gaussian random variable in a discrete output space so that the CNN learns to predict a mass density function [1]. Then, they used a softmax function and the quaternion averaging technique to predict the orientation [10]. URSONet stands out for its generalization capabilities as Proença *et al.* obtained a good score on both synthetic and real test sets. In addition, as they do not rely on keypoints, their method would be able to generalize to objects with unknown geometry using SLAM [1]. However, it comes at the cost of a tremendous amount of parameters (500 million) and high computational complexity as they use an ensemble method based on three ResNet-101 CNNs [1].

In the spirit of MobileNet proposed by Google [11], we propose Mobile-URSONet: a spacecraft pose estimation convolutional neural network adapted to spacecraft onboard computers. Our lightest model has 178 times fewer parameters while degrading accuracy by no more than four times compared to URSONet.

The outline of this paper is as follows: Section II explains the methodology we adopted to optimize URSONet for embedded systems. Section III presents the experimental results obtained, and section IV concludes the paper.

## II. PROPOSED METHOD

Our analysis is based on the ESA competition evaluation metrics [5] which includes: the mean absolute position error

$e_t$  (in meters), the mean absolute orientation error  $e_q$  (in degrees), and the mean ESA score  $E$  (lower is better) which is evaluated on both the synthetic ( $E_{syn}$ ) and real test sets ( $E_{real}$ ).

#### A. Generalization metric

Many solutions proposed during the ESA competition do not obtain as good a score on the real images as on synthetic images. Kisantal *et al.* explains that it comes from the change in statistical distribution between the synthetic and the real images [5]. However, they do not explain the huge differences in generalization between the different models. We propose a generalization metric to characterize these differences:  $G_{factor}$ . It represents the ratio between the mean ESA score obtained on the real and synthetic test sets:

$$G_{factor} = \frac{E_{real}}{E_{syn}} \quad (1)$$

Table I presents the results and generalization factors of the first four participants of the ESA competition [5] (including the baseline solution of Stanford SLAB [6]), and the latest work published on the domain by Black *et al.* [9]. We observe a wide variability in generalization factors: from 2.7 for the third in the competition (Proença *et al.*) to 39.9 for the winners of the competition (Chen *et al.*). Beyond solving the SPE task using a single CNN (without PnP methods), which is a path to pose estimation of unknown objects, Proença *et al.* solution offers the best generalization factor. As robustness to changes in distribution is a key criterion when integrating such networks into embedded systems, we base our work on that of Proença *et al.*

TABLE I  
ESA SCORE ON THE TEST SETS AND CORRESPONDING  
GENERALIZATION FACTOR

Participants	$E_{synth}$	$E_{real}$	$G_{factor}$
Chen <i>et al.</i> [8]	0.0094	0.3752	39.9
EPFL_cvlab	0.0215	0.1140	5.3
Black <i>et al.</i> [9]	0.0409	0.2918	7.13
Proença <i>et al.</i> [1]	0.0571	0.1555	2.7
Sharma <i>et al.</i> [6]	0.0626	0.3951	6.3

#### B. Neural network architecture

To ensure a good ranking in the ESA competition, Proença *et al.* focused on minimizing the ESA score regardless of the complexity of their model. We focused on the trade-off between the ESA score, the number of parameters, and the computational complexity of our model. Figure 1 shows an outline of Mobile-URSONet, the neural network we propose in this work. In the following paragraphs, we will explain in detail our architectural choices.

The first component of a CNN is the backbone that extracts features from the input image (Fig. 1). Proença *et al.* used a ResNet-101 backbone, a CNN that achieves 77% top-1 accuracy on the standard ImageNet benchmark. We use a MobileNet-v2 backbone that achieves only 72% top-1 accuracy on ImageNet but has 13 times fewer parameters.

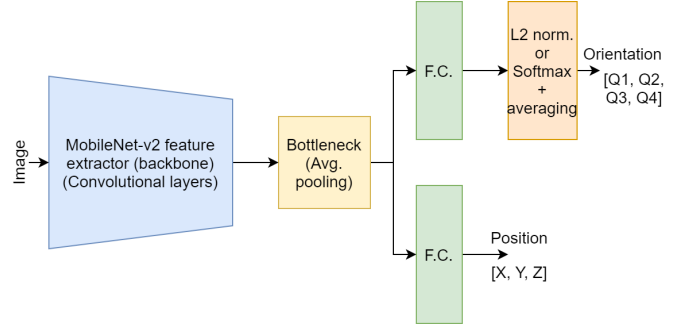


Fig. 1. Mobile-URSONet architecture

Bianco *et al.* show that MobileNet-V2 has a top-1 accuracy density (*i.e.* accuracy per million parameters) more than 10 times better than the ResNet-101 used by Proença *et al.* [12]. They also show that MobileNet-v2 inference requires only 0.3 GFLOPs (floating-point operations), while a ResNet-101 inference requires almost 8 GFLOPs. The computational complexity of our backbone is 26 times less than the original URSONet.

Moreover, MobileNet-v2 is designed to run in real-time on smartphone ARM processors [13]. Table II shows that it also run on weaker ARM processors now used in the space domain. For instance, the Eye-Sat nanosatellite embeds a Zynq 7030 MPSoC (Multiprocessor System on a Chip) developed by Xilinx, which has two ARM A-9 cores [14]. More recent projects consider using MPSoC featuring four ARM A-53 cores [15] [16].

TABLE II  
INFERENCE LATENCY OF MOBILENET-V2 ON VARIOUS ARM  
PROCESSORS [13] [17]

Processor (compiler)	Latency
Qualcomm ARM A-72 (TF-Lite)	75 ms
Xilinx ARM A-53 (TVM)	132 ms
Xilinx ARM A-9 (TVM)	265 ms

The second part of the neural network is the bottleneck (Fig. 1). Many standard CNN architectures uses average pooling such as MobileNets [11] [13] and ResNets [18]. It aims to merge semantically similar features [19] by reducing the spatial dimension of the feature maps. Thus, it reduces the number of parameters and computational complexity of the subsequent layers. Proença *et al.* replaced the standard average pooling layer by a 3x3 convolution layer with a stride of 2 [1]. They show that increasing the number of feature maps in the bottleneck layer decrease the position and orientation error. We summarize their results in table III. The first line of the table (*i.e.* the configuration with eight feature maps) has the same amount of parameters as an average pooling configuration. The table shows that multiplying the number of parameters by six only leads to a 3 degrees improvement in orientation error and 0.24 meters improvement in position error. In our opinion, it is not a suitable trade-off for an embedded system. That is why we kept the original average pooling layer.

TABLE III  
BOTTLENECK SIZE VS. NUMBER OF PARAMETERS, ORIENTATION AND  
POSITION ERROR ON URSONET [1]

# feature maps	# params (M)	Ori err. (°)	Pos err. (m)
8	40	10.2	0.72
128	80	7.8	0.54
512	240	7.2	0.48

Moreover, the number of parameters of the neural network is agnostic to input image resolution thanks to the average pooling layer. Proença *et al.* show that orientation estimation is sensitive to image resolution (*i.e.* increasing image resolution improves accuracy). Thus, increasing image resolution is a more efficient solution to improve accuracy than removing the bottleneck as the power consumption highly depends on memory accesses [20].

The last part of the neural network is the two branches (Fig. 1): the first part estimates orientation, while the other estimates position. Proença *et al.* use two fully connected layers. We use a single fully connected layer to minimize the number of parameters. Our position branch has only 3843 parameters. The number of parameters of our orientation branch depends on the configuration: in regression, it has 5124 parameters; in soft classification, it has between 0.6 and 5 million parameters. It is 39 to 325 times fewer parameters than Proença *et al.*, which has approximately 195 million parameters in their branches.

### C. Loss functions

Target’s position estimation is an easy task solved using direct regression. It allows using ESA metrics as loss functions, as proposed by Proença *et al.*. The same cannot apply to orientation estimation, which explains why we try two methods (*i.e.* regression and soft classification). In ESA metrics, the position error depends on the distance from the target spacecraft; the neural network is increasingly penalized as the target spacecraft is closer. However, this is not the case with the orientation in ESA metrics. We propose a variant of the loss function when orientation estimation is considered a regression task:

$$L_{ori} = \frac{\arccos|\mathbf{q} \cdot \hat{\mathbf{q}}|}{\|\mathbf{t}\|_2} \quad (2)$$

In our experiments, it does not lead to significant improvements compared to the regression loss function proposed by Proença *et al.* Thus, we also estimate orientation using soft classification as Proença *et al.* does. The associated loss function is a standard negative log-likelihood [1].

## III. EXPERIMENTS

### A. Implementation and training details

Networks are trained on one Nvidia Tesla P100, using Pytorch on the SPEED dataset. We use the MobileNet-v2 backbone pre-trained on ImageNet to speed up training. We reserve 15% of the training SPEED dataset as a validation set. Parameters are updated using the SGD algorithm with a

momentum of 0.9. We use a batch size of 32 images resized to 384 \* 240 pixels. The learning rate starts at 0.01 for the first 30 epochs. Then it is decayed to 0.001 for the following 15 epochs. It finishes at 0.0001 for the last five epochs. We employ data augmentation on the training set using OpenCV to rotate the camera across the roll axis for half images with a maximum magnitude of 25°. We also use Pytorch transformations to add a Gaussian blur and randomly change the brightness, contrast, saturation, and hue of training images. When using soft classification, we have set  $\Delta$  which controls the Gaussian width at 3 to act as a regularizer. The number of bins per dimension varies between 8 and 32. All hyper-parameters are tuned on the validation set. Our code is available at [21].

### B. Results

Orientation estimation through regression is done with very few parameters but leads to an average error of 32° on the validation set. Orientation estimation through soft classification implies many more parameters in the orientation branch: it depends on the cube of the number of bins per dimension (we encode rotations as three Euler angles and then convert it to quaternions). However, using soft classification improves the orientation error by a factor of three to five. Table IV shows the results we obtain, with 8 to 32 bins per dimension. The 16-bins model has 2.6 times more parameters than the 8-bins model and an orientation error divided by 2. However, the 32-bins model has six times more parameters than the 16-bins model, with no significant improvement on the orientation error. The 24-bins model also does not bring improvements. It demonstrates a saturation effect on orientation error when increasing the number of bins per dimension. In addition, we notice that the 32-bins model is much more prone to overfitting. Based on these results, we believe that going beyond 16 bins per dimension is not worth it as we focus on the trade-off between the orientation error and the number of parameters of the CNN.

TABLE IV  
EFFECT OF THE NUMBER OF BINS PER DIMENSION ON ORIENTATION  
ERROR AND THE NUMBER OF PARAMETERS

# bins per dim.	# params (M)	$e_q$ train (°)	$e_q$ valid (°)
8	2.8	9.74	11.3
12	4.4	5.69	7.43
16	7.4	4.5	6.29
24	19.9	4.18	6.12
32	44.2	4.92	7.29

Table V shows the orientation and position error of our selected models (from 8 to 16 bins) compared to [1]. As we said before, target position estimation is an easy task solved using direct regression. Position error is the same in all our experiments. It seems to depend only on the number of parameters of the backbone, as increasing the number of parameters in the position branch only increases overfitting. Our backbone uses 13 times fewer parameters than Proença’s backbone while degrading position error by no more than three times. Using soft classification, orientation error highly

depends on the number of parameters of the orientation branch. Proença *et al.* used between 24 and 64 bins per dimension but only published the orientation error for their 24-bins model. The high number of parameters of URSONet causes overfitting that Proença *et al.* mitigates using a data augmentation strategy more refined than ours. It explains why they achieve a better orientation error of  $4.0^\circ$ , while our best model achieves only  $6.3^\circ$  orientation error.

TABLE V  
ORIENTATION AND POSITION ERROR ON VALIDATION SET COMPARED TO PROENÇA *et al.*

# bins per dim.	$e_q$ ( $^\circ$ )	$e_t$ (m)
Ours (8 bins)	11.3	0.54
Ours (12 bins)	7.43	0.51
Ours (16 bins)	6.29	0.56
Proença <i>et al.</i> (24 bins) [1]	4.0	0.17

Figure 2 shows the position and the orientation error of our 12-bins model on the validation set as a function of distance to the target. We see that the position error highly depends on the distance to the target satellite. We observe the same property for the orientation error. It is surprising as the loss function we use does not involve the distance with the target spacecraft (in soft classification configuration). High position and orientation errors appear when the target spacecraft is more than 10 meters apart from the camera. The number of outliers is small enough and occurs only when the target spacecraft is more than 15 meters apart from the camera. The closer the target spacecraft is, the more confidence we can have in the predictions of our model. It is a crucial property for autonomous docking or debris removal applications. It also offers an avenue to reduce both the position and the orientation error: zooming and cropping the image around the target instead of resizing the whole image as we do now.

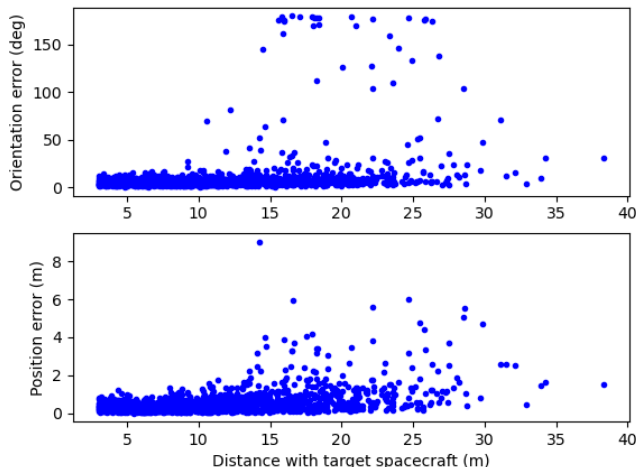


Fig. 2. Position and orientation error by distance for our 12-bins model

Table VI summarizes the results of our models compared to our competitors. We propose the most lightweight spacecraft pose estimation models: ranging from 2.2 to 7.4 million parameters while keeping a good score on the synthetic test

set and a good generalization factor. Our 8-bins model has 178 times fewer parameters while degrading the ESA score by no more than four times compared to URSONet. It has an accuracy density (*i.e.* ESA score per parameters) 139 times higher than the original URSONet. However, we notice that increasing the number of bins per dimension leads the model to overfit the synthetic images. Proença *et al.* demonstrated that a 2.7 generalization factor is achievable while using 24 to 64 bins per orientation dimension. We believe we still have some margin to improve the generalization capabilities of our models by using a more advanced data augmentation technique and by investing more effort in hyper-parameter tuning.

TABLE VI  
ESA SCORE ON TEST SET AND GENERALIZATION FACTOR

Participants	# params (M)	$E_{synth}$	$E_{real}$	$G_{factor}$
Black <i>et al.</i> [9]	6.9	0.0409	0.2918	7.13
Sharma <i>et al.</i> [6]	11.2	0.0626	0.3951	6.31
Proença <i>et al.</i> [1]	500	0.0571	0.1555	2.72
ours (regression)	2.2	0.6160	0.7997	1.30
ours (8 bins)	2.8	0.2520	0.7868	3.12
ours (12 bins)	4.4	0.2104	1.2231	5.81
ours (16 bins)	7.4	0.1947	1.2074	6.20

### C. Future work

Future works will explore in-depth embeddability by using quantization and pruning. It will further optimize the memory footprint of the parameters and the inference computational complexity of Mobile-URSONet. We plan to deploy these models on promising commercial chips for future satellite onboard computers, such as the Xilinx MPSoCs featuring ARM-A53 cores and programmable logic.

## IV. CONCLUSIONS

In this paper, we analyzed URSONet, a popular neural network used for spacecraft pose estimation that seems hardly compatible with the constraints of onboard spacecraft computers. We found that three architectural choices have a dominant effect on both the number of parameters and the computational complexity of the CNN: the backbone, the bottleneck size, and the number of bins per dimension while predicting orientation using soft classification. By analyzing trade-offs for each of these three architectural choices, we were able to propose Mobile-URSONet, a mobile version of URSONet in the spirit of Google MobileNets. We showed that Mobile-URSONet achieves accuracy close to URSONet, while keeping the number of parameters and computational complexity compatible with the constraints of spacecraft onboard computers.

## V. ACKNOWLEDGMENTS

The authors thank the Canadian Space Agency, MITACS and Space Codesign Systems for their financial contributions.

## REFERENCES

- [1] P. F. Proença and Y. Gao, "Deep learning for spacecraft pose estimation from photorealistic rendering," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 6007–6013.
- [2] "Kelvins - Pose Estimation Challenge." [Online]. Available: <https://kelvins.esa.int/satellite-pose-estimation-challenge/home/>
- [3] "ESA commissions world's first space debris removal." [Online]. Available: [https://www.esa.int/Safety\\_Security/Clean\\_Space/ESA\\_commissions\\_world\\_s\\_first\\_space\\_debris\\_removal](https://www.esa.int/Safety_Security/Clean_Space/ESA_commissions_world_s_first_space_debris_removal)
- [4] S. Sharma, C. Beierle, and S. D'Amico, "Pose estimation for non-cooperative spacecraft rendezvous using convolutional neural networks," in *2018 IEEE Aerospace Conference*, Mar. 2018, pp. 1–12.
- [5] M. Kisantal, S. Sharma, T. H. Park, D. Izzo, M. Märtens, and S. D'Amico, "Satellite Pose Estimation Challenge: Dataset, Competition Design and Results," *arXiv:1911.02050 [cs]*, Apr. 2020, arXiv: 1911.02050. [Online]. Available: <http://arxiv.org/abs/1911.02050>
- [6] S. Sharma and S. D'Amico, "Pose Estimation for Non-Cooperative Rendezvous Using Neural Networks," *arXiv:1906.09868 [cs]*, Jun. 2019, arXiv: 1906.09868. [Online]. Available: <http://arxiv.org/abs/1906.09868>
- [7] "Kelvins - Pose Estimation Challenge post mortem - Home." [Online]. Available: <https://kelvins.esa.int/satellite-pose-estimation-challenge/leaderboard/post-mortem-leaderboard>
- [8] B. Chen, J. Cao, A. Parra, and T.-J. Chin, "Satellite Pose Estimation with Deep Landmark Regression and Nonlinear Pose Refinement," *arXiv:1908.11542 [cs]*, Aug. 2019, arXiv: 1908.11542. [Online]. Available: <http://arxiv.org/abs/1908.11542>
- [9] K. Black, S. Shankar, D. Fonseka, J. Deutsch, A. Dhir, and M. R. Akella, "Real-Time, Flight-Ready, Non-Cooperative Spacecraft Pose Estimation Using Monocular Imagery," *arXiv:2101.09553 [cs]*, Jan. 2021, arXiv: 2101.09553. [Online]. Available: <http://arxiv.org/abs/2101.09553>
- [10] F. L. Markley, Y. Cheng, J. L. Crassidis, and Y. Oshman, "Averaging Quaternions," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 4, pp. 1193–1197, Jul. 2007. [Online]. Available: <https://arc.aiaa.org/doi/10.2514/1.28949>
- [11] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv:1704.04861 [cs]*, Apr. 2017, arXiv: 1704.04861. [Online]. Available: <http://arxiv.org/abs/1704.04861>
- [12] S. Bianco, R. Cadene, L. Celona, and P. Napoletano, "Benchmark Analysis of Representative Deep Neural Network Architectures," *IEEE Access*, vol. 6, pp. 64 270–64 277, 2018, arXiv: 1810.00736. [Online]. Available: <http://arxiv.org/abs/1810.00736>
- [13] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. Salt Lake City, UT: IEEE, Jun. 2018, pp. 4510–4520. [Online]. Available: <https://ieeexplore.ieee.org/document/8578572/>
- [14] F. Apper, A. Ressouche, N. Humeau, M. Vuillemin, A. Gaboriaud, F. Viaud, and M. Couture, "Eye-Sat: A 3U student CubeSat from CNES packed with technology," p. 10.
- [15] A. Pérez, A. Rodríguez, A. Otero, D. G. Arjona, Á. Jiménez-Peralo, M. Á. Verdugo, and E. De La Torre, "Run-Time Reconfigurable MPSoC-Based On-Board Processor for Vision-Based Space Navigation," *IEEE Access*, vol. 8, pp. 59 891–59 905, 2020, conference Name: IEEE Access.
- [16] C. M. Fuchs, P. Chou, X. Wen, N. M. Murillo, G. Furano, S. Holst, A. Tavoularis, S.-K. Lu, A. Laat, and K. Marinis, "A Fault-Tolerant MPSoC For CubeSats," in *2019 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*, Oct. 2019, pp. 1–6, iSSN: 2377-7966.
- [17] T. Moreau, T. Chen, L. Vega, J. Roesch, E. Yan, L. Zheng, J. Fromm, Z. Jiang, L. Ceze, C. Guestrin, and A. Krishnamurthy, "A Hardware-Software Blueprint for Flexible Deep Learning Specialization," *arXiv:1807.04188 [cs, stat]*, Apr. 2019, arXiv: 1807.04188. [Online]. Available: <http://arxiv.org/abs/1807.04188>
- [18] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015, arXiv: 1512.03385. [Online]. Available: <http://arxiv.org/abs/1512.03385>
- [19] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015, number: 7553 Publisher: Nature Publishing Group. [Online]. Available: <https://www.nature.com/articles/nature14539>
- [20] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb. 2014, pp. 10–14, iSSN: 2376-8606.
- [21] "Mobile-URSONet code (GitHub)." [Online]. Available: <https://github.com/possoj/Mobile-URSONet>