Resource Constrained Semantic Segmentation for Waste Sorting

Mehrbod Nowrouz
Politecnico di Torino
M.Sc in Data Science and Engineering
Mehrbod.Nowrouz@studenti.polito.it

Mohammad Masoud Khalilian
Politecnico di Torino
M.Sc in Data Science and Engineering
MohammadMasoud.Khalilian@studenti.polito.it

Abstract

This project was carried out to explore and implement tiny deep learning models in the context of waste management and sorting. The first three models were implemented and tested on the project's target dataset in binary segmentation and instance segmentation manners to establish a baseline of all three models for future experiments. The earlier experiments consist of improvements applied to the model itself and the dataset through, in turn, variations of the models' encoder, decoder sizes, and multiple data augmentation techniques applied to the dataset. Later experiments include various personal contributions of the authors, one of which is knowledge distillation, which was experimented on BiSeNet. Dataset's class imbalances were also dealt with by exploring loss functions that tended to stabilize the is- sue. ICNet's large model size was also addressed through pruning and quantization methodologies. The initial optimizer used in the project has also undergone trials with different algorithms. Eventually, with the better versions of all models being ready, the models were evaluated on a well-known dataset such as TACO to showcase the models' performances on a different dataset distribution. The code is available at https://github.com/masoud-khalilian/mldl-waste

1. Introduction

The growing population of the earth has led to frequent problems that are targeting nature's natural cycles, among which growing mass waste production is of high impact and requires humanity to minimize its damage efficiently to nature as it cannot recycle these wastes independently. Therefore, categorizing the waste into different categories to recycle them with respect to their features has been the solution so far. However, recycling of the waste has been performed by humans, a solution that suffers from low productivity and increased health risks. During the past two decades, optical sorters have become popular in industrial material recovery facilities (MRFs). The devices employ a combination of lights and sensors to illuminate and capture images of objects. They use compressed air to move rejected objects to different bins. This technology has difficulty dealing with heavy items. As a means to overcome the limitations of optical sorters, a new trend involves the use of robotic technology. This technology uses Artificial Intelligence and visuals in order to separate the waste. This project also attempts to utilize semantic segmentation models required for waste management and sorting to automate the process of separating the waste. Since this task's target industry is IoT devices that work together through visuals of a running belt, the deep learning models not only should be able to address the accuracy aspect of waste sorting, they are also required to be relatively small to fit into the resource constraints of such devices.

2. Related Works

Semantic Segmentation is the act of classifying every pixel of a given image into the existing classes. In other words, it separates the image into different parts, with each part belonging to a certain class in the original image. This is usually done by assigning a certain number/color to each class that is expected to be found. This process plays a key role in waste sorting as it lets the robots recognize different types of waste from each other in an image (or video) and act with respect to their indicated classes. However, waste sorting is usually done by IoT devices which are highly resource-constrained and would also require these models to be light in order to be implementable on such devices. Subsequently, the concept of Tiny Machine Learning comes into play which exclusively focuses on machine learning models that are small enough to be operable on tiny devices. Example usages of Tiny ML in waste sorting can be seen in [5], which uses it in robotic garbage collectors that tend to search and gather random garbage that is left in the urban areas (Streets, Parks, etc.) and keep them in their waste-type separated mobile storage until later unloaded when they are full. The other would be a more common usage of it in waste processing plants [3] in which vacuum technology is utilized to grab and separate the garbage through visual aids. While the prior used the TACO [7], GarbageNet [8] datasets for training, the latter refused to use these public datasets, since it claimed that these datasets were mostly focused on urban images of waste and would not therefore, suit models that are used in industrial setups as by the time the garbage reach these places, they are strongly deformed, dirty and piled up on top of one another when carried on conveyor belts. Alternatively, they proceeded to create a Synthetic Dataset through mostly data augmentation that was generated by them through the following steps: They first took 4 to 5 different images for each category of waste and applied three geometric transformations (translation, rotation, and scaling). As a result, they had many separate images of different waste with a black background. Afterward, pairs of separate images were randomly chosen and put together on colorful backgrounds in an occasional overlapping manner. The resulting dataset was ”cases with multiple, possibly overlapping objects across a range of diverse backgrounds”. There are also three different models used, which are as follows:

2.1. ENet [6]

The general idea of the publishers of this model revolves around the fact that average semantic segmentation models are quite heavy, and they claim to have produced a model that is ”18x faster, requires 75x less FLOPS, has 79x less parameters, provides similar or better accuracy to existing models”. They have accomplished such results by moving a 512x512 image through an initial block consisting of two branches of a MaxPooling and a 3x3 stride 2 convolution which are then concatenated to produce an overall of 16 feature maps. Then goes through a series of bottleneck modules. There are two branches of the main one being a ResNet followed by maxpooling and padding and the other being a series of layers among which conv is either a regular, dilated, or full convolution (also known as deconvolution) with 3×3 filters, or a 5×5 convolution decomposed into two asymmetric ones. There are a few notable design choices, such as ENet first two blocks heavily reducing the input size and using only a small set of feature maps. The idea behind it is that visual information is highly spatially redundant and thus can be compressed into a more efficient representation. Another design choice is saving indices of elements chosen in max pooling layers and using them to produce sparse upsampled maps in the decoder to tackle the spatial information drop as a result of downsampling. Another approach is the usage of larger encoders and smaller decoders which is motivated by the idea that the encoder should be able to work in a similar fashion to original classification architectures and the decoder should be only in charge of upsampling the output of the encoder, only fine tuning the details. It is also worth mentioning that choosing to apply the pooling operation in parallel with convolutions (instead of before or after it) leads to speeding up inference time in the initial 10 blocks. Based on [2], they understood that an n x n convolution can be decomposed to two smaller 1 x n and n x 1 filters called asymmetric convolution, which helps with increasing the diversity of functions learned by blocks and leads to increasing the receptive field. They have further utilized this practice by using projection, convolution, and projection to decompose large convolution layers, resulting in a great reduction of the number of parameters, making them less redundant, and subsequently large speedups.

2.2. ICNet [11]

The network architecture of ICNet consists of three branches, each having completely different inputs and layers. There are 3 inputs entering the model at the same time, each belonging to its own branch, with the bottom one being a full-resolution image (1024 x 2048) while the middle and top ones being, in turn, a 1/2 and 1/4 downsampled version of the same image. However, since these branches are supposed to work in parallel, the middle and bottom branches, despite having a larger input, include lightweight CNNs. On the contrary, the top branch consists of a PSPNet with a downsampling of 8, which results in a 1/32 feature map. Although the top branch is based on a full segmentation backbone, the input resolution is low, resulting in limited computation. Even for PSPNet with 50+ layers, inference time and memory are 18ms and 0.6GB for the large images. The idea is to get high-quality segmentation from medium and high-resolution branches to help recover and refine the coarse prediction. Though some details are missing and blurry boundaries are generated in the top branch, it already harvests most semantic parts. Then all the extracted feature maps will be, in turn, from top to bottom fused through a cascade-feature-fusion unit and trained with cascade label guidance. These two features are rather too complex to fit into a few lines, but a brief explanation would be that cascade-feature-fusion tends to upsample the smaller feature map (resulting from the upper branch) to match the lower branch, then both feature maps are normalized and accumulated followed by a relu layer. The label guidance system also tends to enhance the learning procedure in each branch through utilizing different scale ground truth labels to guide the learning stage of low- medium and high-resolution branches, with each branch having appended a weighted softmax cross entropy loss.

2.3. BiSeNet [9] [10]

BiSeNet or, more specifically, Bilateral Segmentation Network’s perception of average quick semantic segmentation models is that they, in search of low inference times, tend to sacrifice low-level details and thus the accuracy. To tackle such a trade-off, they propose a model that, on the

Model	Object (mIoU)	GFLOPS	#Parameters	Model Size (MBs)
ICNet	75.37%	452.791	26.245M	108.2
BiSeNet	77.26%	23.614	0.621M	2.57
ENet	85.66%	25.430	0.363M	1.64

Table 1. Binary Segmentation results of ICNet, BiSeNet and ENet models and their characteristics

Model	Paper (mIoU)	Bottle(mIoU)	Aluminum (mIoU)	Nylon (mIoU)	Avg mIoU
ICNet	69.82%	73.85%	82.45%	73.09%	79.59%
BiSeNet	78.23%	78.61%	81.78%	75.57%	81.24%
ENet	81.57%	85.59%	83.25%	86.33%	87.08%

Table 2. Instance Segmentation of ICNet, BiSeNet and ENet and the corresponding results on segmentation of Paper, Bottle, Aluminum and Nylon

one hand, tries to capture low-level details and generate high-resolution feature maps using a Detailed Branch consisting of wide channels and shallow layers. On the other hand, obtain high-level semantic context through a Semantic Branch that includes narrow channels and deep layers. Both of these layers provide completely different information, and a normal combination of these two branches would lead to bad performance and hard optimization. Therefore, these branches are merged using a Guided Aggregation Layer. This layer employs the contextual information of the Semantic Branch to guide the feature response of the Detailed branch and does Vice versa for the Semantic Branch, then accumulates both resulting feature maps into one, then applies a 3x3 convolution at the end.

3. Experiments

3.1. Binary Segmentation Experiments

The dataset ReSort-IT and the Starting Codes were downloaded from the original project repository. Then the models were implemented to match the Starting Codes. All the models were run for 200 epochs with a weight decay of 2e-4 and a learning rate of 5e-4, which decayed every epoch for about 0.995. The results are all reported in table. 1. As you can see ICNet and BiSeNet models are quite similar, with an mIoU of 75% and 77%, with Enet having a significantly better performance of nearly 8-10%. Enet and BiSeNet clearly suffice the tininess criteria of being below 10 Mbs. However, it seems that ICNet's Pytorch implementation is far heavier than expected, with a size of approximately 108 MBs. It is also notable that the GFLOPS associated with ICNet is significantly higher than an average tiny machine learning model that performs semantic segmentation. These problems will be dealt with later during the Quantization and Prunning section.

3.2. Instance Segmentation Experiments

After Binary Segmentation Experiments, the model was updated to support Instance Segmentation in order to differentiate the type of objects (Paper, Bottle, Aluminum, and Nylon) from each other. Subsequently, masking and loss functions were updated to carry out this task. As demonstrated in Table. 2, the results are similarly separated by models, recording only the mIoU of different objects and the overall mIoU.(Spaces with no objects were also included have also influenced the average mIoU calculation)

Evidently, Enet again showed a better average mIoU with 87%, followed by BiSenNet's 0.81% and lastly ICNet's 0.79. The difference in performance of these models is more evident in detecting paper, with ICNet being remarkably lower than the other two with a mIoU of 69% compared to that of Enet and BiSeNet's approximate 80%. The gap between ICNet and Enet becomes even more apparent in detecting Nylon and bottle with the model's mIoU's nearly being in turn 73% and 86%. However, this contrast is less evident in Aluminum detection, with all three models laying in the range 81 to 83%.

4. Improvements and further analysis

4.1. Encoder and Decoder Sizes

As an exploration, the encoder and decoder sizes of all above three models were modified in order to experiment with how encoder and decoder sizes affect the output of these models. Four different alternatives were introduced per each model. Each alternative was coded with the letters D (Decrease), I (Increase), and S (Standard). The prior letter refers to the encoder and the latter to the decoder sizes. Observation of their performance is demonstrated in table. 3. Further details regarding these alternatives' characteristics and performances:

Enet-D-D: is a version of Enet in which the encoder sizes have decreased to 13 bottlenecks, and one bottleneck from

Model	Paper (mIoU)	Bottle(mIoU)	Aluminum (mIoU)	Nylon (mIoU)	Avg mIoU
Enet-D-D	69.82%	61.96%	59.78%	61.96%	67.81%
Enet-S-I	86.29%	89.38%	89.31%	90.12%	90.83%
Enet-I-S	53.31%	64.37%	74.80%	50.23%	68.31%
Enet-I-I	84.20%	88.89%	88.15%	84.36%	88.90%
BiSeNet-D-D	36.00%	27.68%	48.19%	29.92%	46.43%
BiSeNet-D-I	64.04%	71.90%	75.22%	69.17%	71.58%
BiSeNet-I-D	61.51%	66.41%	66.25%	73.85%	67.48%
BiSeNet-I-I	20.46%	5.96%	40.64%	26.84%	32.69%
ICNet-D-D	82.70%	86.66%	86.80%	86.11%	88.11%
ICNet-D-I	79.07%	79.79%	78.43%	80.22%	83.11%
ICNet-I-D	77.27%	81.25%	80.93%	80.57%	83.65%
ICNet-I-I	76.56%	79.46%	81.35%	81.54%	83.41%

Table 3. Alternatives of initial three models and their corresponding results on the initial four classes

sections 4 and 5 of the decoder has been removed. This alteration led to a vanishing gradient with which the model got stuck at 0.6781 despite the fact that no activation function and batch normalization were independently removed which supposedly occurs due to encoders not managing to capture features, since smaller encoders would fail to capture complex features. Therefore, in the next model, the standard encoders were kept, and only the decoder was altered.

Enet-S-I: is a version of Enet in which the encoder was not changed and decoder bottlenecks increased to 12 layers. Despite the paper's conclusion on decoders being only a means of upsampling and are not required to be large, a larger decoder produced better results compared to the original model. This should not be surprising since the decoder size was purposely reduced in the original paper to decrease the model size and performance time. This solution led to better accuracy of 1-2 %; however, increased model size by nearly 16 %.

Enet-I-S: is a version of Enet in which the number of bottlenecks was increased by 4 in section 1 and 8 in sections 2 and 3. The decoder was not changed in this version. This alternate version significantly increased the model size and yet failed to produce better results.

Enet-I-I: is a version of Enet in which the encoder is similar to Enet-I-S and the decoder is similar to Enet-S-I. The results were fairly similar to the original Enet in spite of the fact that the model size was nearly doubled. This could have occurred due to the fact that the original model would suffice to learn the small dataset of this project, and increasing the model size would not necessarily improve the performance. While in the above, encoder and decoder sizes were mostly altered in terms of layers, in the following models, encoder and decoder sizes were mostly changed in terms of channel sizes.

BiSeNet-D-D: is a version of BiSeNet in which many channels were removed from various parts of the model. The results are expectedly disappointing as such removal of channels would lead to the model not being able to capture complex patterns and features.

BiSeNet-D-I: is a version of BiSeNet in which the characteristics of the encoder are the same as BiSeNet-D-D's but 4 layers were added to the feature fusion section. This model's accuracy is the best of all alternatives, yet it underperforms the standard model by 10%, which is reasonable, as previously mentioned.

BiSeNet-I-D: is a version of BiSeNet in which many layers and channels were increased in it, especially in the xception39's part. However, many convolution layers were removed from feature fusion. The accuracy of this alternate version is close to the previous one. It is believed such poor results have appeared due to the fact that increasing channels usually will force the addition of down-sampling layers, which would reduce spatial resolution.

BiSeNet-I-I: is a version of BiSeNet that is similar to BiSeNet-I-D, but instead of removing layers from feature fusion, some were added. The results shockingly deteriorate by more than 30%.

ICNet-D-D: is a version of ICNet in which the input of the sub1 branch was not down-sampled to 1/8, and cascade feature fusion was also reduced in size. The idea behind it was to find out how much accuracy was sacrificed for a faster model. As you can see, the model's performance notably increased from nearly 80 % to approximately 89%. However, this would remarkably slow the model in time of use. It even nearly doubled the training time. However, since this would lead to fewer layers, the model size was decreased by 2 MBs.

ICNet-D-I: is a version of ICNet in which layers were removed from all three branches while cascade feature fusion channels were enlarged.

ICNet-I-D: is a version of ICNet in which layers were increased in sub1 branch from 3 to 6 while output channels of ICHead were reduced in size. The results were slightly

Model	Augmentation	Average(mIoU)
ENet	T1	89.8%
	T2	90.39%
	T3	87.6%
BiSeNet	T1	73.6%
	T2	82.5%
	T3	64.0%
ICNet	T1	79.5%
	T2	85.10%
	T3	67.8%

Table 4. Effect of the three experimented Data Augmentations on each model

better since this branch was purposely light due to having a large input while it is of high potential.
ICNet-I-I: is a version of ICNet in which its encoder layers are similar to ICNet-I-D and its decoders are similar to ICNet-I-D's. The results were almost the same, with the same reasoning as the previous alternatives.

4.2. Data Augmentation

Data Augmentation refers to any sort of expansion of the dataset at hand through applying transformation functions in order to create diversity in our dataset. In other words, instead of training the model on the same dataset, we briefly modify our data every epoch in a random manner. This leads to avoiding over-fitting and results in better generalization of the model, which subsequently means the model would most likely have a better accuracy. It is notable that this methodology helps avoid data collection and labeling costs, as performing the same task through adding new data would be costly. To perform this task, we proposed three different alternative augmentations, and after comparing them for each model, we chose the transformation that had a higher impact on each model's accuracy. The three augmentations (T1, T2 and T3) are as follows:

- **"T1"**: First, an increasing scale of 15% that applies a bilinear interpolation function on the empty area of the image and nearest neighbor on the masks followed by a random crop of the desired image size. Then, as the waste processing plants could have various lighting, a randomized brightness jittering with a range of 60% to 140% and as waste tends to be quite diverse in terms of colors, a randomized contrast, saturation, and hue jittering of in turn 40% to 160%, 70% to 130% and 80% to 120% were also applied. Eventually, a random horizontal flip was applied to further diversify the dataset.

- **"T2"**: Includes all of the above with the exception of a random rotation applied before a random cropping image.

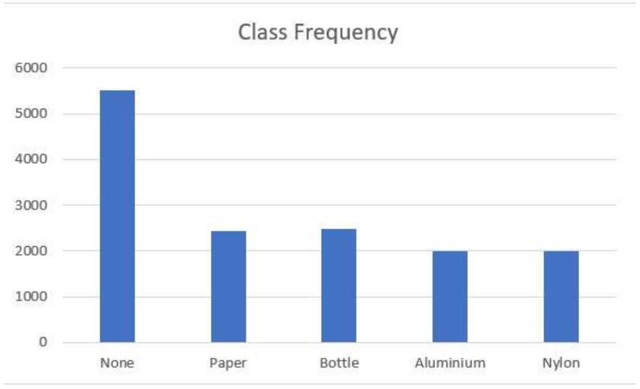

Figure 1. The class frequencies show a portion of imbalance in our data.

- **"T3"**: Includes all of the other two transformations with the exception that random crop was replaced with a random resized crop of ratio 50% to 100% resized back to the image default size with hopes of applying even more diversity to the dataset. However, it was suspected that this transformation might lead to fewer objects appearing during training and result in reduced accuracy.

As demonstrated in Table 4, T2 performed the best among the three and thus was used in the following experimentations as well.

5. Personal Contribution

5.1. Knowledge Distillation

Knowledge distillation is the act of transferring knowledge from a larger model to a smaller one. More accurately, since smaller models would usually have a poorer performance compared to larger models given the same data and computational resources, the larger model (also referred to as teacher) will be trained on the dataset and produce the softmax output while the smaller model (student) would instead of being trained on the dataset's ground truth labels, will be trained on the softmax of much larger models and would learn to imitate the larger model's performance on the dataset. However, this project's teacher and student models are the same model with different backbones. BiSeNet[ResNet18] acts as the teacher, and BiSeNet[Xception39], which is also the baseline model, plays the role of the student. The BiSeNet[ResNet18] managed to reach a 75% average mIoU, and BiSeNet[Xception39] achieved a 72% average mIoU.

5.2. Loss functions

As mentioned earlier, every dataset might be under the effect of some classes being more frequent than others,

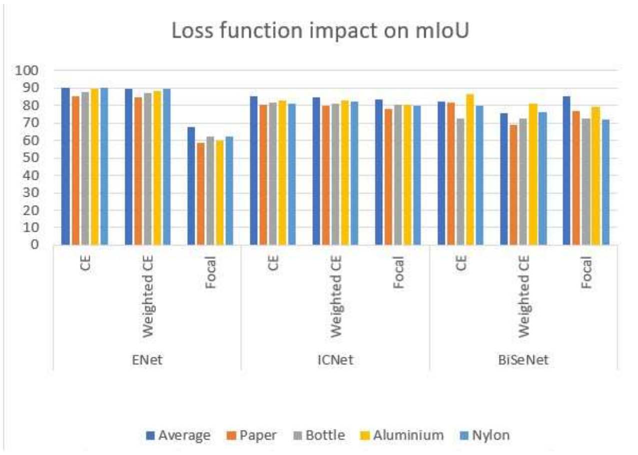

Figure 2. The mIoU results of all models with different losses can be seen above.

and this problem will lead to models being less accurate on classes that are less frequent. There are several ways of overcoming this issue, such as re-sampling the dataset into one that is more balanced in terms of class frequency. This project has also addressed the issue but with a different methodology which addresses the issue through alteration of loss functions. These loss functions were mathematically designed to fix this issue. Two different loss functions were tested:

1.Weighted Cross Entropy: It is similar to the default Cross Entropy Loss. However, a weight is assigned to each class to prioritize the minority class. This weight is usually determined by inversing the class count in the dataset and assigning it as the weight of the class. This leads to the minority class having a larger weight which leads to larger loss values produced by these classes, forcing the algorithm to prioritize the minority.

2.Focal Loss with Alpha [1] [4]: Another approach is to balance the class weights based on their classification difficulty. Therefore, the focal loss is an algorithm that uses the prediction accuracy of classes in the weights assigned to them. The better the accuracy, the simpler it is to classify it, and thus the lower the weight would be and vice versa. However, to further increase the user's control over the loss function, a version of focal loss exists that also incorporates custom class weights into the algorithm as well which is be the focal loss version utilized in this project. Based on [4], a gamma = 2.0 showed better results, and thus it was used in this project as well. Accordingly, the class frequencies were calculated and shown in 1. As demonstrated, there is a slight imbalance in the dataset. To overcome this issue, the above two loss functions were put into practice, and the results were shown in 2. Based on the figure, ENet and BiSeNet showed more balanced results for focal and weighted cross-entropy loss. However, both didn't manage to outperform CE except for BiSeNet's case, in which focal loss managed to show better results. It was also discovered that focal loss tends to produce vanishing gradients during backpropagation. In our case, Enet's focal loss implementation led to the model being stuck at 67.8 % for over 30 epochs.

5.3. Quantization and pruning

Quantization, as the name suggests, refers to the act of mapping continuous and infinite values into smaller sets of finite values, which in the case of deep learning, is applied to the neural network weights. For instance, if there is a weight that is a float, it rounds the float into an integer. This significantly reduces the model size as a float is represented with 32 bits while an integer is represented in 8 bits. In this example, with a rather simple solution, the model size was reduced by a factor of 4. This is a rather cheap approach to tackle model size problems. Despite the fact that this approach increases efficiency, it reduces precision and introduces noise. However, this noise is rather small compared to the efficiency it provides the target model. On the other hand, there also exist countless parameters in the models that do not add any value to model predictions. The removal of these parameters would minimally reduce the accuracy while providing a noticeable reduction in memory, batter, and power consumption. This concept is broadly referred to as pruning. According to above two methodologies, a Post-Training Dynamic/Weight-only Quantization and a global unstructured pruning based on L1 norm of 90% on instances of nn.conv2d and nn.linear were applied which overall managed to reduce model size for about 22% from 120 MBs to 98MBs and GFlops dropping from 452 GFLOPS to only 162 GFLOPS while keeping a 84% average mIoU. Unfortunately, It was realized that unstructured punning does turn weights to 0. However, it was found that most hardware and frameworks are not yet able to accelerate sparse matrices' computation. In other words, no matter how many weights are set to 0, it would not impact the actual cost of the network. Therefore, many tend to do so manually in a structured way. In this project, initial structured L1 and L2 norm pruning were applied to all layers in order to see if this PyTorch library would affect the model size at all. It was realized that no difference occurred despite the change in pruning function.

6. Conclusion

In this paper, the performance of three tiny machine learning models has been showcased, and the importance of encoder and decoder sizes and how they would affect the trade-off between accuracy and inference time has been demonstrated. In the following, it was shown that data augmentation could diversify a small dataset to maximize the value that could be extracted from it. It was also discovered that loss functions could tackle the issue of class imbalance and how algorithms could force the model to focus on objects that are more complex to classify in order to balance the model's prediction accuracy for all objects. Knowledge distillation was also utilized in the case of BiSeNet, to see how a larger model could affect a smaller model's learning process. Finally, quantization and pruning were put into practice to decrease ICNet's initial large model size, and it was revealed how a significant decrease in the number of parameters could decrease the model size and yet have a minor impact on the model's performance.

References

[1] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *CVPR*, 2019. 6

[2] Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. Flattened convolutional neural networks for feedforward acceleration, 2015. 2

[3] Maria Koskinopoulou, Fredy Raptopoulos, George Papadopoulos, Nikitas Mavrakis, and Michail Maniadakis. Robotic waste sorting technology: Toward a vision-based categorization system for the industrial robotic separation of recyclable waste. *IEEE Robotics Automation Magazine*, PP:2–12, 04 2021. 1

[4] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017. 6

[5] Jingyi Liu, Pietro Balatti, Kirsty Ellis, Denis Hadjivelichkov, Danail Stoyanov, Arash Ajoudani, and Dimitrios Kanoulas. Garbage collection and sorting with a mobile manipulator using deep learning and whole-body control. In *2020 IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids)*, pages 408–414, 2021. 1

[6] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *CoRR*, abs/1606.02147, 2016. 2

[7] Pedro F. Proença and Pedro Simões. TACO: trash annotations in context for litter detection. *CoRR*, abs/2003.06975, 2020. 2

[8] Jianfei Yang, Zhaoyang Zeng, Kai Wang, Han Zou, and Lihua Xie. Garbagenet: A unified learning framework for robust garbage classification. *IEEE Transactions on Artificial Intelligence*, 2(4):372–380, 2021. 2

[9] Changqian Yu, Changxin Gao, Jingbo Wang, Gang Yu, Chunhua Shen, and Nong Sang. Bisenet V2: bilateral network with guided aggregation for real-time semantic segmentation. *CoRR*, abs/2004.02147, 2020. 2

[10] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. *CoRR*, abs/1808.00897, 2018. 2

[11] Hengshuang Zhao, Xiaojuan Qi, Xiaoyong Shen, Jianping Shi, and Jiaya Jia. Icnet for real-time semantic segmentation on high-resolution images. *CoRR*, abs/1704.08545, 2017. 2