Incremental Learning for Semantic Segmentation

Francesco Di Salvo
Politecnico di Torino
francesco.disalvo@studenti.polito.it

Gianluca La Malfa
Politecnico diTorino
gianluca.lamalfa@studenti.polito.it

Francesco Lacriola
Politecnico diTorino
francesco.lacriola@studenti.polito.it

Abstract

One of the biggest challenges in Deep Learning is to avoid the loss of past knowledge when new classes appear. This phenomenon is known as catastrophic forgetting and there is an ongoing debate on how to overcome this concern. In this paper we faced this problem in the context of semantic segmentation, where another relevant issue is the background shift. At every incremental step the background semantic considers all the classes that are not present in the current step, even the ones learned before or the ones that will be learned in the next steps. In order to tackle these issues, we started from the "Modeling the Background" protocol, using the BiSeNet architecture for the semantic segmentation task, which preserves the spatial information and generates high-resolution features. Moreover, in order to refresh the past knowledge, we used new synthetic images generated through Deep Inversion from the past incremental steps. It optimizes random noise into a class conditional image with a pre-trained CNN and it improves its sparsity through a competition scheme with a weaker student network. All the experiments have been performed on the PASCAL VOC dataset, that contains 20 annotated classes.

1. Introduction

The goal of Semantic Segmentation is to label each pixel of an image with a corresponding class. It has became fundamental in the fields of autonomous vehicles and medical image analysis. For this reason, always more and more researchers have been studying this topic. Several architecture have been proposed over the years such as the Fully Convolutional Networks [11], U-Net [17], DeepLab [3], BiSeNet [20] and many more. The "Modeling the Background" [2] protocol uses DeepLabV3 [4] for the segmentation task on the Pascal VOC dataset, however we used BiSeNet in order to have a lighter backbone.

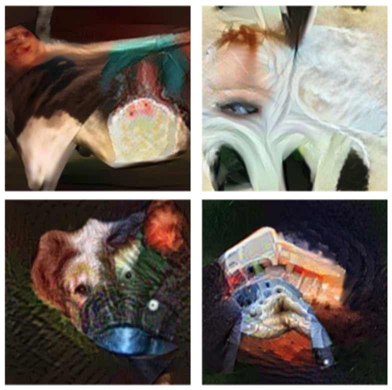
Figure 1. Synthetic images of classes cow, person, dog and bus

Despite the available computational complexity, all these architectures tend to forget the old learned classes in Incremental Learning settings. This phenomenon is called catastrophic forgetting [6]. Moreover, since we are dealing with Semantic Segmentation, another problem to cope with is the background shift. The model at the current iteration classifies as background every pixel that does not belong to the classes that it is learning at the current step. In this way, even the unseen classes may be classified as background. MiB proposed a revisited Knowledge distillation loss [7], introducing two loss terms devoted to the background shift. In order to mitigate the catastrophic forgetting, we generated synthetic images through Deep Inversion [19]. Unlike Generative Adversarial Networks (GANs), Deep Inversion is a data free technique, which means that it does not require an initial dataset. In fact, starting from some randomly initialized noise, it inverts a pre-trained model and extracts significant information from its batch normalization layer to reconstruct an image. Finally, it improves the quality of the image with the Adaptive Deep Inversion module, that explots the disagreement between the pre-trained model (teacher) and a less capable one (student).

2. Related Work

Semantic Segmentation. The semantic segmentation task requires both rich spatial information and a sizable receptive field to be correctly carried out. Given our limited computational resources, we had the necessity to decrease the inference time and therefore, it influenced the choice of the segmentation architecture.

The most commonly used methods for real time semantic segmentation, that try to restrict the input size or prune the channels of the network like SegNet [1], lead to the loss of spatial information and small receptive fields. U-Net [17], on the other hand, tries to recover the edge information lost while doing convolutions and pooling with the skip connections, that allows to combine low and high level data. However, the U-shape structure can slow down the model and not all spatial information might be easily retrieved.

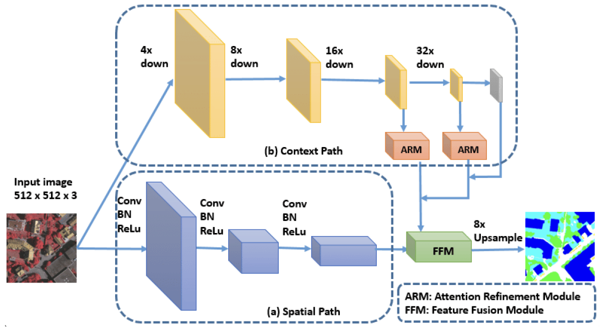

Figure 2. BiSeNet architecture

Contrary to these ones, Changqian Yu et al. proposed the Bilateral Segmentation Network (BiSeNet) [20] for Real-time Semantic Segmentation, based on two components: a Spatial Path to preserve the spatial information and to generate high-resolution features and a Context Path with a fast downsampling strategy to obtain a sufficient receptive field. On top of these two parts, they used a Feature Fusion Module (FFM) to combine the generated features and a final Attention Refinement Module (ARM). Attention mechanisms use every meaningful part of the input images to guide the feed-forward networks. They represent an enhancement of the classic encoder decoder-based system for NLP and it can be seen as a methodological attempt to concentrate on few relevant features. In the BiSeNet architecture, the ARM are used to refine the features of each stage.

The Spatial Path contains only three layers and it extracts the output feature map, that is one-eighth the original image. It encodes a vast amount of spatial information due to the large spatial size of feature maps.

The Context Path, instead, utilizes a lightweight model and a global average pooling to provide a large receptive field. Modern approaches use pyramid pooling [21], like [3], or 'large kernels', like [17], to get a large receptive field, but these methods end up being memory consuming and computation demanding and this could represent a problem for us. The global average provides the maximum receptive field with the context information that we need. Then, the up-sampled output features of this pooling and the ones of the lightweight model are merged together. Considering that the features representation of the two paths is different and they can not be summed, the Feature Fusion Module (FFM) is used to combine them.

Incremental Learning. In modern Artificial Intelligence systems, there is an increasing request for models that are able to incrementally learn new classes when new training data becomes available.

According to Sylvestre-Alvise Rebuffi et al [16] an algorithm can be qualified as class-incremental if:

- it is trainable from different classes at different times
- it provides a multi class classifier at any time
- its computational requirements are limited

In the early stages of Incremental Learning the most commonly used techniques were Feature Extraction, Fine Tuning [5] and Joint Training.

In Feature Extraction the output of one or more layers are used as feature for the new task in training. However, shared parameters fail in discriminating the new classes. Fine Tuning (FT), on the other hand, modifies the parameters of an existing CNN to train a new task. The output layer is extended with randomly initialized weights from a network trained on new classes and finds a new local minimum by optimizing parameters on the new, using a low learning rate. FT still degrades performance on the previously learned task, because the parameters change without any guidance for the original task specific logits. Finally, in Joint Training (JT) all the parameters of a network are optimized using examples from both previous and new tasks. JT becomes really heavy in training as more tasks are learned, and it is not possible to use if training data for previously learned tasks is unavailable.

Learning Without Forgetting (LWF) [10] is combination of fine tuning and distillation. In distillation networks [7] the idea is to learn parameters in a simpler network that produce the same outputs as a more complex ensemble of networks either on the original data set or a large unlabeled data set. The smaller network is trained using a particular cross-entropy loss that encourages responses of the original and new network to be similar.

In Learning Without Forgetting (LWF), using the same data to supervise learning of new tasks and to provide unsupervised output guidance on old tasks, achieves a better accuracy on both tasks, preserving the responses on the task from the original network. However, although preserving

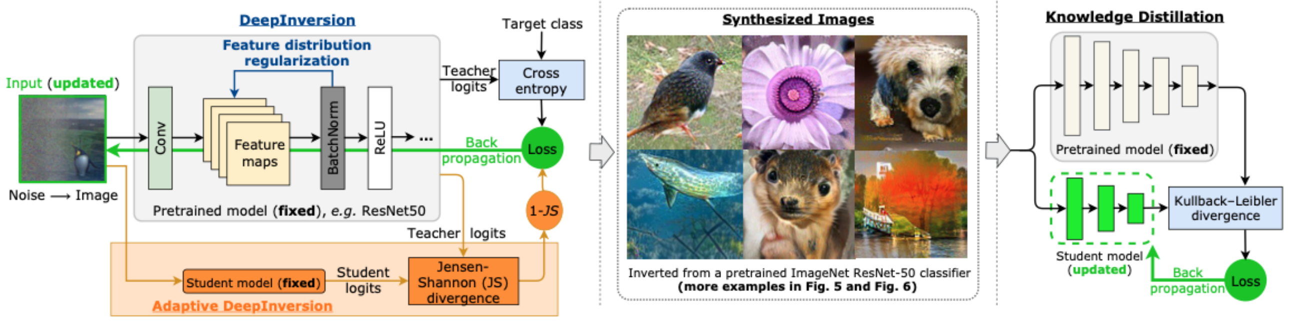

Figure 3. Overview of the original Deep Inversion's framework

outputs for original task domain is still an effective strategy, the performances are not comparable with the ones achieved by the MiB protocol, proposed by Cermelli et al [2] for incremental learning in semantic segmentation.

Even semantic segmentation methods suffer from Catastrophic Forgetting. This phenomenon has been leveraged by MiB thanks to a revisited Cross Entropy and Distillation losses, particularly tailored for the background semantic shift. The new Cross Entropy loss, for the pixels on which the classification is being performed, aggregates the probability to be either background or an old class, as predicted by the old model. As a consequence, we have a new unbiased probability distribution for the predictions utilized to minimize the Cross Entropy Loss. Finally, with the new Distillation Loss, it is possible to tackle the fact that the background could imply classes that in the previous incremental steps were seen as background and are learned in the current step. Since the old model does not know anything about the new classes, it would likely predict them as background. Therefore, the probability of being background and being a new class will be summed up, according to the new model. In this way, the distillation loss is computed between this new probability distribution and the old one.

The last contribution made by the authors is related to the initialization of the classifier. In particular, since the new classes will be more likely classified as background, the distribution of the background is spread over all the new classes and on the background itself.

Synthetic Images Generation. DeepInversion is used to synthesize images from the distribution obtained while training a deep neural network. It is based on the idea of "inverting" [12] a trained network, called *teacher*, in order to create class-conditional input images. Therefore, the goal is to learn how to reconstruct unknown inputs based on just model outputs. To do so, we do not need any additional information other than some injected random noise.

Deep Inversion represents an evolution of the previously introduced Deep Dream [14], that proposed a new method for image synthetization based on the previous work of Model Inversion Attack. Given some randomly initialized noise $\hat{x}$ and a target y, it optimizes the following equation:

$$\min_{x} L(\hat{x}, y) + R(\hat{x}) \tag{1}$$

Deep Inversion improved the quality of the image through using a new feature distribution regularization term $R(\hat{x})$:

$$\begin{aligned} R(\hat{x}) = \sum_{l} \|\mu_l(\hat{x}) - \mathbb{E}(\mu_l(x)|\chi)\|_2 + \\ + \sum_{l} \|\sigma_l^2(\hat{x}) - \mathbb{E}(\sigma_l^2(x)|\chi)\|_2 \end{aligned} \tag{2}$$

where $\mu_l(\hat{x})$ and $\sigma_l^2(\hat{x})$ are the batch-wise mean and variance estimates of feature maps corresponding to the *l-th* convolutional layer.

This distillation technique exploits the property of the batch normalization layer that stores running means and variances. Essentially, they save the history of previously seen data at multiple layers. Hence, in order to estimate the expectations, Deep Inversion [19] exploits the running average statistics of the Batch Normalization layers, that has been captured during the training procedure, assuming that this statistics follow a Gaussian distribution across batches.

To effectively enforce feature similarities at all levels, the idea is to minimize the distance between feature map statistics for $\hat{x}$ and x.

To avoid redundancy and repetitions during the generations, the authors introduced Adaptive Deep Inversion, an image generation scheme based on a competition that encourages the created images to provoke disagreement between the whole image generation process and a student network.

Recently, Zilong Huang et al [8] gave another contribution with SegInversion framework, achieving great results synthesizing the images using the image-level labels.

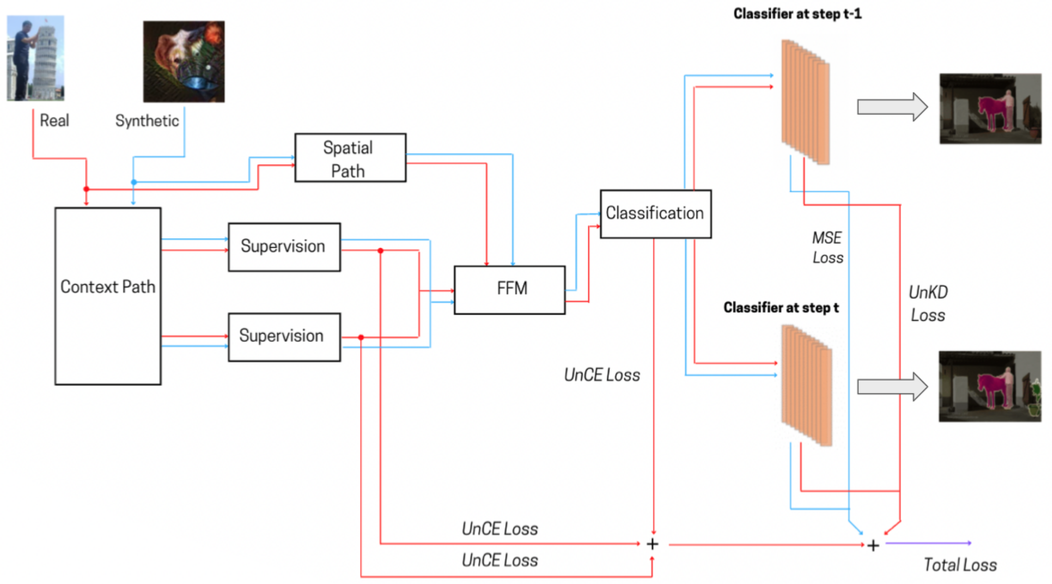

Figure 4. Overview of our contribution. The red path is devoted to the real images whereas the blue one is devoted to the synthetic images. The MSE loss involves the logits for the classification of the synthetic images. The Unbiased Knowledge Distillation loss considers only the output of the real ones. The Unbiased Cross Entropy loss, instead, is used for the features and supervision modules of the BiSeNet.

3. Proposed Methods

Our proposal lightens the incremental segmentation task carried up in the MiB protocol [2], using BiSeNet [20] instead of DeepLabV3 [4]. Next, we generated synthetic images with Deep Inversion [19], starting from the logits of the model at the previous incremental step. Therefore, the generated images belong to the classes already learned in the previous steps. Giving these images to the model at the current incremental step can be helpful to alleviate the catastrophic forgetting.

BiSeNet in incremental setting Concerning the integration of BiSeNet in ”Modeling the Background” (MiB) settings, we moved out from the BiSeNet architecture three convolutional layers to make classifications, exploiting features from the BiSeNet's Context Path and features from the BiSeNet's FFM. They will be used in the incremental protocol, because we are dealing with two classifiers, one for the current step and one for the previous step, all of whom are seeing different classes and therefore they are working on different output channels. Following the BiSeNet pipeline, we used two auxiliary Cross Entropy loss functions (revisited by MiB) to supervise the outputs of the Context Path and we summed up these ones with the Cross Entropy applied to the output of the Feature Fusion Module (FFM).

$$L_B = L_{FFM} + L_{CX1} + L_{CX2} \tag{3}$$

The final loss is driven by both BiSeNet's loss and Knowledge Distillation Loss revisited by Cermelli et al. Then, we get a weighted average of the cross entropy and the knowledge distillation loss:

$$L_{IL}(\theta^t) = \frac{1}{|T^t|} \sum_{(x,y)\in T^t} (L_B^{\theta^t} + \lambda L_{KD}(x,y)^{\theta^t}) \tag{4}$$

where: θ^t represents the network's parameters at the t-th learning step, $|T^t|$ is the training set at the t-th learning step, $\lambda > 0$ is a hyperparameter, whose purpose is to balance the importance of the two terms.

Synthetic Images for the old classes In order to mitigate the catastrophic forgetting, we decided to feed the model with synthetic images of the already seen classes, generated with Deep Inversion [19].

Deep,Inversion's pipeline requires a pre-trained ResNet, therefore we updated the framework in order to be able to use our BiSeNet trained on previous incremental steps. However, ResNet produces an image-wise classification whereas BiSeNet a pixel-wise one. So, we used the Log-Sum-Exp (LSE) pooling like proposed in [8], but in the variant with Lower-bounded Adaptation (LBA) [18]. The goal of LSE is to aggregate pixel-level labels into image-level ones, and according to [15], its main advantage is that pixels with similar scores will have a similar weight. Then, LBA provides a more rubust pooling function towards the numerical underflow.

$$p = \frac{1}{r_0 + \exp\{\beta\}} \log \left[\frac{1}{wh} \sum_{i=1}^{w} \sum_{j=1}^{h} \exp\{[r_0 + \exp(\beta) S_{i,j}]\} \right] \tag{5}$$

where r_0 is a positive constant and β is a learnable parameter. The fake images form a separate dataset that will be used to fed the model separately. Therefore, for each batch of the training set, we extracted another synthetic batch (1:1 proportion), in order to obtain the predictions given by the current and the past model. Obviously, a new loss is required. Thus, as suggested in [9], an MSE loss is more efficient in transferring information from a teacher model $T(\cdot)$ to a student model $S(\cdot)$, compared to a KL loss.

$$MSE(T(\hat{x}), S(\hat{x})) = (T(\hat{x}) - S(\hat{x}))^2 \tag{6}$$

where T is the old model (namely, the teacher) and S is the current model (namely, the student) and $\hat{x}$ is the synthetic image. Hence, the final loss becomes:

$$L(\theta^t) = \frac{1}{|T^t|} \sum_{(x,y)\in T^t} [L_B^{\theta^t} + \lambda L_{KD}(x,y)^{\theta^t} + \mathrm{M}SE(T(\hat{x}), S(\hat{x}))]$$

4. Experiments and result

Three type of experiments have been performed mostly on a P100 and T4 GPU, according to the availability of the cloud service.

4.1. BiSeNet

The first experiment was devoted to the selection on the best hyperparameters and backbone for the BiSeNet's architecture with respect to the training time and the performances. In particular, we trained with two different backbones, namely ResNet18 and ResNet50, different values of batch size, learning rate and augumentation techniques on the images. The experimental results are summarized in Table 1.

Other than the presented hyperparameters and augmentations, another choice has been taken regarding the random resize crop. With 512 pixels, we were able to maximize the usage of the GPU's VRAM. Moreover, we noticed that neither Horizontal nor Vertical Flipping of the input images improved the overall performances.

4.2. Incremental Protocol

Afterwards, we compared five different incremental methods. The results for these baselines are shown in Table 2. These baselines involve two tasks, namely 15-5 and 15-5s. The former consist of two steps. In the first one, the model is fed only with the images from classes 1 to 15, and in the next step with the remaining ones (16-20). On the other hand, in the task 15-5s the model is fed with the remaining classes sequentially.

Table 1. BiSeNet hyperparameters tuning

Backbone	**Batch Size**	**LR**	**Augumentation**	**mIoU**
ResNet18	16	0.025	None	0.48
ResNet18	32	0.025	None	0.53
ResNet18	16	0.01	None	0.61
ResNet18	16	0.001	None	**0.62**
ResNet18	16	0.002	None	0.55
ResNet18	16	0.003	None	0.59
ResNet50	16	0.001	Jittering	0.65
ResNet50	16	0.001	Rotation	**0.72**
ResNet50	16	0.001	Rotation+Jitt	0.65

Table 2. Baseline of incremental methods

	15-5			**15-5s**		
Method	1-15	16-20	All	1-15	16-20	All
MiB [2]	0.65	0.38	0.60	0.35	0.09	0.28
ILT [13]	0.58	0.31	0.53	0.24	0.11	0.23
LWF [10]	0.59	0.42	0.54	0.07	0.11	0.08
FT [5]	0.01	0.29	0.11	0.01	0.02	0.04
Joint	0.74	0.63	0.72	0.74	0.62	0.72

Learning Without Forgetting (LWF) is a data focused method, originally implemented for simple classification tasks and, as we said, it is a combination of a Knowledge Distillation method and Fine Tuning method. On the other hand, ILT [13] is tailored for Incremental Learning in Semantic Segmentation. Like LWF, it uses a distillation loss in the output space, however it adds another one in the feature space. All these proposed methods consider different incremental approaches, except for "Fine Tuning" (FT) and "Joint", that get all the classes within a single step (not incrementally).

4.3. Synthetic images

The synthetic images have been generated through Deep Inversion and a new auxiliary dataset has been defined. To be more precise, a batch of 32 images has been generated for each class with $20k$ iterations each. Using a Tesla T4, the generation of the images for a single class required almost 10 hours. The Mean Squared Error Loss (MSE) has been used on the outputs of the teacher model and student model over the outputs of the same images.

The results are reported in Table 3. In particular a mIoU of 0.37 has been obtained over the first 15 classes whereas the mIoU of all the classes (1-20) was 0.36. Due to the computational complexity involved in the generation of the images, we did not perform the 15-5s test. Otherwise it would have required to generate a new dataset of synthetic images for each incremental step.

Surely, generating a number of images comparable to that of other incremental learning methods would lead to better results as it has been confirmed by the validations carried out in [8] and [19]. In particular, as reported in Deep Inversion, generating 215K ImageNet samples of 224x224 resolution for a ResNet-50 takes 2.8K NVIDIA V100 GPU-hours, or 22 hours on 128 GPUs.

4.4. Ablation studies

Loss function In order to verify what Taehyeon Kim et al. [9] have proposed, we compared the performances of our protocol in a 15-5 test with three different losses: the Mean Squared Error (MSE) loss, the Kullback-Leibler Divergence (KL) loss and the revisited Unbiased Knowledge Distillation loss (proposed in the MiB protocol). The results after 20 epochs of training using the MiB incremental settings are reported in Table 3.

Table 3. Performances with three different losses

mIoU	**15-5**		
Loss	**1-15**	**16-20**	**All**
KL	0.34	0.31	0.32
UKDL	0.32	0.26	0.33
MSE	**0.37**	**0.30**	**0.36**

Quality of synthetic images Since the performances were not as expected, we decided to understand how much the quality of the synthetic images influences the overall performances. Therefore, we compared the mIoU obtained on a 15-5 test with two different datasets. One generated with 10k iterations (for each image) and another one with 20k iterations, both through Deep Inversion. The results can be seen in Table 4. As expected, better images will lead to better performances. However, since we trained our incremental step with a ResNet50 as backbone, we may have low room for improvement. Since Deep Inversion exploits the Batch Normalization layers to compute the layer-wise statistics that it needs, it would surely benefit from a deeper architecture (e.g. ResNet101).

5. Conclusions

We lightened the segmentation task proposed in the MiB protocol, using a BiSeNet instead of a DeepLab V3 and selected the new best hyperparameters, using a ResNet50 as backbone.

Table 4. Performances with different number of iterations per class

mIoU	N. of iterations	
Class	10k	20k
1	0.23	0.29
2	0.16	0.21
3	0.29	0.43
4	0.14	0.26
5	0.26	0.35
6	0.11	0.17
7	0.30	0.41
8	0.50	0.67
9	0.15	0.15
10	0.30	0.32
11	0.17	0.22
12	0.47	0.53
13	0.23	0.36
14	0.18	0.21
15	0.47	0.59

We modified the structure of Deep Inversion in order to use the pixel-wise classification of BiSeNet rather than the image-wise classification of ResNet, as expected. This was possible thanks to the LSE-LBA pooling, for which we aggregate pixel-level labels into image-level ones. Then, the incremental models have been fed with the synthetic images generated before and two new losses have been defined. A revisited Cross Entropy has been used for supervising the Context Path of the BiSeNet and an Unbiased Knowledge Distillation between the classification over the synthetic images made by the old and current model.

Even though the results are not outperforming the previous approaches, we inspected some plausible scenarios to improve them. In particular, as it is possible to observe in the appendix, the classes of the synthetic images are not always easily distinguishable. Therefore, using a deeper architecture can lead to a better reconstruction of the images so that they can be better understood by the models, achieving better results.

References

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation, 2016.

[2] Fabio Cermelli, Massimiliano Mancini, Samuel Rota Bulò, Elisa Ricci, and Barbara Caputo. Modeling the background for incremental learning in semantic segmentation, 2020.

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, 2017.

[4] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation, 2017.

[5] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

[6] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks, 2015.

[7] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[8] Zilong Huang, Wentian Hao, Xinggang Wang, Mingyuan Tao, Jianqiang Huang, Wenyu Liu, and Xian-Sheng Hua. Half-real half-fake distillation for class-incremental semantic segmentation, 2021.

[9] Taehyeon Kim, Jaehoon Oh, NakYil Kim, Sangwook Cho, and Se-Young Yun. Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation, 2021.

[10] Zhizhong Li and Derek Hoiem. Learning without forgetting, 2017.

[11] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2015.

[12] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them, 2014.

[13] Umberto Michieli and Pietro Zanuttigh. Incremental learning techniques for semantic segmentation, 2019.

[14] Alexander Mordvintsev. Deep dream.

[15] Pedro O. Pinheiro and Ronan Collobert. From image-level to pixel-level labeling with convolutional networks, 2015.

[16] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. icarl: Incremental classifier and representation learning, 2017.

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[18] Li Yao, Jordan Prosky, Eric Poblenz, Ben Covington, and Kevin Lyman. Weakly supervised medical diagnosis and localization from multiple resolutions, 2018.

[19] Hongxu Yin, Pavlo Molchanov, Zhizhong Li, Jose M. Alvarez, Arun Mallya, Derek Hoiem, Niraj K. Jha, and Jan Kautz. Dreaming to distill: Data-free knowledge transfer via deepinversion, 2020.

[20] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation, 2018.

[21] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network, 2017.

6. Appendix

In Figure 5 we reported the best image for each class, generated through Deep Inversion with $20k$ iterations. The teacher was the Incremental BiSeNet trained for the first incremental step (step-0) with a ResNet 50 as backbone. In order to improve the spread of the image we used, as a student network, the same BiSeNet but with a ResNet18 as backbone.

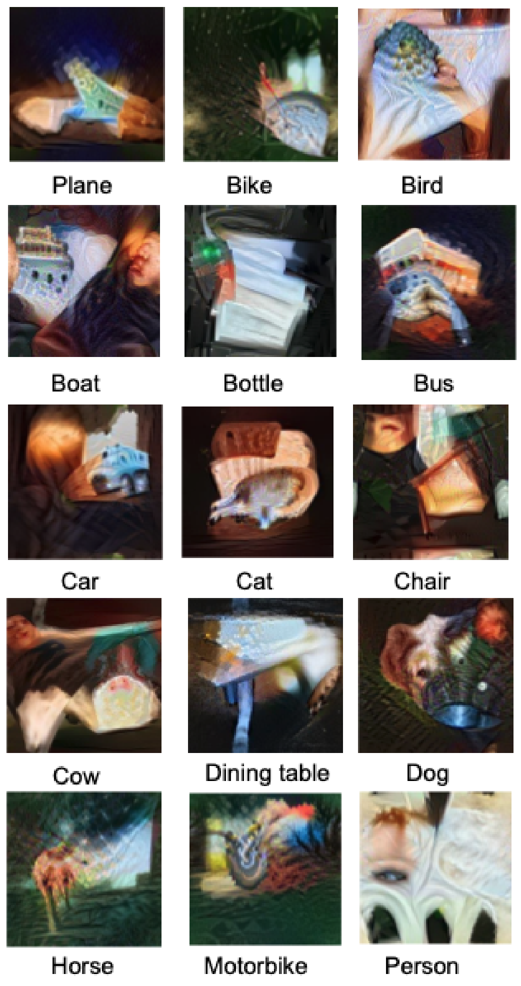

Figure 5. Adaptive DeepInversion