A. Implementation Details

A.1. Models and Weights

All of our experiments are conducted using PyTorch in combination with PyTorch Lightning. We use the PyTorch implementation of ERFNet provided by [13], which can be found at: github.com/Eromera/erfnet_pytorch, the DeepLabV3+ [4] implementation from *Segmentation Models PyTorch* [9] and the SegFormer-B2 [17] implementation from HuggingFace Transformers [16]. The weights for the pre-trained ResNet-50 [8] backbones are taken from:

- DINO [3]: github.com/facebookresearch/dino
- MoCo v3 [5]: github.com/facebookresearch/moco-v3
- BarlowTwins [12]: github.com/facebookresearch/barlowtwins
- SwAV [2]: github.com/facebookresearch/swav

The weights of ERFNet pre-trained with DINO and MoCo v3 can be found on github.com/tobiaskalb/feature-reuse-css.

A.2. Hyperparameter Choice

For each model we start by tuning the LR on Cityscapes. We ran experiments with LR $\in \{0.1, 0.05, 0.01, 0.005, 0.001, 0.0005\}$. We test intermediate LRs between the best and second-best LR. For the pre-trained and augmentation models, we choose the same LR. We chose the parameters for FT and EWC using the Continual Hyperparameter Framework [6].

A.3. Augmentations

For all our augmentations we utilize Albumentations [1]. The augmentation schemes and their specific configurations that were used in our experiments are shown in Tab. 1. The config of the AutoAlbum and further information on the transformation pipelines can be found at: github.com/tobiaskalb/feature-reuse-css. We chose the parameters of Distort to be similar to *PhotometricDistortion* in MMSegmentation.

Method	**Albumentations Parameters**
Distortion	ColorJitter(brightness=0.2, contrast=0.5, saturation=0.5, hue=0.2) ChannelShuffle(p=0.5)
Gaussian Blur	GaussianBlur(blur_limit=(3, 5))
Gaussian Noise	GaussNoise(var_limit=(30, 60))
AutoAlbument	Augementation json config

Table 1. Additional augmentations used in the experiments with there specified arguments and classes using Albumentations [1].

B. Amplitude Spectra of ACDC and Cityscapes

In Fig. 1 and Fig. 2, we compare the mean frequency amplitudes of the Cityscapes dataset with the different ACDC subsets. We observe that ACDC contains much more mid- and high-frequency components in the images, specifically *Snow* and *Rain* contain more higher frequency components. From the ACDC subsets *Night* is most similar to *Cityscapes* in the frequency domain, which could explain why forgetting for *Night* is less. Furthermore, we also see that blurring and the addition of noise to the image have a significant impact in the frequency domain. The goal of adding noise and gaussian blur is to remove the information contained in the high-frequency components of the image so that the model is forced to learn features focusing on low-frequency information that can be reused on the target domain, where the domains are more similar. The plots show that the methods are effectively achieving this. However, we observe that learning color-invariant features are much more effective at mitigating forgetting, which we also confirm for other CNN architectures in Appendix F.

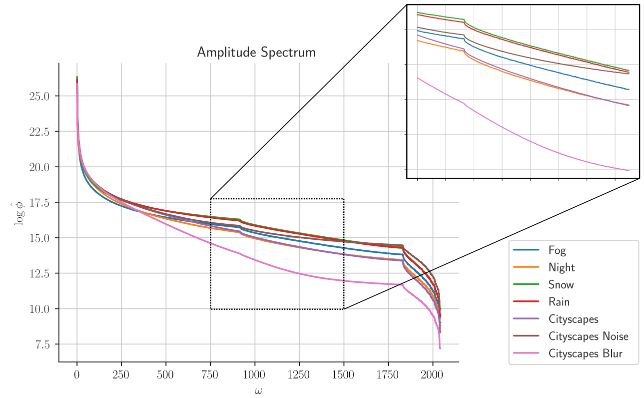

Figure 1. Amplitude Spectra of Cityscapes, Cityscapes Blur, Cityscapes Noise and the ACDC subsets. Specifically, the mid- to high frequent components are increased for *Snow* and *Rain*. In the frequency domain *Cityscapes* is much more similar to *Night* than to any other of the ACDC subsets, specifically in the high-frequent components of the images. We see that Blur is efficiently cutting of high frequency components and that *Snow* and *Rain* contain much more high frequent components.

C. Which BN layers are affected?

In Section 4.1, we found that changing population statistics of BN layers are a significant cause of catastrophic forgetting. To study which BN layer is most affected by the changing population statistics, we re-estimate the BN statistics for one layer at a time. The results are displayed in Fig. 3. We observe that the first BN layer has the most impact on forgetting and that the last BN layer in the first block of each stage (e.g. *layer2.0.downsample.1*) has a comparable impact when the remaining BN layers are not adjusted. These specific layers coincide with blocks that were identified as critical layers by Zhang *et al.* [19]. Interestingly, in a set of freezing experiments in which we freeze the model

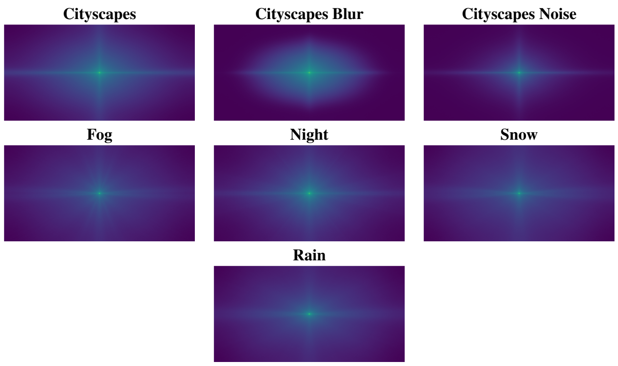

Figure 2. Amplitude spectrum in log-scale for Cityscapes the different ACDC subsets and the Cityscapes dataset using Blur and Noise augmentation.

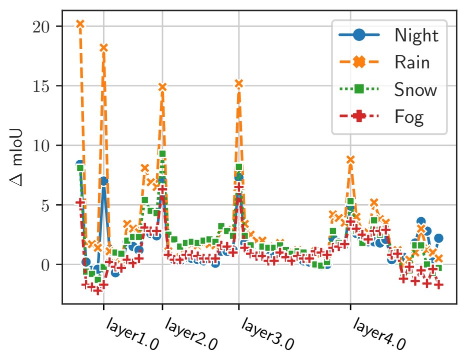

Figure 3. Change in mIoU on the first task after re-estimation of the population statistics of specific BN layers (horizontal axis). Re-Estimation mostly affects the first BN layer and the last BN layers in each stage's first block.

up until specific intermediate layers, we observe that the severe activation drift inside the model is always happening in these specific layers. These results are discussed in the Appendix D.

Furthermore, we also repeat the re-estimation experiment only for the first BN layer. In Tab. 2 we still observe the same trends as for re-estimating the statistics for all layers, but with slightly reduced improvements. These results further demonstrate that the change in population statistics is, in fact, mostly affecting the very first BN layer.

D. Layer freezing experiments

Previous experiments have shown that a major cause of forgetting is the representation shift in the early layers of the model. So naturally the question arises: what happens if we just freeze the early layers and fix the population statistics of the BN layers during incremental training? Therefore, in a set of experiments, we freeze an increasing number of layers during training on *Night* and *Rain* subset, starting from

Method	**CS**	**Rain**		**Night**		
	Test mIoU	*CS forg.*	*Test mIoU*	*CS forg.*	*Rain forg.*	*Test mIoU*
FT	72.0	33.2	57.7	27.8	24.9	45.3
AutoAlb.	72.2	10.7	59.4	15.2	18.2	47.4
Distort	71.7	19.0	60.9	20.8	26.6	47.5
ImageNet	73.9	22.5	60.9	26.1	23.4	46.1
MOCO	75.2	26.8	63.5	18.2	20.1	47.2
DINO	75.0	23.4	64.4	18.3	21.1	49.7
CN	71.2	12.7	58.6	21.1	25.9	43.4
Combined	73.7	6.4	67.8	9.4	16.7	49.8

Table 2. Performance in mIoU [%] on *CS* of the adapted model f_1 after re-estimating the population statistics only of the first BN layer. By measuring and comparing the increase after re-estimating BN statistics (ΔmIoU), we see that re-estimating the population statistics of only the first layer leads to significant improvement on the Cityscapes dataset.

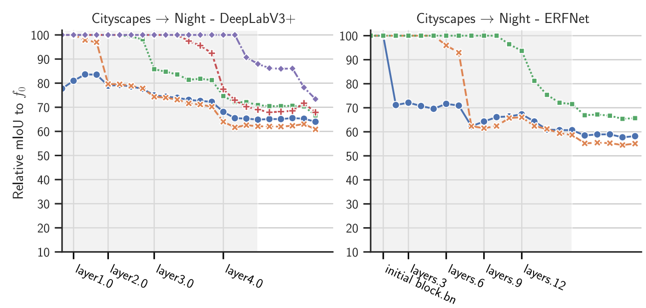

Figure 4. Activation drift between f_1 to f_0 measured by relative mIoU on the first task of the models stitched together at specific layers (horizontal axis). During training on *Night* we froze layers of ERFNet and DeepLabV3+ starting from the very first block. We see that freezing layers during training on the new task fixes early representation shift, but shifts the initial representation shift to later layers.

the very first layer. The results in Tab. 3 show that freezing the first few layers of the encoder has only a minor effect on reducing forgetting or inhibiting learning on the new task. Only when freezing a larger number of layers in the encoder do we observe that the model is less affected by forgetting, but in turn is also inhibited in adapting to *Night*. The reason why the effect is not as prominent for early layers can be seen in the layer stitching plots in Fig. 4. The representational shift of the initial layers is shifted to specific later layers, where the similarity drops down to the level of the non-frozen model. The layers where this representation shift occurs coincide with the layers that were most affected by BN re-estimation. These results indicate that the low-level feature change cannot be addressed by freezing early layers, as it will inhibit learning or shift the activation drift simply to later layers.

Model	Frozen until	Night: *Cityscapes mIoU*	Night: *Night mIoU*	Night: *Forgetting*	Rain: *Cityscapes mIoU*	Rain: *Rain mIoU*	Rain: *Forgetting*
Deep-LabV3+.		45.9	43.6	26.1	38.8	57.8	33.2
	layer1.0	43.6	44.5	28.4	38.5	57.5	33.5
	layer2.1	47.7	42.7	24.3	42.9	56.2	29.1
	layer3.2	48.9	39.0	23.1	50.4	52.6	21.6
	layer4.1	53.1	32.5	18.9	59.1	47.0	12.9
ERFNet		37.1	41.7	31.3	31.9	53.7	36.5
	initial.bn	38.8	41.3	29.6	28.6	54.6	39.8
	layers.5	36.8	37.7	31.6	26.3	52.9	42.1
	layers.10	44.9	36.6	23.5	49.6	48.8	18.8

Table 3. Performance in mIoU [%] on *CS* of the adapted model f_1 after re-estimating the population statistics only of the first BN layer. By measuring and comparing the increase after re-estimating BN statistics (ΔmIoU), we see that re-estimating the population statistics of only the first layer leads to significant improvement on the Cityscapes dataset.

Method	CS: *Test mIoU*	Rain: *CS forg.*	Rain: *Test mIoU*	Night: *CS forg.*	Night: *Rain forg.*	Night: *Test mIoU*
FT	72.0	33.2	57.7	27.8	24.9	45.3
AutoAlb.	72.2	10.7	59.4	15.2	18.2	47.4
Distort	71.7	19.0	60.9	20.8	26.6	47.5
ImageNet	73.9	22.5	60.9	26.1	23.4	46.1
MOCO	75.2	26.8	63.5	18.2	20.1	47.2
DINO	75.0	23.4	64.4	18.3	21.1	49.7
CN	71.2	12.7	58.6	21.1	25.9	43.4
Combined	73.7	6.4	67.8	9.4	16.7	49.8

Table 4. Results for CS $\rightarrow$ *Rain* $\rightarrow$ *Night* with DeepLabV3+. We see that the combination of pre-training with DINO, AutoAlbum and Continual Normalization (denoted as *Combined*) drastically decreases forgetting even in longer task sequences.

E. Longer task sequence

We evaluate the training schemes also on a multi-step domain-increment with *CS*, *Rain* and *Night*, where augmentations are again only used during training on CS. Tab. 4 shows that pre-training and augmentation can decrease forgetting also in a longer task sequences, reducing forgetting not only for the initial task, but for the intermediate task as well. This indicates that the once general low-level features are learned their benefits remain even after the model is fine-tuned on a new domain without the additional augmentations. However, we note that the interaction between these domains can be intricate, as we observe a reduction in forgetting on CS after the model was trained on *Night* when no augmentations are used. Furthermore, we also noticed in our preliminary experiments that the order or similarity of the tasks further impacts the severity of forgetting.

F. Ablation on architectures

We validate our results on the effect of pre-training and augmentation observed previously on DeepLabv3+ also for ERFNet [13], BiSeNet V2 [18], HRNetV2 [15] and RTFormer [14] in Tabs. 5 to 7. We select these networks as they have very distinct architectures compared to DeepLabV3+. HRNetV2 and BiSeNet V2 use multiple parallel branches, ERFNet has significant lower number of parameters, and RTFormer is computationally efficient transformer-based model. Tabs. 5 to 7 show that augmentations and pre-training also significantly reduce forgetting for those selected architectures. Specifically, we see that the combination of pre-training and AutoAlbum leads to significant improvements for all models across all datasets. Furthermore, we see that ERFNet and BiSeNet V2 are much more affected by catastrophic forgetting due to its much smaller size. However, besides this difference, we overall see very similar results, as Distortion and AutoAlbum are the most effective methods to enforce effective feature reuse and thus a reduction of forgetting. Moreover, we make the same observations for ImageNet pre-training, where we achieve higher mIoU on the target dataset but are not as effective at reducing forgetting compared to the models trained with augmentation. The only noticeable difference between the results of BiSeNet V2, ERFNet and DeepLabv3+ is the worse performance on *Snow*, which is drastically worse than the performance of the different subsets, although we use the same training regime as before. Finally, for RTFormer-Base we surprisingly discover results that are similar to CNN architectures than to the results of SegFormer. We hypothesize that this is caused by the use of Batch Normalization instead of Layer Normalization in the Encoder of RTFormer. These results, combined with the observation that SegFormer is less affected by the domain-shift, demonstrate that while our results are applicable to different CNN architectures using BN, catastrophic forgetting significantly varies between architectures, as previous work has pointed out [10, 11].

G. Comparison of Class- and Domain-Incremental Learning

Fig. 5 shows a comparison of layer stitching for class- and domain-incremental learning. In the class-incremental setting, we use PascalVOC2012 [7] with the PascalVOC-15-5 split and in the domain-incremental setting, we use the same Cityscapes to *ACDC* setups as before. We see that during class-incremental learning, the encoder layers up until *layer4.0* are not at all affected by activation drift and the representation shift is only affecting late decoder layers. However, in the domain-incremental setting, we see that primarily the first layers are affected by the activation drift and later layers slightly.

ERFNet

Method	Cityscapes	Night			Rain			Fog			Snow		
	Test mIoU	*Zero Shot*	*Test mIoU*	*Forgetting*	*Zero Shot*	*Test mIoU*	*Forgetting*	*Zero Shot*	*Test mIoU*	*Forgetting*	*Zero Shot*	*Test mIoU*	*Forgetting*
FT	68.4	8.2	41.7	31.3	19.5	53.7	36.5	15.2	58.0	35.3	9.8	57.1	57.4
AutoAlb.	64.0	14.4	42.6	18.9	30.5	54.4	14.7	32.9	56.4	16.4	22.7	55.7	25.1
Distort	65.7	17.7	42.7	19.3	31.0	52.5	18.0	34.9	58.5	19.4	25.3	55.7	22.6
Gaus	65.0	6.1	40.4	27.3	17.3	54.2	41.4	14.1	57.8	28.4	8.1	56.0	43.2
Noise	65.4	3.6	42.7	27.8	20.8	51.8	37.7	18.6	55.6	32.9	15.6	56.4	49.8
ImageNet	70.4	10.7	42.8	29.0	25.7	56.1	36.2	26.1	64.6	30.1	17.8	58.6	59.5
MOCO	71.8	10.2	43.0	28.4	21.7	55.8	34.9	21.3	61.7	30.4	14.0	60.4	38.4
DINO	70.1	7.6	43.3	26.3	24.3	56.6	45.8	20.8	58.9	30.7	15.6	59.6	46.9
CN	70.4	9.6	40.4	21.7	27.5	52.7	15.4	27.8	61.9	17.9	12.2	59.5	20.8
Combined	69.8	11.6	43.2	15.0	37.6	57.5	8.0	44.3	65.5	11.3	32.7	59.8	17.2
Replay	68.4	8.2	39.3	8.8	19.5	53.9	7.7	15.2	58.7	8.0	9.8	58.1	7.2
Offline		40.1	43.1	15.6	50.5	55.1	19.9	58.1	61.5	14.9	53.6	55.8	23.3

Table 5. Results of ERFNet [13] on *CS* $\rightarrow$ *ACDC* in mIoU (%) for each subset of ACDC using different pre-training and augmentations strategies (Augment.). Compared to DeepLabV3+, ERFNet is much more affected by Forgetting, specifically on *Snow*. However, Augmentations and pretraining show the same effects as for the experiments in the main paper.

BiSeNet V2

Method	Cityscapes	Night			Rain			Fog			Snow		
	Test mIoU	*Zero Shot*	*Test mIoU*	*Forgetting*	*Zero Shot*	*Test mIoU*	*Forgetting*	*Zero Shot*	*Test mIoU*	*Forgetting*	*Zero Shot*	*Test mIoU*	*Forgetting*
FT	67.5	4.9	41.2	33.7	18.8	52.1	40.7	14.7	57.3	39.4	9.3	58.1	58.9
AutoAlb.	66.6	12.8	41.0	26.2	35.5	53.5	23.5	39.3	60.2	33.8	27.1	56.6	46.1
Distort	68.2	14.8	42.4	29.7	32.9	52.9	35.3	38.0	58.1	29.2	23.0	58.3	35.8
Gaus	67.1	3.8	40.8	34.2	17.6	52.9	41.2	13.9	59.4	48.1	11.0	59.1	58.9
ImageNet	69.5	7.0	42.1	35.7	20.2	54.7	49.9	14.0	60.8	46.3	13.7	57.9	62.8
CN	68.7	5.4	37.0	26.9	30.4	51.5	18.0	25.5	54.8	23.1	18.7	54.4	25.4
Combined	68.0	13.6	38.6	21.7	36.7	53.2	13.7	44.0	58.8	17.5	29.6	53.2	22.2
Replay	67.5	4.9	40.0	10.7	18.8	51.6	6.2	14.7	50.7	8.3	9.3	58.5	8.3
Offline		39.7	43.8	17.4	52.3	52.4	13.0	59.8	62.9	21.1	56.8	60.3	56.8

Table 6. Results of BiSeNet V2 [18] on *CS* $\rightarrow$ *ACDC* in mIoU (%) for each subset of ACDC using different pre-training and augmentations strategies. Compared to DeepLabV3+, BiSeNet V2 is more affected by Forgetting.

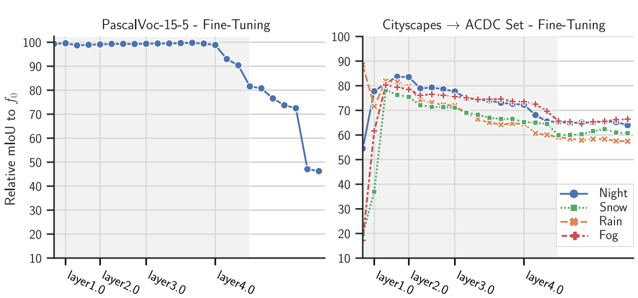

Figure 5. Layer-stitching reveals that during class-incremental learning (PascalVoc-15-5) the encoder layers are mostly stable, only the decoder layers are changing drastically. In domain-incremental learning observe the opposite, early layers show a big discrepancy and later layers do not change as much.

HRNetV2-W48

Method	Cityscapes *Test mIoU*	Night *Zero Shot*	Night *Test mIoU*	Night *Forgetting*	Rain *Zero Shot*	Rain *Test mIoU*	Rain *Forgetting*	Fog *Zero Shot*	Fog *Test mIoU*	Fog *Forgetting*	Snow *Zero Shot*	Snow *Test mIoU*	Snow *Forgetting*
FT	70.7	6.1	42.1	38.0	22.8	59.7	37.2	19.9	67.0	36.2	15.7	62.7	44.1
AutoAlb.	72.4	19.6	44.8	33.1	43.0	58.1	12.6	55.4	68.2	15.2	37.5	61.8	20.4
Distort	70.4	15.7	44.8	21.7	33.3	58.9	13.0	38.6	64.3	11.7	24.5	62.9	18.1
Gaus	69.4	7.8	45.1	28.8	24.3	59.6	32.4	24.5	66.9	26.6	15.7	61.6	40.8
ImageNet	71.1	6.9	46.2	26.2	26.0	58.6	31.9	26.0	66.2	25.8	19.8	60.4	51.0
CN	70.5	9.8	41.9	17.0	29.9	57.0	13.1	28.1	65.9	17.1	19.7	58.0	21.8
Combined	71.8	17.7	41.9	10.4	46.3	60.4	9.3	56.9	66.6	11.1	41.3	62.1	11.3
Replay	70.7	6.1	45.2	9.8	22.8	59.2	3.3	19.9	68.9	4.4	15.7	63.3	5.9
Offline		44.8	45.6	32.5	57.9	57.9	2.4	62	68.8	2.4	58.2	63.1	4.3

Table 7. Results of HRNetv2 [15] on *CS* → *ACDC* in mIoU (%) for each subset of ACDC using different pre-training and augmentations strategies. HRNetV2 performs similar to DeepLabv3+ on Cityscapes, but overall is more impacted by forgetting. The combination of ImageNet pre-training, AutoAlbum. and Continual Normalization (*Combined*) leads to a significant reduction of forgetting.

RTFormer - Base

Method	Cityscapes *Test mIoU*	Night *Zero Shot*	Night *Test mIoU*	Night *Forgetting*	Rain *Zero Shot*	Rain *Test mIoU*	Rain *Forgetting*	Fog *Zero Shot*	Fog *Test mIoU*	Fog *Forgetting*	Snow *Zero Shot*	Snow *Test mIoU*	Snow *Forgetting*
FT	68.8	4.2	42.0	24.7	22.7	57.7	42.5	19.4	65.2	32.2	13.4	60.7	43.7
AutoAlb.	68.5	13.3	41.4	18.2	36.8	56.0	20.6	42.4	61.6	18.9	26.5	58.0	43.1
Distort	70.9	16.3	43.4	15.7	34.5	58.6	21.0	46.4	67.2	17.2	31.6	62.0	31.1
Gaus	66.9	6.9	40.7	25.8	13.6	58.0	40.5	14.4	62.4	25.1	7.4	60.5	47.9
ImageNet	70.8	5.5	42.2	28.1	22.4	59.1	39.3	21.4	65.7	27.2	16.7	61.9	38.9
CN	69.1	4.7	35.0	20.7	16.4	55.1	22.8	14.3	58.2	20.3	10.7	58.8	33.5
Combined	70.8	14.2	41.8	19.5	41.2	59.0	9.7	53.1	65.5	12.0	37.6	61.6	17.2
Replay	68.8	4.2	39.9	5.1	22.7	54.6	2.3	19.4	64.3	3.5	13.4	60.6	4.8
Offline		41.8	42.7	4.2	53.4	58.6	8	60.3	64.3	6.4	60.7	62.7	6.9

Table 8. Results of RTFormer [14] on *CS* → *ACDC* in mIoU (%) for each subset of ACDC using different pre-training and augmentations strategies.

References

[1] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020. 1

[2] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020. 1

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021. 1

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018. 1

[5] Xinlei Chen*, Saining Xie*, and Kaiming He. An empirical study of training self-supervised vision transformers. *arXiv preprint arXiv:2104.02057*, 2021. 1

[6] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021. 1

[7] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, 2012. 3

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015. 1

[9] Pavel Iakubovskii. Segmentation models pytorch. `https://github.com/qubvel/segmentation_models.pytorch`, 2019. 1

[10] Tobias Kalb, Niket Ahuja, Jingxing Zhou, and Jürgen Beyerer. Effects of architectures on continual semantic segmentation. *arXiv preprint arXiv:2302.10718*, 2023. 3

[11] Seyed Iman Mirzadeh, Arslan Chaudhry, Dong Yin, Timothy Nguyen, Razvan Pascanu, Dilan Gorur, and Mehrdad Farajtabar. Architecture matters in continual learning, 2022. 3

[12] PMLR. *Barlow twins: Self-supervised learning via redundancy reduction*, 2021. 1

[13] Eduardo Romera, José M. Álvarez, Luis M. Bergasa, and Roberto Arroyo. Erfnet: Efficient residual factorized convnet for real-time semantic segmentation. *IEEE Transactions on Intelligent Transportation Systems*, 19(1):263–272, 2018. 1, 3, 4

[14] Jian Wang, Chenhui Gou, Qiman Wu, Haocheng Feng, Junyu Han, Errui Ding, and Jingdong Wang. RTFormer: Efficient design for real-time semantic segmentation with transformer. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. 3, 5

[15] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *TPAMI*, 2019. 3, 5

[16] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics. 1

[17] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *Neural Information Processing Systems (NeurIPS)*, 2021. 1

[18] 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. *International Journal of Computer Vision*, 129:3051–3068, 2021. 3, 4

[19] Chiyuan Zhang, Samy Bengio, and Yoram Singer. Are all layers created equal?, 2019. 1