Efficient Self-Ensemble for Semantic Segmentation

Walid Bousselham
bousselh@ohsu.edu
Guillaume Thibault
thibaulg@ohsu.edu
Lucas Pagano
paganol@ohsu.edu
Archana Machireddy
machired@ohsu.edu
Joe Gray
grayjo@ohsu.edu
Young Hwan Chang
chanyo@ohsu.edu
Xubo Song
songx@ohsu.edu

Oregon Health and Science University
Portland, OR, USA

Abstract

Ensemble of predictions is known to perform better than individual predictions taken separately. However, for tasks that require heavy computational resources, *e.g.* semantic segmentation, creating an ensemble of learners that needs to be trained separately is hardly tractable. In this work, we propose to leverage the performance boost offered by ensemble methods to enhance the semantic segmentation, while avoiding the traditional heavy training cost of the ensemble. Our self-ensemble approach takes advantage of the multi-scale features set produced by feature pyramid network methods to feed independent decoders, thus creating an ensemble within a single model. Similar to the ensemble, the final prediction is the aggregation of the prediction made by each learner. In contrast to previous works, our model can be trained end-to-end, alleviating the traditional cumbersome multi-stage training of ensembles. Our self-ensemble approach outperforms, by the time of the publication, the previous state-of-the-art on the benchmark datasets Pascal Context and COCO-Stuff-10K for semantic segmentation and is competitive on ADE20K and Cityscapes. Code is publicly available at `https://github.com/WalBouss/SenFormer`

1 Introduction

Semantic segmentation is the task of assigning each pixel of an image with a semantic category. Its many applications include robotics, autonomous cars, medical application, augmented reality and more. Most segmentation methods follow an Encoder-Decoder scheme.

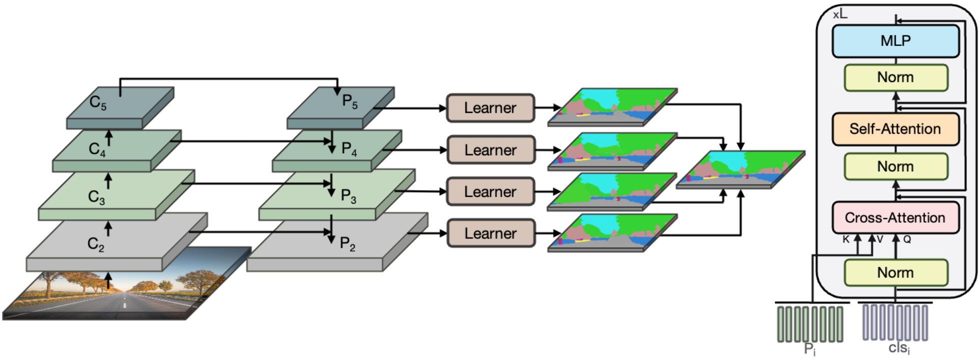

Figure 1: **SenFormer architecture.** (Left): The features extracted by the backbone $\{C_2, C_3, C_4, C_5\}$ are enhanced in a feature pyramid to produce spatially and semantically strong features maps at every level of the pyramid $\{P_2, P_3, P_4, P_5\}$. Each set of features is decoded by a different learner in the ensemble and the learners' predictions are merged. (Right): architecture of the transformer block.

The encoder extracts the relevant features of the image to characterize each pixel, a process usually involving down-sampling the feature maps to increase the receptive field of the model. The decoder up-samples the feature maps to both recover the spatial information and produce a per-pixel classification. In [30] the authors extended this procedure to fully convolutional network (FCN), which paved the way for later work to achieve impressive results on various segmentation datasets and has since dominated the field of semantic segmentation, let it be for medical [17, 34], self-driving cars [35] or robotics applications [18]. Follow up work mainly focused on enhancing FCN to mitigate the inherent locality of the convolution operation. Some examples are the atrous convolution that introduces holes in convolution kernel [3, 4], the pyramid pooling module (PPM) that aggregates context information using different kernel pooling layers, and [44] that combine the PPM and the feature pyramid network (FPN) [27] to capture context information at different resolutions.

The starting observation of this paper was that the combined use of a backbone and an FPN-like method [14, 20, 27, 28, 37, 41] allows extracting multiple features sets at different scales for a single image with a unique forward pass. Furthermore, in [27] the authors show that these features are both semantically and spatially strong at each level of the feature pyramid. Consequently, one has access to multiple features representations of the same image that carry different contextual information at different scale and are loosely correlated (as shown in[41]). This raises questions about the optimal way to use these multi-scales features. A canonical use is UperNet [44], which concatenates the multi-scale feature maps before feeding them to a decoder. However, this paper argues that the "features fusing" strategy consisting of merging the different sets of features maps and letting the model decide which one is important is sub-optimal and often computationally expensive. Indeed, in UperNet, the four pyramid levels are concatenated and merged by a convolution, which by its own involves 155G FLOPs, thus making the "features fusing" strategy FLOPs intensive. Moreover, we hypothesize that a single decoder cannot fully take advantage of the multi-scale features that contain different views of the same objects of interest. Hence, the model may focus on one view and overlook valuable features. This "multi-view" hypothesis is indeed supported by a recent study: in [1] the authors argue that in vision datasets, objects can be recognized using multiple views and show that in the context of image classification, for a given weight initialization, a model will learn to focus on particular views while discarding others.

To overcome the limitations of "features fusion" strategies, we propose and study a different approach to exploit the FPN multi-scale features. Our approach feeds independent decoders with features coming from different levels of the feature pyramid, and then combine the segmentation maps together, hence avoiding expensive features fusion operations. Since the inputs to the learners (i.e., decoders) come from different levels of the feature pyramid that differ in scale and contain different spatial and semantic information and that the learners are independent, our method can be interpreted as a form of self-ensemble segmentation. Usually, the learners of an ensemble must be trained independently. In this work, we show that, in the context of semantic segmentation, this condition can be relaxed and imposed solely to the decoders. Our experiments show that – all else being equal – this strategy improves UperNet performance. However, increasing the number of decoders/learners inevitably increases parameters number. Overall, our observations on self-ensemble performance effectiveness but parameter burden, lead us to design a transformer-based model: SenFormer (**S**elf-**en**semble segmentation trans**Former**). Our motivation for using transformer-based learners is that besides transformers' ability to capture long-range dependencies, it has been observed [10, 22, 24] that recursively applying the same transformer block to the same input features can produce similar – if not better – results than using different blocks while reducing the number of parameters and overfitting. Ultimately, our method has fewer parameters and FLOPs than UperNet and performs better.

Overall, our SenFormer approach achieves excellent results on various benchmark datasets. Specifically, it outperforms similar architectures [44] that use "feature fusion" strategy, suggesting that our self-ensemble approach effectively leverages the expressive power of ensemble methods. In particular, SenFormer achieves 51.5 mIoU on the benchmark dataset COCO-Stuff-10K [2] and 64.0 mIoU on Pascal-Context [31], outperforming the previous state-of-the-art by a large margin of 6 mIoU and 3.0 mIoU respectively. SenFormer is also on par with state-of-the-art methods on Ade20K [48] and Cityscapes [9]. To summarize, our goal is to show that self-ensemble is a more efficient and effective way to leverage multi-scale features for segmentation than "features merging" strategies. To illustrate our point, we proposed SenFormer architecture. We motivate this goal by the observation that each scale of a feature pyramid carries a different amount of contextual information that a single decoder cannot fully exploit. We, therefore, adopted a divide-and-rule policy (self-ensemble), where each learner focuses on one scale. However, increasing the number of learners also increases the parameters, hence the exploration of weight sharing strategies.

2 Method

In this section, we first present the general framework of our method based on self-ensemble as shown in figure 1. Then we detail the different merging strategies. Finally, we describe learners' architecture and the different weight sharing strategies.

Following notations in [16, 27, 44], we denote $C_i \in \mathcal{R}^{d_i \times \frac{H}{2^i} \times \frac{W}{2^i}}$ the output of the i-th stage of the bottom-up network (*i.e.* backbone) which has stride of 2^i pixels with respect to the input image, where $H \times W$ is the spatial dimension of the input image and d_i the number of channels. Similarly, we denote $P_i \in \mathcal{R}^{d \times \frac{H}{2^i} \times \frac{W}{2^i}}$ the output of the i-th stage of the top-down network (*i.e.* output of the FPN), where d is the numbers of channels in all the feature maps of the FPN. We denote N the number of class.

2.1 Self-Ensemble

In this paper, we approach the problem of semantic segmentation as that of a per-pixel classification. Therefore, learners predictions and the merging strategies will be described for an arbitrary pixel and can easily generalize to the whole segmentation map.

An ensemble traditionally consists of M independently trained models called learners. For a given pixel, let denote $X_i \in \mathcal{R}^N$ the random variable parameterized by the output of the i-th learner for that particular pixel, which can be decomposed in $X_i = Y + \varepsilon_i$ where Y is the target and ε_i is the prediction error of the i-th learner.

The most straightforward way to merge different learners' predictions is by averaging them. It is well known [33] [49] [23] that the ensemble performance is usually better than the individual learners.

Classical statistics suggest that when the predictions are roughly independent, the last term in equation 1 is close to zero and therefore averaging greatly reduces the noise.

$$Var(\frac{1}{M}\sum_{i=1}^{M}\varepsilon_i) = \frac{1}{M^2}\sum_{i=1}^{M}Var(\varepsilon_i) + \frac{2}{M^2}\sum_{i<j}Cov(\varepsilon_i, \varepsilon_j). \tag{1}$$

On another note, a recent study suggests that this hypothesis might not hold in the context of deep learning. In [1], Allenzhu *et. al,* acknowledge that for the task of image classification, the different learners learn to detect different views/features of the object of interest depending on their weight initialization. However, there are some images taken from a particular angle where the learned features may be missing. Hence, when the ensemble is large enough, all possible views are captured, thus increasing the model's accuracy. Note, however, that it is not clear in [1] if this result also holds for semantic segmentation. Either way, a key requirement is that the learners' predictions must be independent, let it be for the variance reduction or the multi-view hypothesis.

Total independence of the predictions implies tediously training multiple independent models. In this paper, we aim at relaxing the independence hypothesis to reduce the training cost, while maintaining the performance benefits of ensemble. To do so, the learners/decoders share the same backbone but receive input features coming from different levels of the feature pyramid, *i.e.*, $\{P_2, P_3, P_4, P_5\}$, as shown in figure 1.

Nevertheless, it is observed that if one trains the different learners of an ensemble altogether (i.e., applying the loss on the merged prediction), the performance boost offered by the ensemble disappears [1]. However, we show in our experiments that it is not the case in our setting. We hypothesize that it is because each learner is independently initialized (as in ensemble) and receives different inputs, therefore alleviating the need for separate training. In this manner, several segmentation predictions can be obtained with only a single forward pass of the input image.

2.2 Merging strategies

We describe the different methods considered to merge the different learner predictions (during inference).

Averaging. It is the most commonly used method for prediction merging as no additional trainable parameters are required. The merged prediction X_{avg} of M learners is obtained by: $X_{avg} = \frac{1}{M}\sum_i X_i$

Product. The predicted probability for each pixel is multiplied rather than average. That way, more weight is given to learners with high confidence.The merged prediction X_{prod} of M is given by: $X_{prod} = \prod_{i=1}^{M} X_i$.

Majority vote. Each learner assigns a vote to the class with the largest confidence.

Hierarchical Attention. We borrow the "attention module" from [38] that is used to learn a relative attention mask between adjacent scales. The module consists of $(3 \times 3\,conv) \rightarrow (BatchNorm) \rightarrow (ReLU) \rightarrow (3 \times 3\,conv) \rightarrow (BatchNorm) \rightarrow (ReLU) \rightarrow (1 \times 1\,conv) \rightarrow (Sigmoid)$, where the last convolution output a single (attention) map. In the original paper, the module is fed with the same input features maps of the decoder. Another variant would be to use the segmentation logits (decoder's output) instead. In our experiment, we tried both and found the latter to work better with SenFormer. Since SenFormer has four learners, we need 3 "attention modules" to predict the relative attention maps.

Explicit Attention. We used the same "attention module" as for Hierarchical Attention [38], but trained it to predict a dense mask for each scale rather than a relative mask.

Surprisingly, our experiments found the simple "averaging" strategy to perform better than others, except for the "hierarchical attention" (Table 4). However, given the performance boost of the "attention module" is limited, it does not justify the overhead complexity. Therefore, SenFormer uses the "averaging" as the default merging strategy since it yields high performance without requiring additional parameters.

2.3 Learner architectures

Hereafter, we described the architecture of a single learner/decoder. As depicted in Figure 1, the i^{th} decoder branch takes as input the features coming from the corresponding level of the FPN (with stride s_i) $P_i \in \mathcal{R}^{d \times \frac{H}{2^i} \times \frac{W}{2^i}}$, as well as a set of N learnable embeddings termed as class embeddings, $\mathbf{cls}_i = [cls_i^1, \ldots, cls_i^N] \in \mathcal{R}^{N \times d}$, where N is the number of class. In this respect, there is one learnable class embedding cls_i^k per segmentation class and per level in the feature pyramid.

Each decoder is a transformer composed of L layers whose architecture is inspired by the traditional transformer [39]. Note however that a "pre-norm" strategy is used in place of "post-norm" for the placement of Layer Normalization (LN), *i.e.*, the skip connections inside each transformer block are not affected by the LN [32] (see ablation study in the Annex).

In a nutshell, a single Transformer Decoder block consists of three successive operations: Cross-Attention, Self-Attention and Multi-Layer Perceptron layers. In the Cross-Attention operation the feature map P_i is used as key and value while the class embedding $\mathbf{cls}_i$ is used a query. The Self-Attention and MLP are applied only to the class embeddings.

Finally, each decoder/learner is composed of L layers of decoder block and its prediction is obtained via a dot product between the class embeddings $\mathbf{cls}_i$ and the corresponding feature pyramid feature P_i – see the Annex for more details. However, using multiple decoders greatly increases the number of parameters. To mitigate this, we explore several weight-sharing strategies.

	method	backbone	crop size	#params.	FLOPs	*mIoU*
CNN	UperNet	ResNet-50	512×512	67M	238G	42.05
	SenFormer	ResNet-50	512×512	**55M**	**179G**	**44.38**
	UperNet	ResNet-101	512×512	86M	257G	43.82
	SenFormer	ResNet-101	512×512	**79M**	**199G**	**46.93**
Transformer	UperNet	Swin-T	512×512	60M	236G	44.41
	SenFormer	Swin-T	512×512	**59M**	**179G**	**46.0**
	UperNet	Swin-S	512×512	81M	259G	47.72
	SenFormer	Swin-S	512×512	**81M**	**202G**	**49.2**
	UperNet	Swin-B‡	640×640	121M	471G	50.04
	SenFormer	Swin-B‡	640×640	**120M**	**371G**	**52.21**
	UperNet	Swin-L‡	640×640	234M	647G	52.05
	SenFormer	Swin-L‡	640×640	**233M**	**546G**	**53.08**

Table 1: Self-ensemble SenFormer vs features fusion UperNet on ADE20K validation. Backbones pre-trained on ImageNet-22K are marked with ‡.

2.4 Weight sharing

Weight sharing is a commonly used technique to reduce the number of parameters [10, 22, 24], while also regularizing the optimization by reducing the degree of freedom which mitigates overfitting. However, in the context of Ensemble, special care regarding the kind of weight sharing used must be given.

Two types of weight sharing can be used: inter-learner and intra-learner sharing. The former involves sharing parameters between the different learners, while latter within the learner. A figure depicting the different sharing methods can be found in Annex.

Repeated block. A given learner is composed of a single decoder block recursively used L times. It is a form of "intra-learner sharing" since no parameters are shared between the different learners.

Decoder sharing. The different learners share the same decoder but have their own class embedding. It is a form of "inter-learner sharing".

Class embeddings sharing. The same learnable class embeddings **cls** is used for all the learners. It is also a form of "inter-learner sharing".

Table 3 shows that any "inter-learner sharing" strategy significantly degrades the segmentation performance, confirming the importance of keeping the different learners as independent as possible. Conversely, the "repeated block" strategy performs better than when no sharing is used, while significantly reducing the number of parameters. Hence, SenFormer uses the "repeated block" as the default weight sharing policy.

3 Experiments

Datasets. We evaluate our model performance using four semantic segmentation benchmark datasets, ADE20K [48], Pascal Context [31], COCO-Stuff-10K [2] and Cityscapes [9]. We use ADE20K, which is a challenging scene parsing dataset consisting of 20,210 training images and 2,000 validation images and covers 150 fine-grained labeled classes, for the ablation studies. Please see the Annex for detailed descriptions of all used datasets. We

d_2	d_3	d_4	d_5	*ade*	pascal	coco	city
✓	✗	✗	✗	42.9	51.9	36.5	76.9
✗	✓	✗	✗	42.1	51.8	36.0	77.3
✗	✗	✓	✗	41.4	51.8	37.2	76.5
✗	✗	✗	✓	38.2	46.3	35.3	73.0
✓	✓	✓	✗	44.12	52.8	38.7	78.0
✓	✓	✓	✓	**44.4**	**53.2**	**40.0**	**78.3**

Table 2: Performances by using different learner combinations, where ✓/✗ indicates whether the learner is used for the prediction.

weight sharing setting	# blocks	*mIoU*	#params.	FLOPs
decoder shared	6	42.69	68M	179G
cls embeddings	6	42.91	67M	179G
repeated♣	6	44.38	**55M**	179G
none	1	43.12	**55M**	**111G**
none	6	**44.68**	144M	179G
UperNet		42.05	67M	238G

Table 3: Performance comparisons on ADE20K val of different weight sharing settings for SenFormer.♣ indicates SenFormer's default setting.

merging strategy	mIoU
averaging	44.4
product	40.28
majority vote	39.89
hierarchical att.	**44.5**
explicit att.	39.7

Table 4: Merging strategies.

method	*mIoU*	#params.	FLOPs
UperNet	42.02	67M	238G
SenUperNet♣	42.8	70M	**135G**
FFBaseline	43.1	**52M**	307G
SenFormer♣	**44.3**	55M	179G

Table 5: Comparisons of the features fusion and self-ensemble strategies.♣indicates self-ensemble.

report for every dataset the mean Intersection over Union (mIoU), a standard metric for semantic segmentation.

Baseline model. To demonstrate that the performance improvement of our method is genuinely a result of self-ensemble instead of feature fusion, we introduce a simple decoder baseline module that borrows the features fusion strategy from UperNet [44], but uses our transformer decoder. This way, the FeatureFusionBaseline and SenFormer only differ by the multi-scale fusion strategy – a figure and more details on FFBaseline can be found in Annex.

Training details. We use mmsegmentation [8] library as codebase and follow the standard training practice for each dataset. Moreover, we apply common data augmentation for semantic segmentation, which include left-right flipping, standard random color jittering, random resize with ratio $0.5-2$ and random cropping. Each learner is independently supervised with a cross-entropy loss. We apply the same loss to the ensemble prediction. Following [6] we use AdamW as optimizer and "poly" learning rate scheduler – see Annex for more details.

3.1 Self-ensemble vs Features Fusion

Features at different levels of the pyramid carry different scale of contextual information, and our experiments support that self-ensemble is capable of capturing and integrating such information.

Ensemble effect. We first analyze the output produced by each decoder and assess their performance. Table 2 outlines the *mIoU* scores of independent prediction of each decoder as well as for the ensemble. Notably, the ensemble *mIoU* score is +3.5 (for Ade20K) better than the mean score of the learners taken separately with $\frac{1}{4}\sum_{i=2}^{5} mIoU(d_i) = 41.15$. More surprisingly, even though d_5 taken separately performs significantly worse than the others –

due to its low-resolution inputs – it positively contributes to the ensemble, consistent with traditional ensemble methods where even weak learners can be combined to enhance the overall prediction.

Does the performance boost really comes from self-ensemble? To rule out the performance gain brought by the use of transformer-based decoders rather than convolution, we compare SenFormer and the FeaturesFusionBaseline, since they only differ in the multi-scale fusion strategy (*features fusion vs. self-ensemble*). In Table 5, we observe that SenFormer is +2 *mIoU* better than the baseline. Conversely, we applied the self-ensemble method to UperNet [44] by using the same convolution-based decoder at each level of the feature pyramid rather than merging the features. Likewise, the self-ensemble version (SenUperNet) performs better than the vanilla UperNet, suggesting that our self-ensemble approach is the main driver for improvement.

SenFormer vs UperNet. We compare SenFormer with UperNet architecture for a variety of CNN- and transformer-based backbones. As we can see from Table 1, when using the same standard Swin-Transformer backbone, SenFormer consistently outperforms UperNet regardless of the backbone size. The performance gap is even larger when using convolutional backbones (+3 *mIoU*), suggesting that our transformer-based decoder successfully captures the long-range dependencies missed by the CNN-based backbones.

Thanks to its weight sharing strategy, SenFormer has fewer parameters than UperNet. Furthermore, since SenFormer avoids the computationally expensive features merging operation, it also has substantially fewer FLOPs.

3.2 Comparison to state-of-the-art

In this section we further compare SenFormer to state-of-the-art methods on ADE20K and Pascal Context additional benchmark datasets.

ADE20K. In Table 6 we compare SenFormer to a variety of FCN- and transformer-based decoders using both CNN- and transformer-based backbones. Except for the MaskFormer family, when using standard ResNet backbones, SenFormer outperforms all other methods. The same can be said for per-pixel classification-based models when using transformer-based backbones, where SenFormer even outperforms recently introduced transformer-based decoders like SETR [47], Segmenter [36] and SegFormer [45]. Note however that most instances of MaskFormer [6] are better than SenForme. Indeed, MaskFormer introduces a new approach for semantic segmentation that is based on mask classification (rather than traditional per-pixel classification) and that greatly improves segmentation performances. In fact, MaskFormer [6] significantly outperforms PerPixelBaseline+ [6] while sharing the same architecture and only differing by the problem formulation *(per-pixel vs mask classification)*. We plan to formulate SenFormer as mask classification in our future work, as it has significant potential to improve segmentation.

Pascal Context. In Table 7 we compare SenFormer to current state-of-the-art methods on Pascal Context test dataset, which is obtained by CAA [21] using EfficientNet-B7(EN-B7) as backbone, with a mIoU of 60.5. SenFormer outperforms previous FCN methods when using standard ResNet backbones, as well as recent transformer-based methods. SenFormer outperforms the current state-of-the-art (CAA) when using the same ResNet-101 backbone, showing the benefit of our approach. Moreover, we reach a score of **64.0** mIoU when using Swin-L as backbone. Overall, our approach shows a significant improvement of +3.5 mIoU over the previous state-of-the-art.

method	backbone	*mIoU*	+MS
DeepLabV3+ [5]	R50	44.0	44.9
PerPixelBaseline+[6]	R50	41.9	42.9
MaskFormer [6]	R50	44.5	46.7
SenFormer	R50	44.4	45.2
Mask2Former [7]	R50	**47.2**	**49.2**
OCRNet [46]	R101	-	45.3
DeepLabV3+ [5]	R101	45.5	46.4
MaskFormer [6]	R101	45.5	47.2
SenFormer	R101	**46.9**	**47.9**
SETR-L MLA [47]	ViT-L	-	50.3
Segmenter [36]	ViT-L	50.71	52.25
Segmenter-Mask[36]	ViT-L	51.82	53.63
SegFormer [45]	MiT-B5	51.0	51.8
UperNet [29]	Swin-L	52.05	53.5
SenFormer	Swin-L	53.08	54.2
MaskFormer [6]	Swin-L	54.1	55.6
Mask2Former [7]	Swin-L	**56.1**	**57.3**

Table 6: Benchmark on ADE20K validation set.

method	backbone	*mIoU*	+MS
DANet [11]	R50	-	50.5
EMANet [26]	R50	-	50.5
CAA [21]	R50	50.23	-
SenFormer	R50	**53.18**	**54.3**
DANet [11]	R101	-	52.6
EMANet [26]	R101	-	53.1
DeepLabV3+ [5]	R101	53.2	54.67
OCRNet [46]	R101	-	54.8
CAA [21]	R101	-	55.0
SenFormer	R101	**54.6**	**56.6**
OCRNet [46]	HRNet	-	56.2
CAA [21]	EN-B7	58.40	60.5‡
SETR-L MLA[47]	ViT-L	54.9	55.8
Segmenter [36]	ViT-L	58.1	59.0
SenFormer	Swin-L	**63.1**	**64.5**

Table 7: Benchmark on Pascal Context test. ‡/blue indicates previous/new SOTA.

4 Discussion

Variance reduction. A common explanation for the better performance of the ensemble over its composing elements is that by averaging the variance over the merged prediction is reduced. To test this assumption, for each input image in the Ade20K validation set, we computed for the ensemble and for each learner the variance over the segmentation map prediction for each pixel (*i.e.*, the variance along the channel axis). We then averaged over the entire validation set. As shown in Table 8, the ensemble variance is not significantly smaller than the variance of the individual learners. Consequently, the variance reduction interpretation may not apply in the context of self-ensemble, and more broadly for deep learning models [1].

Output	var. (10^{-3})
ensemble	56.6
d_2	56.0
d_3	56.8
d_4	56.3
d_5	55.2

Table 8: Ensemble and learners variance on Ade20K val.

# learners per scale	total # learner	*mIoU*
1	4	**44.3**
2	8	44.2

Table 9: Effect of increasing the # of learner.

Multi-view approach. A more recent explanation for the success of Ensemble is that the different learners capture multi-views present in the data [1]. However, since the mutli-scale inputs of the learners come from the same backbone, it is very unlikely that they focus on different views of the objects of interest. We rather hypothesize that in SenFormer the boost in performance does not emerge from the different random initialization of the learners that will learn to focus on specific views of the input image, but rather from the different scale information captured by the FPN. Consequently, using more than one learner per level in the feature pyramid will not yield better results. It is indeed confirmed by results in Table 9 where SenFormer performances do not improve with additional learners.

5 Related works

A major limiting factor for building an ensemble of deep learning models is the computational cost during training and testing. Diverse methods were proposed to tackle this issue. By repeatedly applying dropout at inference on an already trained model, Monte Carlo Dropout [12] allows getting many predictions from a single model, ultimately improving its accuracy. BatchEnsemble [42] significantly lower ensemble cost by defining each learner's weights to be the Hadamard product between a shared matrix and a rank-one matrix per learner. Snapshot [19] train a single model to converge to several local minima by leveraging cyclic learning rate scheduling. Other methods for classification include MIMO [15], hyper-batch ensemble [43], late-phase weights [40] or FGE [13]. For segmentation, [38] improves the widely used multi-scale inference by learning relative attention between the scales during training and is used at test-time to greatly improve the performance. However, these methods still require several forward passes of the same image, let it be for training or testing. Perhaps most related to our work is TreeNet [25], which uses multiple classifier branches that share their early layers. Nevertheless, besides being for classification, unlike to our work, all the learners receive the same input, limiting the depth of the shared part. Moreover, in SenFormer, the parameter cost of the ensemble is further reduced through weight sharing within a learner.

6 Conclusions

This paper introduces our self-ensemble approach for semantic segmentation, a simple methodology that benefits from ensemble learning while avoiding the inconvenience and cost of training multiple times the same model. We leveraged the multi-scale feature set produced by FPN-like methods to build an ensemble of decoders within a *single model*, where learners in the ensemble are fed with features coming from different levels of the feature pyramid. We also developed a transformer-based architecture for the learner/decoders. Our approach outperforms current state-of-the-art on Pascal Context and COCO-Stuff-10K datasets and is competitive on Ade20K and Cityscapes datasets for semantic segmentation. It is more efficient in terms of FLOPs and limit the number of parameter thanks to weight sharing.

Acknowledgements

We thank Romain Fabre for insightful discussion without which this paper would not be possible. This work was partially supported by the National Institutes of Health (NIH), National Cancer Institute (NCI) Human Tumor Atlas Network (HTAN) Research Center (U2C CA233280), and and a NIH/NCI Cancer Systems Biology Consortium Center (U54 CA209988).

References

[1] Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning, 2021.

[2] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context, 2018.

[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] 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)*, pages 801–818, 2018.

[6] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation, 2021.

[7] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1290–1299, 2022.

[8] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020.

[9] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding, 2016.

[10] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers, 2019.

[11] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3146–3154, 2019.

[12] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[13] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31, 2018.

[14] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7036–7045, 2019.

[15] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew Mingbo Dai, and Dustin Tran. Training independent subnetworks for robust prediction. In *ICLR*, 2021.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[17] Mohammad Hesam Hesamian, Wenjing Jia, Xiangjian He, and Paul J. Kennedy. Deep learning techniques for medical image segmentation: Achievements and challenges. *Journal of Digital Imaging*, 32:582 – 596, 2019.

[18] Zhang-Wei Hong, Chen Yu-Ming, Shih-Yang Su, Tzu-Yun Shann, Yi-Hsiang Chang, Hsuan-Kung Yang, Brian Hsi-Lin Ho, Chih-Chieh Tu, Yueh-Chuan Chang, Tsu-Ching Hsiao, Hsin-Wei Hsiao, Sih-Pin Lai, and Chun-Yi Lee. Virtual-to-real: Learning to control in visual semantic segmentation, 2018.

[19] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

[20] Shihua Huang, Zhichao Lu, Ran Cheng, and Cheng He. Fapn: Feature-aligned pyramid network for dense image prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 864–873, 2021.

[21] Ye Huang, Di Kang, Wenjing Jia, Xiangjian He, and Liu Liu. Channelized axial attention for semantic segmentation – considering channel relation within spatial attention for semantic segmentation, 2021.

[22] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. Perceiver: General perception with iterative attention, 2021.

[23] Sotiris B Kotsiantis, Ioannis D Zaharakis, and Panayiotis E Pintelas. Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, 26 (3):159–190, 2006.

[24] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020.

[25] Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David Crandall, and Dhruv Batra. Why m heads are better than one: Training a diverse ensemble of deep networks, 2015.

[26] Xia Li, Zhisheng Zhong, Jianlong Wu, Yibo Yang, Zhouchen Lin, and Hong Liu. Expectation-maximization attention networks for semantic segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9167–9176, 2019.

[27] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017.

[28] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018.

[29] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021.

[30] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014. URL http://arxiv.org/abs/1411.4038.

[31] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 891–898, 2014. doi: 10.1109/CVPR.2014.119.

[32] Toan Q Nguyen and Julian Salazar. Transformers without tears: Improving the normalization of self-attention. *arXiv preprint arXiv:1910.05895*, 2019.

[33] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of artificial intelligence research*, 11:169–198, 1999.

[34] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[35] Mennatullah Siam, Sara Elkerdawy, Martin Jagersand, and Senthil Yogamani. Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges, 2017.

[36] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation, 2021.

[37] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.

[38] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation, 2020.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[40] Johannes Von Oswald, Seijin Kobayashi, Joao Sacramento, Alexander Meulemans, Christian Henning, and Benjamin F Grewe. Neural networks with late-phase weights. In *International Conference on Learning Representations*, 2020.

[41] Xinjiang Wang, Shilong Zhang, Zhuoran Yu, Litong Feng, and Wei Zhang. Scale-equalizing pyramid convolution for object detection. 2020.

[42] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. In *International Conference on Learning Representations*, 2019.

[43] Florian Wenzel, Jasper Snoek, Dustin Tran, and Rodolphe Jenatton. Hyperparameter ensembles for robustness and uncertainty quantification. *Advances in Neural Information Processing Systems*, 33:6514–6527, 2020.

[44] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding, 2018.

[45] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers, 2021.

[46] Yuhui Yuan, Xiaokang Chen, Xilin Chen, and Jingdong Wang. Segmentation transformer: Object-contextual representations for semantic segmentation, 2021.

[47] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H. S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers, 2021.

[48] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ade20k dataset, 2018.

[49] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263, 2002.