Robust Data Augmentations for Segformer

Mathieu van Luijken
Eindhoven Technical University
m.f.a.c.v.luijken@student.tue.nl

***Abstract*—In the field of image segmentation, robustness against varying circumstances in the data is an important factor for model performance. In this study we take a pre-trained Segformer model, which is considered state of the art for image segmentation and has displayed strong results for robustness, and use different data augmentations as well as upscaling at test time to improve on robustness. We test on 8 different weather conditions and score the model on each dataset with a DICE and Mean Intersect over Union score. With our current setup we do not find any meaningful changes in performance between the differently trained models.**

***Index Terms*—Image Segementation, Robustness, SegFormer, Data Augmentations, Superpixel Mix**

I. Introduction

Semantic image segmentation is a task in computer vision where the goal is to classify each pixel in an image into a predefined set of categories, such as different objects like cars and people. Unlike object detection, which identifies objects at the bounding box level, semantic segmentation provides a pixel-level predictions on the image. The task plays a crucial role in various applications, including autonomous driving and medical image analysis [1], [2].
In recent years there have been many Vision transformers that come with many improvements to image segmentation and show impressive results on benchmarking datasets such as ADE20K and Cityscapes [3], [4]. One of these models is the Segformer model which at one point was state of the art for image segmentation [5].
These models are often trained on datasets without image corruptions and noise such as different weather conditions, illumination and camera motion which can heavily impact model performance [6], [7]. Segmentation tasks such as autonomous driving however are safety critical and have a reliance on models which are robust against such corruptions [8].
In this work we study the effect of different data augmentations during training on the performance off corrupted data. We evaluate these models on a dataset which contains images with differing weather circumstances and use a variety of upsampling methods to study their interactive effect on image segmentation.
Our results show that the augmentations we use show only very minor to no improvement on the performance on the weather datasets compared to the baseline. We also observe no difference between different upsampling methods on the image segementation performance.

https://github.com/Mathieu-van-Luijken/Robust-Data-Augmentations-For-Segformer

II. Methods

Segformer Model:

The framework we choose to use will be the Segformer architecture. This model architecture introduced in [5] introduces a framework for semantic segmentation that is efficient, accurate and robust. The framework provides a few key novelties which consist of a positional-encoding-free hierarchical Transformer Encoder and a decoder that is consists of all MLP layers.
The hierarchical transformer encoder as can be seen in Figure 1 consists of Transformer Blocks, these transformer blocks each consist of a number of stackable sub-blocks containing Efficient self-attention and MixFFN layers followed by a single overlap patch merging layer. The Efficient self-attention uses the same sequence reduction process the Pyramid Transformer [9] uses, this means a reduction in computational copmlexity from $O(N^2)$ to $O(\frac{N^2}{R})$ where R is a reduction ratio which is set to $64/4^{(Block\_Nr-1)}$. In many ViT a fixed-size positional encoding is used. In image segmenation this positional encoding has to be interpolated, leading to a drop in accuracy. In the model this problem is alleviated by using a CNN taking the self-attention feature and parsing this through a 3x3 convolution and an MLP. Finally we pass through the overlap patch merging, Which takes the concept of unifying a 2x2xCi feature path into a 1x1xCi+1 vector to obtain hierarchical feature maps and uses a patch size, stride and padding size to perform overlapping patch merging.
The end result of the Encoder Transformer Blocks takes in an input of HxWx3 and transforms it into a hierarchical feature map Fi with resolution $\frac{H}{2^{i+1}}$x$\frac{W}{2^{i+1}}$xCi shrinking the image resolution by a factor of 2 for each block.

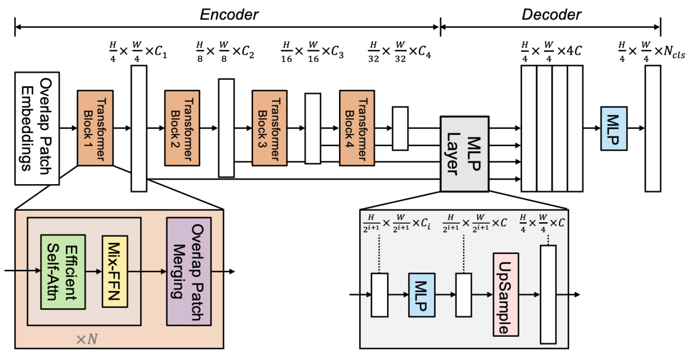

Encoder
Decoder
$\frac{H}{4}\times\frac{W}{4}\times C_1$
$\frac{H}{8}\times\frac{W}{8}\times C_2$
$\frac{H}{16}\times\frac{W}{16}\times C_3$
$\frac{H}{32}\times\frac{W}{32}\times C_4$
$\frac{H}{4}\times\frac{W}{4}\times 4C$
$\frac{H}{4}\times\frac{W}{4}\times N_{cls}$
Overlap Patch Embeddings
Transformer Block 1
Transformer Block 2
Transformer Block 3
Transformer Block 4
MLP Layer
MLP
Efficient Self-Attn
Mix-FFN
Overlap Patch Merging
$\times N$
$\frac{H}{2^{i+1}}\times\frac{W}{2^{i+1}}\times C_i$
$\frac{H}{2^{i+1}}\times\frac{W}{2^{i+1}}\times C$
$\frac{H}{4}\times\frac{W}{4}\times C$
MLP
UpSample

Fig. 1: The model architecture as introduced in [5].

The output of the Encoder is taken by the decoder to produce a segmentation mask of size $\frac{H}{4}$x$\frac{W}{4}$xNcls. The decoder consists of four main steps, the first is an MLP to unify the channel dimension of multi-level features. Then features are upsampled to $\frac{H}{4}$x$\frac{W}{4}$, these upsampled features are then concatenated using an MLP and lastly another MLP is used to create a segmentation mask from the fused features.

Data Augmentations

The original model displays strong performance when comparing to different model frameworks [10] for robustness against Blur, Noise, Digital and Weather. These results were obtained by training the model using only minimal and basic data augmentations. One of the questions we would like to answer then is how does robustness react to training on more complex data augmentation methods. These data augmentations will include: no augmentations to set a baseline for comparison, Basic augmentations as described in the original paper, blurry and digital augmentations and lastly SuperPixel Mix data augmentations.

Basic Data Augmentations:

The original results of the paper were obtained with the following data augmentations:

1) Horizontal Flip: this method flips the image along a vertical axis such that it is now mirrored. This is done with probability p=0.5
2) Random Resize: Randomly resize the image with ratios ranging from 0.5-2.0x the original image size.
3) Random Cropping: Randomly cropping the original image to sizes (512x512) or (1024x1024)

Blurry Data Augmentations:

Three of the original methods for measuring robustness in the model were sub-categorized in Blur, Noise and Digital. To replicate the behaviour of these methods we augment the data in training set using the following data augmentations in addition to the augmentations that we were using in the basic data augmentation method:

1) ColorJitter: jitter the colors in the image with brightness 0.2, contrast 0.2, saturation 0.2 and hue 0.1
2) Gaussian Noise: generate some gaussian noise and lay it over the original image.

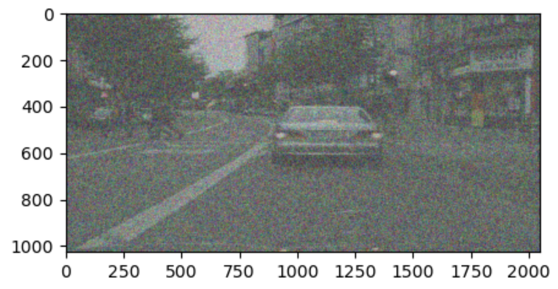
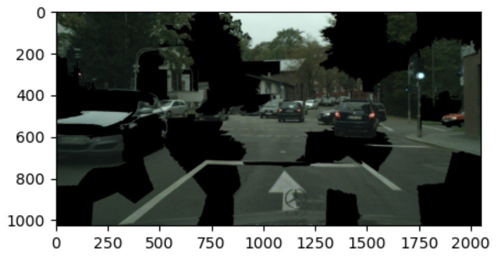


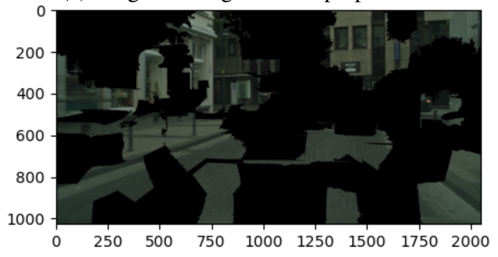
Fig. 2: Blurry data augmentations

Superpixel Mix Data Augmentations:

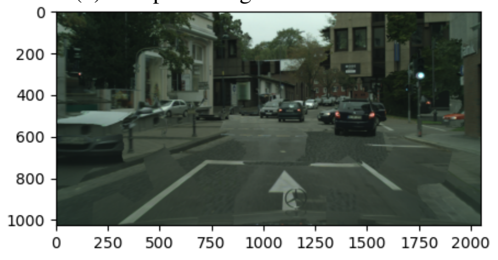
In [11] a novel way of augmenting data based on superpixels is described. The method divides an image in so-called superpixels and mixes those to gain a new image. Superpixels are regions in an image which share characteristics such as color, texture and intensity. The goal of these superpixels is to simplify an image into its most important atomic regions. There are multiple algorithms used for computing these superpixels such as SEEDS[12], SLIC [13] or Watershed [14]. In this research the SLIC method is chosen for superpixel generation, based on its computational simplicity compared to the other methods.



(a) Original Image with superpixel mask



(b) Sampled Image with inverse mask



(c) Final Pixelmix image

Fig. 3: Construction of Superpixel Mix image.

The data augmentation is done in the following manner, first we segment an image based on its superpixels computed by the SLIC algorithm. The algorithm computes at most 100 superpixel regions and numbers each of those. Then from those superpixel regions at random a number of them are chosen and a boolean mask is created for the original image. The resulting image for this boolean mask on the image is displayed in Fig 3a. Then from the dataset another image is sampled at random with exclusion of our original image. We now compute the inverse of the original boolean mask and overlay this on the second randomly sampled image, the result is made visible in Fig 3b.

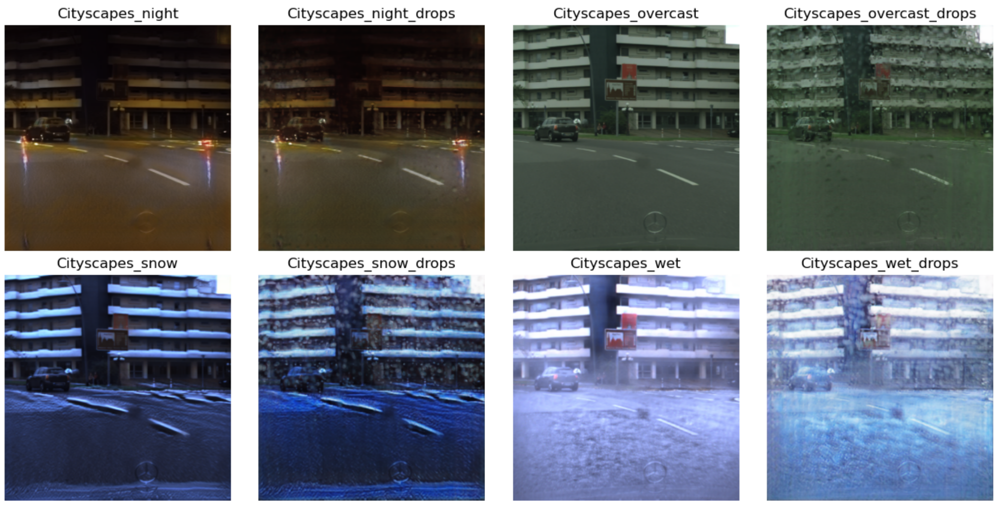


Fig. 4: Example pictures for each of the weather conditions in the evaluation dataset.

Finally we construct our final training image by combining the two images to construct one where each pixel is from either of the two images depending on the chosen superpixel regions and locations of the original image, the final result is visible in Fig 3c. Important to note here is that the target label for this image is also adjusted using the aforementioned masks such that the parts of the image corresponding to each image are replicated in the label as well.

N.B: Originally the intention was to test both the SuperpixelMix and the SuperpixelMix in combination with the Blur augmentations as an ablation study to study the effects of the augmentation. However due to time constraints this was unfortunately infeasible.

Upsampling methods:

The Segformer architecture outputs a segmentation mask, this is crucially different from an image segmentation as the sizes of this mask does not correspond to that of the original image. The mask needs to be upsampled to obtain the sizes corresponding to the original image in order to compute the loss, Mean IoU and Dice scores. There are multiple algorithms which can do this and for this experiment three of those methods are selected each with increasing computational complexity.

Important to note here that at the beginning stage of testing, that the computational complexity of some of these methods makes training take prohibitively long and thus these upsampling methods are only used at test time. The model itself is trained using Billinear upsampling as is proposed in [5].

Bilinear Interpolation:

The bilinear interpolation method as described in [15] describes a method for upsampling images in two dimensions. The interpolation method looks at its neighbouring pixels and uses the distances weighted average of the four nearest pixel values to construct the value of the new pixel. This interpolation method, since it is in 2d space, is computationally very efficient though it might concede some performance.

Bicubic Upsampling:

Bicubic interpolation uses a local polynomial approximation, here it fits a local cubic polynomial function to a neighborhood of pixels surrounding each target pixel in the output. The method is very similar to bilinear interpolation in that it uses pixels in two dimensions to compute the new value for the upsampled pixels. However the difference being of course the interpolation function now being cubic. This higher order polynomial interpolation makes the upsampling method computationally more expensive but in return is better at preserving fine details and smooth transitions.

Lanczos Upsampling:

The interpolation uses a windowed Sinc Filter which is defined as $\frac{sin(x)}{x}$ where x is the distance from the center of the filter and the size of the window is determined by a lanczos kernel. The kernel is determined by two parameters, one controlling the width of the windowed sinc function and one determining the range of neighbouring pixels considered. We use values 4 for both since these values are the base values associated with the library used for the algorithm.

The lanczos interpolation algorithm is computationally very expensive and therefore unfit for training purposes. However it also preserves fine details and edges much better than computationally less expensive methods such as bilinear and bicubic interpolation. This could translate to better performance at test time.

Evaluation

The Evaluation Dataset:

Since the objective here is to increase the robustness of our model, it is very important we test the model in varying circumstances normally not occurring in the dataset. The Multi-Weather City dataset described in [16] describes our original dataset in varying weather conditions. These

Augment	**method**	**Overcast**		**Overcast Drops**		**Night**		**Night Drops**		**Snow**		**Snow Drops**		**Wet**		**Wet Drops**	
		dice	mIoU	dice	mIoU	dice	mIoU	dice	mIoU	dice	mIoU	dice	mIoU	dice	mIoU	dice	mIoU
No	Bil	0.529	0.457	0.362	0.303	0.204	0.156	0.170	0.127	0.399	0.337	0.222	0.183	0.404	0.342	0.212	0.173
	Bic	0.531	0.458	0.363	0.303	0.203	0.156	0.170	0.126	0.401	0.337	0.222	0.183	0.404	0.341	0.212	0.172
	Lanc	0.530	0.456	0.362	0.303	0.203	0.156	0.170	0.127	0.401	0.337	0.222	0.183	0.403	0.341	0.212	0.172
Basic	Bil	0.545	0.471	0.367	0.307	0.219	0.166	0.181	0.134	0.409	0.346	0.231	0.190	0.414	0.350	0.219	0.178
	Bic	0.547	0.472	0.368	0.308	0.217	0.165	0.180	0.133	0.411	0.347	0.231	0.190	0.415	0.350	0.219	0.177
	Lanc	0.544	0.469	0.366	0.306	0.217	0.165	0.181	0.134	0.410	0.346	0.230	0.190	0.415	0.350	0.219	0.177
Blur	Bil	0.548	0.475	0.369	0.306	0.225	0.169	0.182	0.134	0.412	0.344	0.231	0.351	0.416	0.352	0.220	0.177
	Bic	0.547	0.476	0.370	0.306	0.224	0.169	0.183	0.136	0.413	0.343	0.232	0.351	0.417	0.350	0.220	0.179
	Lanc	0.550	0.473	0.369	0.308	0.225	0.171	0.179	0.136	0.410	0.347	0.231	0.351	0.416	0.350	0.220	0.179
Pixelmix	Bil	0.548	0.475	0.368	0.307	0.221	0.168	0.180	0.133	0.414	0.347	0.229	0.189	0.414	0.351	0.219	0.180
	Bic	0.549	0.472	0.368	0.306	0.221	0.168	0.181	0.135	0.413	0.345	0.233	0.191	0.415	0.350	0.219	0.181
	Lanc	0.550	0.475	0.365	0.308	0.222	0.169	0.180	0.135	0.414	0.345	0.231	0.191	0.415	0.350	0.220	0.179

TABLE I: Final Results

conditions are overcast, night, snow and wet. In addition each condition is generated with rain on the lens. An example image can be seen for one of the images originally in the train set in Fig 4. The original paper uses multiple CycleGANs which take an image originally in the Cityscapes dataset called overcast and compute the seven different conditions for this image. Aside from the different weather circumstances the images in the dataset also have a different size to those in the original dataset namely they are resized to 512x1024 and then center cropped to be 512x512.
The Evaluation Metrics:
To evaluate model performance we use two often used metrics called DICE and Mean Intersect over Union.
The DICE Score is computed as:

$$\text{Dice} = \frac{2\times|A\cap B|}{|A|+|B|}$$

The Mean Intersect over Union:

$$\text{IoU} = \frac{\sum_{i=1}^{N}|A_i\cap B_i|}{\sum_{i=1}^{N}|A_i\cup B_i|}$$

III. Results

Training setup:
Training was conducted on a Nvidia A100 GPU for a duration of 25 epochs. The model itself is the pre-trained b1 segformer from nvidia finetuned on the Cityscapes dataset. The loss criterion used is pixel-wise cross entropy and the optimization was done using the Adam optimizer with lr=6e-5. The eventual models are evaluated on the described train set and the metrics used are the DICE and mean IoU metrics.

N.B Originally a training setup was used where the MiT-B1 pre-trained model on ImageNet-1k was used to retrain. This approach had the advantage that it was not fully yet finetuned on the Cityscapes dataset and would thus be more likely to show differences in training for each approach, however was found to need too much compute for convergence.

Model Results:
We report the results of the experiments in Table I. The table displays the twelve different combinations of augmentations and upscaling method for which we note the the DICE and mean IoU scores between each of the different datasets.

To further clarify and condense these many observations in the full final results table we calculate the average DICE and mean IoU scores for each of the augmentation methods and each of the upscaling methods irregardless of datasets in Tab II.

Augment	Dice	mIoU
No	0.313	0.263
Basic	0.323	0.268
Blur	0.325	0.274
Pixelmix	0.325	0.270

Upscale	Dice	mIoU
Bilinear	0.321	0.271
Bicubic	0.322	0.271
Lanczos	0.321	0.271

TABLE II: Average Augmentation and Upscaling Results

IV. Discussion of results

Based on Tables I & II we find that our results show very little difference between different augmentation methods, if any at all. Scores often do not differ and if they do they do so with less than 5% of each other. This result is not in line with earlier results in the papers that were cited. A possible reason for this could be the training approach. By transfer learning a model pre-trained and finetuned on our current dataset and retraining for 25 epochs the model might not have enough time to adjust itself enough to each of the data augmentations.
We further find Upscaling methods have completely no impact as can be seen in Tab II, from this we can surmise that the model prediction is determining factor in the segmentation for the image and the upscaling does not change any semantic labeling from the segmentation mask.
In conclusion we find no differences or improvements between each of the different augmentation methods, though further research with an improved training setup should be conducted for these conclusions to hold.
Future Research:
For future research we propose to take the MiT-b1 model instead of the Cityscapes Finetuned variant and train for a longer time span than 25 epochs until model convergence. We also propose to research research into the combination of all our augmentation methods. Lastly a study into the super-pixelMix augmentation method could be done, for example into the Superpixel algorithm itself or into other methods of augmenting using superpixels.

References

[1] Ç. Kaymak and A. Uçar, "A brief survey and an application of semantic image segmentation for autonomous driving," *Handbook of Deep Learning Applications*, pp. 161–200, 2019.

[2] D. D. Patil and S. G. Deore, "Medical image segmentation: a review," *International Journal of Computer Science and Mobile Computing*, vol. 2, no. 1, pp. 22–27, 2013.

[3] H. Thisanke, C. Deshan, K. Chamith, S. Seneviratne, R. Vidanaarachchi, and D. Herath, "Semantic segmentation using vision transformers: A survey," *Engineering Applications of Artificial Intelligence*, vol. 126, p. 106669, 2023.

[4] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[5] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, "Segformer: Simple and efficient design for semantic segmentation with transformers," *Advances in neural information processing systems*, vol. 34, pp. 12 077–12 090, 2021.

[6] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *2016 eighth international conference on quality of multimedia experience (QoMEX)*. IEEE, 2016, pp. 1–6.

[7] Y. Zhou, S. Song, and N.-M. Cheung, "On classification of distorted images with deep convolutional neural networks," in *2017 IEEE International conference on acoustics, speech and signal processing (ICASSP)*. IEEE, 2017, pp. 1213–1217.

[8] J. Janai, F. Güney, A. Behl, A. Geiger *et al.*, "Computer vision for autonomous vehicles: Problems, datasets and state of the art," *Foundations and Trends® in Computer Graphics and Vision*, vol. 12, no. 1–3, pp. 1–308, 2020.

[9] W. Wang, E. Xie, X. Li, D.-P. Fan, K. Song, D. Liang, T. Lu, P. Luo, and L. Shao, "Pyramid vision transformer: A versatile backbone for dense prediction without convolutions," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 568–578.

[10] C. Kamann and C. Rother, "Benchmarking the robustness of semantic segmentation models," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 8828–8838.

[11] G. Franchi, N. Belkhir, M. L. Ha, Y. Hu, A. Bursuc, V. Blanz, and A. Yao, "Robust semantic segmentation with superpixel-mix," *arXiv preprint arXiv:2108.00968*, 2021.

[12] M. Van den Bergh, X. Boix, G. Roig, and L. Van Gool, "Seeds: Superpixels extracted via energy-driven sampling," *International Journal of Computer Vision*, vol. 111, pp. 298–314, 2015.

[13] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[14] Z. Hu, Q. Zou, and Q. Li, "Watershed superpixel," in *2015 IEEE International Conference on Image Processing (ICIP)*. IEEE, 2015, pp. 349–353.

[15] E. J. Kirkland and E. J. Kirkland, "Bilinear interpolation," *Advanced computing in electron microscopy*, pp. 261–263, 2010.

[16] V. Mușat, I. Fursa, P. Newman, F. Cuzzolin, and A. Bradley, "Multi-weather city: Adverse weather stacking for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 2906–2915.