





Companion Paper: Deep Saliency Map Generators for Multispectral Video Classification

Jens Bayer^{1,2} , David Münch^{1,2} , and Michael Arens^{1,2} 

¹ Fraunhofer Center for Machine Learning, Sankt Augustin, Germany

² Fraunhofer IOSB, Gutleuthausstr. 1, 76275 Ettlingen, Germany

jens.bayer@iosb.fraunhofer.de

Abstract. This is the companion paper for the ICPR 2022 Paper “Deep Saliency Map Generators for Multispectral Video Classification”, that investigates the applicability of three saliency map generators on multispectral video input data. In addition to implementation details of modifications for the investigated methods and the used neural network implementations, the influence of the parameters and a more detailed insight in the training and evaluation process is given.

Keywords: Reproduction Paper · Saliency · Video Classification

1 Introduction

Providing the source code not only gives a better understanding of the statements, it also helps to verify the outcomes of the examined experiments. Nonetheless, even the best-documented source code can still leave questions unanswered, such as how a specific parameterization changes the results or why a selected metric was chosen. In the following, a more detailed view on the examined saliency map generators (Sect. 2) of the addressed Paper [1] is given. In addition, the influence of the methods’ parameters regarding the deletion and insertion metric is shown. Afterwards, the used network implementations are described (Sect. 3). The last section covers the evaluation with the used metrics (Sect. 4).

Our source code can be found at GitHub¹ and requires Python 3.9, PyTorch 1.8.2 LTS, and torchvision 0.9.

2 Deep Saliency Map Generators

The three investigated methods are Grad-CAM [4], Randomized Input Sampling for Explanation (RISE) [3] and the similarity difference and uniqueness method (SIDU) [2]. Since they are usually applied to ordinary images, some adjustments have been made, that are explained in the following. For Table 1 and Table 2, \uparrow (\downarrow) indicates, that a higher (lower) value is better.

¹ <https://github.com/JensBayer/ICPR2022>.

2.1 Grad-CAM

While Grad-CAM not only outperforms its competitors, it is also the simplest to implement. Grad-CAM generates a saliency map by calculating a weighted sum of the forward features of the last convolutional layer. The weights are determined by the gradient of the target class. Grad-CAM requires a forward and backward pass of the input through the network. We therefore register a forward and backward hook at the target layer of our model, to extract the forward features \mathbf{F} and the gradient $\mathbf{G}^c = \frac{\partial y^c}{\partial \mathbf{F}_k}$ for the score y^c of the target class c . Here, k equals the number of forward feature maps, generated by the network. The gradient is summed to obtain the neuron importance weights

$$\alpha_k^c = \sum_{i,j} \mathbf{G}_{kij}^c \quad (1)$$

as described by [4]. The forward features are multiplied with the neuron importance weights and activated via ReLU. The final saliency map

$$\mathbf{S}^c = \text{upscale}(\text{ReLU}(\sum_k \alpha_k^c \mathbf{F}_k)) \quad (2)$$

is given by the upscaled weighted sum of the features. Since 3D ResNet generates three-dimensional forward features, the temporal dimension must also be taken into account in Eq. 1. Additionally, the bilinear interpolation in Eq. 2 changes to a trilinear interpolation.

2.2 RISE

RISE is a Monte Carlo approach that masks the input and calculates a weighted sum of the masks according to the output of the masked input to retrieve a saliency map. For RISE, we largely stick to the official implementation²: First, $n = 1000$ random binary grids of size $\mathbf{s} = 2 \times 8 \times 8$ are sampled in such a way, that the value of a tile of the grid equals one with a probability of $p = 0.1$. Each grid is either bilinear or trilinear, upscaled to a slightly larger size than the input. The resulting grids are randomly cropped to match the input size. After this, the input data is multiplied elementwise with these masks and propagated through the network. The resulting network output \mathbf{P} is then used as a weighting term in the calculation of the final saliency map:

$$\mathbf{S}^c = \sum_k^n \mathbf{P}_k^c \cdot \text{crop}(\text{upscale}(\mathbf{G}_k)), \quad \mathbf{G}_k \in \{0, 1\}^{\mathbf{s}}, \quad P(\mathbf{G}_{kij} = 1) = p \quad (3)$$

As it can be seen in Table 1, the usage of more masks lead to slightly better Deletion and Insertion scores, but comes at the cost of a significant higher computation time. Furthermore, the temporal mask resolution with $\mathbf{s} = 2 \times 8 \times 8$ leads in almost all cases to the best or tied to the best scores. The increased probability of $p = 0.25$ for a grid tile to be nonzero, has also a positive influence and leads to slightly better scores.

² <https://github.com/eclique/RISE>.

Table 1. Deletion and Insertion score for different parameters for RISE, using the IRTV trained network. Only a single parameter is modified, while the remaining two are set to the default values, described in Subject. 2.2.

Image Spectrum	Parameters	3D-ResNet 18		PAN	
		Deletion ↓	Insertion ↑	Deletion ↓	Insertion ↑
TV	$n = 10^2$	0.18 ± 0.13	0.55 ± 0.23	0.27 ± 0.18	0.51 ± 0.22
	$n = 10^3$	0.17 ± 0.09	0.60 ± 0.24	0.22 ± 0.18	0.59 ± 0.22
	$n = 10^4$	0.18 ± 0.08	0.62 ± 0.24	0.22 ± 0.17	0.61 ± 0.24
IR	$n = 10^2$	0.18 ± 0.13	0.48 ± 0.27	0.17 ± 0.14	0.51 ± 0.23
	$n = 10^3$	0.16 ± 0.10	0.54 ± 0.26	0.16 ± 0.14	0.54 ± 0.23
	$n = 10^4$	0.16 ± 0.08	0.57 ± 0.26	0.16 ± 0.13	0.55 ± 0.23
TV	$s = 2 \times 8 \times 8$	0.18 ± 0.10	0.64 ± 0.20	0.23 ± 0.17	0.58 ± 0.24
	$s = 4 \times 8 \times 8$	0.18 ± 0.19	0.56 ± 0.22	0.24 ± 0.17	0.50 ± 0.23
	$s = 8 \times 8 \times 8$	0.16 ± 0.21	0.43 ± 0.30	0.26 ± 0.14	0.45 ± 0.22
IR	$s = 2 \times 8 \times 8$	0.18 ± 0.10	0.61 ± 0.23	0.17 ± 0.14	0.54 ± 0.22
	$s = 4 \times 8 \times 8$	0.18 ± 0.12	0.61 ± 0.24	0.18 ± 0.12	0.54 ± 0.24
	$s = 8 \times 8 \times 8$	0.18 ± 0.16	0.58 ± 0.23	0.22 ± 0.13	0.50 ± 0.23
TV	$p = 0.10$	0.18 ± 0.10	0.61 ± 0.23	0.23 ± 0.18	0.60 ± 0.23
	$p = 0.25$	0.18 ± 0.12	0.61 ± 0.24	0.23 ± 0.19	0.66 ± 0.24
	$p = 0.50$	0.18 ± 0.16	0.58 ± 0.23	0.27 ± 0.21	0.65 ± 0.26
IR	$p = 0.10$	0.18 ± 0.10	0.58 ± 0.23	0.17 ± 0.15	0.53 ± 0.23
	$p = 0.25$	0.17 ± 0.09	0.59 ± 0.23	0.17 ± 0.16	0.62 ± 0.23
	$p = 0.50$	0.18 ± 0.12	0.56 ± 0.24	0.19 ± 0.17	0.60 ± 0.24

2.3 SIDU

SIDU uses the features of the last convolutional layer to mask the input data, propagates the masked input through the network and calculates the similarity differences and a uniqueness scores of the output to finally generate a saliency map. As for RISE, we also stick largely to the official implementation of SIDU³. To extract the forward features, we register a forward hook at the target layer of the used model and perform a forward propagation. The network output $\tilde{\mathbf{P}}$ and the forward features \mathbf{F} are recorded. The masks

$$\mathbf{M} = \text{upscale}(\tilde{\mathbf{M}}), \quad \tilde{M}_{kij} = \begin{cases} 1 & \text{if } F_{kij} > \tau \\ 0 & \text{else} \end{cases} \quad (4)$$

are the result of the binarization of the forward features with a threshold ($\tau = 0.5$), followed by a bilinear or trilinear upscaling. Similar to RISE, the masks are elementwise multiplied with the input data and propagated through the network. The resulting network output \mathbf{P} is used in the calculation of the similarity difference sd and uniqueness u scores. The resulting saliency map

$$\mathbf{S} = u(\mathbf{P}_k) \cdot sd(\mathbf{P}_k, \tilde{\mathbf{P}}) \cdot \mathbf{M}_k \quad (5)$$

is the product of those two scores and the corresponding masks.

³ https://github.com/satyamahesh84/SIDU_XAI.CODE.

Table 2. Deletion and Insertion scores for different τ values.

Image Spectrum	τ	3D-ResNet 18		PAN	
		Deletion ↓	Insertion ↑	Deletion ↓	Insertion ↑
TV	-1	0.35 ± 0.17	0.46 ± 0.19	0.40 ± 0.21	0.49 ± 0.22
	-0.5	0.35 ± 0.17	0.46 ± 0.19	0.40 ± 0.21	0.49 ± 0.22
	0	0.18 ± 0.08	0.67 ± 0.18	0.26 ± 0.14	0.62 ± 0.25
	0.5	0.18 ± 0.09	0.67 ± 0.18	0.23 ± 0.16	0.68 ± 0.22
	1	0.18 ± 0.09	0.67 ± 0.19	0.23 ± 0.15	0.68 ± 0.22
IR	-1	0.29 ± 0.15	0.42 ± 0.19	0.36 ± 0.18	0.48 ± 0.23
	-0.5	0.29 ± 0.15	0.42 ± 0.19	0.36 ± 0.18	0.48 ± 0.23
	0	0.17 ± 0.08	0.64 ± 0.20	0.21 ± 0.12	0.59 ± 0.24
	0.5	0.17 ± 0.08	0.64 ± 0.20	0.18 ± 0.12	0.63 ± 0.21
	1	0.17 ± 0.08	0.63 ± 0.20	0.18 ± 0.12	0.63 ± 0.21

Table 2 shows the influence of τ on the Deletion and Insertion scores. For $\tau \geq 0$, the Deletion and Insertion scores are quite similar.

3 Networks

The investigated network families are 3D-ResNets and the Persistent Appearance Networks (PAN). Since the Multispectral Action Dataset is comparably small, we used the 3D-ResNet 18 provided by torchvision and the official PAN-Lite network implementation⁴.

3.1 3D-ResNet

The target layer for the forward feature extraction of Grad-CAM and RISE, when used with 3D-ResNets, is the output of the last convolutional layer right before the pooling layer. The upscaling for all three investigated methods is trilinear.

3.2 Persistent Appearance Network

Since we use PAN with a ResNet 50 backbone, the forward features are also extracted at the last convolutional layer right before the pooling layer. The upscaling, however, is bilinear for Grad-CAM and SIDU and trilinear for RISE. The trilinear upscaling provides a more temporally stable mask.

⁴ <https://github.com/zhang-can/PAN-PyTorch>.

4 Evaluation

The experiments are evaluated with the Deletion and Insertion metric on sequences of the Multispectral Action Dataset. The fixed train and test split can be found in the git repository, while the dataset can be freely requested.

4.1 Deletion Metric

Given a saliency map $\mathbf{S} \in \mathbb{R}^{t \times h \times w}$ and an input sequence $\mathbf{I} \in \mathbb{R}^{t \times c \times h \times w}$, the Deletion metric (see Algorithm 1) first sorts the indices of the entries of \mathbf{S} in descending order, according to their values. Afterwards, the sorted indices are separated in n coherent parts and the values of \mathbf{S} are successively replaced with a fixed value $v = 0$, according to the partition order. After each part, the classifier f computes the class probability for the target class t of the modified input. The class probability after the i th part is recorded in \mathbf{p}_i . Finally, the area under the curve of the entries of \mathbf{p} and the linear spaced values from 0 to 1 in n steps is returned.

Algorithm 1. Deletion score calculation.

```

1: function DELETIONSCORE( $f, \mathbf{S}, \mathbf{I}, n, v, t$ )
2:    $sorted\_idx \leftarrow argsort(\mathbf{S})$ 
3:    $parts \leftarrow split(sorted\_idx, n)$ 
4:    $\mathbf{p} \leftarrow \mathbf{0}^n$ 
5:   for  $i = 0, \dots, n$  do
6:     for all  $j \in parts_i$  do
7:        $I_j \leftarrow v$ 
8:     end for
9:      $\mathbf{p}_i \leftarrow \sigma(f(\mathbf{I}))_t$  ▷  $\sigma$  is the softmax function
10:  end for
11:   $score \leftarrow AreaUnderCurve(\mathbf{p})$ 
12:  return  $score$ 
13: end function

```

Algorithm 2. Insertion score calculation.

```

1: function INSERTIONSCORE( $f, \mathbf{S}, \mathbf{I}, n, t$ )
2:    $sorted\_idx \leftarrow argsort(\mathbf{S})$ 
3:    $parts \leftarrow split(sorted\_idx, n)$ 
4:    $\tilde{\mathbf{I}} \leftarrow blur(\mathbf{I})$ 
5:    $\mathbf{p} \leftarrow \mathbf{0}^n$ 
6:   for  $i = 0, \dots, n$  do
7:     for all  $j \in parts_i$  do
8:        $\tilde{I}_j \leftarrow I_j$ 
9:     end for
10:     $\mathbf{p}_i \leftarrow \sigma(f(\tilde{\mathbf{I}}))_t$ 
11:  end for
12:   $score \leftarrow AreaUnderCurve(\mathbf{p})$ 
13:  return  $score$ 
14: end function

```

4.2 Insertion Metric

Similar to the Deletion metric, the Insertion metric (see Algorithm 2) successively unblurs a blurred version \tilde{I} of the input data I , according to the importance score of the given saliency map S .

5 Conclusion

This paper shows the impact of different parameter choices for already existing methods, namely Grad-CAM, RISE and SIDU, when applied not to ordinary images but rather video input data in the visual and long-wave infrared spectrum. To quantify the results, the Deletion and Insertion metric are used. While for RISE, a higher number of generated masks seems to improve the scores, a higher temporal mask resolution seems to be counterproductive. The probability parameter used by RISE seems not to have a big impact. For SIDU, the default value for the threshold $\tau = 0.5$ results in most cases in the best or close to the best scores.

Acknowledgements. This work was developed in Fraunhofer Cluster of Excellence “Cognitive Internet Technologies”.

References

1. Bayer, J., Munch, D., Arens, M.: Deep Saliency Map Generators for Multispectral Video Classification. In: 2022 26th International Conference on Pattern Recognition (ICPR), pp. 3757–3764. IEEE (8 2022). <https://doi.org/10.1109/ICPR56361.2022.9955639>. <https://ieeexplore.ieee.org/document/9955639/>
2. Muddamsetty, S.M., Mohammad, N.S.J., Moeslund, T.B.: SIDU: Similarity Difference And Uniqueness Method for Explainable AI. In: 2020 IEEE International Conference on Image Processing (ICIP), pp. 3269–3273. IEEE (10 2020). <https://ieeexplore.ieee.org/document/9190952/>
3. Petsiuk, V., Das, A., Saenko, K.: RISE: Randomized input sampling for explanation of black-box models. In: British Machine Vision Conference (BMVC) (2018)
4. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-CAM: visual explanations from deep networks via gradient-based localization. *Int. J. Comput. Vision* **128**(2), 336–359 (2019). <https://doi.org/10.1007/s11263-019-01228-7>