Cote *et al.*
EURASIP Journal on Image and Video Processing (2023) 2023:13
https://doi.org/10.1186/s13640-023-00613-0

EURASIP Journal on Image and Video Processing

RESEARCH Open Access

Semantic segmentation of textured mosaics

Melissa Cote[1*], Amanda Dash[1] and Alexandra Branzan Albu[1]

*Correspondence:
mcote@uvic.ca
[1] Electrical and Computer Engineering, University of Victoria, Victoria, BC, Canada

Abstract

This paper investigates deep learning (DL)-based semantic segmentation of textured mosaics. Existing popular datasets for mosaic texture segmentation, designed prior to the DL era, have several limitations: (1) training images are single-textured and thus differ from the multi-textured test images; (2) training and test textures are typically cut out from the same raw images, which may hinder model generalization; (3) each test image has its own limited set of training images, thus forcing an inefficient training of one model per test image from few data. We propose two texture segmentation datasets, based on the existing Outex and DTD datasets, that are suitable for training semantic segmentation networks and that address the above limitations: SemSegOutex focuses on materials acquired under controlled conditions, and SemSegDTD focuses on visual attributes of textures acquired in the wild. We also generate a synthetic version of SemSegOutex via texture synthesis that can be used in the same way as standard random data augmentation. Finally, we study the performance of the state-of-the-art DeepLabv3+ for textured mosaic segmentation, which is excellent for SemSegOutex and variable for SemSegDTD. Our datasets allow us to analyze results according to the type of material, visual attributes, various image acquisition artifacts, and natural versus synthetic aspects, yielding new insights into the possible usage of recent DL technologies for texture analysis.

Article highlights

- We propose two texture segmentation datasets that address the limitations of existing texture segmentation datasets.
- Experiments with materials and attributes shed a new light on recent deep learning technologies for texture analysis.
- Our results also suggest that synthetic textures can be used for data augmentation to improve segmentation results.

Keywords: Texture segmentation, Semantic segmentation network, Textures in the wild, Visual attributes, Deep learning, Texture synthesis

1 Introduction

This paper explores the problem of textured mosaic segmentation via a semantic segmentation lens.

1.1 Context

Texture contains vital information about how surfaces are structurally composed and how they relate to their surroundings [1]. It is an important cue that enables *human* vision to identify regions of interest in an image [1]. This is also true for *computer* vision (CV) systems; texture analysis plays a crucial role in many applications, such as medical image analysis (e.g., [2]), remote sensing (e.g., [3]), document image analysis (e.g., [4, 5]), etc. Image segmentation is the partitioning of an image into homogeneous regions or segments. It is a prerequisite for the success of various CV tasks [6], such as content-based image retrieval, scene analysis, quality control, etc. One of the main issues lies in formulating a valid homogeneity criterion, which is especially difficult for textured regions when considering classic texture analysis methods, in which features are extracted via hand-crafted descriptors.

Deep learning (DL) methods and convolutional neural networks (CNNs) are expanding the limits of CV technology by accomplishing tasks that had been considered impossible in the past [7]. In particular, CNNs have been shown to automatically extract features from images suitable for a wide variety of tasks. Most of the work on texture analysis with DL methods has been focused so far on automatic feature extraction for textured image classification, which typically involves single-texture images. The challenges are different for textured image segmentation as it involves multi-texture images, with a need to identify the boundaries between adjacent textures.

In this paper, we investigate DL-based segmentation of textured mosaics by formulating the problem as semantic segmentation. Semantic segmentation identifies which pixels belong to each object class (in our case texture class), effectively partitioning the images into homogeneous regions. Many DL-based semantic segmentation systems (e.g., [8–13]) are available; they have been developed and tested on various types of images, some of which are texture-intensive (e.g., medical, satellite). However, state-of-the-art semantic segmentation networks have not been extensively studied for classic texture segmentation problems, i.e., involving mosaics of materials.

1.2 Contributions

Our contributions are threefold. Firstly, we propose two texture segmentation datasets suitable for training semantic segmentation networks. They possess three critical characteristics that we do not find in existing public texture segmentation datasets, such as Outex [14] and Prague [6] (see Sect. 2.4): (1) training images are similar to test images, being multi-textural (mosaics of textures) instead of single-textural images; (2) training and test images come from different source images to prevent data leakage, which is good practice and allows for the training of models that generalize better and do not overfit; (3) there is one set of training images for the entire set of test images, allowing for the training of a single model that can be tested on all test images (the design of existing datasets forces the training of one fine-tuned model per test image). The first

dataset (SemSegOutex), based on the classic Outex images [14], features mosaics of traditional texture classes (materials, such as "carpet"), as well as a variation that focuses on synthetically generated textures. The second dataset (SemSegDTD), based on the more recent Describable Textures Dataset (DTD) images [15], showcases more difficult cases as the mosaics are constructed from textures in the wild, and the texture classes are attributes, such as "fibrous", instead of materials. Secondly, we study the performance of a state-of-the-art semantic segmentation network (DeepLabv3+ [10]) on the problem of controlled and uncontrolled textured mosaic segmentation. We utilize our datasets to analyze results according to the material type, visual attribute, and natural versus synthetic aspects, yielding new insights into the possible usage of recent DL technologies for texture analysis. Thirdly, we show that texture synthesis is a viable option for data augmentation purposes that is on par with standard image data augmentation techniques, such as random flips and color/contrast variations.

The remainder of the paper is divided as follows. Section 2 reviews related works, Sect. 3 describes our datasets and semantic segmentation network setup, Sect. 4 discusses experimental results, and Sect. 5 offers concluding remarks.

2 Related works

This section provides an overview of the literature on texture analysis from three different angles: classic hand-crafted texture descriptors, which can be used for texture classification and segmentation problems, DL-based texture analyses that include DL-based texture descriptors and other relevant DL work, and available texture datasets. We conclude the section with the main takeaways from the literature.

2.1 Classic texture analyses

Hand-crafted texture descriptors are typically used in conjunction with traditional supervised machine learning techniques to accomplish texture classification or segmentation tasks. A wide variety of descriptors have been proposed over the last decades to characterize textured images; the interested reader is referred to the work of Humeau-Heurtier [16] for a recent thorough survey of texture feature extraction methods, which they group into seven categories: statistical, structural, transform-based, model-based, graph-based, learning-based, and entropy-based approaches. Statistical, transform-based (also known as spectral), and learning-based methods are arguably the most studied ones. Statistical descriptors describe the spatial distribution of gray-level values within an image region. Examples of popular statistical texture descriptors include the gray-level co-occurrence matrices (GLCM) [1], from which various measures can be derived (contrast, correlation, energy, entropy, etc.), and the local binary pattern (LBP) operator [17] and its many variations. GLCM are easy to implement but are very sensitive to the choice of the distance parameter. LBPs have the advantage of combining structural and statistical information and of being invariant to monotonic illumination changes, but are sensitive to noise in near-uniform regions. Transform-based descriptors typically rely on filter banks designed to focus on various ranges of frequencies and local spatial interactions. Gabor filter banks [18], utilized in many texture segmentation applications, are particularly interesting due to their optimality for minimizing the joint two-dimensional uncertainty in space and frequency [19], but yield a high dimensional

feature space, which can be further reduced using for instance sparseness measures [20]. The work by Yuan et al. [21] constitutes an example of successful use of filter banks for texture segmentation. Learning-based descriptors are built on the concept of dictionaries of visual patterns, pioneered by Leung and Malik [22]: a repeated pattern or texton is described by features clustered in the same visual word in the feature space. Bags of textons are then used to describe textures.

2.2 Deep learning-based texture analyses

Contrary to classic texture descriptors, CNN-based features are automatically extracted via supervised learning and thus do not require (as much) hand-crafted design. In many works, CNN layers are viewed as filter banks of increasing complexity with respect to the layer depth. These methods build orderless representations on top of CNN activations [23]. Expanding on the idea that the overall shape information extracted by the fully connected (FC) layers of a CNN is of minor importance in texture analysis, Andrearczyk and Whelan [24] proposed Texture CNN (T-CNN), pooling an energy measure from the last convolution layer which is connected to an FC layer. T-CNN showed good results, but they were tested for texture classification only. Cimpoi et al. [25, 26] proposed FV-CNN, extracting CNN features from convolutional layers at multiple scale levels and then performing orderless Fisher Vector pooling. Texture segmentation is accomplished by generating region proposals from low-level cues and then utilizing a traditional classifier (support vector machine (SVM)) on the FV-CNN features for region classification. Compared to local binary patterns and other CNN-based descriptors, FV-CNN was the best for textures with considerable appearance variations, using classic SVM classifiers [27]. Lin et al. [28] aggregated second order statistics of CNN features to construct an orderless texture representation in their bilinear CNN (B-CNN). Tested for texture classification problems, B-CNN showed comparable performance with FV-CNN. Bello-Cerezo et al. [29] compared traditional hand-crafted descriptors against off-the-shelf CNN-based features for the classification of different types of textures under a range of imaging conditions. They found CNNs to have a marked superiority for the classification problem, particularly for non-stationary textures and in the presence of multiple changes in the acquisition conditions.

Few works have featured DL semantic segmentation networks for texture segmentation. Andrearczyk and Whelan [30] have shown that fully convolutional networks (FCNs) [9] can learn to perform semantic segmentation from classic texture recognition datasets with non-segmented images and outperform classic descriptors. Karabağ et al. [31] compared five classic descriptors with U-Net [8] and found that U-Net performed better in four out of six textured mosaics. Yamada et al. [32] extracted features from CNNs and Siamese networks and performed texture segmentation via hierarchical region merging based on a region adjacency graph. Compared to U-Net, their method performed best for unknown texture. Zhu et al. [33], noting that texture features should reflect local structure and include global statistical knowledge of input images, proposed Statistical Texture Learning (STL)-Net based on texture enhancing and pyramid texture feature extraction modules. Their approach was not evaluated on texture datasets.

2.3 Texture datasets

Many texture datasets have been proposed, but most are for texture classification of single-texture images, with CUReT, KTH-TIPS2, and DTD among the most popular ones. CUReT (Columbia-Utrecht Reflectance and Texture Database) [34] contains 61 classes spanning a range of geometric and photometric properties. KTH-TIPS2 (KTH Textures under varying Illumination, Pose and Scale) [35], covering 11 materials, has decent intra-class variety due to the use of multiple distinct samples of different materials. DTD (Describable Textures Dataset) [15] contains textural images in the wild (from Google and Flickr) organized in 47 human-centric attributes, as opposed to materials. For texture segmentation, two datasets stand out in terms of popularity: Outex and Prague. Outex's texture segmentation suite [14] contains 100 test mosaics of the same template, with five among 12 possible texture classes. Outex's raw data of macro- and micro-surface textures (319 texture classes organized in 29 categories) were acquired under controlled variations of illumination, rotation and spatial resolution. Prague (Prague Texture Segmentation Datagenerator and Benchmark) [6] is the most comprehensive benchmark for texture segmentation algorithms, offering a suite of metrics and tailored computer-generated texture mosaic sets for download. The images have three to 12 textures and cover a range of geometric and photometric properties. Prague also offers rotation-, scale-, and illumination-invariant sets and noisy versions.

2.4 Takeaways

Classic texture descriptors present a significant downside: they need to be manually tailored for each dataset. CNN-derived approaches can automatically learn features from raw data, assuming sufficient annotated training data are available. DL approaches based on semantic segmentation networks utilize older architectures (like U-Net and FCN), except for STL-Net which was not assessed on texture datasets. Existing popular texture segmentation datasets (Outex [14] and Prague [6]), designed prior to the DL era, have several limitations: (1) training and test textures are typically taken from different regions of the same raw image, which may prevent models from performing well on previously unseen data; (2) a single training image per class present in a test mosaic is made available, with each test mosaic image considered as a separate problem, thus impractically requiring the training of several models from few data; (3) training images are single-textured, thus very different from actual test mosaics. Our work distinguishes itself by exploring the performance of a state-of-the-art semantic segmentation network, DeepLabv3+, on two new texture segmentation datasets that address these limitations.

3 Methods

We base our study of textured mosaic image segmentation on the state-of-the-art semantic segmentation network DeepLabv3+ [10]. Focusing on a single architecture allows us to thoroughly study the segmentation performance with respect to specific variables, such as the type of material, visual attributes, and image acquisition artifacts. This section describes our two datasets, SemSegOutex and SemSegDTD, and our DeepLabv3+ setup for their semantic segmentation.

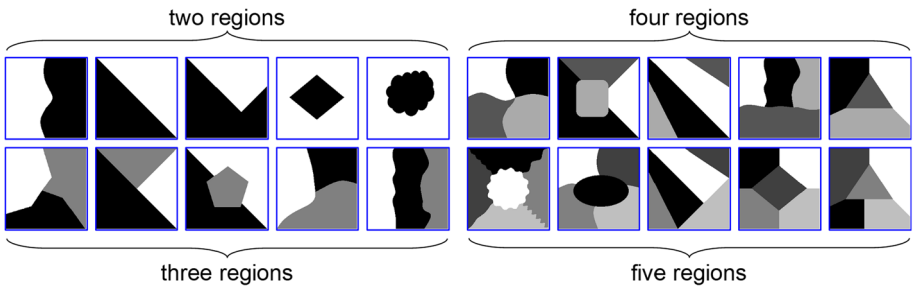

Fig. 1 All 20 templates used to generate training and test mosaics for SemSegOutex and SemSegDTD

3.1 Datasets

We created two datasets from existing texture datasets to address segmentation via textured mosaics, which are made publicly available online[1][2] The need for new data stems from the limitations of existing public datasets (Sect. 2.4), relating to the need for training and test images from separate source images, for one set of training images for all test images, and multi-textured training images.

We generate mosaics similar to the Outex texture segmentation suite and the Prague dataset, but we do this for both training and test images, utilizing croppings from different source images for the training and test mosaics. One could argue that computer-assembled textured mosaics may not be a realistic representation of the assemblages of textured surfaces found in real-life situations, or that training with artificial mosaics could bias the network with respect to boundary decisions at inference time [30]. However, Mikes and Haindl [6] have shown that the ranking of segmentation approaches on such computer-assembled mosaics correlates well with experiments on natural scenes. In addition, two sizable advantages of computer-assembled mosaics are that we have access to the exact ground truth, and we can generate large training and test sets with varied combinations of textures [6]. Both SemSegOutex and SemSegDTD utilize the 20 mosaic templates shown in Fig. 1, which are 224×224 pixels and have two to five regions. One template is from the Outex texture segmentation suite (left-most in five-region group), one is from the Prague dataset (left-most in three-region group), and the others are new. The width and coverage of each region and the type of boundary (wavy, curvy, straight) vary across the templates.

3.1.1 Materials: SemSegOutex

SemSegOutex is created from source images from the classic Outex dataset [14] of materials. This allows us to study the performance of a state-of-the-art semantic segmentation network in the context of traditional texture segmentation problems. Outex includes textures from various pixel pattern categories (as defined in [36]): random, periodic, mixed.

We design two main experiments, one with five classes ("5-class") and one with 10 classes ("10-class") which includes five additional classes, to test scalability in terms of

[1] http://web.uvic.ca/~mcote/TextureMosaics/SemSegOutex_v1p0.zip.

[2] http://web.uvic.ca/~mcote/TextureMosaics/SemSegDTD_v1p0.zip.

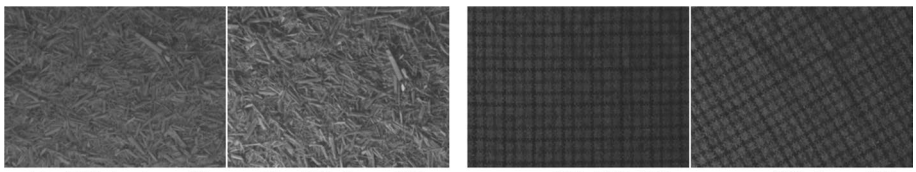

Fig. 2 Sample source images from Outex acquired under various illumination and rotation for the "chips010" and "canvas005" classes, used to generate mosaics in SemSegOutex

number of classes. For each class, 12 different source images are cropped (224×224 central region) and split into 10/2 to create training and test mosaics, respectively. The 12 source images, all of the same size, are acquired under three controlled variations of illumination ("tl84", "inca", "horizon") and four controlled variations of rotation (0, 30, 60, 90 degrees), yielding to (some) differences in shadows and gray-level values. The 10 classes ("barleyrice002", "canvas005", "canvas022", "carpet005", "chips010", "paper004", "pasta006", "plastic005", "seeds004", "wallpaper006") were randomly selected from the available classes, covering small to large inter-class variations. Figure 2 shows sample source images. For the 5-class experiment, each mosaic template (see Fig. 1) is utilized 80 and 20 times (the 100 figure is an arbitrary choice) to create a total of 1600 training and 400 test mosaics, respectively. For the 10-class experiment, as there are twice the classes, each mosaic template is utilized 160 and 40 times for twice the training (3200) and test (800) images being generated, respectively. Each time a mosaic template is used, its regions are assigned a random class label from [1:5] or [1:10]; which cropped training or test image is used to fill the region space is also selected randomly.

We create a variation of SemSegOutex, called SemSegOutex-Synth, that focuses on synthetically generated textures from reference natural textures. The goal is to allow for the study of segmentation performance with respect to the synthetic aspect of textures. In particular, this will allow us to verify whether DeepLabv3+ performs better, equal, or worse for synthetically generated textures compared to natural textures, and also to explore the use of synthetically generated textures as a means to replace traditional data augmentation based on random image operations.

After experimenting with a few texture synthesis approaches, we opted to use the recent method proposed by Brochard et al. [37] based on wavelets. In that work, the authors propose a family of statistics built upon non-linear wavelet-based representations that bridges the gap between wavelet-based classic models and CNN-based state-of-the-art models. Their approach allows for the production of textures of similar quality to state-of-the-art models with more interpretable representations. One sizable advantage of their approach is that it accepts input reference textures, thereby allowing us to generate synthetic versions of each Outex source image for sound natural versus synthetic comparisons.

We used Brochard et al.'s PyTorch implementation [38] to generate five randomly seeded different synthetic textures for each (natural) cropped source image. Figure 3 shows sample natural source textures and one of their synthetic versions. We can see that the synthetic textures are highly realistic. However, they present some slight synthetic irregularities in the case of regular (stationary) reference textures (e.g., akin to

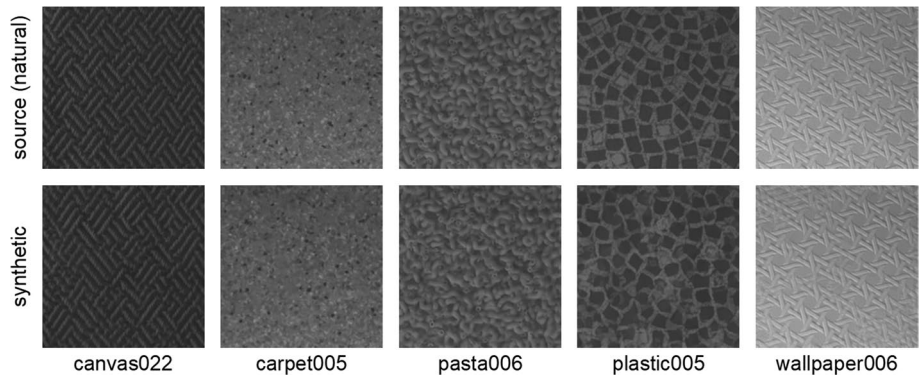

Fig. 3 Sample cropped source images from Outex, used to generate mosaics in SemSegOutex (top row), and one corresponding synthetic version, used to generate mosaics in SemSegOutex-Synth (bottom row)

an amateur crochet version in "canvas022" or "wallpaper006") and may include some slightly distorted shapes (e.g., weirdly shaped macaroni in "pasta006" or dark fragments in "plastic005"). These differences make for an interesting comparative analysis (see Sect. 4.4). As for SemSegOutex, we create training and test mosaics from the synthetic textures for the 5- and 10-class experiments. Since we have five different synthetic versions of each cropped source image, we create five different subsets per experiment, one for each seed, yielding five times the data in SemSegOutex-Synth compared to SemSegOutex. These synthetic data also allow us to explore their use for data augmentation purposes (see Sect. 4.5).

3.1.2 Attributes: SemSegDTD

SemSegDTD is created from source images from the challenging DTD dataset [15], featuring attributes as class labels for textures in the wild. Utilizing the same three pixel pattern categories as in Outex to qualify each class does not apply in the case of DTD, as the source images contain many non-stationary textures in all classes, for instance textures with large-scale irregular structures, textures exhibiting spatial variance in color, local orientation, and local scale, as well as heterogeneous textures, such as weathered surfaces [39]. We are in fact referring to data non-stationarity within cropped images here, as opposed to visual non-stationarity (containing multiple types of textures) [40], which we find in the texture mosaics. Thus, it is impossible to assign one category to each class that would fit all its samples, as some samples are random, some are periodic, and others are mixed. SemSegDTD allows us to study the performance of a state-of-the-art semantic segmentation network in the context of more difficult texture segmentation problems.

As for SemSegOutex, we designed experiments with five and 10 classes, but this time, there are two versions: *standard* ("5-class" and "10-class"), in which the sources images present intra- and inter-class variations with minimal distortions and occlusions, and *hard* ("5-class (hard)" and "10-class (hard)"), which features the same classes but with completely different source images presenting high intra- and inter-class variations with distortions and some occlusions. For each class, 12 different source images (same quantity as for SemSegOutex), all of varying sizes and selected randomly from the available

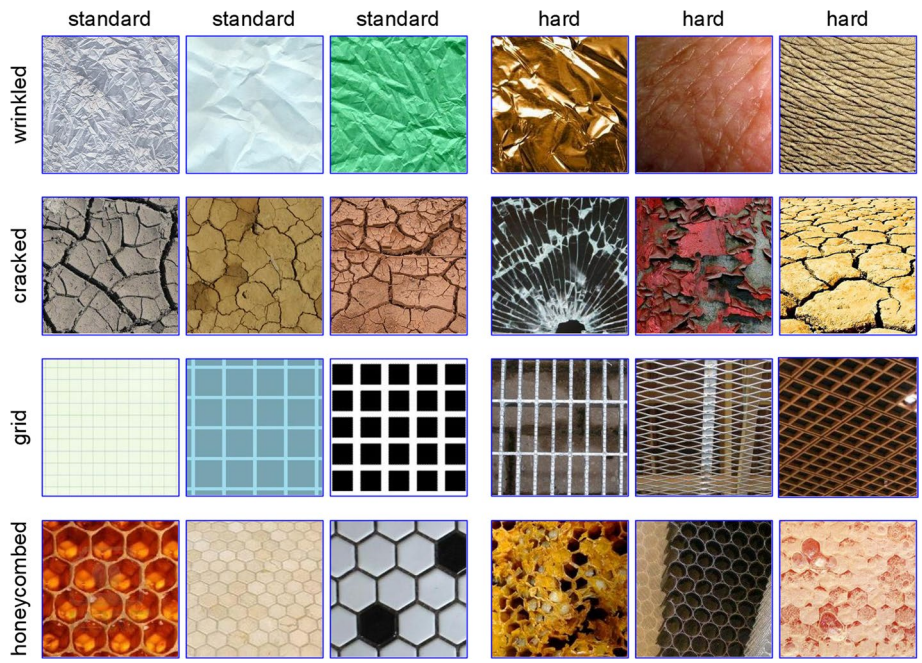

Fig. 4 Sample standard and hard cropped images from DTD acquired in the wild for the "wrinkled", "cracked", "grid", and "honeycombed" classes, used to generate mosaics in SemSegDTD. Many samples are non-stationary textures

images in DTD, are cropped (224 × 224 central region) and split into 10/2 to create training and test mosaics, respectively. The 10 classes ("banded", "chequered", "woven", "wrinkled", "fibrous", "cracked", "grid", "honeycombed", "polka-dotted", and "zigzagged") were randomly selected from the available attribute classes and then vetted so that no two classes were close synonyms (e.g., "banded" and "lined"). The training and test mosaics were created in the same way as for SemSegOutex. Figure 4 shows representative sample cropped images used to generate mosaics, for both the standard and hard versions. Looking at the "wrinkled" class, the standard samples are from different objects but similar in material (crumpled paper), whereas the hard samples are from different objects of different materials and have different reflection properties (glossy metallic foil, matte human and elephant skin). The "cracked" hard samples are the results of different phenomena and materials (cracked glass, peeling paint, and dry soil), whereas the standard samples are from similar materials but different objects. The "grid" standard samples are from different materials (graph paper and computer-generated) but share the same perpendicular view, whereas the hard samples are from angled views with background objects visible through the grids, which have different basic shapes. Finally, the "honeycombed" hard samples may show significant degradation in the patterns compared to the standard samples.

3.2 Semantic segmentation network

DeepLabv3+ [10] is a well-understood and well-designed model that is still considered among the state-of-the-art models for semantic segmentation on natural images (PASCAL VOC and Cityscapes datasets) [41] and that remains highly popular despite

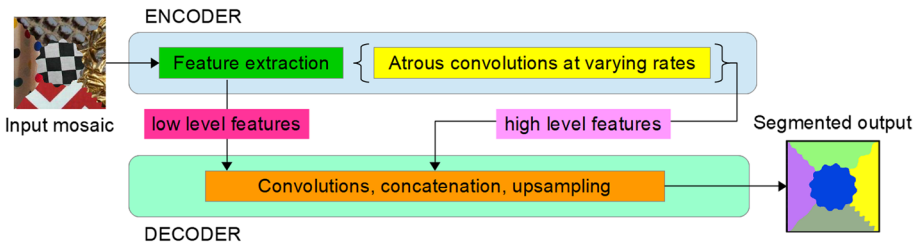

Fig. 5 Applying DeepLabv3+ on textured mosaics

being several years old. It also does not require as much data as transformer-based networks (e.g., [13]) and does not tend towards memorization/overfitting as much as the U-Net family of semantic segmentation networks (e.g., [8, 12]). We thus selected it to study the impact of materials, attributes, and image acquisition artifacts on semantic segmentation performance.

DeepLabv3+ is the latest iteration of a series of DeepLab models by the original authors, combining the advantages of spatial pyramid pooling and encoder–decoder structures. The former encodes multi-scale contextual information by probing features with pooling (or filter) operations at multiple rates and effective fields-of-view (DeepLabv3+ applies several parallel atrous convolutions with different rates, exploiting multi-scale information well). The latter contain two main modules: an encoder that gradually reduces the feature maps, capturing higher semantic information, and a decoder gradually recovering the spatial information. DeepLabv3+ utilizes DeepLabv3 as the encoder module and adds a simple yet effective decoder module to obtain sharper segmentations along the object boundaries. Recovering precise boundaries in computer-generated textured mosaics (as in SemSegOutex and SemSegDTD) can be challenging due to the arbitrary cropping of the textural patterns filling the mosaic regions, compared to the more organic transitions in natural scenes. Figure 5 illustrates the high-level DeepLabv3+ structure in the context of segmenting textured mosaics.

We used a PyTorch implementation [42] of DeepLabv3+ with a ResNet-101 backbone pre-trained on ImageNet [43], without separable convolutions for the sake of simplicity. The model was trained for 100 epochs with a stochastic gradient descent optimizer (1e-4 weight decay and 0.9 momentum) with a polynomial learning rate scheduler (0.9 power factor), batch size of 16, output stride of 16, and learning rates of 0.01 (backbone) and 0.1 (rest of the net). We used a standard cross-entropy loss function and standard data augmentations, applied in a pipeline: horizontal flip (50% probability), random crop, random color jitter, and random brightness contrast. The color jitter and brightness contrast each have three parameters selected from a uniform distribution when the augmentation is applied. For color jitter, the distribution is N (0, 0.5) for 3 parameters and for brightness contrast, it is N (0, 0.2) for 2 parameters. For cropping, the images are resized by adding 30 pixels (height and width) and then randomly cropped back to the original size of 224×224. We also normalized the images using mean and standard deviation values of (0.485, 0.456, 0.406) and (0.229, 0.224, 0.225), respectively, derived from ImageNet.

4 Experimental results and discussion

In this section, we evaluate DeepLabv3+'s performance for materials (SemSegOutex), attributes and various additional challenges (SemSegDTD), the number of regions present, natural versus synthetic materials (SemSegOutex-Synth), and using synthetic data versus standard data augmentation. We also provide comparisons with a classic texture-based image segmentation method and with another recent DL semantic segmentation network. For a quantitative analysis, we rely on the following metrics, which are widely used for semantic segmentation problems [44, 45]: pixel accuracy (PA, i.e., proportion of correctly classified pixels), mean pixel accuracy (mPA, i.e., per-class PA averaged over all classes), per-class intersection over union (IoU_i for class i), and mean intersection over union (mIoU, i.e., IoU_i averaged over all classes). All experiments were carried out on a hardware setup of four NVIDIA Tesla P100 GPUs, except for the ones related to the classic method, which were carried out on a hardware setup of one NVIDIA GeForce RTX 3060 GPU.

4.1 Performance with respect to materials

SemSegOutex allows us to evaluate DeepLabv3+'s performance for material types. Table 1 shows the performance metrics on SemSegOutex for the 5- and the 10-class experiments, also indicating the pixel pattern category for each class. Looking at the PA, mPA, and mIoU values, DeepLabv3+ yields a high accuracy in both experiments. Interestingly enough, increasing the number of classes from five to 10 does not decrease performance. IoU_i values are all very high (> 0.98), with the highest for "canvas005" and "wallpaper006" in the 5- and 10-class experiments, respectively, and the lowest for "chips010" in both experiments. Since IoU_i values are all similar and computed globally for all images, we verify if the difference in performance between classes is statistically significant via a two-sample t-test applied to each class pair, looking at individual test images: we compare the set of IoU values of the test images containing the class pair, allowing us to verify whether there is any statistically significant difference on a per-image level. Considering a significance threshold of .05, for the 5-class experiment, the only class pairs to have no statistical difference are "barleyrice002-carpet005", "barleyrice002-chip010", and "canvas005-canvas022", with p = .22, .06 and .86, respectively. This means that the best-performing class "canvas005" ($\text{IoU}_2 = 0.9888$) rightly outperforms "barleyrice002", "carpet005", and "chips010". For the 10-class experiment, out of 45 class pairs, 10 have no statistical difference. The best class "wallpaper006" ($\text{IoU}_{10} = 0.9936$) is the only one to show a statistically significant difference with all other classes. Confusion matrices are not shown due to very low misclassifications (≤ 0.004 and ≤ 0.002 in the 5- and 10-class experiments, respectively). Looking at the pixel pattern categories for the 10-class experiment, as expected, the three textures with periodic patterns yield the highest performances (IoU_i in the 0.99 s), followed by the class with a mixed pattern. The classes with random patterns consistently yield the lowest performances (although still high with IoU_i in the 0.98 s). For the 5-class experiment, we can observe a similar behavior with the mixed and periodic patterns yielding the top performances.

Table 1 Evaluation on the 5-class and 10-class test sets of SegSemOutex

Metric	Class	Pixel pattern	Experiment	
			5-class	10-class
PA	All	N/A	0.9929	0.9940
mPA	All	N/A	0.9929	0.9939
mIoU	All	N/A	0.9860	0.9880
IoU_1	barleyrice002	Random	0.9845	0.9867
IoU_2	canvas005	Mixed	**0.9888**	0.9894
IoU_3	canvas022	Periodic	0.9886	0.9906
IoU_4	carpet005	Random	0.9856	0.9871
IoU_5	chips010	Random	*0.9824*	*0.9846*
IoU_6	paper004	Periodic	N/A	0.9905
IoU_7	pasta006	Random	N/A	0.9851
IoU_8	plastic005	Random	N/A	0.9847
IoU_9	seeds004	Random	N/A	0.9875
IoU_{10}	wallpaper006	Periodic	N/A	**0.9936**

Best and worst IoU_i shown in bold and italics, respectively

Figure 6 shows representative sample results of DeepLabv3+ on SegSegOutex 10-class, covering all 20 mosaic templates and all 10 material classes. The results look very similar to the ground truth, with correct labels for all regions. Most errors come from boundaries, as the transitions between regions sometimes share common features, such as similar intensities, especially in the case of textures with random pixel patterns like "barleyrice002" and "chips010". Examples 4 and 5 in the two-region group illustrate this, with minor errors on the boundaries between the central and encircling regions; example 5 is difficult even to the human eye.

4.2 Performance with respect to attributes

We evaluate DeepLabv3+'s performance on SemSegDTD with a focus on material-transcending attributes. We expect this to be a more complex problem than the classic texture segmentation problem centered on materials due to the different levels of abstraction of attributes and the larger intra-class variations. Table 2 shows the performance metrics on SemSegDTD for the 5- and the 10-class experiments in their standard and hard versions. For the standard version, DeepLabv3+ yields high accuracy in both the 5- and 10-class experiments, with only a slight drop in performance when increasing the number of classes from five to 10, i.e., 0.2 p.p. in terms of accuracy (PA, mPA) and 0.4 p.p. in terms of mIoU. IoU_i values are all very high (> 0.97), with the highest for "banded" and "woven" and the lowest for "chequered" and "zigzagged" in the 5- and 10-class experiments, respectively. Interestingly, compared to the material classes of SegSemOutex, the attribute classes of SegSemDTD (standard) do not pose any additional difficulty, with PA and mPA values in the 0.99 s and mIoU in the 0.98 s. Using again a two-sample t-test and considering a significance threshold of .05, for the 5-class experiment, four class pairs have no statistical difference in their performance: "banded-woven", "chequered-wrinkled", "chequered-fibrous" and "wrinkled-fibrous", with $p = .47$, .44, .46 and .98, respectively. For the 10-class experiment, out of 45 class pairs, 12 have no statistical difference, with the best-performing class ("woven") being the only

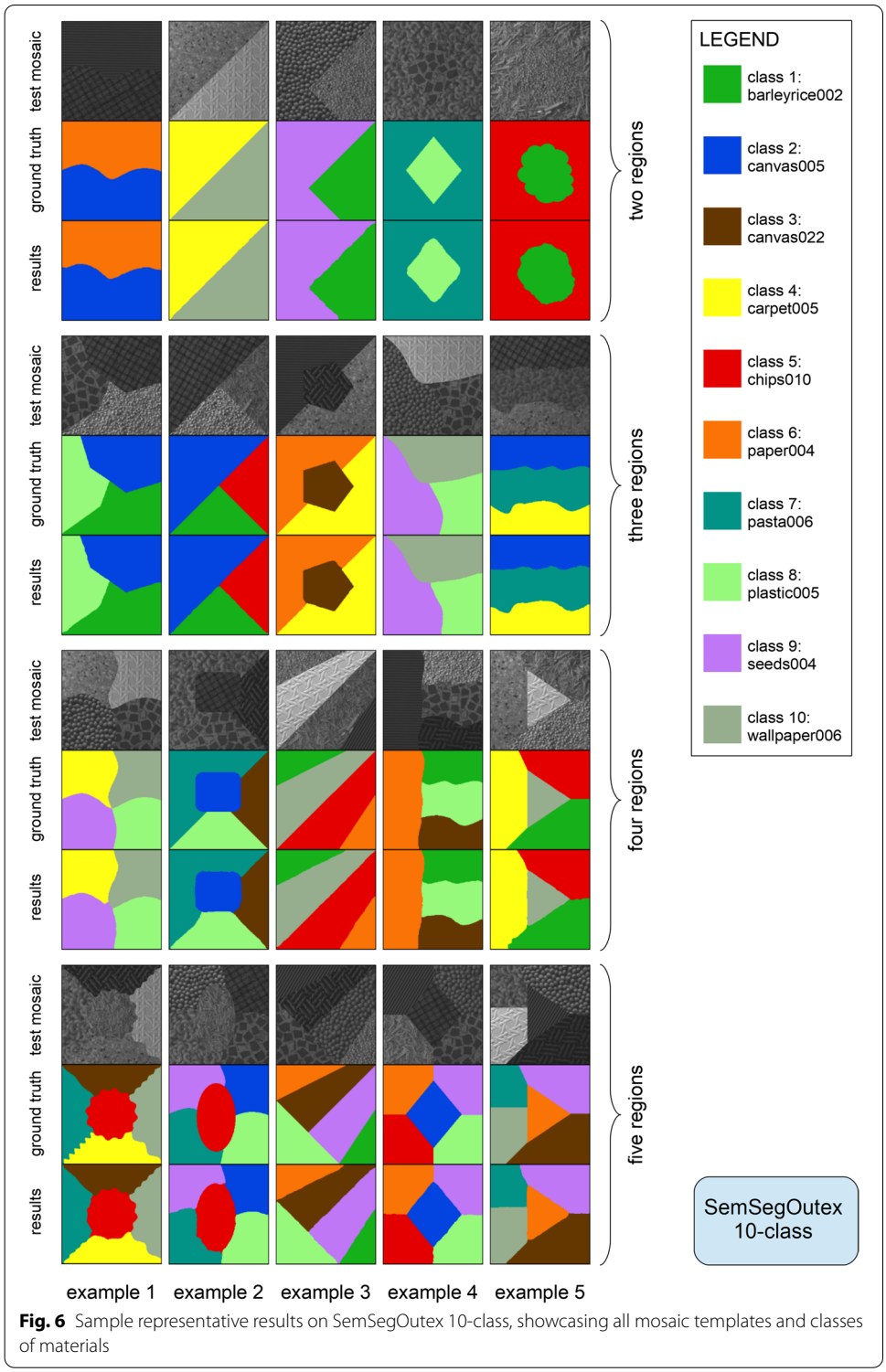

Fig. 6 Sample representative results on SemSegOutex 10-class, showcasing all mosaic templates and classes of materials

one to show a statistically significant difference with all other classes. Confusion matrices are not shown for the standard versions due to very low misclassifications among existing classes, from 0.000 to 0.002 for both the 5-class and 10-class experiments. For the 10-class experiment, some pixels are predicted as background (0.7% and 0.3% of

Table 2 Evaluation on the 5-class and 10-class standard and hard test sets of SegSemDTD

Metric	Class	Experiment			
		5-class	10-class	5-class (hard)	10-class (hard)
PA	All	0.9945	0.9924	0.9622	0.8228
mPA	All	0.9945	0.9923	0.9621	0.8244
mIoU	All	0.9891	0.9849	0.9281	0.7088
IoU_1	Banded	**0.9907**	0.9896	0.9450	0.6640
IoU_2	Chequered	*0.9878*	0.9872	**0.9654**	0.8735
IoU_3	Woven	0.9903	**0.9908**	0.9004	0.5831
IoU_4	Wrinkled	0.9883	0.9884	*0.8713*	0.7100
IoU_5	Fibrous	0.9884	0.9832	0.9583	0.7064
IoU_6	Cracked	N/A	0.9872	N/A	0.7827
IoU_7	Grid	N/A	0.9886	N/A	0.6275
IoU_8	Honeycombed	N/A	0.9761	N/A	*0.4594*
IoU_9	Polka-dotted	N/A	0.9852	N/A	**0.9350**
IoU_{10}	Zigzagged	N/A	*0.9732*	N/A	0.7462

Best and worst IoU_i shown in bold and italics, respectively

"zigzagged" and "honeycombed" pixels, respectively). Even though there are no actual background pixels in the training/test data, the network misclassifies some pixels as background due to the cross-entropy loss function, which utilizes a one-hot encoding setup (with a binary value for each possible class).

The hard version of SemSegDTD generates a noticeable drop in DeepLabv3+'s performance. Comparing 5-class hard with 5-class standard, we see drops of 3.2 p.p. for accuracy (PA, mPA) and 6.1 p.p. for mIoU. The drops are higher for the 10-class hard versus 10-class standard, with 17 p.p. and 28 p.p. for accuracy and mIoU, respectively. Increasing the number of classes from five to 10 for the hard set noticeably lowers the performance, which was not the case for the standard set nor SemSegOutex. This decrease can be explained in part by the compounded effects of the distortions and acquisition artifacts of the hard sets. IoU_i values vary substantially, from 0.8713 for "wrinkled" to 0.9654 for "chequered" (5-class) and from 0.4594 for "honeycombed" to 0.9350 for "polka-dotted" (10-class). The worst-performing class for the 5-class standard is also the best-performing class for the 5-class hard ("chequered"), as the drop from standard to hard is only 2.2 p.p. The largest drop from standard to hard is for "honeycombed" (52 p.p.), which can be explained by the substantial degradation in the pattern themselves (presence of honey) in the hard set. Confusion matrices for the 5- and 10-class (hard) experiments are given in Fig. 7. The largest misclassifications (5-class hard) are for "wrinkled" predicted as "woven" (0.062) and as background (0.031); other misclassifications are substantially lower. The largest misclassifications (10-class hard) happen for pixels predicted as background, particularly severe for "honeycombed" (0.342). Misclassifications among classes happen mostly for "woven" predicted as "grid" (0.074), "banded" predicted as "zigzagged" (0.071), "honeycombed" predicted as "woven" (0.067), "fibrous" predicted as "grid" (0.053), "honeycombed" predicted as "grid" (0.047), and "wrinkled" predicted as "fibrous" (0.047). DeepLabv3+ appears to view regular structures from different attributes as being alike and attributes related to random textures as alike.

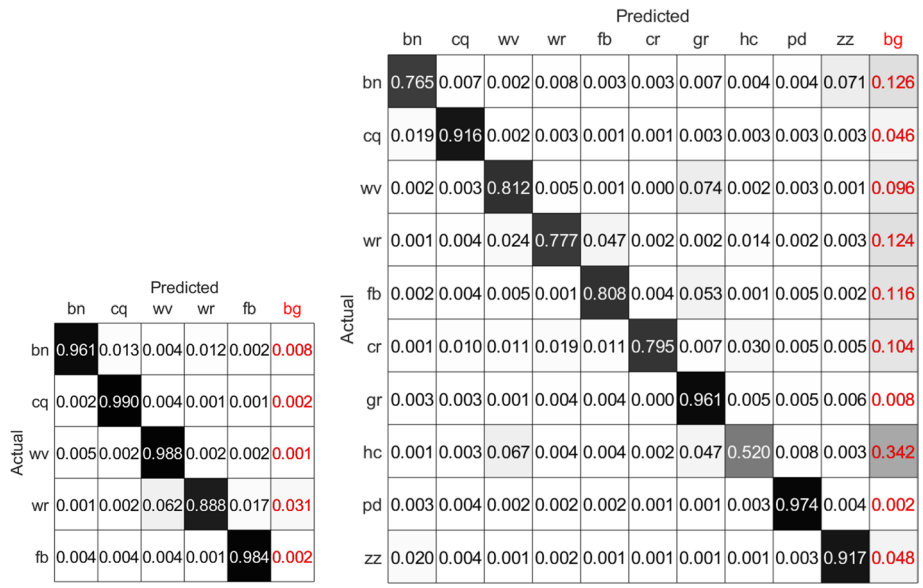

Fig. 7 Confusion matrices for the 5-class (left) and 10-class (right) test sets of SegSemDTD in their hard version. There is no actual background, but some pixels are predicted as background, hence the "background" column

Figures 8 and 9 show representative sample results of DeepLabv3+ on SemSegDTD 10-class for the standard and hard versions, respectively, covering all 20 mosaic templates and all 10 attribute classes. For the standard version (Fig. 8), the results are visually generally excellent, with three main types of errors when errors do occur: (1) centered on boundary regions (as was the case for SemSegOutex), such as polka dots adjacent to a "chequered" area (example 4 in the two-region group); (2) covering larger areas, e.g., parts of the "honeycombed" structures predicted as "fibrous" (examples 3 and 4 in the five-region group); (3) pixels predicted as background (example 2 in the two-region group). For the hard version (Fig. 9), those same types of errors appear more frequently and strongly, mainly due to angled views, degraded patterns, and larger intra-class variations. Although many textured mosaics are correctly segmented, we do see more errors at boundary areas (e.g., example 4 in the two-region group, which is also difficult for the human eye), over large areas (e.g., example 2 in the four-region group, where the misclassified "wrinkled" region shares similar randomness with the central "fibrous" region), and related to the background (e.g., example 5 in the two-region group). As per the confusion matrix (Fig. 7, right), most errors related to the background are for "honeycombed", especially when we can see through the patterns (example 4 in the three-region group, examples 2 and 5 in the five-region group).

4.3 Performance with respect to the number of regions present

Table 3 shows the influence of the number of regions present in a test mosaic image on DeepLabv3+'s performance. The last column shows the global trend as the number of regions increases. As expected, although mIoU values remain very high except for the

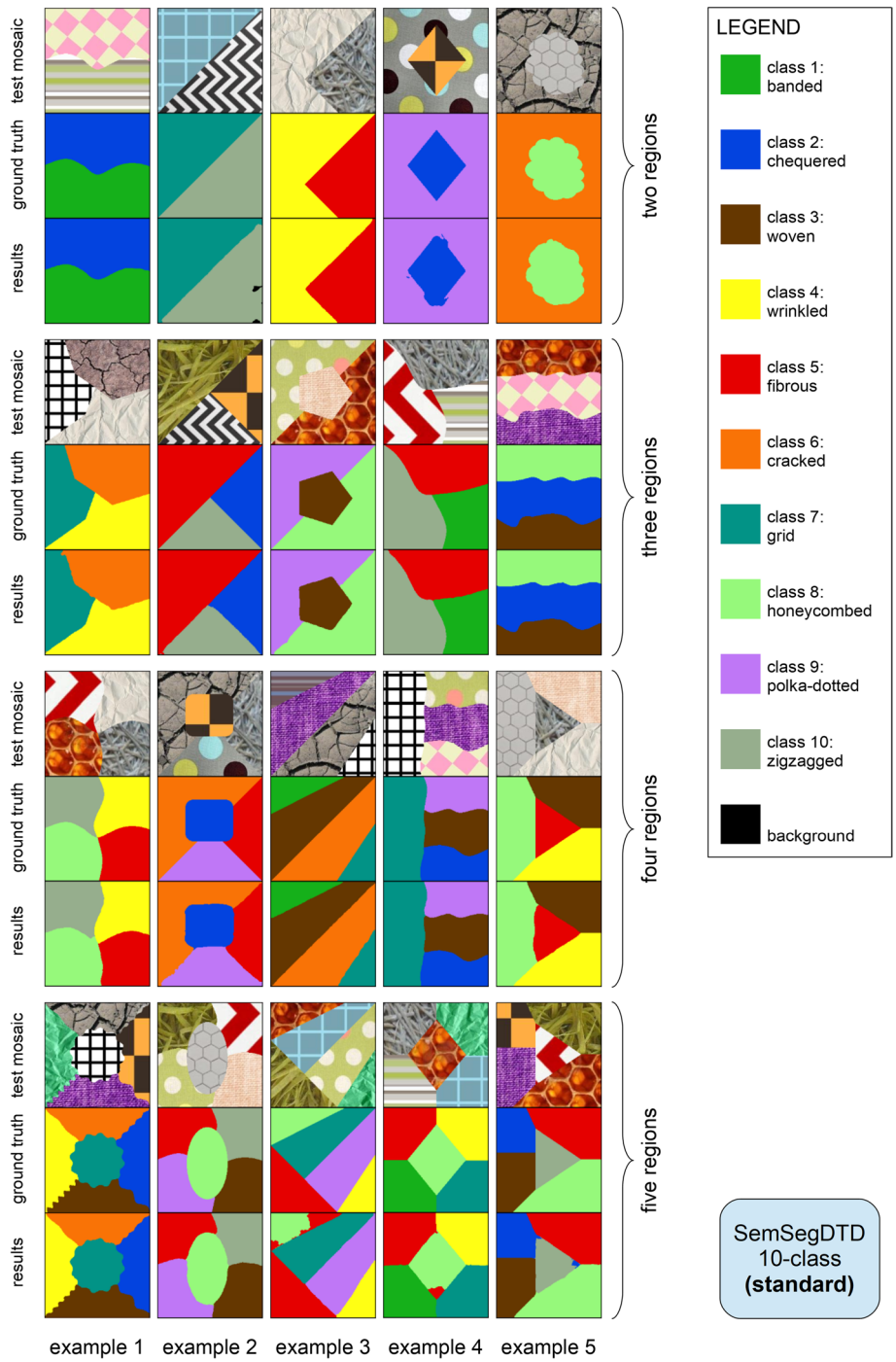

Fig. 8 Sample representative results on SemSegDTD 10-class (standard), showcasing all mosaic templates and classes of materials

hard version of SemSegDTD, an increase in the number of regions yields a monotonic decrease in performance, observed in all cases but SemSegDTD 10-class, which shows a non-monotonic decrease.

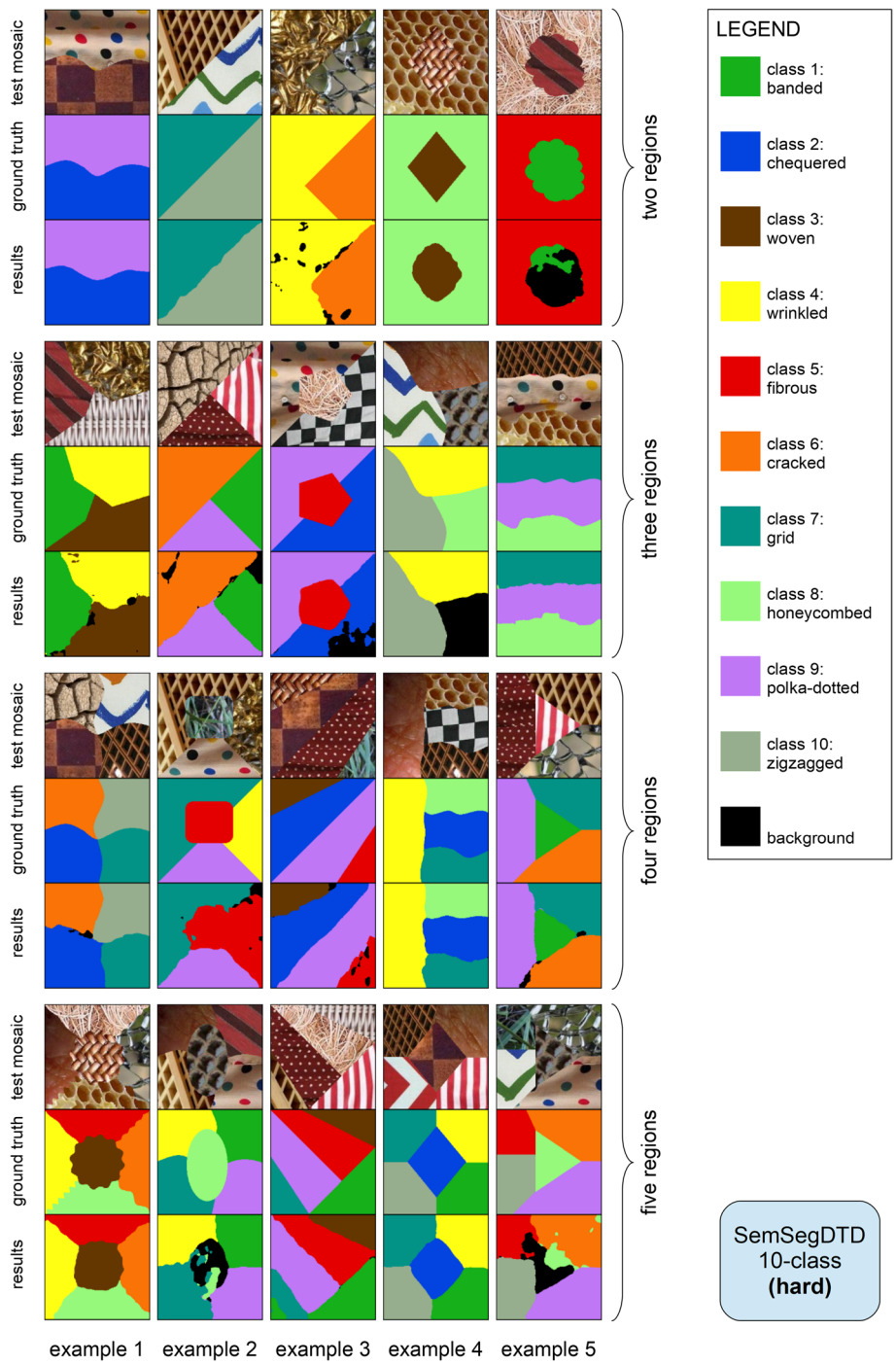

Fig. 9 Sample representative results on SemSegDTD 10-class (hard), showcasing all mosaic templates and classes of materials

Table 3 Influence of the number of regions (2, 3, 4, or 5) in test mosaics on the performance of DeepLabv3+ in terms of mIoU

Dataset	Experiment	# Regions				Global trend
		2	3	4	5	
SegSemOutex	5-class	0.9888	0.9870	0.9840	0.9801	⇓
	10-class	0.9911	0.9890	0.9860	0.9828	⇓
SegSemDTD	5-class	0.9912	0.9901	0.9872	0.9846	⇓
	10-class	0.9869	0.9889	0.9839	0.9796	⇓
	5-class (hard)	0.9568	0.9474	0.9237	0.9100	⇓
	10-class (hard)	0.7905	0.7747	0.7527	0.7471	⇓

Table 4 Evaluation on the 10-class test sets of SegSemOutex and SegSemOutex-Synth

Metric	Class	Case				
		N-N[1]	S-S-Ind[2]	S-S-All[3]	S-N-Ind[4]	S-N-All[5]
PA	All	0.9940	0.9955 ± 0.0002	**0.9976**	0.9938 ± 0.0006	0.9969
mPA	All	0.9939	0.9954 ± 0.0002	**0.9976**	0.9937 ± 0.0006	0.9969
mIoU	All	0.9880	0.9909 ± 0.0005	**0.9951**	0.9874 ± 0.0012	0.9938
IoU_1	barleyrice002	0.9867	0.9889 ± 0.0012	**0.9946**	0.9783 ± 0.0056	0.9923
IoU_2	canvas005	0.9894	0.9912 ± 0.0003	**0.9951**	0.9892 ± 0.0002	0.9939
IoU_3	canvas022	0.9906	0.9925 ± 0.0003	**0.9958**	0.9907 ± 0.0003	0.9951
IoU_4	carpet005	0.9871	0.9896 ± 0.0011	**0.9952**	0.9812 ± 0.0051	0.9935
IoU_5	chips010	0.9846	0.9896 ± 0.0006	**0.9942**	0.9857 ± 0.0006	0.9920
IoU_6	paper004	0.9905	0.9924 ± 0.0005	**0.9952**	0.9914 ± 0.0003	0.9949
IoU_7	pasta006	0.9851	0.9897 ± 0.0006	**0.9946**	0.9874 ± 0.0005	0.9930
IoU_8	plastic005	0.9847	0.9896 ± 0.0003	**0.9943**	0.9871 ± 0.0002	0.9929
IoU_9	seeds004	0.9875	0.9905 ± 0.0003	**0.9948**	0.9893 ± 0.0002	0.9938
IoU_{10}	wallpaper006	0.9936	0.9949 ± 0.0002	**0.9974**	0.9939 ± 0.0002	0.9968

Best metric values shown in bold

[1]: N-N: model trained and tested on natural (SemSegOutex) textures

[2]: S-S-Ind: average and standard deviation of five seeded models trained and tested on synthetic (SemSegOutex-Synth) textures, one seed per model

[3]: S-S-All: single model trained and tested on synthetic (SemSegOutex-Synth) textures, all seeds combined

[4]: S-N-Ind: average and standard deviation of five seeded models trained on synthetic (SemSegOutex-Synth) textures (one seed per model) and tested on natural (SemSegOutex) textures

[5]: S-N-All: single model trained on synthetic (SemSegOutex-Synth) textures (all seeds combined) and tested on natural (SemSegOutex) textures

4.4 Performance on natural versus synthetic textures

SemSegOutex-Synth allows us to evaluate DeepLabv3+'s performance for synthetically generated textures, which can then be compared with that for natural textures from SemSegOutex. We focus the discussion on the more complex 10-class experiment, but similar results are obtained for the 5-class experiment. Table 4 shows the performance metrics for the 10-class experiment for five different cases:

1 SemSegOutex (as reported in Table 1), i.e., the model is trained and tested on natural textures (N-N).

2 SemSegOutex-Synth with five seeded models trained and tested on synthetic textures, one seed per model (S-S-Ind). Average and standard deviation figures are given.
3 SemSegOutex-Synth with one model trained and tested on synthetic textures, all seeds combined (S-S-All). This model is thus trained on five times the data.
4 SemSegOutex-Synth/SemSegOutex with five seeded models trained on synthetic textures, one seed per model, and tested on natural images (S-N-Ind). Average and standard deviation figures are given.
5 SemSegOutex-Synth/SemSegOutex with one model trained on synthetic textures, all seeds combined, and tested on natural textures (S-N-All). This model is thus trained on five times the data.

In all five cases, the models were trained using the hyperparameters specified in Sect. 4.4. DeepLabv3+ performs as well or slightly better on synthetically generated textures compared to natural textures: up to 0.5 p.p. when comparing S-S-Ind with N-N figures, and up to approximately 1 p.p. when comparing S-S-All with N-N figures, with the highest results for the S-S-All case which includes the largest quantity of training data. Interestingly, testing natural textures using models trained on synthetic textures (the seeded models in S-N-Ind) yield similar performances compared to the model trained on natural textures (N-N), and a slight increase in performance consistent across all classes when testing natural textures using the model trained on all synthetic data (S-N-All). If we consider the pixel pattern categories (see Table 1), the largest performance increases for the S-N-All case, compared to the N-N case, come from the classes with random pixel patterns, which does not apply to the seeded models (S-N-Ind). This seems to indicate that more synthetic data allow for a better capture of random variations.

4.5 Performance with respect to synthetic data versus standard data augmentation

The excellent results of DeepLabv3+ for synthetic data in Sect. 4.4 prompted us to explore the use of synthetic textures for training models evaluated on natural textures as an alternative means to standard data augmentation based on random image operations. Table 5 compares the performance metrics for the 10-class experiment for natural textures from the test set of SegSegOutex under various conditions: using models trained with and without standard data augmentation (see Sect. 3.2) on natural or synthetic textures. The synthetic models were trained on textures from all five seeds combined.

Results for the natural models are similar in both cases of data augmentation (N-N-DA) and no data augmentation (N-N-NoDA), with a slight advantage for no data augmentation. The best overall performance (PA, mPA, mIoU) is obtained for the synthetic model using data augmentation (S-N-DA); however, on a class-by-class basis (IoU_i), the advantage goes to the synthetic model without data augmentation (S-N-NoDA), with two outliers ("barleyrice002" and "carpet005"). Comparing the performance of the synthetic model without data augmentation (S-N-NoDA) with that of the natural model with data augmentation (N-N-DA) is the crucial test as to whether synthetic data can serve a similar purpose as standard data augmentation. Here, we see that synthetic data yield a slightly better performance than data augmentation overall and for most classes, with the same two outliers previously identified. This seems to support our theory that

Table 5 Performance comparison between standard data augmentation and synthetic data usage for the 10-class test set of SegSemOutex

Metric	Class	Case			
		N-N-NoDA[1]	N-N-DA[2]	S-N-NoDA[3]	S-N-DA[4]
PA	All	0.9950	0.9940	0.9948	**0.9969**
mPA	All	0.9950	0.9939	0.9948	**0.9969**
mIoU	All	0.9900	0.9888	0.9895	**0.9938**
IoU_1	barleyrice002	0.9886	0.9867	0.9692	**0.9923**
IoU_2	canvas005	0.9909	0.9894	**0.9940**	0.9939
IoU_3	canvas022	0.9927	0.9906	**0.9957**	0.9951
IoU_4	carpet005	0.9896	0.9871	0.9718	**0.9935**
IoU_5	chips010	0.9865	0.9846	**0.9924**	0.9920
IoU_6	paper004	0.9926	0.9905	**0.9953**	0.9949
IoU_7	pasta006	0.9877	0.9851	**0.9932**	0.9930
IoU_8	plastic005	0.9874	0.9847	**0.9933**	0.9929
IoU_9	seeds004	0.9894	0.9875	**0.9940**	0.9938
IoU_{10}	wallpaper006	0.9946	0.9936	**0.9968**	**0.9968**

Best metric values shown in bold

[1]: N-N-NoDA: model trained (no data augmentation) and tested on natural (SemSegOutex) textures

[2]: N-N-DA: model trained (with data augmentation, see Sect. 3.2) and tested on natural (SemSegOutex) textures—same as column N-N in Table 4

[3]: S-N-NoDA: single model trained (no data augmentation) on synthetic (SemSegOutex-Synth) textures (all seeds combined) and tested on natural (SemSegOutex) textures

[4]: S-N-DA: single model trained (with data augmentation, see Sect. 3.2) on synthetic (SemSegOutex-Synth) textures (all seeds combined) and tested on natural (SemSegOutex) textures—same as column S-N-All in Table 4

the transient nature of random data augmentations are inferior when fully representational permanent synthetic data are available. In the case of the two outliers, the texture synthesis method proposed by Brochard et al. [37] seems unable to generate sufficient representational examples of the similar random outlier textures, although additional tests would be necessary for any meaningful conclusion on that matter.

The results in Table 5 suggest that testing natural textures with a model trained on a larger number of synthetic textures (here five times the training data) is a viable option, offering a comparable performance (or better in eight out of 10 classes) compared to using standard random data augmentation. Combining both strategies of synthetic training textures and standard random data augmentation yields the best overall choice, i.e., with no particular class of interest.

4.6 Traditional versus deep learning: comparative analysis

While the core of our study is focused on the performance of the popular state-of-the-art DeepLabv3+ for textured mosaic segmentation, here we offer comparisons with a baseline method based on classic image processing techniques (LBPs [17], which we introduced in Sect. 2.1), as well as with another recent DL semantic segmentation network (SegNeXt [46]). We center the discussion on the more complex 10-class experiment for SemSegOutex and SemSegDTD (standard and hard).

For the classic LBP-based technique, we followed a strategy similar to [47], which performed pixel-level classification for robust texture image classification. Here, we also perform pixel-level classification but for the purpose of image segmentation. Given a

grayscale image, the classic LBP code of a pixel is computed by comparing it to the pixels in the (P, R) neighborhood, i.e., P neighbors circularly sampled around the pixel with a radius R. The standard rotation invariant uniform patterns mapping is used. For each pixel of each training and test image, we create a window centered on the current pixel of size up to $(2 * HS + 1) \times (2 * HS + 1)$, with HS the window half size. We then compute the LBP code histogram over the window, normalize the histogram and use it as feature vector for the pixel. The normalization is particularly handy as it allows for the processing of boundary pixels for which the window is not square. Similarly to [47], we select a number T of training pixels from each class in each training image with uniform sampling (in [47], each training image had only one class whereas here we have between two and five classes depending on the mosaic template that was used to generate the image). We could, of course, use all 224×224 training pixels from each image (i.e., roughly 160 million pixels), but that would unnecessarily increase the classification process complexity [47]. We use $T = 10$ and experiment with all combinations of typical (P, R) neighborhoods and HS values, i.e., $(P, R) = \{(8, 1), (16, 2), (24, 3)\}$ and $HS = \{12, 24, 36, 48\}$. Pixel-level classification based on the feature vectors can be done via various traditional machine learning classifiers. Since we do not know beforehand which classifier will perform best, we train five different types of classifiers: (1) decision trees [48], (2) naive Bayes (see [49]), (3) SVMs [50], (4) k-nearest neighbors (KNNs) [51], and (5) ensembles (see [52]). Using Matlab's *Classification Learner* application, we thus train several variations of each type of classifier for each (P, R) and HS combination, and use the validation accuracy with fivefold cross-validation to retain the best classifier. We report on the combination yielding the best results, which varies with the dataset.

SegNeXt [46] is a recent (and thus less investigated and not as widely popular as DeepLabv3+—yet) convolutional network architecture for semantic segmentation. Arguing that a successful semantic segmentation model should have a strong backbone network as encoder, multi-scale information interaction, spatial attention, and low computational complexity, the authors designed SegNeXt to have spatial attention via multi-scale convolutional features in the encoder part and a decomposition-based Hamburger module [53] for global information extraction in the decoder part. We used the official PyTorch implementation of SegNeXt [54], the same data augmentation pipeline that we used for DeepLabv3+ (see Sect. 3.2), and the training parameters that the authors used on the ADE20K dataset [55]; however, given the smaller size of our datasets, we only trained for 200 epochs. Multiple sizes of the SegNeXt architecture are available; during testing, we determined that the "base" model produced the best results.

Table 6 compares the performance metrics on the test sets of SemSegOutex, SemSegDTD, and SemSegDTD (hard) 10-class experiments. As expected, the classic method (LBP descriptor combined with a traditional classifier) yields the lowest performance across all experiments and classes. It performs best for the traditional, material-related classes of texture (SemSegOutex) and has difficulties coping with the uncontrolled nature of SemSegDTD. Part of the explanation is the scale of the textured patterns in SemSegDTD, which has considerable intra-class variation. In some cases, the degradation in the hard version of SemSegDTD, compared to the standard version, causes a sizable drop in performance for the classic LBP method, for instance a drop in IoU_i of 68 p.p. for "banded" and of 53 p.p. for "woven". Interestingly, some classes see an increase in

Table 6 Comparison of LBP + traditional classifier, SegNeXt, and DeepLabv3+ performance on 10-class test sets of SemSegOutex and SegSemDTD (standard and hard)

Metric	SemSegOutex 10-class			SemSegDTD 10-class			SemSegDTD 10-class (hard)		
	LBP[1]	SegNeXt	DLV3+	LBP[2]	SegNeXt	DLV3+	LBP[3]	SegNeXt	DLV3+
PA	0.9416	**0.9959**	0.9940	0.5715	0.9881	**0.9924**	0.2929	**0.8706**	0.8228
mPA	0.8635	**0.9958**	0.9939	0.5359	0.9884	**0.9923**	0.2713	**0.8732**	0.8244
mIoU	0.8122	**0.9917**	0.9880	0.3669	0.9775	**0.9849**	0.1650	**0.7859**	0.7088
IoU_1	0.8863	**0.9915**	0.9867	0.7088	**0.9912**	0.9896	0.0218	**0.9071**	0.6640
IoU_2	0.8951	**0.9918**	0.9894	0.4681	0.9233	**0.9872**	0.1512	**0.9778**	0.8735
IoU_3	0.8859	**0.9922**	0.9906	0.5510	**0.9913**	0.9908	0.0162	**0.7952**	0.5831
IoU_4	0.8688	**0.9918**	0.9871	0.2715	**0.9921**	0.9884	0.1221	0.5266	**0.7100**
IoU_5	0.8064	**0.9916**	0.9846	0.3230	**0.9925**	0.9832	0.1209	**0.8389**	0.7064
IoU_6	0.9412	**0.9923**	0.9905	0.4730	**0.9924**	0.9872	0.1052	0.4482	**0.7827**
IoU_7	0.8879	**0.9914**	0.9851	0.3933	0.9178	**0.9886**	0.4165	**0.8677**	0.6275
IoU_8	0.8868	**0.9914**	0.9847	0.1028	**0.9921**	0.9761	0.2857	**0.5691**	0.4594
IoU_9	0.9273	**0.9917**	0.9875	0.1674	**0.9921**	0.9852	0.3407	**0.9890**	0.9350
IoU_{10}	0.9005	0.9915	**0.9936**	0.5297	**0.9907**	0.9732	0.1875	**0.9396**	0.7462

Best metric values shown in bold. Figures for DLV3+ are repeated from Tables 1 and 2

[1] Best LPB descriptor: $(P, R) = (24, 3)$, $HS = 24$; best classifier: medium Gaussian SVM

[2] Best LPB descriptor: $(P, R) = (24, 3)$, $HS = 36$; best classifier: cubic SVM

[3] Best LPB descriptor: $(P, R) = (24, 3)$, $HS = 48$; best classifier: weighted KNN

performance in the hard version compared to the standard one: "grid", "honeycombed", and "polka-dotted". Regarding the DL-based methods, there is no clear winner that yields the best performance across all experiments and classes. SegNeXt outperforms DeepLabv3+ for the overall metrics on SemSegOutex and SemSegDTD (hard), whereas DeepLabv3+ performs best for the overall metrics on SemSegDTD. Looking at the class-wise metrics, SegNeXt's performance tends to surpass that of DeepLabv3+ in most cases, with notable exceptions for "chequered" and "grid" (SemSegDTD) and "wrinkled" and "cracked" (SemSegDTD hard). The difference in performance for those few classes is such that for SemSegDTD, they allow DeepLabv3+ to come in first place overall. While SegNeXt appears to be the better choice for traditional material-based textured mosaics obtained in controlled conditions (SemSegOutex), the choice between the two DL-based methods is not so clear for attribute-based textured mosaics obtained in uncontrolled conditions (SemSegDTD), even if DeepLabv3+ is about four years older than SegNeXt.

5 Conclusion

This paper studies textured mosaic segmentation with a semantic segmentation lens. We propose two texture segmentation datasets (SemSegOutex and SemSegDTD) suitable for training semantic segmentation networks, possessing critical features missing from existing texture segmentation datasets. Experiments are focused on the state-of-the-art semantic segmentation network DeepLabv3+, which allows us to analyze the segmentation performance for materials, visual attributes, image acquisition artifacts, etc.

SemSegOutex results showcase the excellent performance of DeepLabv3+, whatever the material type, and its scalability to the number of classes. Considering that scaling the number of classes from five to 10 did not affect the performance at all, it is reasonable to believe that DeepLabv3+ could handle a much larger number of

material types, and that the eventual degradation would be graceful. More experiments are needed to confirm this hypothesis. Minor errors are centered on transition regions that share common visual features. SemSegDTD results show that DeepLabv3+ performs extremely well with visual attributes when the source images present intra- and inter-class variations but minimal visual distortions and occlusions. The hard version of SemSegDTD, with larger variations, distortions and occlusions, yields lower performance metrics, with pixels classified as background being problematic for specific classes, such as "honeycombed". DeepLabv3+ performs as well or slightly better on synthetically generated textures (SemSegOutex-Synth); this also applies to natural textures tested on models trained on synthetic versions of the natural textures. Given that the performance is already very high on SemSegOutex, the increase in performance using larger quantities of synthetic textures is limited to less than 2 p.p. Nonetheless, our results point towards a beneficial use of synthetic data either as an alternative means or in addition to standard random data augmentation. Finally, DeepLabv3+ fares extremely well when compared with a classic method utilizing a texture descriptor along with a machine learning-based classifier, and reasonably well when compared with another recent DL method.

Future work will expand the datasets with more material types and attributes, extend experiments on texture synthesis for data augmentation purposes, address misclassifications related to the background, and experiment with feature fusion to yield more reliable segmentations at region boundaries.

Abbreviations

B-CNN	Bilinear convolutional neural network
bg	Background
bn	Banded
CNN	Convolutional neural network
cq	Chequered
cr	Cracked
CUReT	Columbia-Utrecht reflectance and texture database
CV	Computer vision
DL	Deep learning
DLV3+	DeepLabv3+
DTD	Describable textures dataset
fb	Fibrous
FC	Fully connected
FCN	Fully convolutional network
FV-CNN	Fisher vector convolutional neural network
GLCM	Gray-level co-occurrence matrix
GPU	Graphics processing unit
gr	Grid
hc	Honeycombed
HS	Window half size
IoU	Intersection over union
KNN	K-nearest neighbors
KTH-TIPS2	Kungliga Tekniska högskolan (Royal Institute of Technology) textures under varying illumination, pose and scale
LBP	Local binary pattern
mIoU	Mean intersection over union
mPA	Mean pixel accuracy
N	Normal distribution
P	Number of pixels in the neighborhood
PA	Pixel accuracy
pd	Polka-dotted
p.p.	Percentage point
R	Neighborhood radius
STL	Statistical texture learning
SVM	Support vector machine

T-CNN	Texture convolutional neural network
VOC	Visual object classes
wr	Wrinkled
wv	Woven
zz	Zigzagged

Acknowledgements
Not applicable.

Author contributions
All authors made contributions to the current work. MC reviewed the literature, conceived the study, created the datasets, conducted some of the experiments, analyzed and interpreted the data, and drafted the manuscript. AD conceived the study, designed and conducted the majority of the experiments, collected the data, and helped modify the manuscript. ABA conceived the study, participated in its coordination, and helped modify the manuscript. All authors read and approved the final manuscript.

Funding
There was no funding for the research reported.

Availability of data and materials
The datasets used and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Received: 24 June 2022 Accepted: 7 August 2023
Published online: 14 August 2023

References

1. R.M. Haralick, K. Shanmugam, I. Dinstein, Textural features for image classification. IEEE Trans. Syst. Man Cybern. **6**, 610–21 (1973)
2. O. Faust, U.R. Acharya, K.M. Meiburger, F. Molinari, J.E. Koh, C.H. Yeong, P. Kongmebhol, K.H. Ng, Comparative assessment of texture features for the identification of cancer in ultrasound images: a review. Biocybern. Biomed. Eng. **38**(2), 275–96 (2018)
3. M.-T. Pham, S. Lefèvre, F. Merciol, Attribute profiles on derived textural features for highly textured optical image classification. IEEE Geosci. Remote Sens. Lett. **15**(7), 1125–9 (2018)
4. M. Cote, A.B. Albu, Texture sparseness for pixel classification of business document images. Int. J. Doc. Anal. Recognit. (IJDAR) **17**(3), 257–73 (2014)
5. M. Mehri, P. Héroux, P. Gomez-Krämer, R. Mullot, Texture feature benchmarking and evaluation for historical document image analysis. Int. J. Doc. Anal. Recognit. (IJDAR) **20**(1), 1–35 (2017)
6. S. Mikes, M. Haindl, Texture segmentation benchmark. IEEE Trans. Pattern Anal. Mach. Intell. **44**(9), 5647–5663 (2021)
7. Z. Li, F. Liu, W. Yang, S. Peng, J. Zhou, A survey of convolutional neural networks: analysis, applications, and prospects. IEEE Trans. Neural Netw. Learning Syst. (2021)
8. O. Ronneberger, P. Fischer, T. Brox, U-Net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI), pp. 234–41 (2015). Springer
9. J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3431–40 (2015)
10. L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, A. Hartwig, Encoder-decoder with atrous separable convolution for semantic image segmentation. In: European Conference on Computer Vision (ECCV), pp. 801–18 (2018)
11. C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, N. Sang, Learning a discriminative feature network for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1857–66 (2018)
12. Z. Zhou, M.M. Rahman Siddiquee, N. Tajbakhsh, J. Liang, UNet++: A nested U-Net architecture for medical image segmentation. In: 4th International Workshop on Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support (DLMIA), pp. 3–11 (2018). Springer
13. E. Xie, W. Wang, Z. Yu, A. Anandkumar, J.M. Alvarez, P. Luo, Segformer: Simple and efficient design for semantic segmentation with transformers. In: Advances in Neural Information Processing Systems, vol. 34, pp. 12077–12090 (2021)
14. T. Ojala, T. Maenpaa, M. Pietikainen, J. Viertola, J. Kyllonen, S. Huovinen, Outex-new framework for empirical evaluation of texture analysis algorithms. In: International Conference on Pattern Recognition (ICPR), vol. 1, pp. 701–6 (2002). IEEE
15. M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, A. Vedaldi, Describing textures in the wild. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3606–13 (2014)
16. A. Humeau-Heurtier, Texture feature extraction methods: a survey. IEEE Access **7**, 8975–9000 (2019)

17. T. Ojala, M. Pietikäinen, D. Harwood, A comparative study of texture measures with classification based on featured distributions. Pattern Recogn. **29**(1), 51–9 (1996)
18. M. Clark, A.C. Bovik, W.S. Geisler, Texture segmentation using gabor modulation/demodulation. Pattern Recogn. Lett. **6**(4), 261–7 (1987)
19. J.G. Daugman, Complete discrete 2-d gabor transforms by neural networks for image analysis and compression. IEEE Trans. Acoust. Speech Signal Process. **36**(7), 1169–79 (1988)
20. M. Cote, A.B. Albu, Sparseness-based descriptors for texture segmentation. In: International Conference on Pattern Recognition (ICPR), pp. 1108–13 (2014). IEEE
21. J. Yuan, D. Wang, A.M. Cheriyadat, Factorization-based texture segmentation. IEEE Trans. Image Process. **24**(11), 3488–97 (2015)
22. T. Leung, J. Malik, Representing and recognizing the visual appearance of materials using three-dimensional textons. Int. J. Comput. Vision **43**(1), 29–44 (2001)
23. T.-Y. Lin, S. Maji, Visualizing and understanding deep texture representations. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2791–9 (2016)
24. V. Andrearczyk, P.F. Whelan, Using filter banks in convolutional neural networks for texture classification. Pattern Recogn. Lett. **84**, 63–9 (2016)
25. M. Cimpoi, S. Maji, A. Vedaldi, Deep filter banks for texture recognition and segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3828–36 (2015)
26. M. Cimpoi, S. Maji, I. Kokkinos, A. Vedaldi, Deep filter banks for texture recognition, description, and segmentation. Int. J. Comput. Vision **118**(1), 65–94 (2016)
27. L. Liu, P. Fieguth, X. Wang, M. Pietikäinen, D. Hu, Evaluation of LBP and deep texture descriptors with a new robustness benchmark. In: European Conference on Computer Vision (ECCV), pp. 69–86 (2016). Springer
28. T.-Y. Lin, A. RoyChowdhury, S. Maji, Bilinear convolutional neural networks for fine-grained visual recognition. IEEE Trans. Pattern Anal. Mach. Intell. **40**(6), 1309–22 (2017)
29. R. Bello-Cerezo, F. Bianconi, F. Di Maria, P. Napoletano, F. Smeraldi, Comparative evaluation of hand-crafted image descriptors vs. off-the-shelf CNN-based features for colour texture classification under ideal and realistic conditions. Appl. Sci. **9**(4), 738 (2019)
30. V. Andrearczyk, P.F. Whelan, Texture segmentation with fully convolutional networks. arXiv preprint arXiv:1703.05230 (2017)
31. C. Karabağ, J. Verhoeven, N.R. Miller, C.C. Reyes-Aldasoro, Texture segmentation: an objective comparison between five traditional algorithms and a deep-learning u-net architecture. Appl. Sci. **9**(18), 3900 (2019)
32. R. Yamada, H. Ide, N. Yudistira, T. Kurita, Texture segmentation using siamese network and hierarchical region merging. In: International Conference on Pattern Recognition (ICPR), pp. 2735–40 (2018). IEEE
33. L. Zhu, D. Ji, S. Zhu, W. Gan, W. Wu, J. Yan, Learning statistical texture for semantic segmentation. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 12537–46 (2021)
34. K.J. Dana, B. Van Ginneken, S.K. Nayar, J.J. Koenderink, Reflectance and texture of real-world surfaces. ACM Trans. Graphics (TOG) **18**(1), 1–34 (1999)
35. B. Caputo, E. Hayman, P. Mallikarjuna, Class-specific material categorisation. In: IEEE International Conference on Computer Vision (ICCV), vol. 2, pp. 1597–1604 (2005). IEEE
36. R. Gonzalez, R. Woods, *Digital Image Processing*, 4th edn. (Pearson Education Limited, Harlow, 2018)
37. A. Brochard, S. Zhang, S. Mallat, Generalized rectifier wavelet covariance models for texture synthesis. In: International Conference on Learning Representations (ICLR) (2022)
38. GitHub—abrochar/wavelet-texture-synthesis: Code for the paper: "Generalized Rectifier Wavelet Covariance Model For Texture Synthesis". https://github.com/abrochar/wavelet-texture-synthesis. Accessed: 2022-Jun-07
39. Y. Zhou, Z. Zhu, X. Bai, D. Lischinski, D. Cohen-Or, H. Huang, Non-stationary texture synthesis by adversarial expansion. arXiv preprint arXiv:1805.04487 (2018)
40. M. Conni, H. Deborah, P. Nussbaum, P. Green, Visual and data stationarity of texture images. J. Electron. Imaging **30**(4), 043001 (2021)
41. S. Asgari Taghanaki, K. Abhishek, J.P. Cohen, J. Cohen-Adad, G. Hamarneh, Deep semantic segmentation of natural and medical images: a review. Artif. Intell. Rev. **54**(1), 137–78 (2021)
42. GitHub—VainF/DeepLabV3Plus-PyTorch. https://github.com/VainF/DeepLabV3Plus-Pytorch. Accessed: 2023-Jun-21
43. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: a large-scale hierarchical image database. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 248–255 (2009). IEEE
44. A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, J. Garcia-Rodriguez, A survey on deep learning techniques for image and video semantic segmentation. Appl. Soft Comput. **70**, 41–65 (2018)
45. S. Hao, Y. Zhou, Y. Guo, A brief survey on semantic segmentation with deep learning. Neurocomputing **406**, 302–21 (2020)
46. M.-H. Guo, C.-Z. Lu, Q. Hou, Z.-N. Liu, M.-M. Cheng, S.-m. Hu, SegNeXt: Rethinking convolutional attention design for semantic segmentation. In: Oh, A.H., Agarwal, A., Belgrave, D., Cho, K. (eds.) Advances in Neural Information Processing Systems (2022)
47. M. Cote, A.B. Albu, Robust texture classification by aggregating pixel-based lbp statistics. IEEE Signal Process. Lett. **22**(11), 2102–2106 (2015)
48. L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, *Classification and Regression Trees* (Routledge, New York, 2017)
49. C.D. Manning, P. Raghavan, H. Schutze, *Introduction to Information Retrieval* (Cambridge University Press, New York, 2008)
50. C. Cortes, V. Vapnik, Support-vector networks. Mach. Learn. **20**(3), 273–297 (1995)
51. T. Cover, P. Hart, Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13**(1), 21–27 (1967)
52. L. Rokach, Ensemble-based classifiers. Artif. Intell. Rev. **33**(1), 1–39 (2010)

53. Z. Geng, M.-H. Guo, H. Chen, X. Li, K. Wei, Z. Lin, Is attention better than matrix decomposition? In: International Conference on Learning Representations (ICLR) (2021)
54. GitHub—Visual Attention Network/SegNeXt: Official Pytorch implementations. https://github.com/Visual-Attention-Network/SegNeXt. Accessed: 2023-Jun-21
55. B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, A. Torralba, Scene parsing through ADE20K dataset. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 633–641 (2017)

Publisher's Note