ResUNet-a: a deep learning framework for semantic segmentation of remotely sensed data

Foivos I. Diakogiannis[a,1], François Waldner[b], Peter Caccetta[a], Chen Wu[c]

[a]*Data61, CSIRO, Floreat WA*
[b]*CSIRO Agriculture & Food, St Lucia, QLD, Australia*
[c]*ICRAR, The University of Western Australia, Crawley, WA*

Abstract

Scene understanding of high resolution aerial images is of great importance for the task of automated monitoring in various remote sensing applications. Due to the large within-class and small between-class variance in pixel values of objects of interest, this remains a challenging task. In recent years, deep convolutional neural networks have started being used in remote sensing applications and demonstrate state of the art performance for pixel level classification of objects. Here we present a novel deep learning architecture, ResUNet-a , that combines ideas from various state of the art modules used in computer vision for semantic segmentation tasks. We analyse the performance of several flavours of the Generalized Dice loss for semantic segmentation, and we introduce a novel variant loss function for semantic segmentation of objects that has better convergence properties and behaves well even under the presence of highly imbalanced classes. The performance of our modeling framework is evaluated on the ISPRS 2D Potsdam dataset. Results show state-of-the-art performance with an average F1 score of 92.9% over all classes for our best model.

Keywords: convolutional neural network, loss function, architecture, data augmentation, very high spatial resolution

1. Introduction

Semantic labelling of very high resolution (VHR) remotely-sensed images, *i.e.*, the task of assigning a category to every pixel in an image, is of great interest for a wide range of urban applications including land-use planning, infrastructure management, as well as urban sprawl detection (Matikainen and Karila, 2011; Zhang and Seto, 2011; Lu et al., 2017; Goldblatt et al., 2018). Labelling tasks generally focus on extracting one specific category, *e.g.*, building, road, or certain vegetation types (Li et al., 2015; Cheng et al., 2017; Wen et al., 2017), or multiple classes all together (Paisitkriangkrai et al., 2016; Längkvist et al., 2016; Liu et al., 2018; Marmanis et al., 2018).

Extracting spatially consistent information in urban environments from remotely-sensed imagery remains particularly challenging for two main reasons. First, urban classes often display a high within-class variability and a low between-class variability. On the one hand, man-made objects of the same semantic class are often built in different materials and with different structures, leading to an incredible diversity of colors, sizes, shapes, and textures. On the other hand, semantically-different man-made objects can present similar characteristics, *e.g.*, cement rooftops, cement sidewalks, and cement roads. Therefore, objects with similar spectral signatures can belong to completely different classes. Second, the intricate three-dimensional structure of urban environments is favourable to interactions between these objects, *e.g.*, through occlusions and cast shadows.

Circumventing these issues requires going beyond the sole use of spectral information and including geometric elements of the urban class appearance such as pattern, shape, size, context, and orientation. Nonetheless, pixel-based classifications still fail to satisfy the accuracy requirements because they are affected by the salt-and-pepper effect and cannot fully exploit the rich information content of VHR data (Myint et al., 2011; Li and Shao, 2014). GEographic Object-Based Imagery Analysis (GEOBIA) is an alternative image processing approach that seeks to group pixels into meaningful objects based on specified parameters (Blaschke et al., 2014). Popular image segmentation algorithm in remote sensing include watershed segmentation (Vincent and Soille, 1991), multi-resolution segmentation (Baatz and Schäpe, 2000) and mean-shift segmentation (Comaniciu and Meer, 2002). In addition, GEOBIA also allows to compute additional attributes related to the texture, context, and shape of the objects, which can then be added to the classification feature set. However, there is no universally-accepted method to identify the segmentation parameters that provide optimal pixel grouping, which implies the GEOBIA is still highly interactive and includes subjective trial-and-error methods and arbitrary decisions. Furthermore, image segmentation might fail to simultaneously address the wide range of object sizes that one typically encounters in urban landscapes ranging from finely structure objects such as cars and trees to larger objects such as buildings. Another drawback is that GEOBIA relies on pre-selected features for which the maximum attainable accuracy is *a priori* unknown. While several methods have been devised to extract and select features, these methods are not themselves learned from the data, and are thus potentially sub-optimal.

In recent years, deep learning methods and Convolutional Neural Networks (CNNs) in particular (LeCun et al., 1989)

[1]foivos.diakogiannis@data61.csiro.au

have surpassed traditional methods in various computer vision tasks, such as object detection, semantic, and instance segmentation (see Rawat and Wang, 2017, for a comprehensive review). Some of the key advantages of CNN-based algorithms is that they provide end-to-end solutions, that require minimal feature engineering which offer greater generalization capabilities. They also perform object-based classification, *i.e.*, they take into account features that characterize entire image objects, thereby reducing the salt-and-pepper effect that affects conventional classifiers.

Our approach to annotate image pixels with class labels is object-based, that is, the algorithm extracts characteristic features from whole (or parts of) objects that exist in images such as cars, trees, or corners of buildings and assigns a vector of class probabilities to each pixel. In contrast, using standard classifiers such as random forests, the probability of each class per pixel is based on features inherent in the spectral signature only. Features based on spectral signatures contain less information than features based on objects. For example, looking at a car we understand not only it's spectral features (color) but also how these vary as well as the extent these occupy in an image. In addition, we understand that it is more probable a car to be surrounded by pixels belonging to a road, and less probable to be surrounded by pixels belonging to buildings. In the field of computer vision, there is a vast literature on various modules used in convolutional neural networks that make use of this idea of "per object classification". These modules, such as atrous convolutions (Chen et al., 2016) and pyramid pooling (He et al., 2014; Zhao et al., 2017), boost the algorithmic performance on semantic segmentation tasks. In addition, after the residual networks era (He et al., 2015) it is now possible to train deeper neural networks avoiding to a great extent the problem of vanishing (or exploding) gradients.

Here, we introduce a novel Fully Convolutional Network (FCN) for semantic segmentation, termed `ResUNet-a` . This network combines ideas distilled from computer vision applications of deep learning, and demonstrates competitive performance. In addition, we describe a modeling framework consisting of a new loss function that behaves well for semantic segmentation problems with class imbalance as well as for regression problems. In summary, the main contributions of this paper are the following:

1. A novel architecture for understanding and labeling very high resolution images. The architecture combines many state-of-the-art components that boost the performance of per-pixel classification. We present two variants of the basic architecture, a single task and a multi-task one.
2. We analyze the performance of various flavours of the Dice coefficient for semantic segmentation. Based on our findings, we introduce a variant of the Dice loss function that speeds up the convergence of semantic segmentation tasks and improves performance. Our results indicate that the new loss function behaves well even when there is a large class imbalance. This loss can also be used for continuous variables when the target domain of values is in the range [0,1].

In addition, we also present a data augmentation methodology, where the input is viewed in multiple scales during training by the algorithm, that improves performance and avoids overfitting. The performance of `ResUNet-a` was tested using the Potsdam data set made available through the ISPRS competition (ISPRS). Validation results show that `ResUNet-a` achieves state-of-the-art results.

This article is organized as follows. In section 2 we provide a short review of related work on the topic of semantic segmentation focused on the field of remote sensing. In section 3, we detail the model architecture and the modeling framework. Section 4 describes the data set we used for training our algorithm. In section 5 we provide an experimental analysis that justifies the design choices for our modeling framework. Finally, section 6 presents the performance evaluation of our algorithm and comparison with other published results. Readers are referred to sections Appendix A for a description of our software implementation and hardware configurations, and to section Appendix B for the full error maps on unseen test data.

2. Related Work

The task of semantic segmentation has attracted significant interest in the latest years, not only in the field of computer vision community but also in other disciplines (e.g. biomedical imaging, remote sensing) where automated annotation of images is an important process. In particular, specialized techniques have been developed over different disciplines, since there are task-specific peculiarities that the community of computer vision does not have to address (and vice versa).

Starting from the computer vision community, when first introduced, Fully Convolutional Networks (hereafter FCN) for semantic segmentation (Long et al., 2014), improved the state of the art by a significant margin (20% relative improvement over the state of the art on the PASCAL VOC (Everingham et al., 2010) 2011 and 2012 test sets). The authors replaced the last fully connected layers with convolutional layers. The original resolution was achieved with a combination of upsampling and skip connections. Additional improvements have been presented with the use of deeplab models (Chen et al., 2016, 2017), that first showcased the importance of atrous convolutions for the task of semantic segmentation. Their model uses also a conditioned random field as a post processing step in order to refine the final segmentation. A significant contribution in the field of computer vision came from the community of biomedical imaging and in particular, the U-Net architecture (Ronneberger et al., 2015) that introduced the encoder-decoder paradigm, for upsampling gradually from lower size features to the original image size. Currently, the state of the art on the computer vision datasets is considered to be mask-rcnn (He et al., 2017), that performs various tasks (object localization, semantic segmentation, instance segmentation, pose estimation etc). A key element of the success of this architecture is its multitasking nature.

There has been a quick uptake of the approach in the remote sensing community and various solutions based on deep learning have been presented recently (*e.g.*, Audebert et al.,

2018, 2017, 2016; Längkvist et al., 2016; Li et al., 2015; Li and Shao, 2014; Volpi and Tuia, 2017; Liu et al., 2018, 2017a,b; Pan et al., 2018a,b; Marmanis et al., 2018; Wen et al., 2017). A comprehensive review of deep learning applications in the field of remote sensing can be found in Gu et al. (2019). Lately, for FCNs, Liu et al. (2017a) introduced the Hourglass-shape network for semantic segmentation on VHR images, that included an encoder-decoder style network, utilizing inception like modules. Pan et al. (2018b) presented the Dense Pyramid Network. The authors, utilized a dense-unet like architecture, where they also incorporated group convolutions to process independently the Digital Surface Model from the true orthophoto, presenting an interesting data fusion approach. Liu et al. (2018) introduced the CASIA network, that consists of a deep encoder and self-cascaded convolutional units. The architecture achieved state of the art performance on the ISPRS Potsdam and Vaihingen data. This architecture departs from the traditional encoder-decoder prototype and can utilize pre-trained networks.

3. The `ResUNet-a` framework

In this section, we introduce the architecture of `ResUNet-a` in full detail (section 3.1), a novel loss function design to achieve faster convergence and higher performance (section 3.2), data augmentation methodology (section 3.3) as well as the methodology we followed on performing inference on large images (section 3.4). The training strategy and software implementation characteristics can be found in Appendix A.

3.1. Architecture

Our architecture combines the following set of modules encoded in our models:

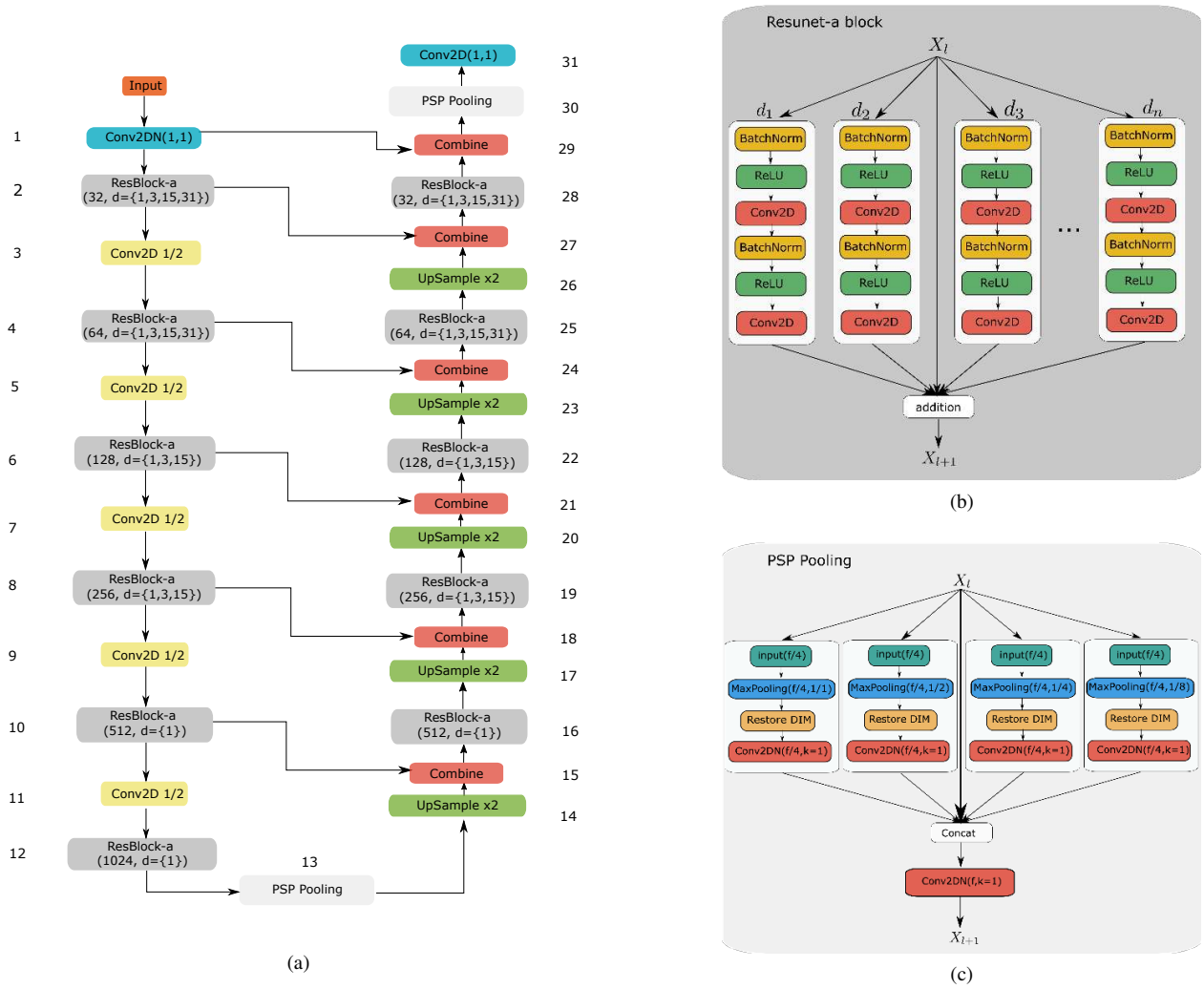
1. A UNet (Ronneberger et al., 2015) backbone architecture, *i.e.*, the encoder-decoder paradigm, is selected for smooth and gradual transitions from the image to the segmentation mask.
2. To achieve consistent training as the depth of the network increases, the building blocks of the UNet architecture were replaced with modified residual blocks of convolutional layers (He et al., 2016). Residual blocks remove to a great extent the problem of vanishing and exploding gradients that is present in deep architectures.
3. For better understanding across scales, multiple parallel atrous convolutions (Chen et al., 2016, 2017) with different dilation rates are employed within each residual building block. Although it is not completely clear why atrous convolutions perform well, the intuition behind their usage is that they increase the receptive field of each layer. The rationale of using these multiple-scale layers is to extract object features at various receptive field scales. The hope is that this will improve performance by identifying correlations between objects at different locations in the image.
4. In order to enhance the performance of the network by including background context information we use the pyramid scene parsing pooling (Zhao et al., 2017) layer. In shallow architectures, where the last layer of the encoder has a size no less than 16x16 pixels, we use this layer in two locations within the architecture: after the encoder part (*i.e.*, middle of the network) and the second last layer before the creation of the segmentation mask. For deeper architectures, we use this layer only close to the last output layer.
5. In addition to the standard architecture that has a single segmentation mask layer as output, we also present two models where we perform multi-task learning. The algorithm learns simultaneously four complementary tasks. The first is the segmentation mask. The second is the common boundary between the segmentation masks that is known to improve performance for semantic segmentation (Bertasius et al., 2015; Marmanis et al., 2018). The third is the distance transform[2] (Borgefors, 1986) of the segmentation mask. The fourth is the actual colored image, in HSV color space. That is, the identity transform of the content, but in a different color space.

We term our network `ResUNet-a` because it consists of residual building blocks with multiple atrous convolutions and a UNet backbone architecture. We present two basic architectures, `ResUNet-a` d6 and `ResUNet-a` d7, that differ in their depth, i.e. the total number of layers. In `ResUNet-a` d6 the encoder part consists of six `ResBlock-a` building blocks followed by a `PSPPooling` layer. In `ResUNet-a` d7 the encoder consists of seven `ResBlock-a` building blocks. For each of the d6 or d7 models, there are also three different output possibilities: a single task semantic segmentation layer, a multi-task layer (mtsk), and a conditioned multi-task output layer (cmtsk). The difference between the mtsk and cmtsk output layers is how the various complementary tasks (i.e. the boundary, the distance map, and the color) are used for the determination of the main target task, which is the semantic segmentation prediction. In the following we present in detail these models, starting from the basic `ResUNet-a` d6.

3.1.1. `ResUNet-a`

The `ResUNet-a` d6 network consists of stacked layers of modified residual building blocks (`ResBlock-a`), in an encoder-decoder style (UNet). The input is initially subjected to a convolution layer of kernel size (1, 1) to increase the number of features to the desired initial filter size. A (1, 1) convolution layer was used in order to avoid any information loss from the initial image by summarizing features across pixels with a larger kernel. Then follow the residual blocks. In each residual block (Fig. 1b), we used as many as three in parallel atrous convolutions in addition to the standard set of two convolutions of the residual network architecture, *i.e.*, there were up to four parallel branches of sets of two stacked convolutional layers. After the convolutions, the output is added to the initial input in the

[2]The result of the distance transform on a binary segmentation mask is a gray level image, that takes values in the range [0,1], where each pixel value corresponds to the distance to the closest boundary. In OpenCV this transform is encoded in `cv::distance_transform`.



(a) (b) (c)

Figure 1: Overview of the `ResUNet-a` d6 network. (a) The left (downward) branch is the encoder part of the architecture. The right (upward) branch is the decoder. The last convolutional layer has as many channels as there are distinct classes. (b) Building block of the `ResUNet-a` network. Each unit within the residual block has the same number of filters with all other units. Here $d_1, \ldots, d_n$ designate different dilation rates, (c) Pyramid scene parsing pooling layer. Pooling takes place in 1/1, 1/2, 1/4 and 1/8 portions of the original image.

spirit of residual building blocks. We decided to sum the various atrous branches (instead of concatenating them) because it is known that the residual blocks of two successive convolutional layers demonstrate constant condition number of the Hessian of the loss function, irrespective of the depth of the network Li et al. (2016). Therefore the summation scheme is easier to train (in comparison with the concatenation of features). In the encoder part of the network, the output of each of the residual blocks is downsampled with a convolution of kernel size of one and stride of two. At the end of both the encoder and the decoder part, there exists a `PSPooling` operator (Zhao et al., 2017). In the `PSPPooling` operator (Fig. 1c), the initial input is split in channel (feature) space in 4 equal partitions. Then we perform max pooling operation in successive splits of the input layer, in 1, 4, 16 and 64 partitions. Note that in the middle layer (Layer 13 has size: [batch size]$\times 1024 \times 8 \times 8$), the split of 64 corresponds to the actual total size of the input (so we have no additional gain with respect to max pooling from the last split). In Fig. 1a we present the full architecture of `ResUNet-a` (see also Table 1). In the decoder part, the upsampling is being done with the use of nearest neighbours interpolation followed by a normed convolution with a kernel size of one. By normed convolution, denoted with `Conv2DN`, we mean a set of a single 2D convolution followed by a `BatchNorm` layer. This approach for increasing the resolution of the convolution features was used in order to avoid the chequerboard artifact in the segmentation mask (Odena et al., 2016). The combination of layers from the encoder and decoder parts is being performed with the `Combine` layer (Table 2). This module concatenates the two inputs and subjects them to a normed convolution that brings the number of features to the desired size.

The `ResUNet-a` d7 model is deeper than the corresponding d6 model, by one resunet building block both in the encoder and decoder parts. We have tested two versions of this deeper architecture that differ in the way the pooling takes place in the middle of the network. In version 1 (hereafter d7v1) the `PSPPooling` Layer (`Layer` 13) is replaced with one additional building block, that is a standard resnet block (see Table 3 for

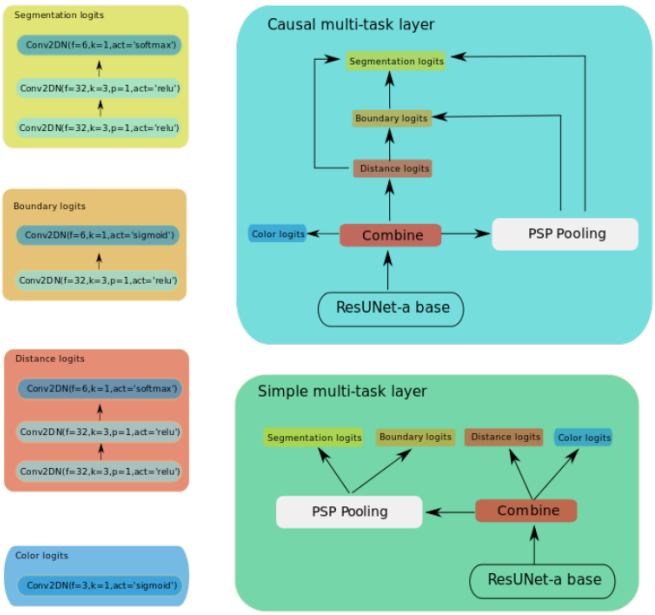


Figure 2: Multi-task output layer of the `ResUNet-a` network. Referring to the example of `ResUNet-a` d6, Layers 29, 30 and 31 are replaced with one of two variants of the multi-task layer. The first one, the conditioned multitask layer, combines the various intermediate products progressively so as the final segmentation layer to take a "decision" based on inference from previous results. The simple multi-task layer keeps the tasks independent.

details). There is, of course, a corresponding increase in the layers of the decoder part as well, by one additional residual building block. In more detail (Table 3), the `PSPPooling` layer in the middle of the network is replaced by a standard residual block at a lower resolution. The output of this layer is subjected to a `MaxPooling2D(kernel=2, stride=2)` operation the output of which is rescaled to its original size and then concatenated with the original input layer. This operation is followed by a standard convolution that brings the total number of features (i.e. the number of channels) to their original number before the concatenation. In version 2 (hereafter d7v2), again the `Layer 12` is replaced with a standard resnet block. However, now the `MaxPooling` operation following this layer is replaced with a smaller `PSPPooling` layer that has three parallel branches, performing pooling in 1/1, 1/2, 1/4 scales of the original filter (Fig. 1c). The reason for this is that the filters in the middle of the d7 network cannot sustain 4 parallel pooling operations due to their small size (therefore, we remove the 1/8 scale pooling), for an initial input image of size 256x256.

With regards to the model complexity, `ResUNet-a` d6 has ~ 52M trainable parameters for an initial filter size of 32. `ResUNet-a` d7 that has greater depth has ~ 160M parameters for the same initial filter size. The number of parameters remains almost identical for the case of the multi-task models as well.

3.1.2. Multitasking `ResUNet-a`

This version of `ResUNet-a` replaces the last layer (Layer 31) with a multitasking block (Fig. 2). The multiple tasks are complementary. These are (a) the prediction of the semantic segmentation mask, (b) the detection of the common

Table 1: Details of the `ResUNet-a` layers for the d6 model. Here `f` stands for the number of output channels (or features, the input number of features is deduced from the previous layers). `k` is the convolution kernel size, `d` is the dilation rate, and `s` the stride of the convolution operation. In all convolution operations we used appropriate zero padding to keep the dimensions of the produced feature maps equal to the input feature map (unless downsampling).

Layer #	Layer Type
1	`Conv2D(f=32, k=1, d=1, s=1)`
2	`ResBlock-a(f=32, k=3, d={1,3,15,31}, s=1)`
3	`Conv2D(f=64, k=1, d=1, s=2)`
4	`ResBlock-a(f=64, k=3, d={1,3,15,31}, s=1)`
5	`Conv2D(f=128, k=1, d=1, s=2)`
6	`ResBlock-a(f=128, k=3, d={1,3,15}, s=1)`
7	`Conv2D(f=256, k=1, d=1, s=2)`
8	`ResBlock-a(f=256, k=3, d={1,3,15}, s=1)`
9	`Conv2D(f=512, k=1, d=1, s=2)`
10	`ResBlock-a(f=512, k=3, d=1, s=1)`
11	`Conv2D(f=1024, k=1, d=1, s=2)`
12	`ResBlock-a(f=1024, k=3, d=1, s=1)`
13	`PSPPooling`
14	`UpSample (f=512)`
15	`Combine (f=512, Layers 14 & 10)`
16	`ResBlock-a(f=512, k=3, d=1, s=1)`
17	`UpSample (f=256)`
18	`Combine (f=256, Layers 17 & 8)`
19	`ResBlock-a(f=256, k=3, d=1, s=1)`
20	`UpSample (f=128)`
21	`Combine (f=128, Layers 20 & 6)`
22	`ResBlock-a(f=128, k=3, d=1, s=1)`
23	`UpSample (f=64)`
24	`Combine (f=64, Layers 23 & 4)`
25	`ResBlock-a(f=64, k=3, d=1, s=1)`
26	`UpSample (f=32)`
27	`Combine (f=32, Layers 26 & 2)`
28	`ResBlock-a(f=32, k=3, d=1, s=1)`
29	`Combine (f=32, Layers 28 & 1)`
30	`PSPPooling`
31	`Conv2D (f = NClasses, k=1, d=1, s=1)`
32	`Softmax(dim = 1)`

Table 2: Details of the `Combine(Input1,Input2)` layer.

Layer #	Layer Type
1	`Input1`
2	`ReLU(Input1)`
3	`Concat(Layer 2,Input2)`
3	`Conv2DN(k=1, d=1, s=1)`

Table 3: Details of the replacement of the middle `PSPPooling` layer (`Layer 13` from Table 1) for the `ResUNet-a` d7 model.

Layer #	Layer Type
input	`Layer 12`
A	`Conv2D(f=2048, k=1, d=1, s=2)(input)`
B	`ResBlock-a(f=2048, k=3, d=1, s=1)(Layer A)`
C	`MaxPooling(kernel=2, stride=2)(Layer B)`
D	`UpSample(Layer C)`
E	`Concat(Layer D, Layer B)`
F	`Conv2D(f=2048,kernel=1)(Layer E)`

boundaries between classes, (c) the reconstruction of the distance map and (e) the reconstruction of the original image in HSV color space. Our choice of using a different color space than the original input was guided by the principle that: (a) we wanted to avoid the identity transform in order to exclude the algorithm recovering trivial solutions and (b) the HSV (or HSL) colorspace matches closely the human perception of color A. Vadivel (2005). It is important to note that these additional labels are derived using standard computer vision libraries from the initial image and segmentation mask, without the need for additional information (e.g. separately annotated boundaries). The idea here is that all these tasks are complementary and should help the target task that we are after. Indeed, the distance map provides information for the topological connectivity of the segmentation mask as well as the extent of the objects (for example if we have an image with a "car" (object class) on a "road" (another object class), then the ground truth of the mask of the "road" will have a hole exactly to the location of the pixels corresponding to the "car" object). The boundary helps in better understanding the extent of the segmentation mask. Finally, the colorspace transformation provides additional information for the correlation between color variations and object extent. It also helps to keep "alive" the information of the fine details of the original image to its full extent until the final output layer. The rationale here is similar with the idea behind the concatenation of higher order features (first layers) with lower order features that exist in the UNet backbone architecture: the encoder layers have finer details about the original image as closely as they are to the original input. Hence, the reason for concatenating them with the layers of the decoder is to keep the fine details necessary until the final layer of the network that is ultimately responsible for the creation of the segmentation mask. By demanding the network to be able to reconstruct the original image, we are making sure that all fine details are preserved[3] (an example of input image, ground truth and inference for all the tasks in the conditioned multitasking setting can be seen in Fig. 12).

We present two flavours of the algorithm whose main difference is how the various tasks are used for the target output that we are interested in. In the simple multi-task block (bottom right block of Fig 2), the four tasks are produced simultaneously and independently. That is, there is no direct usage of the three complementary tasks (boundary, distance, and color) in the construction of the target task that is the segmentation. The motivation here is that the different tasks will force the algorithm to identify new meaningful features that are correlated with the output we are interested in and can help in the performance of the algorithm for semantic segmentation. For the distance map, as well as the color reconstruction, we do not use the PSPPooling layer. This is because it tends to produce large squared areas with the same values (due to the pooling operation) and the depth of the convolution layers in the logits is not sufficient to diminish this.

The second version of the algorithm uses a conditioned inference methodology. That is, the network graph is constructed in such a way so as to take advantage of the inference of the previous layers (top right block of Fig 2). We first predict the distance map. The distance map is then concatenated with the output of the PSPPooling layer and is used to calculate the boundary logits. Then both the distance map and the prediction of the boundary are concatenated with the PSPPooling layer and the result is provided as input to the segmentation logits for the final prediction.

[3] However, the color reconstruction on its own does not guarantee that the network learns meaningful correlations between classes and colors.

3.2. Loss function

In this section, we introduce a new variant of the family of Dice loss functions for semantic segmentation and regression problems.

3.2.1. Introducing the Tanimoto loss with complement

When it comes to semantic segmentation tasks, there are various options for the loss function. The Dice coefficient (Dice; Sørensen, 1948), generalized for fuzzy binary vectors in a multiclass context (Milletari et al., 2016, see also Crum et al. 2006; Sudre et al. 2017), is a popular choice among practitioners. It has been shown that it can increase performance over the cross entropy loss (Novikov et al., 2017). The Dice coefficient can be generalized to continuous binary vectors in two ways: either by the summation of probabilities in the denominator or by the summation of their squared values. In the literature, there are at least three definitions which are equivalent (Crum et al., 2006; Milletari et al., 2016; Drozdzal et al., 2016; Sudre et al., 2017):

$$D_1(\mathbf{p},\mathbf{l}) = \frac{2\sum_i p_i l_i}{\sum_i p_i + \sum_i l_i} \tag{1}$$

$$D_2(\mathbf{p},\mathbf{l}) = \frac{2\sum_i p_i l_i}{\sum_i (p_i^2 + l_i^2)} \tag{2}$$

$$D_3(\mathbf{p},\mathbf{l}) = \frac{\sum_i p_i l_i}{\sum_i (p_i^2 + l_i^2) - \sum_i (p_i l_i)} \tag{3}$$

where $\mathbf{p} \equiv \{p_i\}$, $p_i \in [0,1]$ is a continuous variable, representing the vector of probabilities for the i-th pixel, and $\mathbf{l} \equiv \{l_i\}$ are the corresponding ground truth labels. For binary vectors, $l_i \in \{0,1\}$. In the following we will represent (where appropriate) for simplicity the set of vector coordinates, $\mathbf{p} \equiv \{p_i\}$, with their corresponding tensor index notation, i.e $\mathbf{p} \equiv \{p_i\} \rightarrow p_i$.

These three definitions are numerically equivalent, in the sense that they map the vectors (p_i, l_i) to the continuous domain $[0,1]$, i.e. $D(p_i, l_i) : \Re^2 \rightarrow [0,1]$. The gradients however, of these loss functions behave differently for gradient based optimization, *i.e.*, for deep learning applications, as demonstrated in Milletari et al. (2016). In the remainder of this paper, we call Dice loss, the loss function with the functional form with the summation of probabilities and labels in the denominator (Eq. 1). We also use the name Tanimoto for the D_3 loss function (Eq. 3) and designate it with the letter $T \equiv D_3$.

We found empirically that the loss functions containing squares in the denominator behave better in pointing to the ground truth

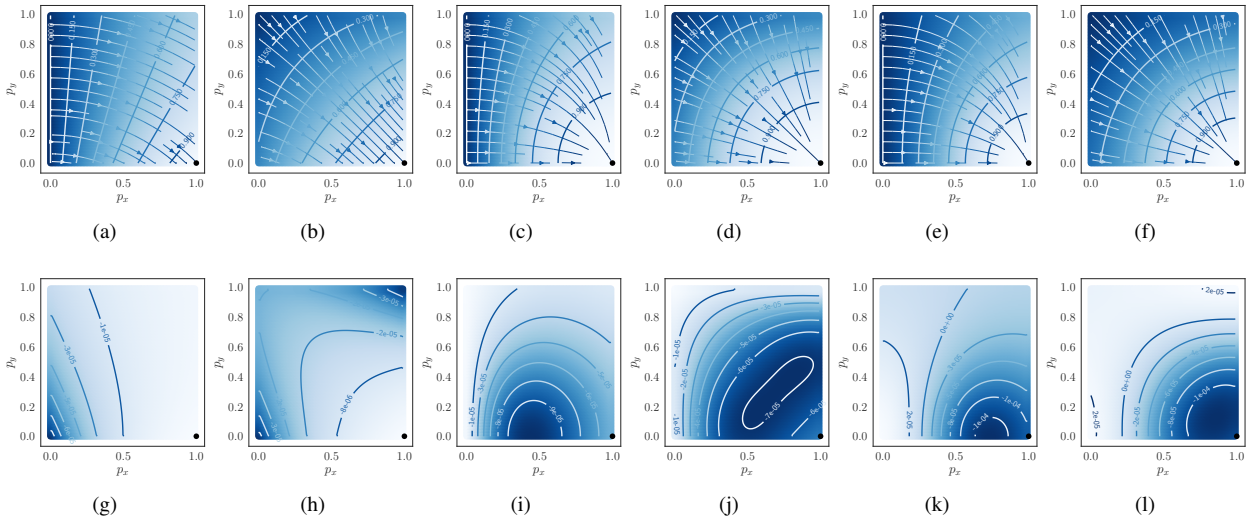


Figure 3: Contour plots of the gradient flow (top row) and Laplacian operator (bottom row) for the various versions of the Dice loss functions (Eq 1–3), D_i, as well as the functional forms with complements, $\tilde{D}_i$. The black dot corresponds to the ground truth value $(1, 0)$. From left to right we have a) D_1, b) $\tilde{D}_1$, c) D_2, d) $\tilde{D}_2$, e) D_3, f) $\tilde{D}_3$. The bottom panels, (g) – (l) are the corresponding Laplacian operators of these.

irrespective of the random initial configuration of weights. In addition, we found that we can achieve faster training convergence by complementing the loss with a dual form that measures the overlap area of the complement of the regions of interest. That is, if p_i measures the probability of the ith pixel to belong in class l_i, the complement loss is defined as $T(\mathbf{1}-p_i, \mathbf{1}-l_i)$, where the subtraction is performed element-wise, e.g. $\mathbf{1} - p_i = \{1 - p_1, 1 - p_2, \ldots, 1 - p_n\}$ etc. The intuition behind the usage of the complement in the loss function comes from the fact that the numerator of the Dice coefficient, $\sum_i p_i l_i$, can be viewed as an inner product between the probability vector, $\mathbf{p} = \{p_i\}$ and the ground truth label vector, $\mathbf{l} = \{l_i\}$. Then, the part of the probabilities vector, p_i, that corresponds to the elements of the label vector, l_i, that have zero entries, does not alter the value of the inner product[4]. We, therefore, propose that the best flow of gradients (hence faster training) is achieved using as a loss function the average of $T(p_i, l_i)$ with its complement, $T(\mathbf{1} - p_i, \mathbf{1} - l_i)$:

$$\tilde{T}(p_i, l_i) = \frac{T(p_i, l_i) + T(\mathbf{1} - p_i, \mathbf{1} - l_i)}{2}. \tag{4}$$

3.2.2. Experimental comparison with other Dice loss functions

In order to justify these choices, we present an example with a single 2D ground truth vector, ($l = (1, 0)$), and a vector of probabilities $p = (p_x, p_y) \in [0, 1]^2$. We consider the following six loss functions:

1. the Dice coefficient, $D_1(p_i, l_i)$ ((Eq. 1))
2. the Dice coefficient with its complement:

$$\tilde{D}_1(p_i, l_i) = (D_1(p_i, l_i) + D_1(\mathbf{1} - p_i, \mathbf{1} - l_i))/2$$

3. The Dice coefficient $D_2(p_i, l_i)$ (Eq. 2).
4. the Dice coefficient with its complement, $\tilde{D}_2$.
5. the Tanimoto coefficient, $T(p_i, l_i)$ (Eq. 3).
6. the Tanimoto coefficient with its complement, $\tilde{T}(p_i, l_i)$ (Eq. 4).

In Fig. 3 we present the gradient fields of the various loss functions (top panels), as well as the Laplacians of these (i.e. 2nd order derivatives, bottom panels). The ground truth is marked with a black dot. What is important in these plots is that for a random initialization of the weights for a neural network, the loss function will take a (random) value in the area within $[0, 1]^2$. The quality of the loss function then, as a suitable criterion for training deep learning models, is whether the gradients, from *every point of the area* in the plot, direct the solution towards the ground truth point. Intuitively we also expect that the behavior of the gradients is even better, if the local extrema of the loss on the ground truth, is also a local extremum of the Laplacian of the loss. As it is evident from Fig. 3 this is not the case for all loss functions.

In more detail, in Fig. 3a, we plot the gradient field of the Dice coefficient based loss, $D_1(p, l)$, in Fig. 3b the Dice loss with its complement, $\tilde{D}_1(p_i, l_i)$. From Fig. 3a, it is evident that for a random initialization (which is often the case in deep learning) at some point (p_x, p_y), the gradients of the loss with respect to p_x, p_y will not necessary direct to the ground truth point in $(1, 0)$. In this respect, the generalized Dice loss with complement (Fig. 3b) behaves better. However, it is apparent that the gradient flow lines do not pass through the ground truth point for all possible pairs of values (p_x, p_y). In Fig 3c, we present the second functional form of the dice loss, D_2 and in Fig. 3d the dice with complement $\tilde{D}_2$. In Fig 3e, we present the Tanimoto-based loss, i.e. D_3, and in Fig. 3f the Tanimoto with complement, $\tilde{T}$. In all these cases, the gradient flow lines pass through the ground truth point. However, the functional forms

[4] As a simple example, consider four dimensional vectors, say $p = (p_1, p_2, p_3, p_4)$ and $l = (1, 1, 0, 0)$. The value of the inner product term is $\mathbf{p}\cdot\mathbf{l} = p_1 + p_2$, and therefore the information contained in p_3 and p_4 entries is not apparent to the numerator of the loss. The complement inner product provides information for these terms: $(\mathbf{1}-\mathbf{p})\cdot(\mathbf{1}-\mathbf{l}) = (\mathbf{1}-\mathbf{p})\cdot(0, 0, 1, 1) = 2-(p_3 + p_4)$.

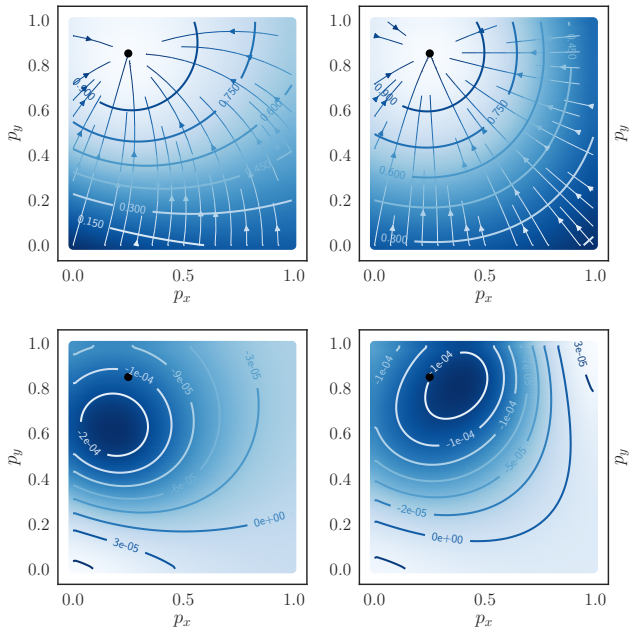


Figure 4: Top panels: gradient flow of the Tanimoto (top left) and Tanimoto with complement (top right) loss functions for a continuous target value. Bottom panels: corresponding Laplacian of the gradients. The "ground truth", $(0.25, 0.85)$ is represented with a black dot.

with complement, $\tilde{D}_2, \tilde{T}$, have the gradient lines flow straight towards the ground truth irrespective of the (random) initialization point. The Laplacians of these loss functions are presented in the panels Fig. 3g - Fig. 3l. It is clear that the extremum of the Laplacian operator is closer to the ground truth values only for the cases where we consider the loss functions with complement. In addition, the Tanimoto with complement has steeper gradients close to the ground truth. Interestingly, the Laplacian of the Tanimoto functional form (D_3) has extremum values closer to the ground truth point in comparison with the D_2 functional form.

In summary, the Tanimoto loss with complement has gradient flow lines that are straight lines (geodesics, i.e. they follow the shortest path) pointing to the ground truth from any random initialization point, and the second order derivative has extremum on the location of the ground truth. This demonstrates, according to our opinion, the superiority of the Tanimoto with complement as a loss function, among the family of loss functions based on the Dice coefficient, for training deep learning models.

3.2.3. Tanimoto with complement as a regression loss

It should be stressed, that if we restrict the output of the neural network in the range $[0, 1]$ (with the use of softmax or sigmoid activations) then the Tanimoto loss can be used to recover also continuous variables in the range $[0, 1]$. In Fig. 4 we present an example of this, for a ground truth vector of $l = (0.25, 0.85)$. In the top panels, we plot the gradient flow of the Tanimoto (left) and Tanimoto with complement (right) functions. In the bottom panels, we plot the corresponding

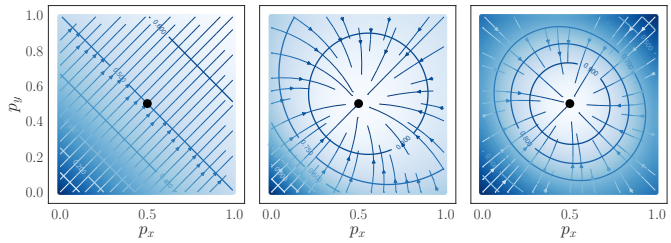


Figure 5: Gradient flow of the Dice family of losses for three different functional forms. From left to right: standard Dice loss (D_1, Eq. (1), Sudre et al. 2017, Dice loss with squares in the denominator (D_2, Eq. (2), Milletari et al. 2016), Tanimoto with complement (Eq, (4), this work). The "ground truth", $(0.5, 0.5)$ is represented with a black dot. It is clear that Tanimoto with complement has gradient flow (i.e. gradient magnitudes and direction) that is symmetric around the ground truth point, thus making it suitable for continuous regression problems. In contrast the Dice loss D_1 is not suitable for this usage, while D_2 has a clear assymetry that affects the gradients magnitude around the ground truth.

functions obtained after applying the Laplacian operator to the loss functions. This is an appealing property for the case of multi-task learning, where one of the complementary goals is a continuous loss function. The reason being that the gradients of these components will have similar magnitude scale and the training will be equally balanced to all complementary tasks. In contrast, when we use different functional form functions for different tasks in the optimization, we have to explicitely balance the gradients of the different components, with the extra cost of having to find the additional hyperparameter(s). For example, assuming we have two complementary tasks described by two different functional form functions, L_1 and L_2, then the total loss must be balanced with the usage of some (unknown) hyperparameter a that needs to be calculated: $L_{\text{total}} = L_1 + aL_2$.

In Fig. 5 we plot the gradient flow for three different members of the Dice family loss functional forms. From left to right we plot the standard Dice loss with summation in the denominator (D_1, Eq. (1), Sudre et al. 2017), the Dice loss with squares in the denominator (D_2, Eq. (2), Milletari et al. 2016) and Tanimoto with complement (Eq, (4) that we introduce in this work. It is clear that the Tanimoto with complement has the highest degree of symmetric gradients in both magnitude and direction around the ground truth point (for this example, the "ground truth" is $(p_x, p_y) = (0.5, 0.5)$). In addition, it also has steeper gradients as this is demonstrated from the distance of isocontours. The above help achieving faster convergence in problems with gradient descent optimization.

3.2.4. Generalization to multiclass imbalanced problems

Following the Dice loss modification of Sudre et al. (2017) for including weights per class, we generalize the Tanimoto loss for multi-class labelling of images:

$$T(p_{iJ}, l_{iJ}) = \frac{\sum_{J=1}^{N_{\text{class}}} w_J \sum_{i=1}^{N_{\text{pixels}}} p_{iJ} l_{iJ}}{\sum_{J=1}^{N_{\text{class}}} w_J \sum_{i=1}^{N_{\text{pixels}}} \left(p_{iJ}^2 + l_{iJ}^2 - p_{iJ} l_{iJ}\right)}. \quad (5)$$

Here w_J are the weights per class J, p_{iJ} is the probability of pixel i belonging to class J and l_{iJ} is the label of pixel i belonging to class J. Weights are derived following the inverse

"volume" weighting scheme per Crum et al. (2006):

$$w_J = V_J^{-2}, \tag{6}$$

where V_J is the total sum of true positives per class J, $V_J = \sum_{i=1}^{N_{\text{pixels}}} l_{iJ}$. In the following we will exclusively use the weighted Tanimoto (Eq. 5) with complement, $\tilde{T}(p_{iJ}, l_{iJ}) = (T(p_{iJ}, l_{iJ}) + T(\mathbf{1} - p_{iJ}, \mathbf{1} - l_{iJ}))/2$, and we will refer to it simply as the Tanimoto loss.

3.3. Data augmentation

To avoid overfitting, we relied on geometric data augmentation so that, in each iteration, the algorithm never sees the exact same set of images (i.e. the batch of images is always different). Each pair of image and ground truth mask are rotated at a random angle, with a random centre and zoomed in/out according to a random scale factor. The parts of the image that are left out from the frame after the transformation are filled in with reflect padding. This data augmentation methodology is particularly useful for aerial images of urban areas due to the high degree of reflect symmetry these areas have by design. We also used random reflections in x, y directions as an additional data augmentation routine.

The regularization approach is illustrated in Fig. 6 for a single datum of the ISPRS Potsdam data set (top row, FoV×4 dataset). From left to right, we show the false color infrared image of a 256x256 image patch, the corresponding digital elevation model, and the ground truth mask. In rows 2-4, we provide examples of the random transformations of the original image. By exposing the algorithm to different perspectives of the same objects scenery, we encode the prior knowledge that the algorithm should be able to identify the objects for all possible affine transformations. That is, we make the segmentation task invariant in affine transformations. This is quite similar to the functionality of the Spatial Transformer Network (Jaderberg et al., 2015), with the difference that this information is hard-coded in the data rather than the internal layers of the network. It should be noted that several authors report performance gains when they use inputs viewed at different scales, *e.g.*, Audebert et al. (2016) and Yang et al. (2018).

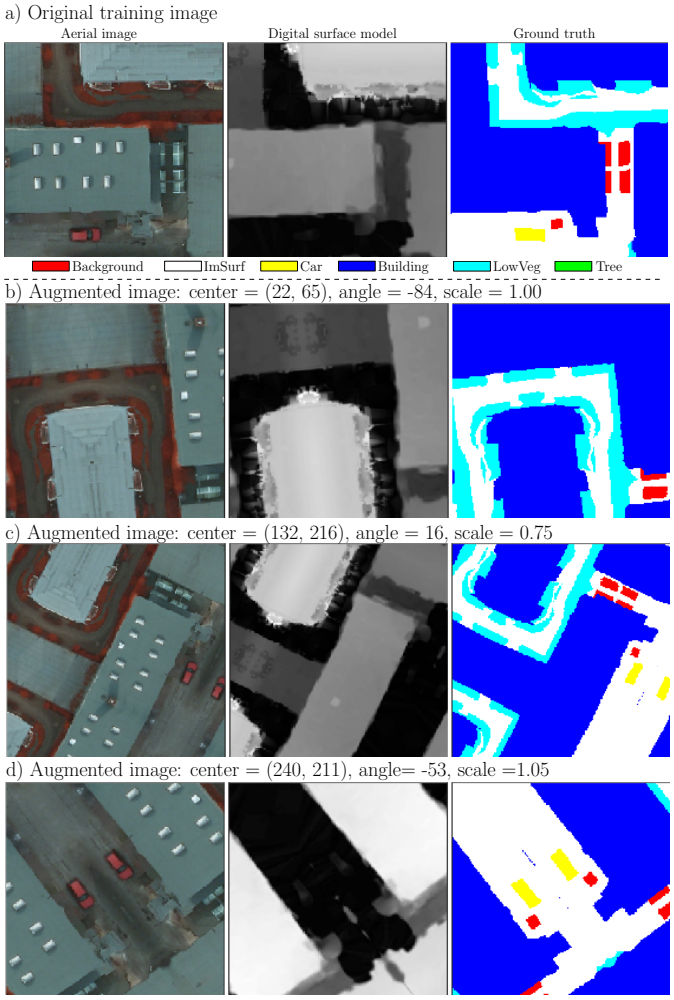


Figure 6: Example of data augmentation on image patches of size 256x256 (FoV×4 dataset). Top row: original image, subsequent rows: random rotations with respect to (random) center and at a random scale (zoom in/out). Reflect padding was used to fill the missing values of the image after the transformation.

3.4. Inference methodology

In this section, we detail the approach we followed for performing inference over large true orthophoto that exceeds the 256x256 size of the image patches we use during training.

As detailed in the introduction, FCNs such as `ResUNet-a` use contextual information to increase their performance. In practice, this means that in a single 256x256 window for inference, the pixels that are closer to the edges will not be classified as confidently as the ones close to the center because more contextual information is available to central pixels. Indeed, contextual information for the edge pixels is limited since there is no information outside the boundaries of the image patch. To further improve the performance of the algorithm and provide seamless segmentation masks, the inference is enhanced with multiple overlapping inference windows. This is like deciding on the classification result from multiple views (sliding windows) of the same objects. This type of approach is also used for large-scale land cover classification to combine classification in a seamless map (Lambert et al., 2016; Waldner et al., 2017).

Practically, we perform multiple overlapping windows passes over the whole tile and store the class probabilities for each pixel and each pass. The final class probability vector ($\tilde{p}_i(x, y)$) is obtained using the average of all the prediction views. The sliding window has size equal to the tile dimensions (256x256), however, we step through the whole image in strides of 256/4 = 64 pixels, in order to get multiple inference probabilities for each pixel. In order to account for the lack of information outside the tile boundaries, we pad each tile with reflect padding at a size equal to 256/2 = 128 pixels (Ronneberger et al., 2015).

4. Data and preprocessing

We sourced data from the ISPRS 2D Semantic Labelling Challenge and in particular the Potsdam data set (ISPRS). The data consist of a set of true orthophoto (TOP) extracted from a larger mosaic, and a Digital Surface Model (DSM). The TOP consists of the four spectral bands in the visible (VIS; red (R), green (G), and blue (G) and in the near infrared (NIR) and the ground sampling distance is 5 cm. The normalized DSM layer provides information on the height of each pixel as the ground elevation was subtracted. The four spectral bands (VISNIR) and the normalized DSM were stacked (VISNIR+DSM) to be used to train the semantic segmentation models. The labels consist of six classes, namely impervious surfaces, buildings, cars, low vegetation, trees, and background.

Unlike conventional pixel-based (e.g. random forests) or GEOBIA approaches, CNNs have the ability to "see" image objects in their contexts, which provides additional information to discriminate between classes. Thus, working with large image patches maximizes the competitive advantage of CNNs, however, limits to the maximum patch size are dictated by memory restrictions of the GPU hardware. We have created two versions of the training data. In the first version, we resampled the image tiles to half their original resolution and extracted image patches of size 256x256 pixels to train the network. The reduction of the original tile size to half was decided with the mindset that we can include more context information per image patch. This resulted in image patches with four times larger Field of View (hereafter FoV) for the same 256x256 patch size. We will refer to this dataset as (FoV×4) as it includes 4 times larger Field of View (area) in a single 256x256 image patch (in comparison with 256x256 image patches extracted directly from the original unscaled dataset). In the second version of the training data, we kept the full resolution tiles and again extracted image patches of size 256x256 pixels. We will refer to this dataset as FoV×1. The 256x256 image patch size was the maximum size that the memory capacity of our hardware configuration could handle (see Appendix A) so as to process a meaningfully large batch of datums. Each of the 256x256 patches used for training was extracted from a sliding window swiped over the whole tile at a stride, *i.e.*, step, of 128 pixels. This approach guarantees that all pixels at the edge of a patch become central pixels in subsequent patches. After slicing the original images, we split[5] the 256x256 patches into a training set, a validation set, and a test set with the following ratios: 0.8-0.1-0.1.

The purpose of the two distinct datasets is: the FoV×4 is useful in order to understand how much (if any) the increased context information improves the performance of the algorithm. It also allows us to perform more experiments much faster due to the decreased volume of data. The FoV×4 dataset is approximately ~50GB, with ~10k of pairs of images, masks. The FoV×1 has volume size of ~250GB, and ~40k pairs of images, masks. In addition, the FoV×4 is a useful benchmark on how the algorithm behaves with a smaller amount of data than the one provided. Finally, the FoV×1 version is used in order to compare the performance of our architecture with other published results.

[5] Making sure there is no overlap between the image patches of the training, validation and test sets.

4.1. Accuracy assessment

For each tile of the test set, we constructed the confusion matrix and extracted the several accuracy metrics such as the overall accuracy (OA), the precision, the recall, and the F1-score (F_1):

$$\text{OA} = \frac{TP + TN}{FP + FN} \tag{7}$$

$$\text{precision} = \frac{TP}{TP + FP} \tag{8}$$

$$\text{recall} = \frac{TP}{TP + FN} \tag{9}$$

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{10}$$

where TP, FP, FN, and TN are the is true positive, false positive, false negative and true negative classifications, respectively.

In addition, for the validation dataset (for which we have ground truth labels), we use the Matthews Correlation Coefficient (hereafter MCC, Matthews 1975):

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \tag{11}$$

5. Architecture and Tanimoto loss experimental analysis

In this section, we perform an experimental analysis of the `ResUNet-a` architecture as well as the performance of the Tanimoto with complement loss function.

5.1. Architecture ablation study

In this section, we design two experiments in order to evaluate the performance of the various modules that we use in the `ResUNet-a` architecture. In these experiments we used the FoV×1 dataset, as this is the dataset that will be the ultimate testbed of `ResUNet-a` performance against other modeling frameworks. Our metric for understanding the performance gains of the various models tested is the model complexity and training convergence: if model A has greater (or equal) number of parameters than model B, and model A converges faster to optimality than model B, then it is most likely that will also achieve the highest overall score.

In the first experiment we test the convergent properties of our architecture. In this, we are not interested in the final performance (after learning rate reduction and finetuning) which is a very time consuming operation, but how `ResUNet-a` behaves during training for the same fixed set of hyperparameters and epochs. We start by training a baseline model, a modified `ResUNet` (Zhang et al., 2017) where in order to keep the number of parameters identical with the case with atrous, we use the same `ResUNet-a` building blocks with dilation rate equal to 1

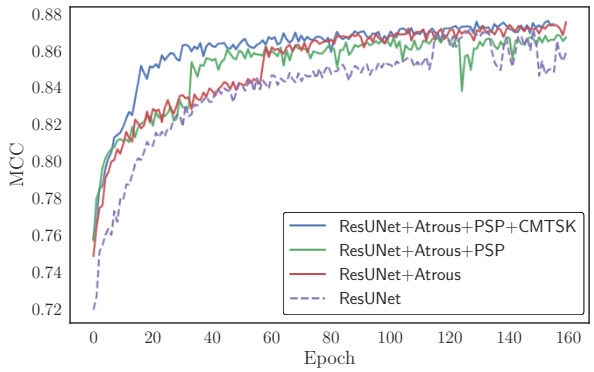


Figure 7: Convergence performance of the `ResUNet-a` architecture. Starting from a baseline wide-ResUNet, we add components keeping all training hyperparameters identical.

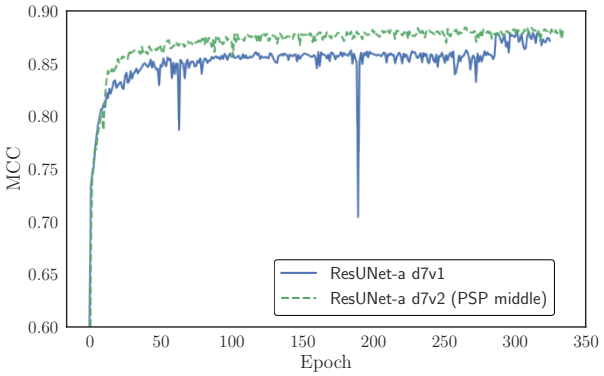


Figure 8: Convergence performance evaluation for the PSPPooling layer in the middle. We compare two architectures `ResUNet-a` d7v1 architecture without PSPPooling layer (blue solid line) and with PSPPooling layer in the middle (i.e. d7v2, dashed green line). It is clear that the insertion of the PSPPooling layer in the middle of the architecture boosts convergence performance of the network.

for all parallel residual blocks (i.e. there are no atrous convolutions). This is similar in philosophy with the wide residual networks (Zagoruyko and Komodakis, 2016), however, there are no dropout layers. Then, we modify this baseline by increasing the dilation rate, thus adding atrous convolutions (model: `ResUNet + Atrous`). It should be clear that the only difference between the models `ResUNet` and `ResUNet + Atrous` is that the latter has different dilation rates than the former, i.e. they have identical number of parameters. Then we add PSPPooling in both the middle and the end of the framework (model: `ResUNet + Atrous + PSP`), and finally we apply the conditioned multitasking, i.e. the full `ResUNet-a` model (model: `ResUNet + Atrous + PSP + CMTSK`). The differences in performance of the convergence rates is incremental with each module addition. This performance difference can be seen in Fig. 7 and is substantial. In Fig. 7 we plot the Matthews correlation coefficient (MCC) for all models. The MCC was calculated using the success rate over all classes. The baseline `ResUNet` requires approximately 120 epochs to achieve the same performance level that `ResUNet-a - cmtsk` achieves in epoch $\sim$ 40. The mere change from simple (`ResUNet`) to atrous convolutions (model `ResUNet+` Atrous) almost doubles the convergence rate. The inclusion of the PSP module (both middle and end) provides additional learning capacity, however, it also comes with training instability. This is fixed by adding the conditioned multitasking module in the final model. Clearly, each module addition: (a) increases the complexity of the model since it increases the total number of parameters and (b) it improves the convergence performance.

Next, we are interested in evaluating the importance of the PSPPooling layer. In our experiments we found this layer to be more important in the middle of the network than before the last output layers. For this purpose, we train two `ResUNet-a` d7 models, the d7v1 and d7v2, that are identical in all aspects except that the latter has a PSPPooling layer in the middle. Both models are trained with the same fixed set of hyperparameters (i.e. no learning rate reduction takes place during training). In Fig 8 we show the convergence evolution of these networks. It is clear that the model with the PSPPooling layer in the middle (i.e. v2) converges much faster to optimality, despite the fact that it has greater complexity (i.e. number of parameters) than the model d7v1.

In addition to the above, we have to note that when the network performs erroneous inference, due to the PSPPooling layer in the end, this may appear in the form of square blocks, indicating the influence of the pooling area in square subregions of the output. The last PSPPooling layer is in particular problematic when dealing with regression output problems. This is the reason why we did not use it in the evaluation of color and distance transform modules in the multitasking networks.

Comparing `ResUNet-a-mtsk` and `ResUNet-a-cmtsk` models (on the basis of the d7v1 feature extractor), we find that the latter demonstrates smaller variance in the values of the loss function (and in consequence, the performance metric) during training. In Fig. 9 we present an example of the comparative training evolution of the `ResUNet-a` d7v1 mtsk versus the `ResUNet-a` d7v1 cmtsk models. It is clear that the conditioned inference model demonstrates smaller variance, and that, despite the random fluctuations of the MCC coefficient, the median performance of the conditioned multitasking model is higher than the median performance of the simple multitasking model. This helps in stabilizing the gradient updates and results slightly better performance. We have also found that the inclusion of the identity reconstruction of the input image (in HSV colorspace) helps further to reduce the variance of the performance metric.

Our conclusion is that the greatest gain in using the conditioned multi-task model is in faster and consistent convergence to optimal values, as well as better segmentation of the boundaries (in comparison with the single output models).

5.2. Performance evaluation of the proposed loss function

In order to demonstrate the performance difference between the Dice loss as defined in Crum et al. (2006) and the Tanimoto loss, and the Tanimoto with complement (Eq. 4) we train three

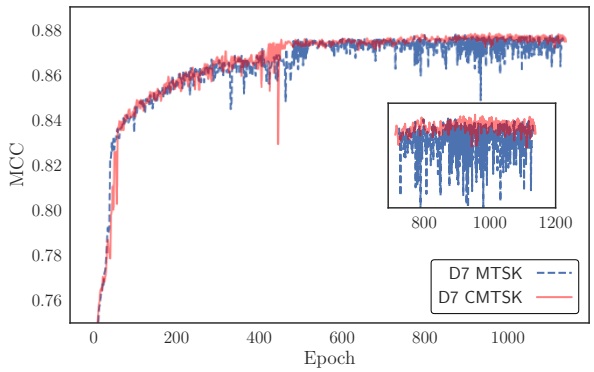


Figure 9: Training evolution of the conditioned vs the standard multi-task models for the `ResUNet-a` d7v1 family of models. The conditioned model (cmtsk) is represented with a red solid line, while the standard multi-task (mtsk) one with a dashed blue line. The mtsk demonstrates higher variance during training especially closer to the final convergence.

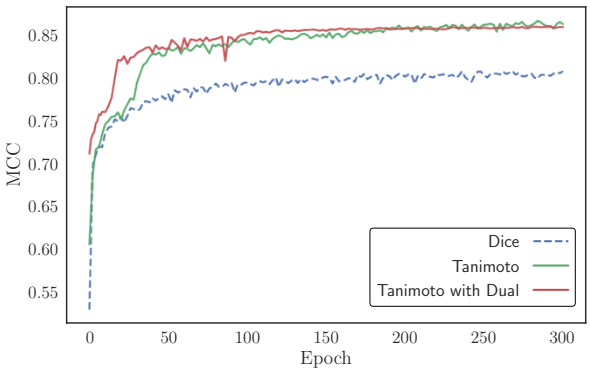


Figure 10: Training evolution of the same model using three different loss functions. The Tanimoto with complement loss (solid red line, this work), the Tanimoto (solid green line), and the Dice loss (dashed blue line, Sudre et al., 2017).

identical models with the same set of hyper-parameters. The weighting scheme is the same for all losses. In this experiment we used the FoV×4 dataset, in order to complete it shorter time. As the loss function cannot be responsible for overfitting (only the model capacity can lead to such behavior) our results persist also with the larger FoV×1 dataset. In Fig. 10 we plot the Matthews correlation coefficient (MCC). In this particular example, we are not interested in achieving maximum performance by reducing the learning rate and pushing the boundaries of what the model can achieve. We are only interested to compare the relative performance for the same training epochs between the different losses with an identical set of fixed hyper-parameters. It is evident that the Dice loss stagnates to lower values, while the Tanimoto loss with complement converges faster to an optimal value. The difference in performance is significant: the Tanimoto loss with complement achieves for the same number of epochs an MCC = 85.99, while the Dice loss stagnates at MCC = 80.72. The Tanimoto loss without complement (Eq. 5) gives a similar performance with the Tanimoto with complement, however, it converges relatively slower and demonstrates greater variance. In all experiments we performed, the Tanimoto with complement gave us the best performance.

6. Results and discussion

In this section, we present and discuss the performance of `ResUNet-a`. We also compare the efficiency of our model with results from architectures of other authors. We present results in both the FoV×4 and FoV×1 versions of the ISPRS Potsdam dataset. It should be noted that the ground truth masks of the test set were made publicly available on June 2018. Since then, the ISPRS 2D semantic label online test results are not being updated. The ground truth labels used to calculate the performance scores are the ones with the eroded boundaries.

6.1. *Design of experiments*

In Section 5 we tested the convergence properties of the various modules that `ResUNet-a` uses. In this section, our goal is to train the best convergent models until optimality and compare their performance. To this aim we document the following set of representative experiments: (a) `ResUNet-a` d6 vs `ResUNet-a` d6 cmtsk. The relative comparison of this will give us the performance boost between single task and conditioned multitasking models, keeping everything else the same. (b) `ResUNet-a` d7v1 mtsk vs `ResUNet-a` d7v1 conditioned mtsk. Here we are trying to see if conditioned multitasking improves performance over simple multitasking. In order to reduce computation time, we train these models with the FoV×4 dataset. Finally, we train the two best models, `ResUNet-a` d6 cmtsk and `ResUNet-a` d7v2 cmtsk on the FoV×1 dataset, in order to see if there are performance differences due to different Field of Views as well as compare our models with the results from other authors.

6.2. *Performance of `ResUNet-a` on the FoV×4 dataset*

`ResUNet-a` d6 (i.e. the model with no multi-task output) shows competitive performance in all classes (Table 4). The worst result (excluding the class "Background") for this model comes in the class "Trees", where it seems that `ResUNet-a` d6 systematically under segments the area close to their boundary. This is *partially* owed to the fact that we reduced the size of the original image, and fine details required for the detailed extent of trees cannot be identified by the algorithm. In fact, even for a human, the annotated boundaries of trees are not always clear (e.g. see Fig. 11). The `ResUNet-a` d6 cmtsk model provides a significant performance boost over the single task `ResUNet-a` d6 model, for the classes "Bulding", "LowVeg" and "Tree". In these classes it also outperforms the deeper models d7v1 (which, however, do not include the `PSPPooling` layer at the end of the encoder). This is due to the explicit requirement for the algorithm to reconstruct also the boundaries and the distance map and use them to further refine the segmentation mask. As a result, the algorithm gains a "better understanding" of the

Table 4: Potsdam comparison of results (F1 score and overall accuracy - OA) for the various `ResUNet-a` models trained on the FoV×4 and FoV×1 datasets. Highest score is marked with bold. The average F1-score was calculated using all classes except the "Background" class. The overall accuracy, was calculated including the "Background" category.

Methods	DataSet	ImSurface	Building	LowVeg	Tree	Car	Avg. F1	OA
`ResUNet-a` d6	(FoV×4)	92.7	97.1	86.4	85.8	95.8	91.6	90.1
`ResUNet-a` d6 cmtsk	(FoV×4)	91.4	**97.6**	87.4	88.1	95.3	91.9	90.1
[2pt] `ResUNet-a` d7v1 mtsk	(FoV×4)	92.9	97.2	86.8	87.4	96.0	92.1	90.6
`ResUNet-a` d7v1 cmtsk	(FoV×4)	92.9	97.2	87.0	87.5	95.8	92.1	90.7
`ResUNet-a` d6 cmtsk	(FoV×1)	93.0	97.2	87.5	88.4	96.1	92.4	91.0
`ResUNet-a` d7v2 cmtsk	(FoV×1)	**93.5**	97.2	**88.2**	**89.2**	**96.4**	**92.9**	**91.5**

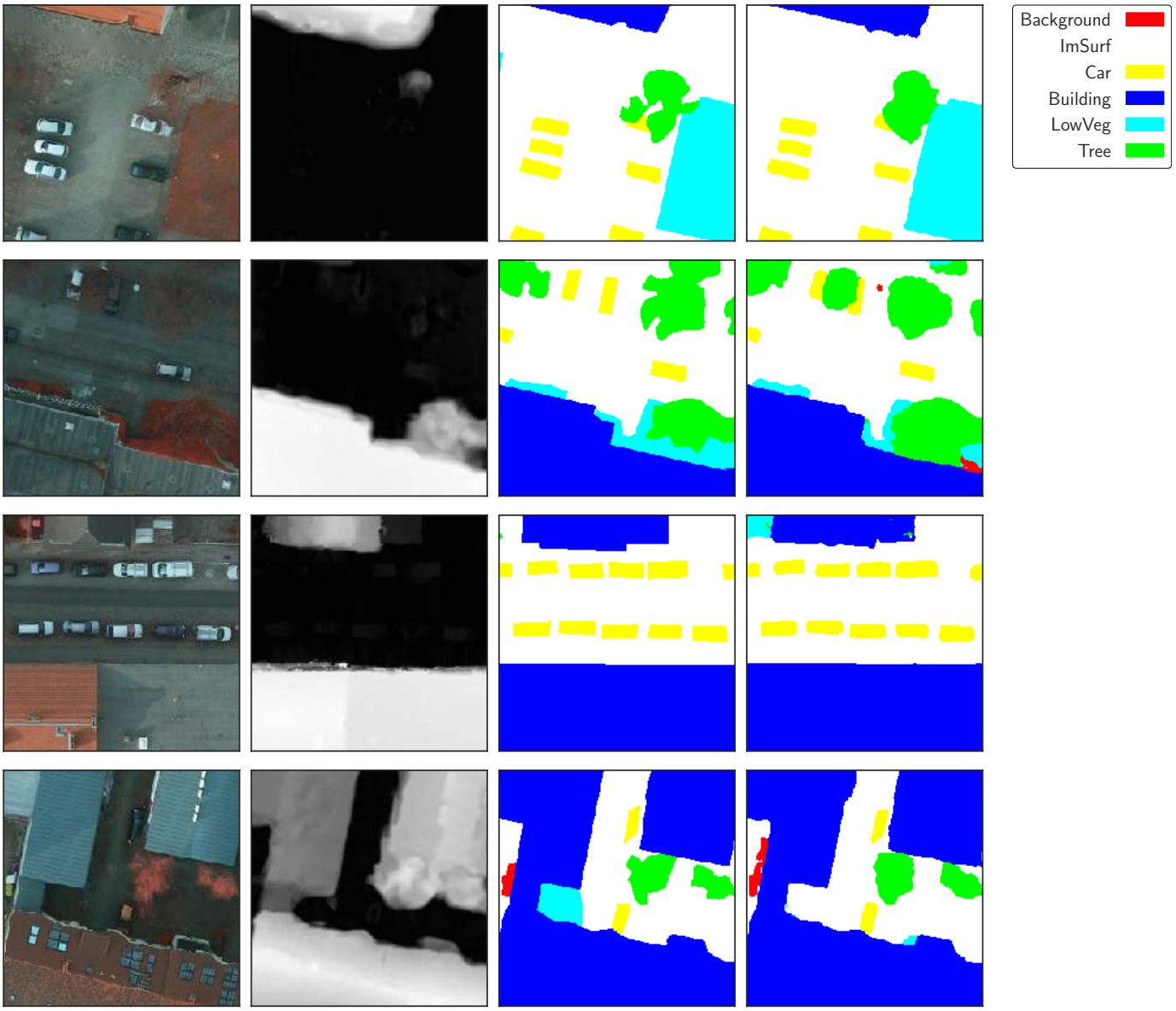


Figure 11: `ResUNet-a` d7v1 cmtsk inference on unseen test patches of size 256x256 (FoV×4). From left to right: rgb image, digital elevation map, ground truth, and prediction.

fine details of objects, even if in some cases it is difficult for humans to clearly identify their boundaries.

The `ResUNet-a` d7v1 cmtsk model demonstrates slightly increased performance over all of the tested models (Table 4, although differences are marginal for the FoV×4 dataset, and vary between classes). In addition, there are some annotation

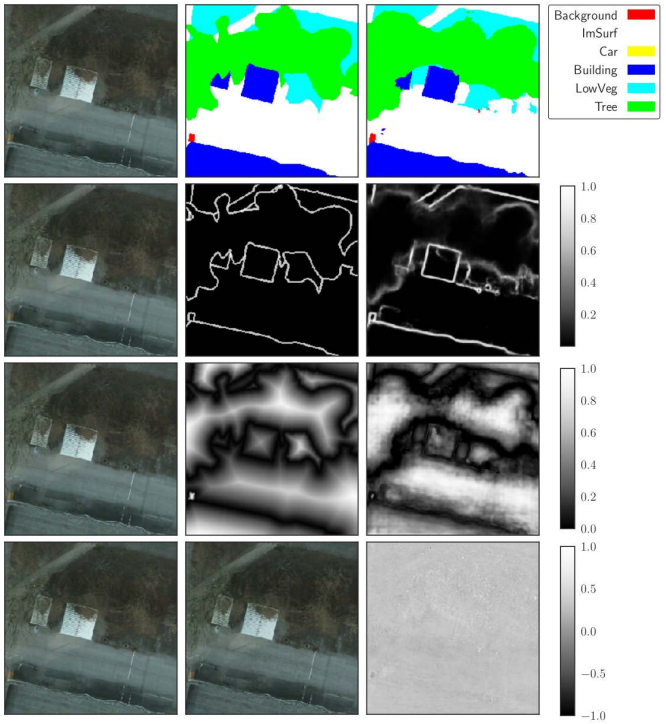


Figure 12: `ResUNet-a` d7v1 cmtsk all tasks inference on unseen test patches of size 256x256 for the FoV×4 dataset. From left to right, top row: input image, ground truth segmentation mask, predicted segmentation mask. Second row: input image, ground truth boundaries, predicted boundaries (confidence). Third row: input image, ground truth distance map, inferred distance map. Bottom row: input image, reconstructed image, difference between input and predicted image.

errors to the dataset that eventually prove to be an upper bound to the performance. In Fig. 11 we give an example of inference on 256x256 patches of images on unseen test data. In Fig. 12 we provide an example of the inference performed by `ResUNet-a` d7v1 cmtsk for all the predictive tasks (boundary, distance transform, segmentation, and identity reconstruction). In all rows, the left column corresponds to the same ground truth image. In the first row, from left to right: input image, ground truth segmentation mask, inference segmentation mask. Second row, middle and right: ground truth boundary and inference heat map of the confidence of the algorithm for characterizing pixels as boundaries. The more faint the boundaries are, the less confident is the algorithm for their characterization as boundaries. Third row, middle and right: distance map and inferred distance map. Last row, middle: reconstructed image in HSV space. Right image: average error over all channels between the original RGB image and the reconstructed one. The reconstruction is excellent suggesting that the Tanimoto loss can be used for identity mappings, whenever these are required (as a means of regularization or for Generative Adversarial Networks training (Goodfellow et al., 2014), e.g. Zhu et al. (2017)).

Finally, in Table 4, we provide a relative comparison between models trained in the FoV×4 and FoV×1 versions of the datasets. Clearly, there is a performance boost when using the higher resolution dataset (FoV×1) for the classes that require finer details. However, for the class "Building" the score is actually better with the wider Field of View (FoV×4, model d6 cmtsk) dataset.

6.3. Comparison with other modeling frameworks

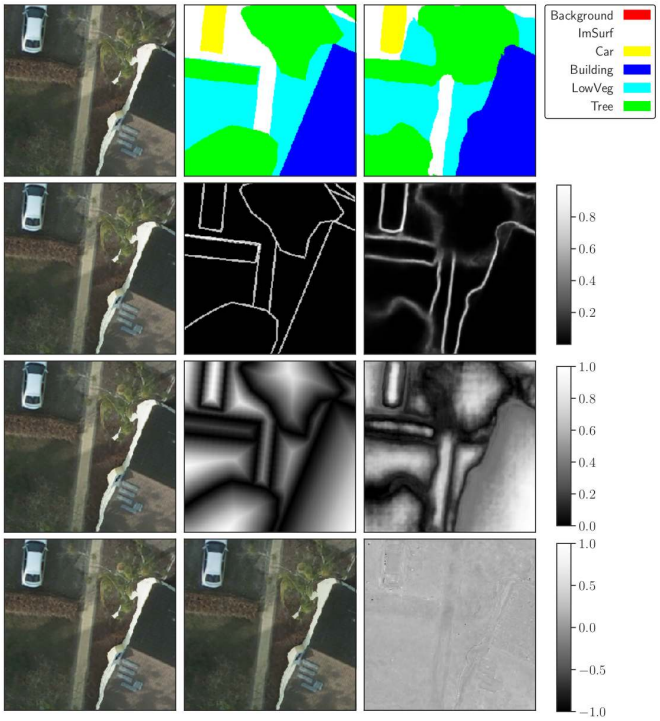


Figure 13: Same as Fig. 12 for the `ResUNet-a` d7v2 cmtsk trained on the FoV×1 dataset. It is clear that finer details are present especially for the class "Trees" and "LowVeg", that improve the performance of the algorithm over the FoV×4 dataset.

In this section, we compare the performance of the `ResUNet-a` modeling framework with a representative sample of (peer reviewed) alternative convolutional neural network models. For this comparison we evaluate the models trained on the FoV×1 dataset. The modeling frameworks we compare against `ResUNet-a` have published results on the ISPRS website. These are: UZ_1 (Volpi and Tuia, 2017), RIT_L7 (Liu et al., 2017b), RIT_4 (Piramanayagam et al., 2018), DST_5 (Sherrah, 2016), CAS_Y3 (ISPRS), CASIA2 (Liu et al., 2018), DPN_MFFL (Pan et al., 2018b), and HSN+OI+WBP (Liu et al., 2017a). To the best of our knowledge, at the time of writing this manuscript, these consist of the best performing models in the competition. For comparison, we provide the F1-score per class over all test tiles, the average F1-score over all classes over all test tiles, and the overall accuracy. Note that the average F1-score was calculated using all classes except the "Background" class. The overall accuracy, for `ResUNet-a` , was calculated including the "Background" category.

In Table 5 we provide the comparative results, per class, as well as the average F1 score and overall accuracy for the `ResUNet-a` d6 cmtsk and `ResUNet-a` d7v2 cmtsk models as well as results from other authors. `ResUNet-a` d6 performs very well in accordance with other state of the art modeling

Table 5: Potsdam comparison of results (based on per class F1 score) with other authors. Best values are marked with bold, second best values are underlined, third best values are in square brackets. Models trained with FoV×1 were trained on 256x256 patches extracted from the original resolution images.

Methods	ImSurface	Building	LowVeg	Tree	Car	Avg. F1	OA
UZ_1 (Volpi and Tuia, 2017)	89.3	95.4	81.8	80.5	86.5	86.7	85.8
RIT_L7 (Liu et al., 2017b)	91.2	94.6	85.1	85.1	92.8	89.8	88.4
RIT_4 (Piramanayagam et al., 2018)	92.6	97.0	86.9	87.4	95.2	91.8	90.3
DST_5 (Sherrah, 2016)	92.5	96.4	86.7	88.8	94.7	91.7	90.3
CAS_Y3 (ISPRS)	92.2	95.7	87.2	87.6	95.6	91.7	90.1
CASIA2 (Liu et al., 2018)	93.3	97.0	[87.7]	[88.4]	96.2	92.5	91.1
DPN_MFFL (Pan et al., 2018b)	92.4	[96.4]	87.8	88.0	95.7	92.1	90.4
HSN+OI+WBP (Liu et al., 2017a)	91.8	95.7	84.4	79.6	88.3	87.9	89.4
`ResUNet-a` d6 cmtsk (FoV×1)	[93.0]	**97.2**	87.5	[88.4]	[96.1]	[92.4]	[91.0]
`ResUNet-a` d7v2 cmtsk (FoV×1)	**93.5**	**97.2**	**88.2**	**89.2**	**96.4**	**92.9**	**91.5**

frameworks, and it ranks overall 3rd (average F1). It should be stressed that for the majority of the results, the performance differences are marginal. Going deeper, the `ResUNet-a` d7v2 model rank 1st among the representative sample of competing models, in all classes, thus clearly demonstrating the improvement over the state of the art. In Table 6 we provide the confusion matrix, over all test tiles, for this particular model.

It should be noted that some of the contributors (*e.g.*, CASIA2, RIT_4, DST_5) in the ISPRS competition used networks with pre-trained weights on external large data sets (e.g. ImageNet, Deng et al., 2009) and fine-tuning, i.e. a methodology called transfer learning (Pan and Yang, 2010, see also Penatti et al. (2015), Xie et al. (2015) for remote sensing applications). In particular, CASIA2, that has the 2nd highest overall score, used as a basis a state of the art pre-trained ResNet101 (He et al., 2016) network. In contrast, `ResUNet-a` was trained from random weights initialization only on the ISPRS Potsdam data set. Although it has been demonstrated that such a strategy does not influence the final performance, i.e. it is possible to achieve the same performance without pre-trained weights (He et al., 2018), this comes at the expense of a very long training time.

To visualize the performance of `ResUNet-a` , we generated error maps that indicate incorrect (correct) classification in red (green). All summary statistics and error maps were created using the software provided on the ISPRS competition website. For all of our inference results, we used the ground truth masks with eroded boundaries as suggested by the curators of the ISPRS Potsdam data set (ISPRS). This allows interested readers to have a clear picture of the strengths and weaknesses of our algorithm in comparison with online published results[6]. In Fig. 14 we provide the input image (left column), the error map between the inferred and ground truth masks (middle column) and the inference (right column) for a sample of four test tiles. In Appendix B we present the evaluation results for the rest of the test TOP tiles, per class. In all of these figures, for each row, from left to right: original image tile, error map and inference using our best model (`ResUNet-a-cmtsk` d7v2).

[6]For comparison, competition results can be found online.

7. Conclusions

In this work, we present a new deep learning modeling framework, for semantic segmentation of high resolution aerial images. The framework consists of a novel multitasking deep learning architecture for semantic segmentation and a new variant of the Dice loss that we term Tanimoto.

Our deep learning architecture, `ResUNet-a`, is based on the encoder/decoder paradigm, where standard convolutions are replaced with ResNet units that contain multiple in parallel atrous convolutions. Pyramid scene parsing pooling is included in the middle and end of the network. The best performant variant of our models are conditioned multitasking models which predict among with the segmentation mask also the boundaries of the various classes, the distance transform (that provides information for the topological connectivity of the objects) as well as the identity reconstruction of the input image. The additionally inferred tasks, are re-used internally into the network before the final segmentation mask is produced. That is, the final segmentation mask is conditioned on the inference result of the boundaries of the objects as well as the distance transform of their segmentation mask. We show experimentally that the conditioned multitasking improves the performance of the inferred semantic segmentation classes. The ground truth labels that are used during training for the boundaries, as well as the distance transform, can be both calculated very easily from the ground truth segmentation mask using standard computer vision software (OpenCV).

We analyze the performance of various flavours of the Dice loss and introduce a novel variant of this as a loss function, the Tanimoto loss. This loss can also be used for regression problems. This is an appealing property that makes this loss useful for the case of multitasking problems in that it results in balanced gradients for all tasks during training. We show experimentally that the Tanimoto loss speeds up the training convergence and behaves well under the presence of heavily imbalanced data sets.

The performance of our framework is evaluated on the 2D semantic segmentation ISPRS Potsdam data set. Our best model, `ResUNet-a` d7v2 achieves top rank performance in comparison with other published results (Table 5) and demonstrates a clear improvement over the state of the art. The combination

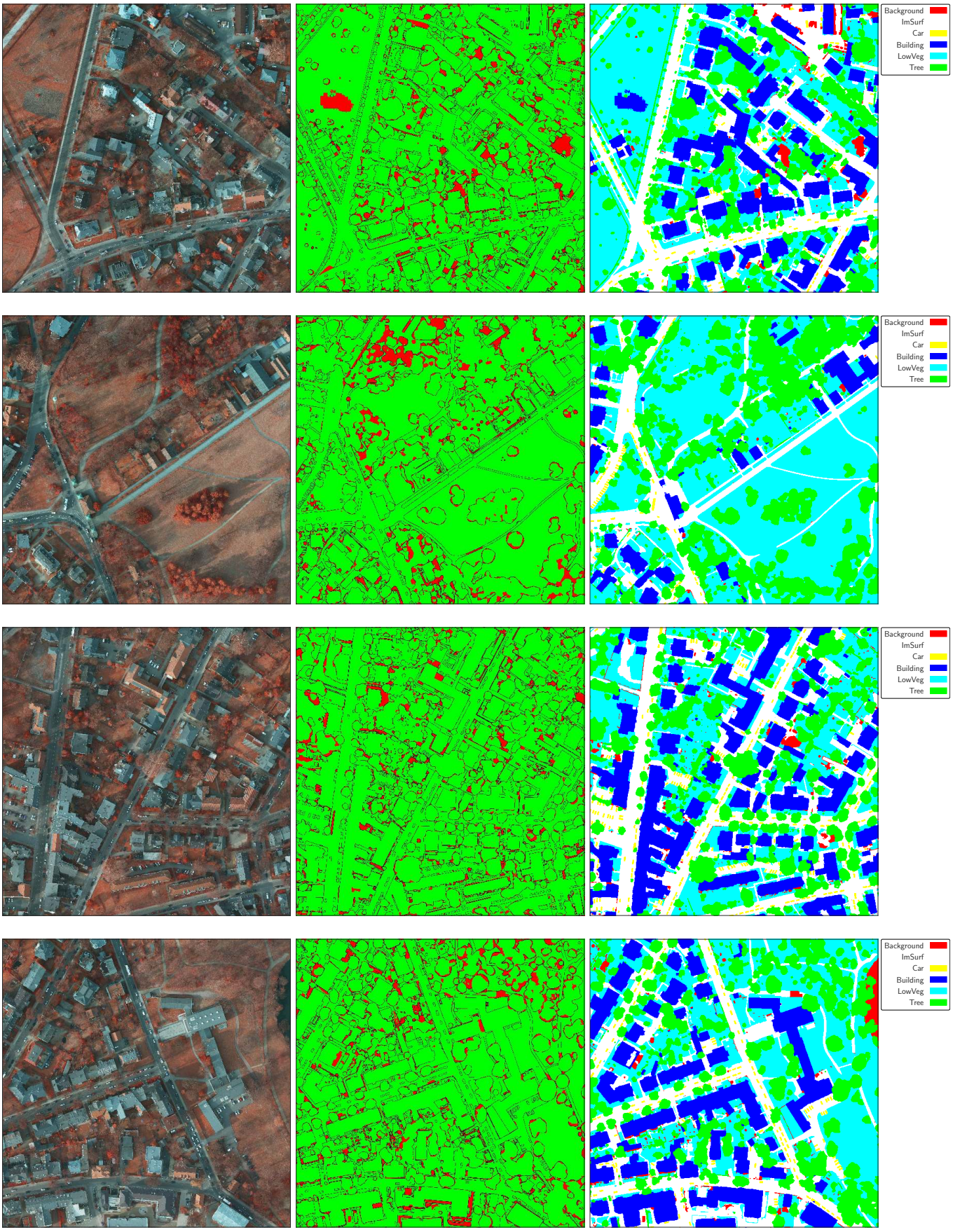


Figure 14: `ResUNet-a` best model results for tiles 2-13, 2-14, 3-13 and 3-14. From left to right, input image, difference between ground truth and predictions, inference map.

Table 6: Potsdam summary confusion matrix over all test tiles for ground truth masks that do not include the boundary. The results correspond to the best model, `ResUNet-a-cmtsk` d7v2, trained on the FoV×1 dataset. The overall accuracy achieved is **91.5%**

.

Predicted \ Reference	ImSurface	Building	LowVeg	Tree	Car	Clutter/Background
ImSurface	**0.9478**	0.0085	0.0247	0.0117	0.0002	0.0071
Building	0.0115	**0.9765**	0.0041	0.0025	0.0001	0.0053
LowVeg	0.0317	0.0057	**0.9000**	0.0532	0.0000	0.0095
Tree	0.0223	0.0036	0.0894	**0.8807**	0.0016	0.0024
Car	0.0070	0.0016	0.0002	0.0091	**0.9735**	0.0086
Clutter/Background	0.2809	0.0844	0.1200	0.0172	0.0090	**0.4885**
Precision/Correctness	0.9220	0.9679	0.8640	0.9030	0.9538	0.7742
Recall/Completeness	0.9478	0.9765	0.9000	0.8807	0.9735	0.4885
F1	0.9347	0.9722	0.8816	0.8917	0.9635	0.5990

of `ResUNet-a` conditioned multitasking with the proposed loss function is a reliable solution for performant semantic segmentation tasks.

Acknowledgments

The authors acknowledge the support of the Scientific Computing team of CSIRO, and in particular Peter H. Campbell and Ondrej Hlinka. Their contribution was substantial in overcoming many technical difficulties of distributed GPU computing. The authors are also grateful to John Taylor for his help in understanding and implementing distributed optimization using Horovod (Sergeev and Balso, 2018). The authors acknowledge the support of the mxnet community, and in particular Thomas Delteil, Sina Afrooze and Thom Lane. The authors acknowledge the provision of the Potsdam ISPRS dataset by BSF Swissphoto[7].

References

References

A. Vadivel, Shamik Sural, A.K.M., 2005. Human color perception in the hsv space and its application in histogram generation for image retrieval. URL: https://doi.org/10.1117/12.586823, doi:10.1117/12.586823.

Audebert, N., Le Saux, B., Lefèvre, S., 2018. Beyond rgb: Very high resolution urban remote sensing with multimodal deep networks. ISPRS Journal of Photogrammetry and Remote Sensing 140, 20–32.

Audebert, N., Le Saux, B., Lefvre, S., 2017. Segment-before-detect: Vehicle detection and classification through semantic segmentation of aerial images. Remote Sensing 9. URL: http://www.mdpi.com/2072-4292/9/4/368, doi:10.3390/rs9040368.

Audebert, N., Saux, B.L., Lefèvre, S., 2016. Semantic segmentation of earth observation data using multimodal and multi-scale deep networks. CoRR abs/1609.06846. URL: http://arxiv.org/abs/1609.06846, arXiv:1609.06846.

Baatz, M., Schäpe, A., 2000. Multiresolution segmentation: An optimization approach for high quality multi-scale image segmentation (ecognition) , 12–23.

Bertasius, G., Shi, J., Torresani, L., 2015. Semantic segmentation with boundary neural fields. CoRR abs/1511.02674. URL: http://arxiv.org/abs/1511.02674, arXiv:1511.02674.

[7] http://www.bsf-swissphoto.com/unternehmen/ueber_uns_bsf.

Blaschke, T., Hay, G.J., Kelly, M., Lang, S., Hofmann, P., Addink, E., Feitosa, R.Q., Van der Meer, F., Van der Werff, H., Van Coillie, F., et al., 2014. Geographic object-based image analysis–towards a new paradigm. ISPRS journal of photogrammetry and remote sensing 87, 180–191.

Borgefors, G., 1986. Distance transformations in digital images. Comput. Vision Graph. Image Process. 34, 344–371. URL: http://dx.doi.org/10.1016/S0734-189X(86)80047-0, doi:10.1016/S0734-189X(86)80047-0.

Chen, L., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L., 2016. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. CoRR abs/1606.00915. URL: http://arxiv.org/abs/1606.00915, arXiv:1606.00915.

Chen, L., Papandreou, G., Schroff, F., Adam, H., 2017. Rethinking atrous convolution for semantic image segmentation. CoRR abs/1706.05587. URL: http://arxiv.org/abs/1706.05587, arXiv:1706.05587.

Chen, T., Li, M., Li, Y., Lin, M., Wang, N., Wang, M., Xiao, T., Xu, B., Zhang, C., Zhang, Z., 2015. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. arXiv preprint arXiv:1512.01274 .

Cheng, G., Wang, Y., Xu, S., Wang, H., Xiang, S., Pan, C., 2017. Automatic road detection and centerline extraction via cascaded end-to-end convolutional neural network. IEEE Transactions on Geoscience and Remote Sensing 55, 3322–3337.

Comaniciu, D., Meer, P., 2002. Mean shift: A robust approach toward feature space analysis. IEEE Transactions on pattern analysis and machine intelligence 24, 603–619.

Crum, W.R., Camara, O., Hill, D.L.G., 2006. Generalized overlap measures for evaluation and validation in medical image analysis. IEEE Trans. Med. Imaging 25, 1451–1461. URL: http://dblp.uni-trier.de/db/journals/tmi/tmi25.html#CrumCH06.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L., 2009. ImageNet: A Large-Scale Hierarchical Image Database, in: CVPR09.

Dice, L.R., . Measures of the amount of ecologic association between species. Ecology 26, 297–302. doi:10.2307/1932409.

Drozdzal, M., Vorontsov, E., Chartrand, G., Kadoury, S., Pal, C., 2016. The importance of skip connections in biomedical image segmentation. CoRR abs/1608.04117. URL: http://arxiv.org/abs/1608.04117, arXiv:1608.04117.

Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A., 2010. The pascal visual object classes (voc) challenge. International Journal of Computer Vision 88, 303–338.

Goldblatt, R., Stuhlmacher, M.F., Tellman, B., Clinton, N., Hanson, G., Georgescu, M., Wang, C., Serrano-Candela, F., Khandelwal, A.K., Cheng, W.H., et al., 2018. Using landsat and nighttime lights for supervised pixel-based image classification of urban land cover. Remote Sensing of Environment 205, 253–275.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets, in: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (Eds.), Advances in Neural Information Processing Systems 27. Curran Associates, Inc., pp. 2672–2680. URL: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf.

Goyal, P., Dollár, P., Girshick, R.B., Noordhuis, P., Wesolowski, L., Kyrola, A.,

Tulloch, A., Jia, Y., He, K., 2017. Accurate, large minibatch SGD: training imagenet in 1 hour. CoRR abs/1706.02677. URL: http://arxiv.org/abs/1706.02677, arXiv:1706.02677.

Gu, Y., Wang, Y., Li, Y., 2019. A survey on deep learning-driven remote sensing image scene understanding: Scene classification, scene retrieval and scene-guided object detection. Applied Sciences 9. URL: https://www.mdpi.com/2076-3417/9/10/2110, doi:10.3390/app9102110.

He, K., Girshick, R.B., Dollár, P., 2018. Rethinking imagenet pre-training. CoRR abs/1811.08883. URL: http://arxiv.org/abs/1811.08883, arXiv:1811.08883.

He, K., Gkioxari, G., Dollár, P., Girshick, R.B., 2017. Mask R-CNN. CoRR abs/1703.06870. URL: http://arxiv.org/abs/1703.06870, arXiv:1703.06870.

He, K., Zhang, X., Ren, S., Sun, J., 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. CoRR abs/1406.4729. URL: http://arxiv.org/abs/1406.4729, arXiv:1406.4729.

He, K., Zhang, X., Ren, S., Sun, J., 2015. Deep residual learning for image recognition. CoRR abs/1512.03385. URL: http://arxiv.org/abs/1512.03385, arXiv:1512.03385.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Identity mappings in deep residual networks. CoRR abs/1603.05027. URL: http://arxiv.org/abs/1603.05027, arXiv:1603.05027.

Ioffe, S., Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. CoRR abs/1502.03167. URL: http://arxiv.org/abs/1502.03167, arXiv:1502.03167.

ISPRS, . International society for photogrammetry and remote sensing (isprs) and bsf swissphoto: Wg3 potsdam overhead data. http://www2.isprs.org/commissions/comm3/wg4/tests.html.

Jaderberg, M., Simonyan, K., Zisserman, A., Kavukcuoglu, K., 2015. Spatial transformer networks. CoRR abs/1506.02025. URL: http://arxiv.org/abs/1506.02025, arXiv:1506.02025.

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. CoRR abs/1412.6980. URL: http://arxiv.org/abs/1412.6980, arXiv:1412.6980.

Lambert, M.J., Waldner, F., Defourny, P., 2016. Cropland mapping over sahelian and sudanian agrosystems: A knowledge-based approach using proba-v time series at 100-m. Remote Sensing 8, 232.

Längkvist, M., Kiselev, A., Alirezaie, M., Loutfi, A., 2016. Classification and segmentation of satellite orthoimagery using convolutional neural networks. Remote Sensing 8, 329.

LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E., Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. Neural Computation 1, 541–551. URL: https://doi.org/10.1162/neco.1989.1.4.541, doi:10.1162/neco.1989.1.4.541.

Li, E., Femiani, J., Xu, S., Zhang, X., Wonka, P., 2015. Robust rooftop extraction from visible band images using higher order crf. IEEE Transactions on Geoscience and Remote Sensing 53, 4483–4495.

Li, S., Jiao, J., Han, Y., Weissman, T., 2016. Demystifying resnet. CoRR abs/1611.01186. URL: http://arxiv.org/abs/1611.01186, arXiv:1611.01186.

Li, X., Shao, G., 2014. Object-based land-cover mapping with high resolution aerial photography at a county scale in midwestern usa. Remote Sensing 6, 11372–11390.

Liu, Y., Fan, B., Wang, L., Bai, J., Xiang, S., Pan, C., 2018. Semantic labeling in very high resolution images via a self-cascaded convolutional neural network. ISPRS Journal of Photogrammetry and Remote Sensing 145, 78–95.

Liu, Y., Minh Nguyen, D., Deligiannis, N., Ding, W., Munteanu, A., 2017a. Hourglass-shapenetwork based semantic segmentation for high resolution aerial imagery. Remote Sensing 9. URL: http://www.mdpi.com/2072-4292/9/6/522, doi:10.3390/rs9060522.

Liu, Y., Piramanayagam, S., Monteiro, S.T., Saber, E., 2017b. Dense semantic labeling of very-high-resolution aerial imagery and lidar with fully-convolutional neural networks and higher-order crfs, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Honolulu, USA.

Long, J., Shelhamer, E., Darrell, T., 2014. Fully convolutional networks for semantic segmentation. CoRR abs/1411.4038. URL: http://arxiv.org/abs/1411.4038, arXiv:1411.4038.

Lu, X., Yuan, Y., Zheng, X., 2017. Joint dictionary learning for multispectral change detection. IEEE Trans. Cybernetics 47, 884–897.

Marmanis, D., Schindler, K., Wegner, J.D., Galliani, S., Datcu, M., Stilla, U., 2018. Classification with an edge: Improving semantic image segmentation with boundary detection. ISPRS Journal of Photogrammetry and Remote Sensing 135, 158–172.

Matikainen, L., Karila, K., 2011. Segment-based land cover mapping of a suburban areacomparison of high-resolution remotely sensed datasets using classification trees and test field points. Remote Sensing 3, 1777–1804.

Matthews, B., 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. Biochimica et Biophysica Acta (BBA) - Protein Structure 405, 442 – 451. URL: http://www.sciencedirect.com/science/article/pii/0005279575901099, doi:https://doi.org/10.1016/0005-2795(75)90109-9.

Milletari, F., Navab, N., Ahmadi, S., 2016. V-net: Fully convolutional neural networks for volumetric medical image segmentation. CoRR abs/1606.04797. URL: http://arxiv.org/abs/1606.04797, arXiv:1606.04797.

Myint, S.W., Gober, P., Brazel, A., Grossman-Clarke, S., Weng, Q., 2011. Per-pixel vs. object-based classification of urban land cover extraction using high spatial resolution imagery. Remote sensing of environment 115, 1145–1161.

Novikov, A.A., Major, D., Lenis, D., Hladuvka, J., Wimmer, M., Bühler, K., 2017. Fully convolutional architectures for multi-class segmentation in chest radiographs. CoRR abs/1701.08816. URL: http://arxiv.org/abs/1701.08816, arXiv:1701.08816.

Odena, A., Dumoulin, V., Olah, C., 2016. Deconvolution and checkerboard artifacts. Distill URL: http://distill.pub/2016/deconv-checkerboard/.

Paisitkriangkrai, S., Sherrah, J., Janney, P., van den Hengel, A., 2016. Semantic labeling of aerial and satellite imagery. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 9, 2868–2881.

Pan, S.J., Yang, Q., 2010. A survey on transfer learning. IEEE Trans. on Knowl. and Data Eng. 22, 1345–1359. URL: http://dx.doi.org/10.1109/TKDE.2009.191, doi:10.1109/TKDE.2009.191.

Pan, X., Gao, L., Marinoni, A., Zhang, B., Yang, F., Gamba, P., 2018a. Semantic labeling of high resolution aerial imagery and lidar data with fine segmentation network. Remote Sensing 10. URL: http://www.mdpi.com/2072-4292/10/5/743, doi:10.3390/rs10050743.

Pan, X., Gao, L., Zhang, B., Yang, F., Liao, W., 2018b. High-resolution aerial imagery semantic labeling with dense pyramid network. Sensors 18. URL: http://www.mdpi.com/1424-8220/18/11/3774, doi:10.3390/s18113774.

Penatti, O.A., Nogueira, K., dos Santos, J.A., 2015. Do deep features generalize from everyday objects to remote sensing and aerial scenes domains?, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 44–51. URL: doi.ieeecomputersociety.org/10.1109/CVPRW.2015.7301382, doi:10.1109/CVPRW.2015.7301382.

Piramanayagam, S., Saber, E., Schwartzkopf, W., Koehler, F.W., 2018. Supervised classification of multisensor remotely sensed images using a deep learning framework. Remote Sensing 10. URL: http://www.mdpi.com/2072-4292/10/9/1429, doi:10.3390/rs10091429.

Rawat, W., Wang, Z., 2017. Deep convolutional neural networks for image classification: A comprehensive review. Neural Computation 29, 2352–2449. doi:10.1162/neco_a_00990. pMID: 28599112.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. CoRR abs/1505.04597. URL: http://arxiv.org/abs/1505.04597, arXiv:1505.04597.

Sergeev, A., Balso, M.D., 2018. Horovod: fast and easy distributed deep learning in TensorFlow. arXiv preprint arXiv:1802.05799 .

Sherrah, J., 2016. Fully convolutional networks for dense semantic labelling of high-resolution aerial imagery. CoRR abs/1606.02585. URL: http://arxiv.org/abs/1606.02585, arXiv:1606.02585.

Smith, L.N., 2018. A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay. CoRR abs/1803.09820. URL: http://arxiv.org/abs/1803.09820, arXiv:1803.09820.

Sørensen, T., 1948. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. Biol. Skr. 5, 1–34.

Sudre, C.H., Li, W., Vercauteren, T., Ourselin, S., Cardoso, M.J., 2017. Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations. CoRR abs/1707.03237. URL: http://arxiv.org/abs/1707.03237, arXiv:1707.03237.

Vincent, L., Soille, P., 1991. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. IEEE Transactions on Pattern Analysis & Machine Intelligence , 583–598.

Volpi, M., Tuia, D., 2017. Dense semantic labeling of subdecimeter resolution images with convolutional neural networks. IEEE Transactions on Geoscience and Remote Sensing 55, 881–893.

Waldner, F., Hansen, M.C., Potapov, P.V., Löw, F., Newby, T., Ferreira, S., Defourny, P., 2017. National-scale cropland mapping based on spectral-temporal features and outdated land cover information. PloS one 12, e0181911.

Wen, D., Huang, X., Liu, H., Liao, W., Zhang, L., 2017. Semantic classification of urban trees using very high resolution satellite imagery. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 10, 1413–1424.

Xie, S.M., Jean, N., Burke, M., Lobell, D.B., Ermon, S., 2015. Transfer learning from deep features for remote sensing and poverty mapping. CoRR abs/1510.00098. URL: http://arxiv.org/abs/1510.00098, arXiv:1510.00098.

Yang, H., Wu, P., Yao, X., Wu, Y., Wang, B., Xu, Y., 2018. Building extraction in very high resolution imagery by dense-attention networks. Remote Sensing 10. URL: http://www.mdpi.com/2072-4292/10/11/1768, doi:10.3390/rs10111768.

Zagoruyko, S., Komodakis, N., 2016. Wide residual networks. CoRR abs/1605.07146. URL: http://arxiv.org/abs/1605.07146, arXiv:1605.07146.

Zhang, H., Dana, K., Shi, J., Zhang, Z., Wang, X., Tyagi, A., Agrawal, A., 2018. Context encoding for semantic segmentation, in: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Zhang, Q., Seto, K.C., 2011. Mapping urbanization dynamics at regional and global scales using multi-temporal dmsp/ols nighttime light data. Remote Sensing of Environment 115, 2320–2329.

Zhang, Z., Liu, Q., Wang, Y., 2017. Road extraction by deep residual u-net. CoRR abs/1711.10684. URL: http://arxiv.org/abs/1711.10684, arXiv:1711.10684.

Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J., 2017. Pyramid scene parsing network, in: CVPR.

Zhu, J., Park, T., Isola, P., Efros, A.A., 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. CoRR abs/1703.10593. URL: http://arxiv.org/abs/1703.10593, arXiv:1703.10593.

Appendix A. Software implementation and training characteristics

`ResUNet-a` was built and trained using the MXNET deep learning library (Chen et al., 2015), under the GLUON API. Each of the models trained on the FoV×4 dataset was trained with a batch size of 256 on a single node containing 4 NVIDIA Tesla P100 GPUs in CSIRO HPC facilities. Due to the complexity of the network, the batch size in a single GPU iteration cannot be made larger than ~ 10 (per GPU). In order to increase the batch size we used manual gradient aggregation[8]. For the models trained on the FoV×1 dataset we used a batch size of 480 in order to speed up the computation. These were trained in a distributed scheme, using the ring allreduce algorithm, and in particular it's implementation on HOROVOD (Sergeev and Balso, 2018) for the MXNET (Chen et al., 2015) deep learning library. The optimal learning rate for all runs was set by the methodology developed in Smith (2018). In particular, by monitoring the loss error during training for a continuously increasing learning rate, starting from a very low value. An example is shown in Fig. A.15: The optimal learning rate is approximately

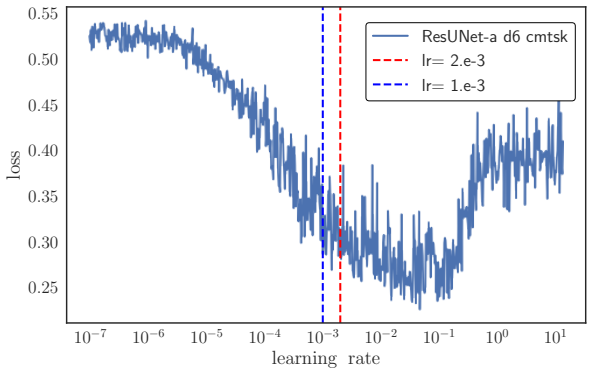


Figure A.15: Learning rate finder process for the FoV×1 dataset. The model used was `ResUNet-a` d6 cmtsk. For this particular training profile, our learning rate choice was 0.001, although a little higher values are possible (see Smith 2018 for details.

the point of steepest decent of the loss functions. This process was complete in approximately 1 epoch and it can be applied in a distributed scheme as well. We found it more useful than the linear learning rate scaling that is used for large batch size (Goyal et al., 2017) in distributed optimization.

For all models , we used the Adam (Kingma and Ba, 2014) optimizer, with an initial learning rate of 0.001 (initial learning rate can also be set higher for this dataset, see Fig. A.15), momentum parameters $(\beta_1, \beta_2) = (0.9, 0.999)$. The learning rate was reduced by an order of magnitude whenever the validation loss stopped decreasing. Overall we reduced the learning rate 3 times. We have also experimented with smaller batch sizes. In particular, with a batch size of 32, the training is unstable. This is owed mainly to the fact that we used 4 GPUs for training, therefore the batch size per GPU is 8, and this is not sufficient for the Batch Normalization layers that use only the data per GPU for the estimation of running means of their parameters. When we experimented with synchronized Batch Normalization layers (Ioffe and Szegedy, 2015; Zhang et al., 2018), this increased the stability of the training dramatically even with a batch size as small as 32. However, due to the GPU synchronization, this was a slow operation that proved to be impractical for our purposes.

A software implementation for the `ResUNet-a` models that relate to this work can be found on github[9].

Appendix B. Inference results

In this section, we present classification results and error maps for all the test TOP tiles of the Potsdam ISPRS dataset.

[8] A a short tutorial on manual gradient aggregation with the GLUON API in the MXNET framework can be found online.

[9] https://github.com/feevos/resuneta

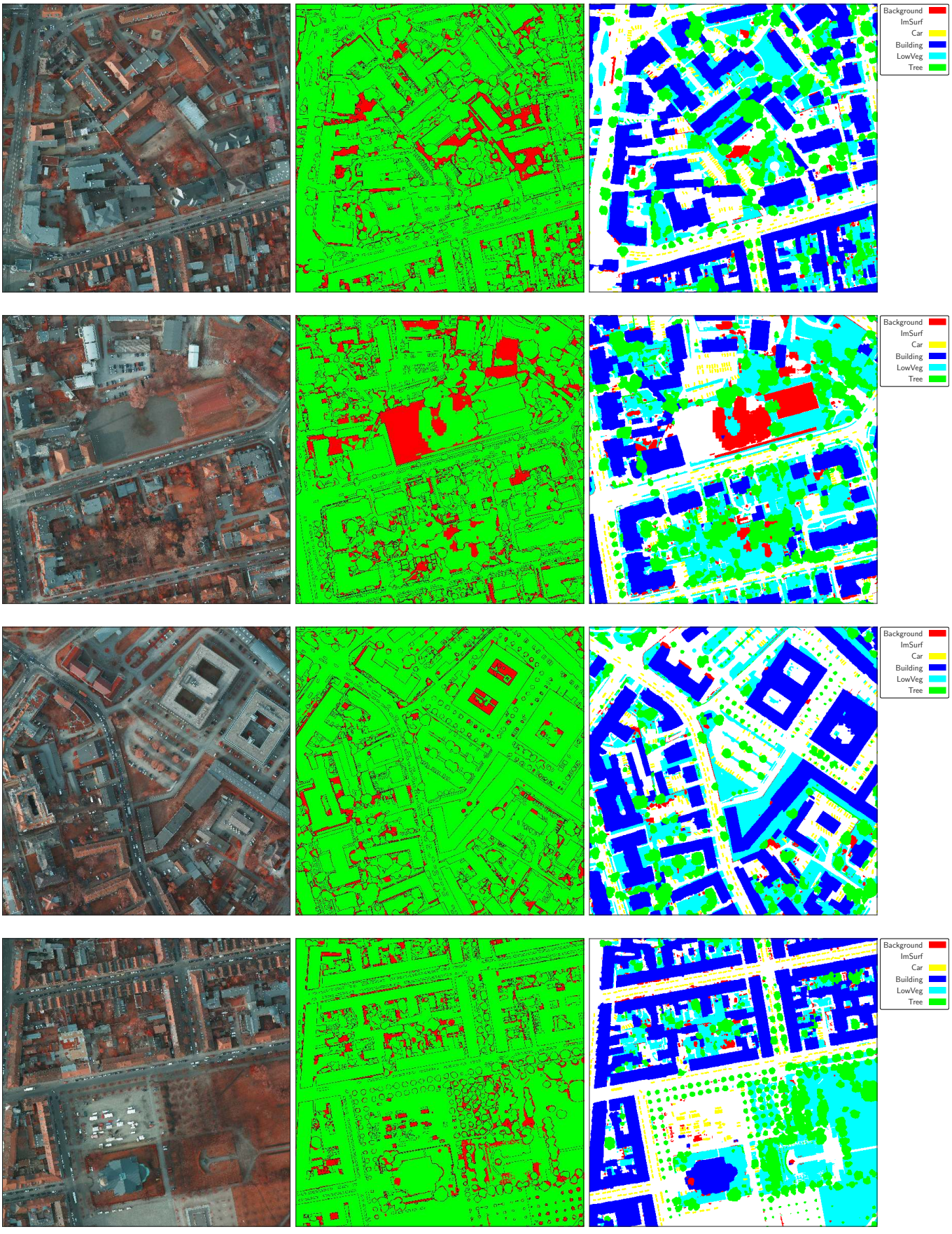


Figure B.16: `ResUNet-a` best model results for tiles 4-13, 4-14, 4-15, 5-13. From left to right, input image, difference between ground truth and predictions, inference map.

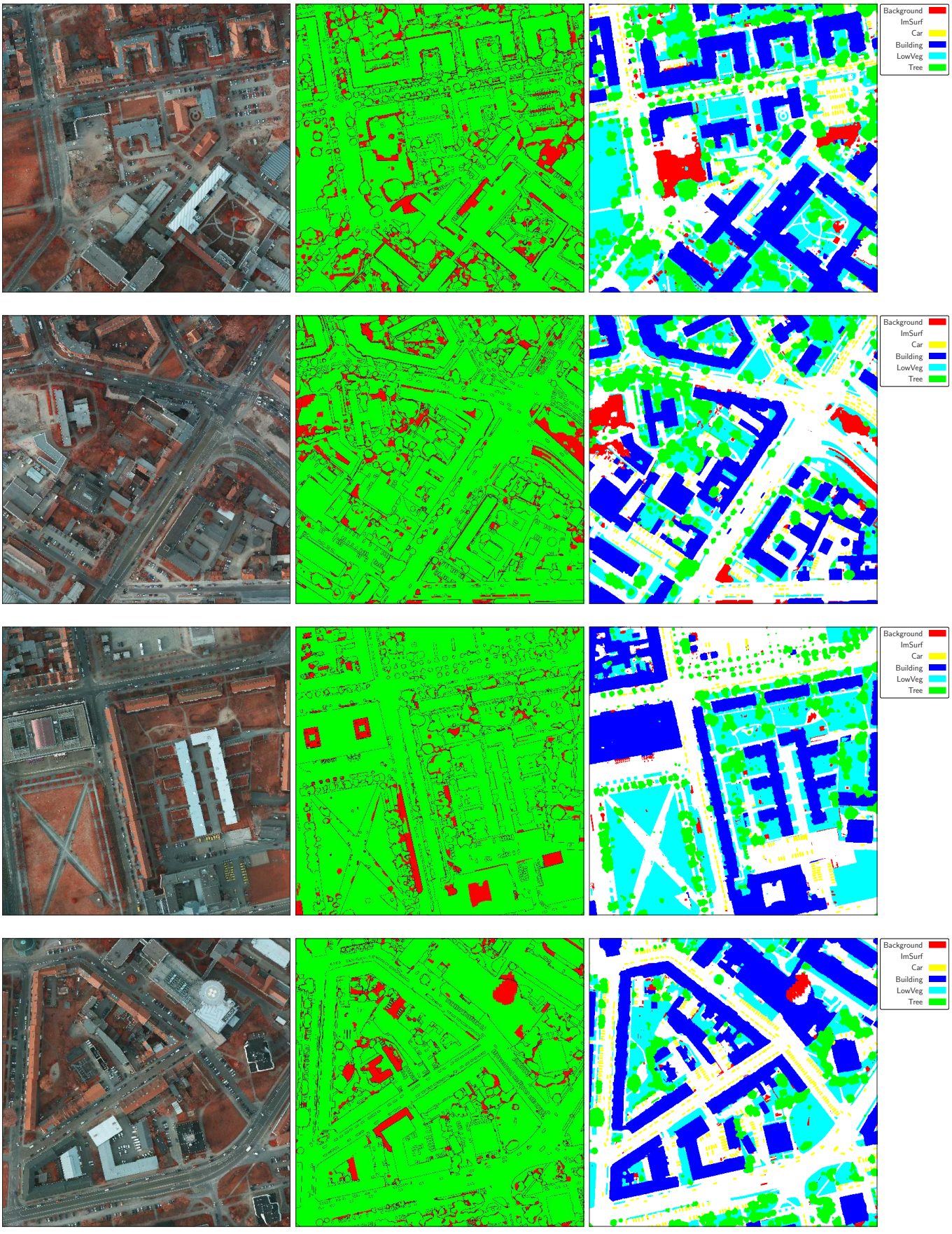


Figure B.17: As Fig. B.16 for tiles 5-14, 5-15, 6-13, 6-14

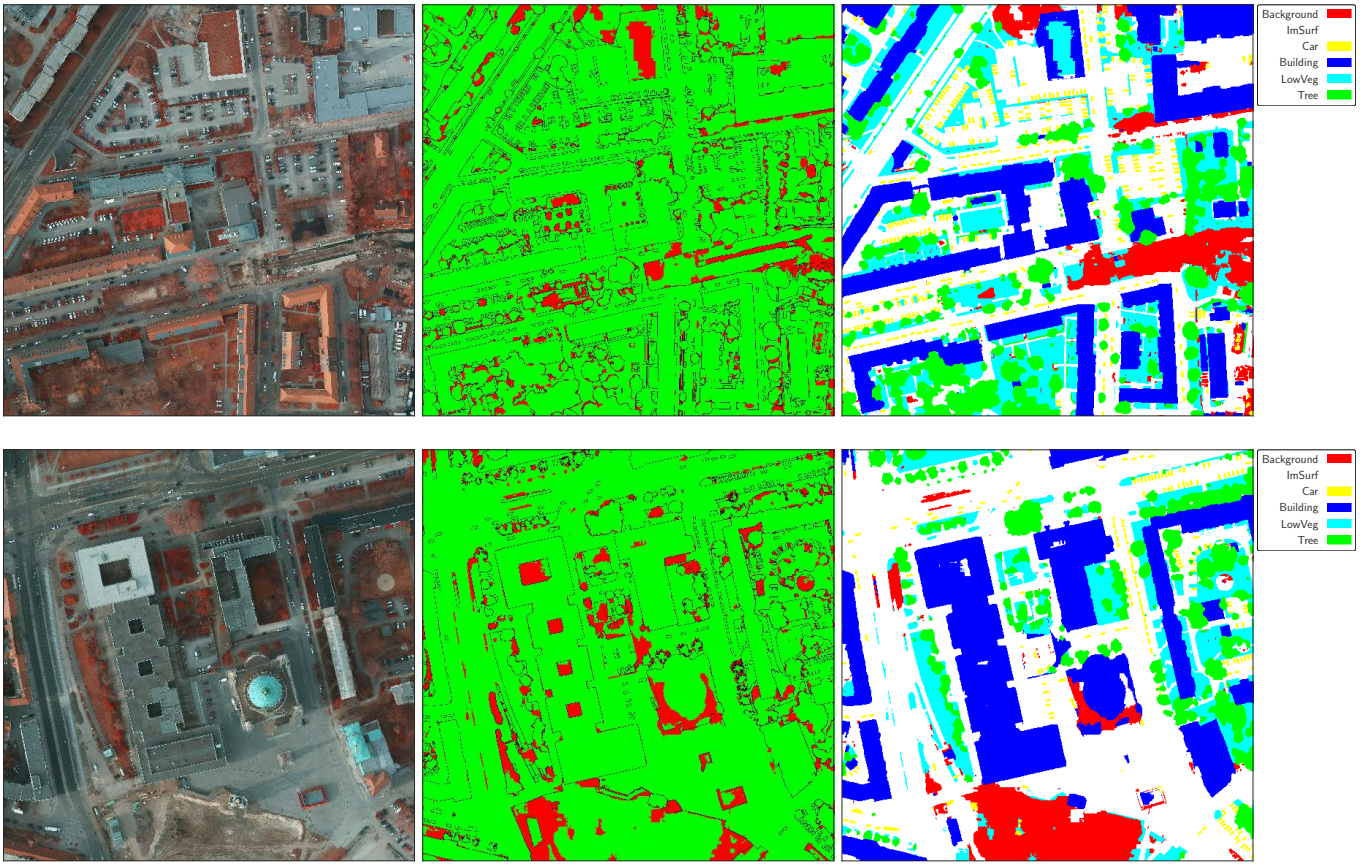


Figure B.18: As Fig. B.16 for tiles 6-15, 7-13