Approaches to Semantic Segmentation of Aerial Images

Noemi Manara
noemi.manara@studenti.unipd.it

Giuseppe Simionato
giuseppe.simionato.1@studenti.unipd.it

Nicholas Sinigaglia
nicholas.sinigaglia@studenti.unipd.it

Abstract

Image segmentation is a topic of paramount importance in our society, it finds applications from computer vision to medical image analysis, robotic perception, video surveillance, and many others. Currently, there are various algorithms deployed for the semantic segmentation task and the most common are deep learning models such as convolutional neural networks (CNNs). In this report, we present, implement, and study three different algorithms to perform semantic segmentation of aerial images under the constraint of limited data and few classes. The first approach is a fully CNN (FCNN) designed by us taking inspiration from U-Net. The second approach is to adapt the Xception pretrained classifier using transfer learning and fine-tuning (XTFT). The third and last approach is a Random Forest Classifier (RF). The models are trained over the same dataset and in the same environment (same system specifics). Thanks to this, we provide a complete comparison of the three models, seeing how the best approach in our case is a FCNN with a contained number of parameters.

1. Introduction

Image segmentation is a key component in many computer vision systems. It involves partitioning images into multiple segments or objects [11] performing a pixel-wise classification. Segmentation plays a central role in a broad range of applications, including autonomous vehicles [4], medical image analysis [2], video surveillance and augmented reality. The amount of high resolution satellite images available daily is a great resource that can and should be exploited to tackle today's world problems, such as deforestation, arson, overbuilding, protection of water resources and poverty. For this reason, our interest is focused on building an algorithm that performs well on the task of semantic segmentation of aerial images.

Numerous image segmentation algorithms have been developed in the literature, from histogram-based bundling to k-means clustering, to much more advanced techniques [5].

As it is well known, convolutional neural networks (CNNs) are part of these powerful visual models, since they are designed to find common structures in different images. For this reason, fully CNNs are capable of obtaining stunning performances in the image segmentation task [8].

Another common strategy used to learn visual tasks is transfer learning followed by fine-tuning. This setup is especially beneficial when dealing with limited hardware availability as described in [8].

A less common approach is the usage of a Random Forest (RF), which is an ensemble learning technique where multiple classifiers are trained and a combination of their hypotheses is used. The classifiers are decision trees [7].

In this paper, we compare three different models under the same conditions (data and resources) to provide a full comparison based on our experience: from architecture design to memory usage, training time, and finally performance.
The first model is a fully CNN (FCNN) inspired by U-Net [10]. The second is an encoder-decoder structure, where the encoder is Xception pretrained on ImageNet and the decoder is a NN based on transposed convolution layers. The third is a Random Forest based on a simplified version of the work of B. Kang and T.Q. Nguyen [9].

We show how the FCNN and the fine-tuned Xception + decoder have compatible performances over the dataset, but the FCNN is less expensive in terms of memory usage and training time. The random forest is the best model in terms of training time, but its performances are by far the worst.

2. Related works

The work of Long et al. [8] is our main reference, as it paves the way for semantic segmentation with fully convolutional neural networks. It suggests to perform the upsampling in-network to have end-to-end learning by backpropagation, and in this work we do so with an extensive implementation of Convolutional Transpose layers. Even if their work underlines the benefits of wholeimage training,

in our case we face the issue of a small dataset, so we revert to patchwise training to perform some data augmentation through cuts.

Another reference is the recent work of Wurm et al. on semantic segmentation of Mumbai satellite images [12]. They aim at classifying categories similar to ours (urban, vegetation, water, and slums), and as well take advantage of patchwise training. The performances that they encounter are very high, but they only investigate the transfer learning approach. In our work, we broaden the framework by considering also two other kinds of models.

We have not found any related work that used as well the Dubai satellite dataset, so it is not possible to directly compare our results with others, and we set this work as the first step towards the solution of this task.

3. Dataset

The dataset used is taken from Kaggle[1] and it is made of 72 aerial RGB images of Dubai with associated semantic RGB masks. There are six classes: building, land, road, vegetation, water, and unlabeled. The images are divided in different folders based on the location, they have been merged and the images have been randomly shuffled once loaded.

In order to implement a loss function suited for the task (for example, Categorical Crossentropy), we use the one-hot encoding of the masks in grayscale: instead of three color channels, we created six "class channels", in which a '0' means that the corresponding pixel does not belong to that class and instead a '1' that it does.

3.1. Pre processing

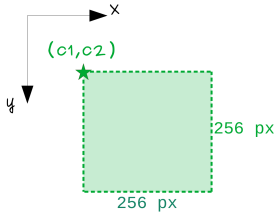


Figure 1: A cut (and the patch) is defined by the coordinates *(c1,c2)* of its upper-left corner

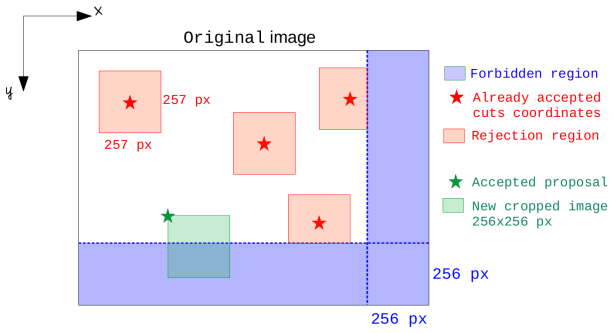
Every image has a different shape, but since the goal of this work is not to create models that take inputs of arbitrary shapes, we decide to fix the input shape of our models to 256x256 px. Reshaping all the images to the input shape is not a wise choice: the transformation would change mask pixels, adding shades and making impossible the one-hot encoding. Moreover, it would imply a loss of resolution and thus information. A much better option than reshaping is extracting 256x256 px patches from every image performing a direct cut, which is defined in Figure 1. One way to do it is to cut **n** casual patches from each image.

[1] https://www.kaggle.com/humansintheloop/semantic-segmentation-of-aerial-imagery

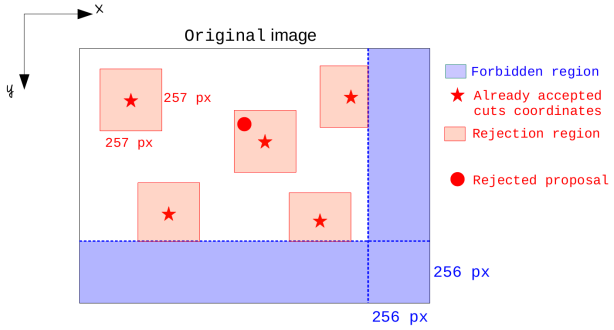
However, performing completely random cuts can lead to obtain very similar patches that hold the same information. For this reason, we design a "smart" casual cropper (SCC), which has memory of already extracted patches. In this way, it avoids to crop two almost identical patches.

The SCC follows a simple idea: it randomly extracts a cut proposal *(r1,r2)* and it accepts it only if it is far enough from all the others accepted proposals. In figures 2a and 2b this iterative procedure is illustrated more in detail. In our pre-processing SCC has been iterated until there were no more acceptable proposals, consequently extracting the maximum number of patches from each image.

In this way, we hope to achieve higher variance batches and a subsequent improvement of the training process.



(a) n-th step of SCC, the proposal is accepted



(b) (n+1)-th step of SCC, the proposal is rejected

Figure 2: Representation of SCC iterative procedure. The blue forbidden region represents unsuitable proposals because the cut would overextend the original image. The red rejection region is a 257x257 px region centered around accepted cuts coordinates, defined in Figure 1.

3.2. Training samples augmentation

Once we have pre-processed the training images thanks to SCC, we double the obtained 256x256 RGB images through to the following augmentation: for each RGB image a copy is created, which is randomly flipped horizontally or vertically and then a gamma correction is applied,

where *gamma* has a random value $\in [0.5, 2]$ with mean value 1, generated thanks to equation 1. The mask associated to the new image is a flipped version of the original.

$$gamma = \frac{2}{1+3\cdot r} \quad , \quad r \sim Unif[0,1] \qquad (1)$$

4. Methods

We tackle the problem with three different approaches, which are presented in this section.

4.1. Fully Convolutional Neural Network (FCNN)

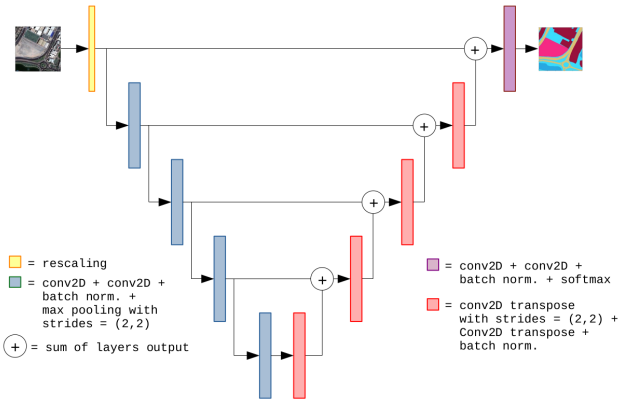


Figure 3: The architecture of the FCNN model. After each blue block the image dimension is halved, while after each red block it is doubled. This implies that the "bottleneck" dimension is 16x16xN, where N is the number of filters. Every conv2D and conv2D transpose uses padding and has 16 filters with few exceptions, this is due to the fact that layers ouput must have same shape in order to be summed. The kernel size of conv2D and conv2D transpose changes depending on the block.

CNNs in their original form were designed for recognition, i.e. assigning a single label to an entire image. The bottleneck when using them for semantic segmentation (where we label each single pixel) is the loss of the spatial coordinate. Extensions of the basic CNN architecture which mitigate this problem have been developed. Thanks to deconvolutional layers that learn to reverse the down-sampling and together with direct connections from lower layers that "skip" parts of the network, so called Fully Convolutional Networks make it possible to predict at a finer spatial resolution than would be possible after multiple rounds of pooling. Our FCNN is implemented with deconvolutional layers, which perform a learned upsampling of the previous layer (they are the reverse of a convolution layer) and in this way the representation is "upsampled back" to the original resolution. We implemented also skip connections between the up-convolution part and the down-convolution part to increase the convergence speed and improve the segmentation performance.

The structure of the FCNN is represented in Figure 3 and it is inspired by U-net architecture [10]. We mimic this architecture because of its success in the semantic segmentation task altogether. The model is fully convolutional and even in the upsampling portion there are learnable parameters thanks to the implementation of Conv2D Transpose (see [1]), as suggested in [8].

4.2. Xception Transfered and Fine Tuned (XTFT)

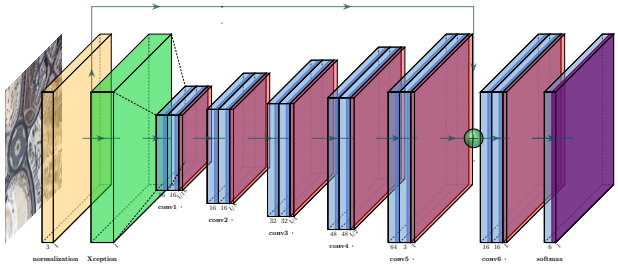


Figure 4: The architecture of the XTFT model. First of all, in yellow we have the normalization layer, followed by the Xception base model in green. In light blue are represented the Conv2DTranspose modules. Their activation functions are in the right side of the box: in blue the ReLu and in purple the softmax. At the end of each conv block, Batch Normalization (in red) is performed. The Residual layer is represented with the green sphere with a "+" sign on it.

The state of the art approach of transfer learning for Semantic Segmentation provides for a classification network to be decapitated, by removing the final classifier layer and replacing it with an architecture that gives an output of the same size as the input. The training of the net should be performed keeping fixed the base model that accounts for the feature extraction part of the network. Finally, it is also possible to fine-tune the network by unfreezing the weights of the base model for a few final epochs with a lower learning rate, to prevent overfitting.

The classifier of our choice is Xception [3], which outperforms Inception V3 on the ImageNet classification task while keeping the same number of parameters. It is adapted for our input size and we load the weights of its version pretrained on the ImageNet database, as it is beneficial compared to the random initialization of weights, as stated in literature [14]. For the decoder, we use a deconvolutional neural network with a single residual module, as shown in Figure 4 . With this approach we aim at exploiting existing results, as in filters and low-level representations already learnt with a thorough training by Xception, in order to gain both in terms of generalization power and feature representation accuracy.

We use multiple conv. blocks with small kernel to have fewer parameters than learning large kernels at once. Through upsampling, these filters manage to have an effective window size that goes from coarse-grained to finely resolute.

4.3. Random Forest (RF)

One of the most powerful and widely applied ideas in modern machine learning, is the use of ensemble methods that combine predictions from multiple, often weak, statistical models to improve predictive performance. Facing the problem of classifying every pixel of an image, we tried to use a Random Forest to label each single element thanks to this ensemble technique. The classifiers are decision trees where each inner node exploits one or more features to decide in which branch to descend. This tree uses a series of questions to hierarchically partition the data: each branch consists of a question that splits the data into smaller subsets, with the leaves (end points) of the tree corresponding to the ultimate partitions of the data. Each leaf is a class. As F. Schroff et al. highlight in their paper [6], this method can lead to good results for class-based pixel-wise segmentation of images. It is robust to noisy and variant data because of the combination of multiple trees with different features and splitting criteria. Moreover, it is computationally less complex than typical neural networks since in the inference procedure each input data is processed using only a small portion of the forest. The random forest framework is usually selected to achieve good efficiency in a short amount of time (i.e. for real-time processing).

Each decision tree in the random forest is composed of a root node, splitting nodes, and leaf nodes (as we can see in Figure 5). Given the input at a root node, the input data is classified to a child node based on the splitting criteria $f_n(\cdot) \lessgtr \theta$ where $f_n(\cdot)$ extracts a learned feature at node n. The classification to a descendant node is terminated when the input data reaches a leaf node. At the leaf node, it is learned the conditional probability $p(c|f_n(x)) \lessgtr \theta$ for every n and for every class c. Hence, in the training stage, the random forest learns the splitting criteria at each splitting node n and a conditional probability distribution for every n until the leaf node of being each class c at each leaf node. Given the trained random forest, each data point x on an image X is classified to child nodes using each tree until it reaches a leaf node. The chosen parameters for our Random Forest are reported in Table 1.

n estimators	**max depth**	**max leaf nodes**
5	25	6
max features	**min samples split**	**min samples leaf**
'auto'	0.1	0.1

Table 1: Settings of Random Forest Classifier

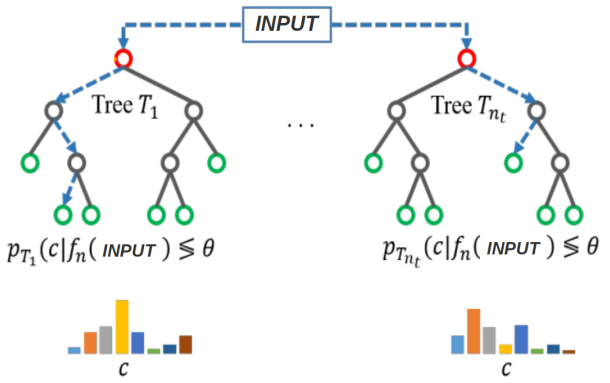


Figure 5: Red, grey, and green circles denote root nodes, split nodes, and leaf nodes, respectively. At leaf nodes, the forest estimates and uses the conditional probability of being each class given the specific leaf node [9]

5. Experiment

In order to provide a fair comparison, each model is trained in the same environment (the one offered by Kaggle notebook) which has the following specifics:
- RAM: 13 GB
- GPU model name: NVidia K80
- CPU model: Intel(R) Xeon(R). 2.20GHz, 2 cores (2 threads per core)

Moreover, the performances of every model are evaluated using a 4-fold cross-validation. Since the dataset is composed by 72 images, every fold has a training set of 54 non pre processed images and a validation set of 18 non pre processed images; their composition is the same for every model. Furthermore, the pre-processing is done every time with the same random seed, in this way the SCC will provide the same validation and training sets of 256x256 images for every model. The augmentation described in section **4.1** is performed on the training set depending on the occupied RAM.

All models are trained through backpropagation of the categorical crossentropy pixel-wise loss and their performances are reported in Table 2.

Note: we do not take into account class imbalance. As in [8] it is declared unnecessary, we do not regret this choice.

5.1. FCNN results

As evident from the results reported in Table 2, FCNN is the best model out of the three because it reaches both accuracy and IoU scores compatible with XTFT, but it has fewer parameters and requires a shorter time interval for the training. This model takes great advantage of not using dense layers; indeed the same architecture with a bottleneck made of a dense layer was attempted, but it greatly increased the number of trainable parameters without any improvement in performance.

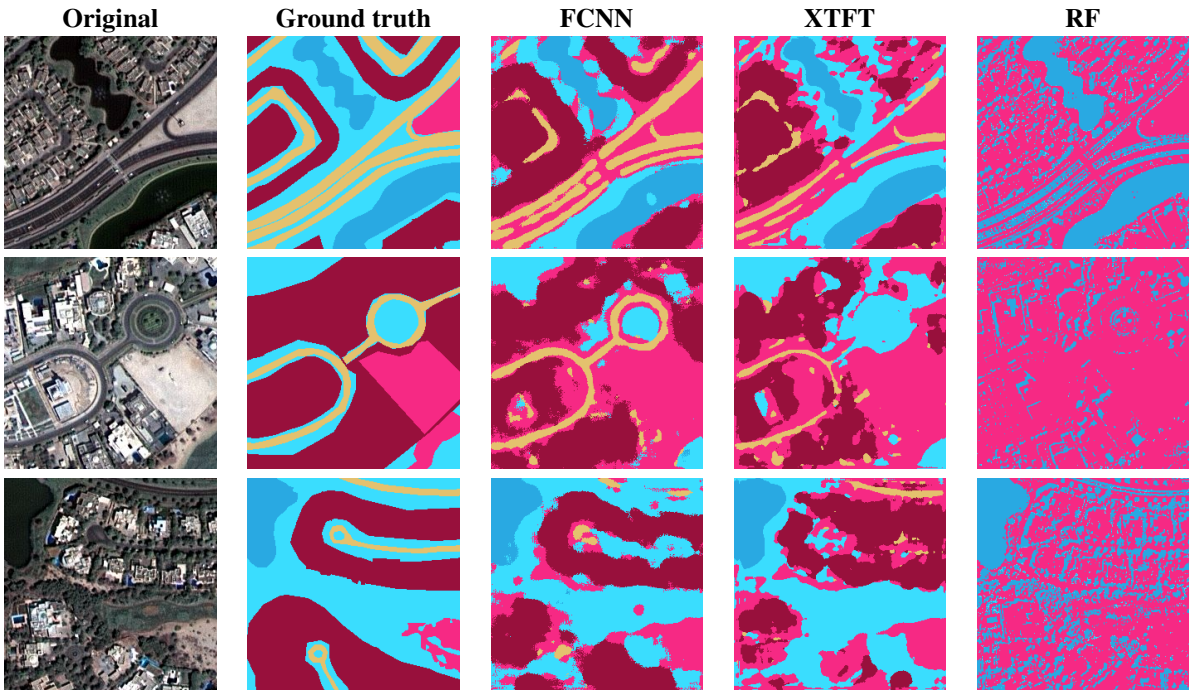


Figure 6: Original and prediction for some cropped test images. The predictions of FCNN are in some case more accurate than the mask itself (1st image, the land surrounding the street, not vegetation as in the ground truth). Streets are well recognized even if thin when edges are neat (2nd image), but not when interrupted by vegetation or shadows (3rd image). The predictions of XTFT are similar and sometimes less accurate than FCNN. RF masks instead present very neat edges, but lack in correctly categorizing less frequent classes, as streets, which here are totally misclassified as vegetation.

Model $\rightarrow$	**FCNN**	**XTFT**	**RF**
N_P	107k	22M	-
Augmentation	yes	yes	no
M (GB)	7.1	9.8	11.2
t (min)	50	112	16
Cat Acc (%)	81 ± 3	79 ± 3	64 ± 1
IoU (%)	60 ± 6	60 ± 7	25 ± 1
Precision (%)	72 ± 4	74 ± 4	29 ± 2
Recall (%)	78 ± 5	75 ± 5	35 ± 1

Table 2: Results of 4 fold cross validation. N_P = number of trainable parameters (the parameters of the random forest are the variables and thresholds used to split each node learned during training); **M** = occupied RAM during one fold (max is 13 GB); **t** = run time of 4-fold cross validation. All the following metrics are obtained from the average of 4-fold cross validation: **Cat Acc** = mean top1 categorical accuracy; **IoU** = mean of correctly classified pixels over total for each class; **Precision** = true positives over all predicted positives; **Recall** = true positives over all real positives. Each error is computed as standard deviation.

Increasing the number of filters per layer does not lead to better results and just increases the computational effort. Moreover, the augmentation comes with negligible improvement in performance. As we will see, these behaviours may be due to having already reached the optimal performance of this architecture on the dataset.

5.2. XTFT results

The training of this architecture was highly demanding in terms of computational time, with respect to the other two architectures, as one can see in Table 2. This is due to the number of parameters, two orders of magnitude higher than the FCNN. The number of parameters though does not seem to be beneficial for this simple task: the performances are overall compatible with the ones of FCNN.

Many attempts and tunings of hyperparameters have been performed for the decoder, which led to the current architecture. In particular, the variables taken into consideration are: kernel size, number of convolutional blocks, number of filters per layer, stride dimension, and number of residual modules. In all cases, an increase or decrease with respect to the current parameters resulted in a drop in performance. It has to be noted that, due to the high number of parameters of the Xception architecture, small changes in the decoder cannot bring to significant changes in performance, as the classifier stays the main component in the

network.

Allowing the decoder architecture to have a large number of filters has been the most impacting choice as is.

Another consideration that we make is that more parameters need more data to be properly inferred: this architecture rapidly reaches the peak of the accuracy metric (around epoch 15th), and easily falls in overfitting. As for a fact, just a couple of epochs were enough for the fine tuning, as the training loss dropped while the validation increased almost immediately.

Finally, a possible reason for the lack of accuracy with respect to FCNN may be the limited skip connection layers that could be inserted due to the architecture of Xception. We conclude that the benefits of transfer learning can not single-handedly overcome the noticeable advantages of connections between multiple encoding stages and corresponding decoding levels.

5.3. RF results

We trained five trees using the proposed framework and the chosen parameters. In Figure 6, we can notice that the predicted masks of the random forest present only two labels. This may be due to the fact that these two classes are the most frequent in the dataset.

We have noticed that predicting using a random forest is a computationally and temporally efficient process, but training large forests with a huge amount of data is computationally expensive. It is especially time and memory consuming during the training procedure since each node needs to be optimized depending on the previous nodes (considering the maximum depth of 25). Hence, training this type of model on the whole augmented dataset would be impossible. For this reason, the RF is trained using only the images cropped in more parts, without the augmentation.

A notable aspect is that the Random Forest has a time advantage since the computational complexity is controlled by adjusting the number of trees. In particular, from Table 2 we notice that the processing time is about 3 times smaller than FCNN and about 7 times smaller than XTFT. However, its performances are limited and not competitive compared to the other models. This is not surprising since the random forest classifies exploiting only the three color channels information of the single pixel.

An other possible explanation of this difference in performances comes from the paper of B. Kang and T.Q. Nguyen [9]. Firstly, the deep neural networks can describe nonlinear representations by using nonlinear activation functions, while the representation of the RF is a linear representation of the input image. Moreover, the other proposed methods are trained by using back-propagation, while the RF is trained in order from a parent node to child nodes. In this way, when all nodes are trained, it has not any feedback to improve its performance. A way to improve RF performance and obtain more competitive results is to combine it with a deep-based architecture [13] or to feed it with more informative features, such as the colors of nearest neighbour pixels.

5.4. Final considerations

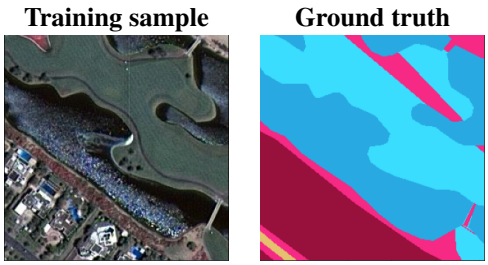


Figure 7: An example where the ground truth image is not accurate. We can see that part of the "main green area" (cyan in the ground truth) is wrongly classified as land (pink in the ground truth). Moreover all the vegetation in the lower left corner is consider as land and buildings (red in the ground truth).

As a last note, we consider aspects and challenges common to all architectures. From table 2 it is possible to notice that the performances of two fairly different networks, as FCNN and XTFT are compatible. We find the explanation to be in the upper limit of the accuracy due to the mask resolution, as in many cases masks are not pixel-wise accurate, as we can see in Figure 7.

For this reason, we state that the performances we obtain are biased by the quality of the masks. To obtain further improvements, more accurate ground-truths would be needed.

Finally, we underline how the skip connection module is very efficient for this task because it reduces the convergence speed and increases the performances as we expected.

6. Conclusions

We have presented the implementation of three different algorithms to perform semantic segmentation of aerial images. We have seen how the FCNN and the fine-tuned Xception + decoder lead to similar performances over the small considered dataset, but the FCNN is less expensive in terms of memory usage and training time. The random forest is the best model out of the three in terms of training time, but its performances are not competitive with the other models. Future work could include a comparative study of those algorithms on publicly available datasets. The study of more advanced models as an attention-based approach would be an interesting option to explore in order to improve accuracy and to obtain a faster training. Another possible model to study is the combination of different CNNs in a multiscale approach to avoid the loss of the spatial information.

References

[1] Keras documentation. https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2DTranspose.

[2] Grady jensen Baris Kayalibay and Patrick van der Smagt. Cnn-based segmentation of medical imaging data. We provide the code we used in our experiments on https://github.com/BRML/CNNbasedMedicalSegmentation.

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions, 2017.

[4] Mennatullah Siam et al. Deep semantic segmentation for automated driving: Taxonomy, roadmap and challenges. 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC).

[5] Shervin Minaee et al. Image segmentation using deep learning: A survey, 2015.

[6] A. Zisserman1 F. Schroff, A. Criminisi. Object class segmentation using random forests, 2008. https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/Criminisi_bmvc2008.pdf.

[7] Ned Horning. Random forests: An algorithm for image classification and generation of continuous fields data sets, 2010. http://wgrass.media.osaka-cu.ac.jp/gisideas10/papers/04aa1f4a8beb619e7fe711c29b7b.pdf.

[8] Evan Shelhamer Jonathan Long and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2015. UC Berkeley.

[9] B. Kang and T.Q. Nguyen. Random forest with learned representations for semantic segmentation, 2019. https://arxiv.org/pdf/1901.07828.pdf.

[10] Philipp Fischer Olaf Ronneberger and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. Computer Science Department and BIOSS Centre for Biological Signalling Studies, University of Freiburg, Germany ronneber@informatik.uni-freiburg.de, WWW home page: http://lmb.informatik.uni-freiburg.de/ paper: https://arxiv.org/pdf/1505.04597.pdf.

[11] R. Szeliski. Computer vision: algorithms and applications, 2010. Springer, Science & Business Media.

[12] Michael Wurm, Thomas Stark, Xiao Xiang Zhu, Matthias Weigand, and Hannes Taubenböck. Semantic segmentation of slums in satellite images using transfer learning on fully convolutional neural networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 150:59–69, 2019.

[13] Tom Drummond Yan Zuo. Fast residual forests: Rapid ensemble learning for semantic segmentation, 2017. http://proceedings.mlr.press/v78/zuo17a/zuo17a.pdf.

[14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks?, 2014.