Fully Residual Neural Networks for Aerial Image Segmentation

Dinh Viet Sang
Hanoi University of Science
and Technology
Email: sangdv@soict.hust.edu.vn

Nguyen Duc Minh
FPT Technology Research Institute
FPT University
Email: minhnd33@fpt.com.vn

***Abstract*—Semantic segmentation from aerial imagery is one of the most essential tasks in the field of remote sensing with various potential applications ranging from map creation to intelligence service. One of the most challenging factor of these tasks is the very heterogeneous appearance of artificial objects like buildings, cars and natural entities such as trees, low vegetation in very high-resolution digital images. In this paper, we propose a very deep feature learning approach with fully convolutional networks (FCN) for semantic segmentation. Particularly, we introduce one more skip connection while upsampling the feature maps in FCN architecture using the famous ResNet101 encoding network. The benchmark results from ISPRS which our proposed method attains much better accuracies in comparison with other recent state-of-the-art methods on the well-known Vaihingen and Potsdam datasets from ISPRS Working Group II/2.**

I. Introduction

In the last 20 years, a huge amount of effort has been spent on solving semantic segmentation problem in aerial photography. However, the task is still considered not being solved yet. Many advanced machine learning techniques have been applied to train and learn efficient models and many promising results have been made in the 2D Semantic Labeling Contest. Furthermore, with the rise of generally available computational resources, a large number of deep learning models which can learn so much useful insight and pattern from raw data without providing the models with much human domain knowledge to produce even better results.

Semantic segmentation in aerial images is a key task in remote sensing with many applications. For instance, data acquired by airborne sensors can be applied to urban planning to identify different urban areas such as vegetation, buildings, transportation infrastructure or even traffic flow to help design and develop a town or city with a long-term vision. However, to accurately attain a correctly segmented images from raw data from aerial cameras is a very severe and painful task which requires so much domain knowledge in both remote sensing and computer science. To our knowledge, no fully automated methods have been devised to solve this very core problem yet because different data from different regions for different tasks needs different approaches. Since 2014, a 2D semantic labelling contest has been held by ISPRS WG III/4 is meant to resolve is issue. The first place of this competition [1] to date on both Vaihingen and Potsdam datasets is HUSTW team with 91.6% overall accuracy on the hidden test set.

In this paper, we propose a powerful deep learning approach which can unleash the segmentation potential of ResNet which has won the 1st place on the ILSVRC 2015 classification task. Particularly, we combine ResNet101 with FCN for solving the semantic segmentation task. Moreover, we experiment two different FCN architectures with 2 and 3 skip connections and both of them demonstrate the same overall accuracy on Vaihingen dataset and achieve the 4th position globally with 91% accuracy on the hidden test set by ISPRS. Similarly, our 3-skip model also obtain 91.1% accuracy on the hidden test set from Potsdam dataset with the 3rd position globally.

The rest of the paper is organized as follows. In section 2, we review related work in deep learning and semantic segmentation. In section 3, we briefly introduce our proposed approach. Finally, in section 4, we show the experimental results on both Vaihingen and Potsdam dataset and compare our approach with recent state-of-the-art methods.

II. Related work

Classical approaches for semantic segmentation from aerial images are often based on many extracted features and building typical classification models such as random forest and conditional random field. Despite the fact that these and combination of the two can perform with such a small amount of time, but many deep learning models can achieve better overall accuracies.

In recent years, deep learning has proved its potential in many field, especially in computer vision. These neural network based models can learn much of the hand-engineering features automatically as long as provided with enough good labelled data. The first known convolutional neural network is LeNet used to solve the classification task of ten handwritten digits from 0 to 9. However, until 2012, AlextNet came to public with the winner of ImageNet ILSVRC challenge. Since then, many typical recognition CNNs has been proposed and performed impressively in one of the most pretigious computer vision competitions - the annual ImageNet Large Scale Visual Recognition Challenge (ILSVRC). These nets tend to be both wider and deeper with the heavy use of batch normalisation

after every weight layer to improve the optimization ability and curb overfitting problem of these deep learning models.

Those classification networks are not only used in classification tasks. Each of them can also act as an image encoder. In [4], Long propose a new semantic segmentation architecture, which is called fully convolutional neural network (FCN). To encode a raw image as the input data, the FCN encodes the raw image into a set of feature maps which retain only semantic information about objects and textures of the input image using any visual recognition network without any fully-connected layers among it.

In 2D semantic labelling contest on ISPRS, the top teams use many variations of fully conditional networks that set the state-of-the-are on both Vaihingen and Potsdam datasets. In [**?**] and [**?**], the author successfully apply a FCN using VGG19 to 2D semantic segmentation problem and achieve better overall accuracy than previous methods.

III. Our proposed approach

A. Deep residual network

Deep residual networks (ResNets) [2] achieved 1st-place winning entries in all five main tracks of the ImageNet and COCO 2015 competitions, which covered image classification, object detection and semantic segmentation. ResNet architectures are much deeper than the preceeded ones which are VGG architectures (with both 16 and 19 weight layers). With the heavy use of batch normalization and the introduction of residual blocks periodically stacked throughout the network.

A typical ResNet consists of multiple *residual blocks* and heavy application of *batch normalization* between every convolutional layer and ReLU layer.

Residual blocks: Instead of directly approximate $H(x)$ underlying mapping by stacking a few more intermediate layers, another mapping of $F(x) = H(x) - x$ is fit by the stacked nonlinear layers. The original function is now recast into $F(x) + x$. Therefore, it is easier to optimize the residual mapping than to optimize the original.

The fundamental building block of ResNet architectures in Fig. 6 has been reformulated under the assumption that the optimal function a block is trying to model is closer to an identity mapping than to a zero mapping, and that it should be easier to find the perturbations with reference to an identity mapping than to a zero mapping. This simplifies the optimization of the network at almost no cost. Subsequent blocks in the model are thus responsible for fine-tuning the output of a previous block, instead of having to generate the desired output from scratch.

Batch normalization:

Batch normalization draws its strength from making normalization a part of the model architecture and performing the normalization for each training mini-batch. Batch Normalization allows us to use much higher learning rates and be less careful about initialization. It also acts as a regularizer, in some cases eliminating the need for Dropout.

Batch normalization is similar to Dropout in the sense that it multiplies each hidden unit by a random value at each step of training. In this case, the random value is the standard deviation of all the hidden units in the mini-batch. Because different examples are randomly chosen for inclusion in the mini-batch at each step, the standard deviation randomly fluctuates. Batch normalization also subtracts a random value (the mean of the mini-batch) from each hidden unit at each step.

Both of these sources of noise mean that every layer has to learn to be robust to a lot of variation in its input, just like with Dropout.

B. Fully convolutional network

Fully convolutional networks (FCN) popularized CNN architectures for dense predictions without any fully-connected layers. This allowed segmentation maps to be generated for image of any size and was also much faster compared to the patch classification approach. Almost all the subsequent state-of-the-art approaches on semantic segmentation adopted this paradigm.

A FCN architecture can be broadly thought of as an encoder network followed by a decoder network:

- The **encoder** is usually a pre-trained classification network like VGG/ResNet followed by a decoder network;
- The task of the **decoder** is to semantically project the discriminative features (lower resolution) learned by the encoder onto the pixel space (higher resolution) to get a dense classification;

One issue in this specific FCN is that by propagating through several alternated convolutional and pooling layers, the resolution of the output feature maps is downsampled. Therefore, the direct predictions of FCN are typically in low resolution, resulting in relatively fuzzy object boundaries.

C. Cross-entropy loss

Cross-entropy loss is widely used for training deep neural networks. Suppose that N is the number of pixels in the training data; K is the number of classes; l_i is the correct class label of i-th sample; $\mathbf{y}_i$ is the one-hot encoding of the correct class label of i-th sample ($\mathbf{y}_i(l_i) = 1$); and $\widehat{\mathbf{y}}_i$ is the probability distribution of i-th sample over classes. More importantly, this semantic segmentation is mathematically a classification problem over every pixel which each pixel is classified into K classes. The cross-entropy loss is defined as follows:

$$L_S = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K}\mathbf{y}_i(j)log(\widehat{\mathbf{y}}_i(j)), \tag{1}$$

where $\mathbf{y}_i(j) \in \{0, 1\}$ indicates whether j is the correct label of i-th sample; $\widehat{\mathbf{y}}_i(j) \in [0, 1]$ expresses the probability that j is the correct label of i-th sample.

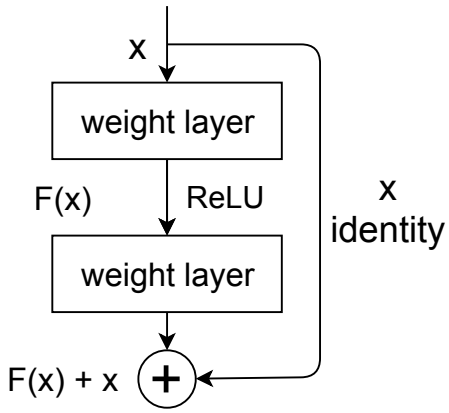

Fig. 1: A residual block - the fundamental building block of residual networks [2].

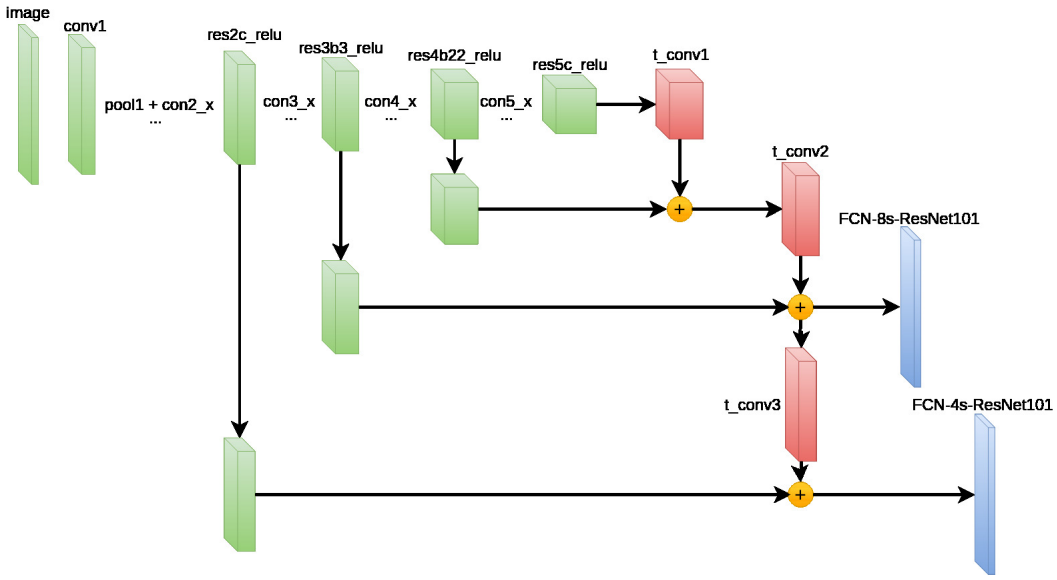

Fig. 2: FCN-ResNet.

D. Our Fully Convolutional Networks with ResNet101

In this work, we propose 2 different FCN-ResNet101 architectures to deal with the semantic segmentation problem on very high resolution data, namely FCN-ResNet101-8s and FCN-ResNet101-4s. The encoder for both of them is a pre-trained ResNet101 network on the ImageNet dataset for 1000-class classification task. However, two different decoders are experimented for the semantic segmentation task.

We first add a 1×1 convolution layer on top of `res5c_relu` layer to produce additional class predictions. Then we upsamples stride 2 predictions to get double-sized feature maps by performing transpose convolution with the receptive field of 4×4. To combine predictions from both upsampled volumes and the `res4b22_relu` layer by implementing a skip connection to fuse two of them which is just simply an addition operation. While to build FCN-ResNet101-8s model just one more $2\times$ upsampling and skip connection are applied before the stride 8 predictions are upsampled back to the image, FCN-ResNet101-4s model requires upto two more same $2\times$ upsampling and skip connection are applied before the stride 4 predictions are upsampled back to the image. The receptive field of the stride 8 predictions and the stride 4 predictions are 16×16 and 8×8.

For both models, we introduce Dropout to reduce the effect of overfitting by applying only `res5c_relu` layer due to very high channel dimension.

IV. Experiments and Evaluation

A. Datasets

Vaihingen dataset and Potsdam dataset are provided on the 2D Semantic Labelling Contest of ISPRS Working Group III/4 and both of them cover different urban scenes in Germany. While Vaihingen is a quite small village with mostly detached homes and multi-storey buildings, Potsdam shows a typical historic city with large building blocks, narrow streets and dense settlement structures.

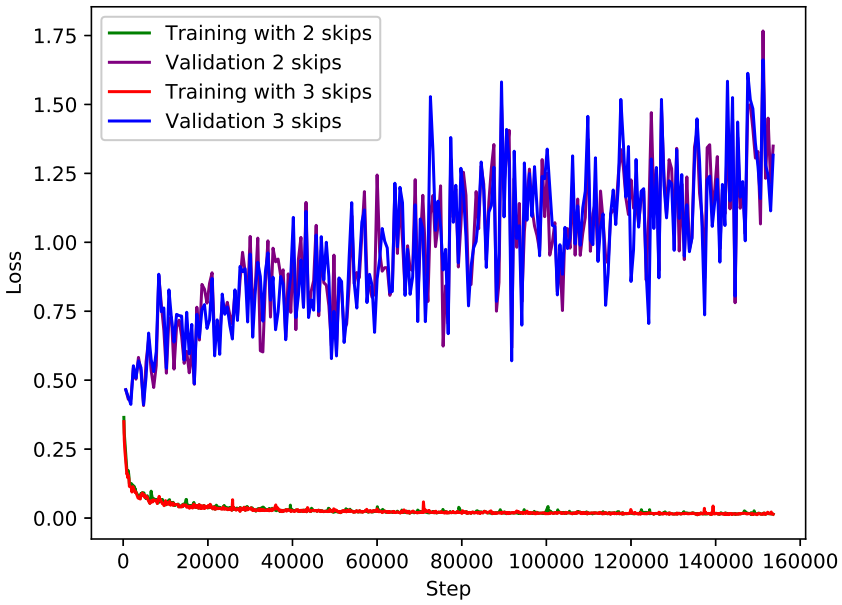

Fig. 3: Losses of 4s and 8s.

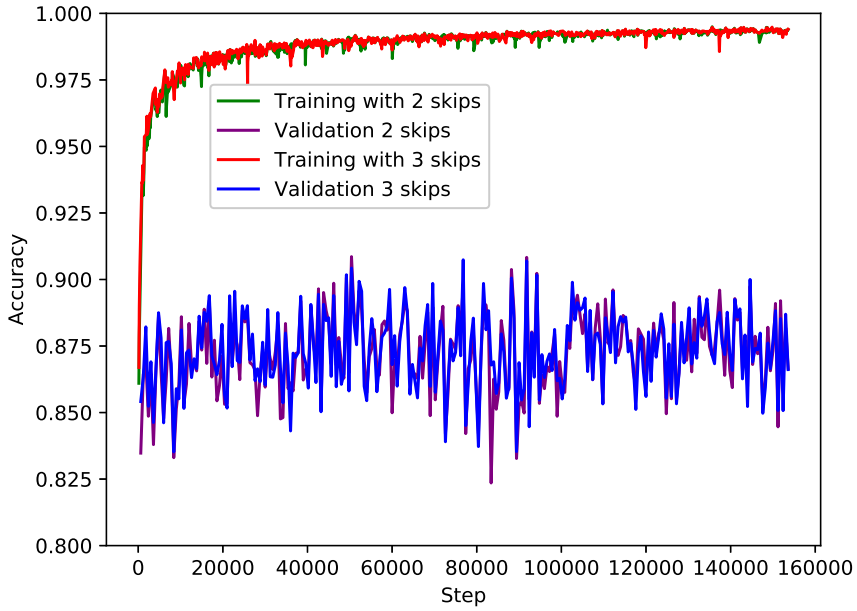

Fig. 4: Accuracies of 4s and 8s.

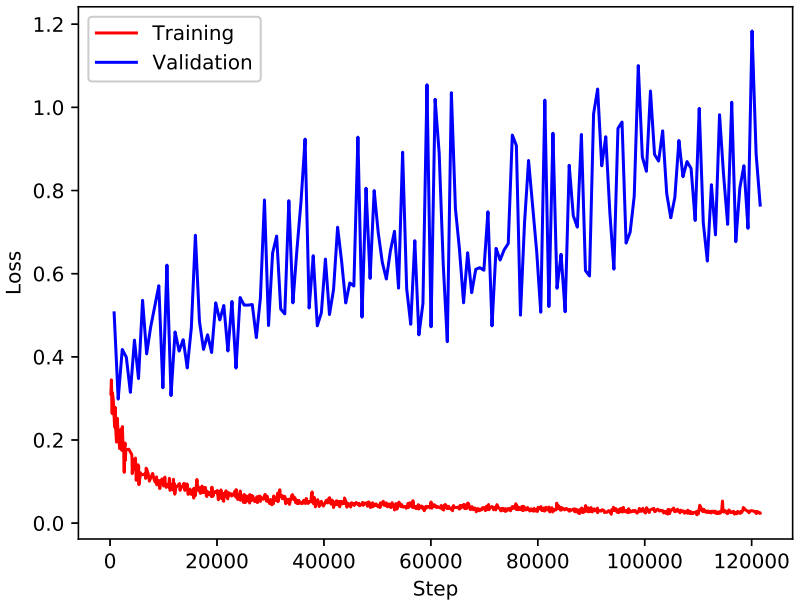

Fig. 5: FCN-4s-ResNet101_Potsdam_loss.

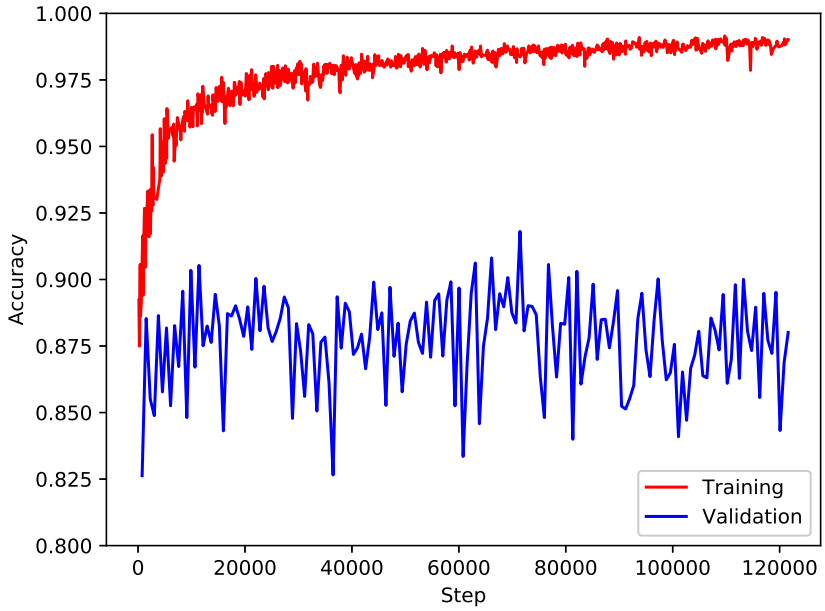

Fig. 6: FCN-4s-ResNet101_Potsdam_acc.

Labelled ground truth is provided for only one part of the data. The ground truth of the remaining scenes will remain unreleased and stays with the benchmark test organizers to be used for evaluation of submitted results. While Vaihingen dataset contains 33 patches of different sizes and 16 of them are provided with ground truth, Potsdam dataset consists of 38 patches which all of them are of the same size and 24 of them are provided with ground truth, 6000×6000. Each patch from both datasets consists of a true orthophoto (TOP) extracted from a larger TOP mosaic.

We use 5-channel data when training on Vaihingen dataset (IR, R, G, nDSM and DSM) and 6-channel data for Potsdam dataset (R, G, B, I, R, nDSM and DSM). nDSM data from Potsdam dataset is generated by LAStools and supplied by ISPRS.

Each spatial position on the ground truth is labelled by one of six urban objects: impervious surfaces, building, low vegetation, tree, car, clutter/background (Fig.**??**).

B. Data augmentation

Due to very high resolution data and small number of patches in two datasets, we use data augmentation techniques to generate more new data with out resizing given images for the training phase. These techniques help us to reduce information loss and, hence, to train a more robust network.

We used 3 main methods for data augmentation as follows:

- Randomly crop: Every TOP patch is randomly cropped for 4096 times with the spatial dimension of 224×244 using uniform distribution;
- Entirely flip an image from left to right and upside down;

In practice, we find that applying augmentation techniques greatly improves the capacity of the model.

C. Implementation details

Our experiments are conducted by using Python 3 with Tensorflow 1.4 framework on a deep learning server with the following specifications: Intel Xeon CPU E5-2699 v4 88-core 2.20GHz Processor, Ubuntu Server 16.04.3 LTS 64-bit Operating System, 756GB of RAM and GPU NVIDIA GeForce GTX1080i with 11GB of GPU memory and compute capability 6.1.

Preparing dataset: Since the images in both Vaihingen and Potsdam dataset are cropped already, the preprocessing step is quite simple. Firstly, we compute the mean image over the training set. Each pixel in the mean image is computed from the average of all corresponding pixels (i.e. with the same coordinates) across all training images. For each training image, we then subtract from each pixel its mean value, and then set the standard deviation of each pixel over all training images to 1.

Optimization: We train our models by Adam optimization algorithm with the learning rate of 10^{-4} [3]. We set the batch size equal to 32 when training on Vaihingen dataset and 64 for Potsdam dataset. Dropout is set at the rate of 50%. All parameters in every batch normalization layer are also retrained with Exponential Moving Average and decay 0.9.

Fine-tuning phase: All weights of two models are initialized with pre-trained weight trained on ImageNet dataset including means, variances, offsets, scales for batch normalization layers and also mean pixel values for the first 3 input channels. Those which are additional layers such as the 2 or 3 last input layers (when using images with more than 3 channels), 1×1 convolution layer on top of `res5c_relu` and filters for transpose convolution are initialized randomly with a normal distribution which has mean and standard deviation of zero and 0.02 respectively. Furthermore, mean pixel value for DSM and nDSM are computed separately from pretrained weights, using data from diven datasets. Finally, we fine-tune all layers by backpropagation through the whole net.

Testing phase: In the testing phase, we ensemble different checkpoints obtained during the training phases to increase the robustness and the power of the learned classifier. In this paper, we simply keep the last fifteen checkpoints corresponding to the last fifteen training epochs for inference.

D. Experimental results

V. Conclusion

In this paper, we propose an effective approach to ...
In the future, we would like to exploit ...

References

[1] 2d semantic labeling contest.

[2] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[3] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[4] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.