INVESTIGATION ON THE SEGNET DEEP NEURAL NETWORK FOR AERIAL IMAGE SEGMENTATION AND CLASSIFICATION

Nima A. Gard[1], Zotlan Koppanyi[1], Charles K. Toth[1]

[1]OSU Department of Civil, Environmental and Geodetic Engineering
470 Hitchcock Hall, 2070 Neil Avenue, Columbus, OH 43210
ajamgard.1@osu.edu, koppanyi.1@osu.edu, toth.2@osu.edu

ABSTRACT

By now, modern photogrammetric workflows are fully automatized from image orientation to dense point cloud, mesh and DSM generation. However, extracting the semantic information of the images is still a challenge. Segmentation and segment classification are labor intensive, and therefore, the automation of this step of the photogrammetric workflow is of interest for researchers and practitioners. Recently, deep learning, in particular, convolutional neural networks (CNN) have become one of the promising approaches to tackle this problem. CNN has already achieved high accuracy for various image classification, segmentation and labeling tasks, however, current efforts are typically focused on segmentation of close-range, mobile or indoor imagery. The goal of this paper is to investigate the performance of an encoder-decoder convolutional neural network, called SegNet, on aerial images for image segmentation and classification. The encoder-decoder network structure allows for pixel-wise segmentation of an image, where all pixels are annotated with a label by the network. The performance analysis is conducted on the ISPRS's aerial benchmark dataset, and includes the investigation of various hyper parameters, such as learning rate, input image size, the effect of training set size and pretrained weights on the segmentation and classification accuracy.

Key words: point cloud reconstruction, motion estimation, object tracking, performance analysis

INTRODUCTION

In photogrammetric applications, the goal of image segmentation and classification is the task to annotate all pixels with land use categories, such as forests, buildings, infrastructures, etc. This task is a labor intensive and expensive, therefore the automatization of this process has been a long studied problem in photogrammetry. Many techniques have been developed in the last decades including simpler threshold based, edge based and region based methods, and more complex methods, such as Markov random fields, fuzzy models, watershed models, and neural networks (Dey, Zhang, and Zhong 2010; Pal and Pal 1993). The developments of deep learning and deep neural networks are boosted by the growing computation power and the availability of large datasets required for training these models. Recent advances in deep learning further pushed the accuracy of image segmentation methods.

The first deep neural networks for image segmentation and classification are developed by the computer vision community. These algorithms are used on various types of indoor and outdoor images. Aerial images taken from UAS or airborne platforms are different than the images investigated in computer vision studies. First, in many cases, the objects, such as buildings, present as small patches on the images and the patterns of these patches are significantly different than other objects in close-range indoor or outdoor images. Second, the image resolution is higher than what it is typical in computer vision applications that requires multi-scale approaches to reduce the problem space. Third, airborne sensors might capture larger spectral bandwidth, and thus, the images might have multiple channels or have different channels, for instance, infra-red instead of red, as opposed to single RGB images. And finally, the number of object classes in photogrammetric applications is typically smaller. For these reasons, the investigation of deep neural models on aerial images are required.

In this paper, we investigate the SegNet deep neural network (Badrinarayanan, Kendall, and Cipolla 2015). SegNet is one of the most successful and relatively simple deep model for image segmentation and classification. In general, deep neural network has several hyper-parameters, such as mini-batch size, learning rate, weight decay, that has relevant impact on the classification accuracy. The goal of this paper is to investigate the impact of the SegNet's hyper-parameters on the classification accuracy using airborne images, and ultimately, to provide guideline for appropriately choose them.

THEORY

SegNet

SegNet is a deep convolutional neural network (CNN) developed for image segmentation and classification. Deep learning libraries define the tensor data structure, which is basically a multi-dimension matrix. For CNNs, tensors typically have three dimensions: width, height and depth. The input of the investigated SegNet network is a RGB image represented as a 224x224x3 tensor. The output of the network is a tensor which has the same width and height as the input tensor, i.e. 224x224, and the depth channels indicate the scores for each label class. The label with the highest score along the channel dimension is chosen as the label for a pixel.

The SegNet network consists of two parts. The first part is a VGG-16 network (Simonyan and Zisserman 2014), see Figure 1. VGG-16 is developed for image classification problem, where the task is to label the content of the whole image into subcategories. VGG-16 applies a series of convolution, ReLU, batch normalization, max-pooling, and dropout layers. The height and width dimensions decrease while the depth dimension increases as the network gets deeper. The deeper depth channels are interpreted as different features of the scene. The tail of the VGG-16 network is a classifier head that transforms 512 features into label scores utilizing a fully connected and SoftMax layers.

As opposed to VGG-16, SegNet annotates all pixels, and thus, the output of the network is not a single label for the entire image, but labels for all pixels. Therefore, SegNet replaces the classifier tail of VGG-16 with a “mirrored” VGG-16 network architecture, see Figure 1. The original VGG-16 network is called encoder, and the “mirrored” counterpart is the decoder. In this decoder network, the max-pooling operator is replaced by max-unpooling. Unpooling is the inverse operator of pooling that increases the height and width of the layer's input tensor and decreases the number of depth channels. Thus, this allows the network to transform the features produced by the VGG-16 network into the original image resolution. One of the main advantage of using VGG-16 is that large datasets are available for image classification, and therefore, VGG-16 pre-trained model are more accurate. These pre-trained model, then, can be used in SegNet via transfer learning. Finally, the output of the SegNet is the label scores for all pixels of the image.

In general, it is noteworthy that there exists other pixel-level labeling networks, such as U-Net (Ronneberger, Fischer, and Brox 2015), or DeepLab (Chen et al. 2018), that achieve better accuracy on notable benchmark datasets, however its relatively simple network structure allows for easy implementation. Note that almost all popular deep learning libraries, such as TensorFlow or PyTorch, provide pre-defined SegNet architecture for the user.

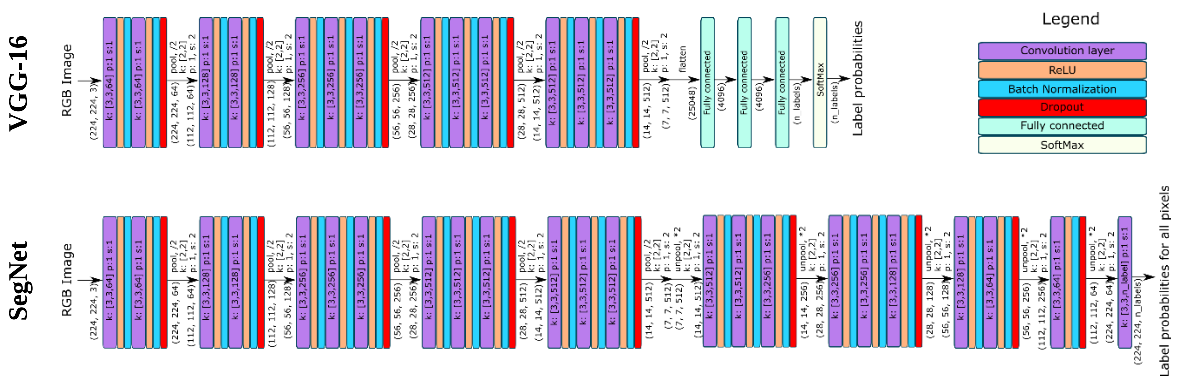


Figure 1 The network architecture of VGG-16 and SegNet.

METHODS

The performance of a neural network is directly dependent upon its hyper-parameters such as number of layers, activation functions, learning rate, loss function, and etc. Finding the optimal parameters for a network is a tedious task; therefore, we study several parameter tuning techniques including hand-tuning, grid search, randomization, and sequential model-based global optimization.

Hand-Tuning

Arguably, the simplest hyper-parameter tuning technique in neural networks is hand-tuning. Assume that the only influential hyperparameter is the learning rate, and all other parameters have a negligible effect on the performance of the network. Intuitively, one can start with start with few different learning rate values and finally pick the one that results in the best performance of the network. However, a more effective technique is to observe the behavior of the measuring index for several epochs. Depending on what measuring index is chosen for performance evaluation, change of hyper-parameters, e.g. learning rate, will result in a different behavior/trend in the measuring index. For instance, Fig. 2 illustrates the trend in loss for a low/right/high/very high learning rate value. In hand-tuning technique, plotting the values provides valuable insights about the network. Hand-tuning requires some fundamental knowledge about the response of a neural network to change in hyper-parameters. The more complex the network is, the more difficult it is to picture how a measuring index will change, especially if the chosen measuring index is the hybrid of several other measuring indices such as f1-score. Another drawback in hand-tuning technique is values selected for a training round. For instance, humans might only select round numbers. The implicit bias in selecting values could lead to ignoring the right values for the network.

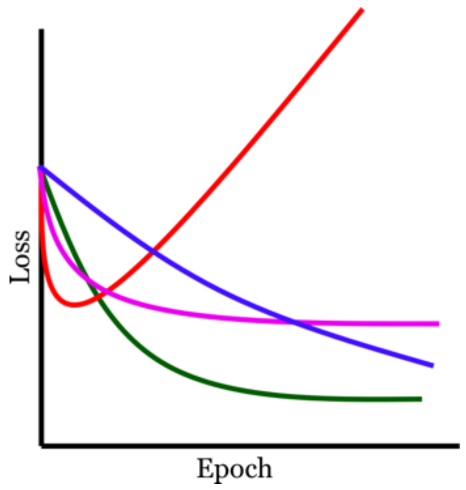


Figure 2 Loss vs. Epoch: Colors (---), (---), (---), and (---) represent the trend of loss over several epochs for a very high, low, high, and appropriate learning rate.

Grid Search

A more elegant parameter tuning technique is to use a systematic way of selecting values within a certain interval. The grid search technique divides the domain of parameters into equal regions. Assume that learning rate values fall between 1e-6 and 1. The grid search technique takes as input, the number of grids, e.g. 11. A naïve gird search divides the given interval into 10 equal subintervals and outputs the following values: 1e-6, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1. All the values between 1e-6 and 0.1 are neglected. A better approach is to uniformly divide the logarithm of the given interval. In such case, values between 1e-6 and 0.1 will also be sampled from.

Once the interval is sampled, the network is trained for all the combination of sampled parameters. For n hyper-parameters, with m sampled values for each, there are m^n possible combinations. Since training deep neural networks could take several hours or days, blindly using all possible combinations of hyper-parameter values will take a long time.

Randomization

A major drawback of using grid search is the approach taken in sampling values for hyper-parameters. Randomly sampling a subset of all the possible combinations reduces the time required for training. Many distributions such as normal distribution can be used for this purpose. However, a subtlety about random sampling is that samples should not be drawn with replacement and should not be close to other values. The former is handled easily by ignoring duplicates. The latter is done through a systematic sampling method such as van der Corput sequence, which is a one-dimensional low discrepancy sequence. As opposed to a uniform distribution, which is unbiased and generates samples all with the same probability, a sample generated by low discrepancy sequence is aware of all the previous generated samples. Thus, the interval is sampled (filled) more uniformly compared to the same interval sampled by a regular uniform distribution. Other examples of low discrepancy sequence are Hammersley set, Halton sequence, and Sobol sequence.

Sequential Model-based Global Optimization

In this work, we use a sequential model-based global optimization approach (SMBO) algorithm (Bergstra et al., 2011) since optimizing the actual problem is very expensive. SMBO is a model-based algorithm that evaluates an approximate function, which is cheaper than the true function. The parameters that optimize the approximate function are proposals used to evaluate the true function. Following the suggestion of Bergstra et al. (Bergstra et al., 2011), we use the criterion of Expected Improvement (EI) as the as optimization objective.

Expected improvement is defined as the expectation of function $f: X \to \mathbb{R}^N$ that $f(x)$ will exceed some threshold τ:

$$EI(x) = \int_{-\infty}^{\infty} \max(\tau - y, 0)\, p(y|x) dy. \tag{1}$$

We use Tree-structed Parzen Estimator (TPE) approach in order to model $p(y|x)$ via $p(x|y)$ and $p(y)$. TPE collects parameters $\{x^{(1)}, \dots, x^{(k)}\}$, then it decides which hyper-parameters it should use in the next iteration. The decision-making process requires a prior distribution to sample hyper-parameters from.

The TPE algorithm categorizes hyper-parameters into two groups. The first group is composed of hyper-parameters that correspond to f exceeding the threshold τ, and the second group which contains all other hyper-parameters. With collecting these observations, we are able to construct a distribution for hyper-parameters that improve EI. Formally, TPE defines $p(x|y)$ as:

$$p(x|y) = \begin{cases} l(x) & if\ y \leq \tau \\ g(x) & if\ y \geq \tau, \end{cases} \tag{2}$$

where $l(x)$ and $g(x)$ are density functions constructed based on parameters resulting in $f(x^{(i)}) = y$ being less than and larger than τ, respectively. We can incorporate the TPE algorithm into Equation 1 as:

$$\begin{aligned} EI(x) &= \int_{-\infty}^{\infty} \max(\tau - y, 0)\, p(y|x) dy = \int_{-\infty}^{\tau} \max(\tau - y, 0)\, p(y|x) dy + \int_{\tau}^{\infty} \max(\tau - y, 0)\, p(y|x) dy \\ &= \int_{-\infty}^{\tau} \max(\tau - y, 0)\, p(y|x) dy = \int_{-\infty}^{\tau} \max(\tau - y, 0) \frac{p(x|y)p(y)}{p(x)} dy \\ &= \int_{-\infty}^{\tau} (\tau - y) \frac{l(x)p(y)}{p(x)} dy = l(x) \int_{-\infty}^{\tau} (\tau - y) \frac{p(y)}{p(x)} dy. \end{aligned} \tag{3}$$

We can further simplify Eq. 3 by denoting $p(y < \tau)$ as λ and substituting $p(x)$ by its definition

$$\begin{aligned} p(x) &= \int_{-\infty}^{\infty} p(x|y)p(y) dy = l(x) \int_{-\infty}^{\tau} p(y) dy + g(x) \int_{\tau}^{\infty} p(y) dy \\ &= l(x)p(y < \tau) + g(x)\big(1 - p(y < \tau)\big) = \lambda l(x) + g(x)(1 - \lambda), \end{aligned} \tag{4}$$

into the following expression:

$$EI(x) = l(x)\int_{-\infty}^{\tau}(\tau - y)\frac{p(y)}{p(x)}dy = \frac{l(x)}{\lambda l(x) + g(x)(1-\lambda)}\left(\tau\int_{-\infty}^{\tau}p(y)dy - \int_{-\infty}^{\tau}yp(y)dy\right) \propto \left(\lambda + \frac{g(x)}{l(x)}(1-\lambda)\right)^{-1}. \tag{5}$$

It is clear that $EI(x)$ is maximized when the probability of hyper-parameters belonging to $g(x)$ is low and high for $l(x)$.

EXPERIMENT

Hyper-parameter optimization

The SegNet neural network is implemented in PyTorch. The weights of the encoder part of the network are transferred from a pre-trained VGG-16 network available from the PyTorch Model Zoo. The stochastic gradient descent optimizer was used during the training and the loss function is defined as cross entropy. For faster evaluation, we apply early stopping and finish the training of each instance after 75 epochs.

Batch size, dropout probability, learning rate and weight decay hyper-parameters selected for the optimization process, see Table 1. Since we are investigating SegNet's architecture, parameters such as the number of layers, activation functions, which define the architecture are not included in the optimization. Table 1 presents the searching boundaries of the hyper parameters.

Table 1 Searching boundaries for the hyper-parameters.

Hyper-parameter	Searching boundaries	Comments
Learning rate	[1e-6, 1]	Controls how much we are adjusting the weights of the network with respect to the loss
Weight decay	[1e-6, 1]	Plays important role in network regularization.
Mini-batch size	[1, 16]	Controls the accuracy of the gradient step.
Dropout probability	[0, 1]	It is a regularization technique that deactivates certain number of neurons during training

Dataset

We used the ISPRS Vaihingen dataset for the experiment. The images have a size of 2000 x 2500 pixels. This image size is too large to directly feed into the neural network, therefore all images in the datasets are cropped to 224 x 224 with 60% overlap resulting in 12,664 images. In order for faster training, we used only the 25% of these images (3166 images) that were randomly chosen. 85% of these images are used for training and 15% was the validation dataset. The validation dataset is used to measure the segmentation accuracy during the stochastic gradient descent after each epoch and we chose the best model, accordingly.

Evaluation Metrics

In sematic segmentation tasks, there are many evaluation measures. In this study, we use global accuracy:

$$\frac{1}{N}\sum_{i=1}^{K} TP_i \tag{6}$$

where TP_i is the number of true positive pixels in class i, N is the number of all pixels in all images and K is the number of classes. Note that global accuracy measures only the overall success, but does not emphesize classess with smaller samples. For instance, images might contain larger number of pixels that represent vegetation, but lower number of building pixels. In this case, if a calssifier better extracts vegetation, it might outperform another calssifer that is able to better extract buildings but weaker for vegetations. Therefore, we use macro F1 score that amplifies segmentation success of label categories with lower number of examples in biased datasets:

$$
\begin{aligned}
p_i &= \frac{TP_i}{TP_i + FP_i}, \\
r_i &= \frac{TP_i}{TP_i + FN_i}, \\
F_1 &= \frac{1}{K}\sum_{i=1}^{K} 2\frac{p_i r_i}{p_i + r_i},
\end{aligned} \tag{7}
$$

where FP_i, FN_i are the false positive and false negative pixels in class i, and p_i, r_i is called precision and recall, respectively.

RESULTS AND DISCUSSIONS

Figure 3 shows the results, where X, Y, and Z axes are the batch size, droput probability and learning rate on logarithmic scale. The dots are colorized based on F1-score. Note that the dots are not evenly distributed due to the the Bayes model based searching strategy. The best F1-scores are located at around 1e-4 learning rate, batch size with value 2 and dropout at around 22%. Figure 4 shows the results for global accuracy. Here, the best solution indicates two magnitudes of order higher learning rate and ~50% drouput probability, which is significantly different than parameters indicated by F1-score. Same results can be seen in Figure 5, where F1-score and global accuracy is presented as function of droput probability and learning rate. For F1- score, the best hyper-parameters are located at around ceratin spot, see Figure 5a. As opposed to F1-score, Figure 5b shows that above ~1e-3 learning rate, good global accuracy can be achived that does not depend on the droput probablity. Therefore, the results indicate that the hyper-parameters are more flexible for maximizing the overall (global) accuracy, and careful hyper-parameter tuning is required to maximize the segmentation accuracy within the classes. The numerical results for the best parameters that miximized the F1-score as well as the global accuracy are presented in Table 2.

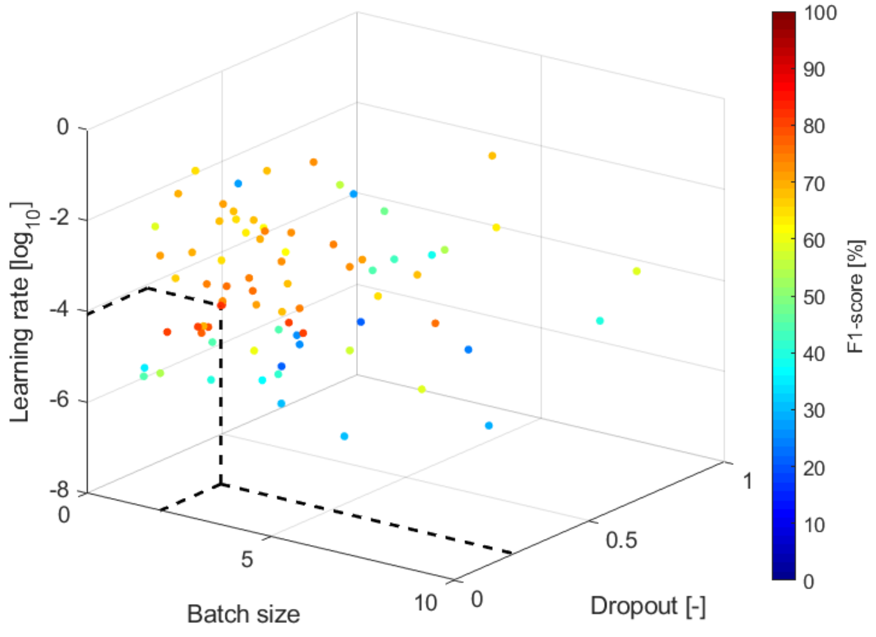


Figure 3 Batch size, droput probaility and learning rate. The dots are colorized based on F1 score.

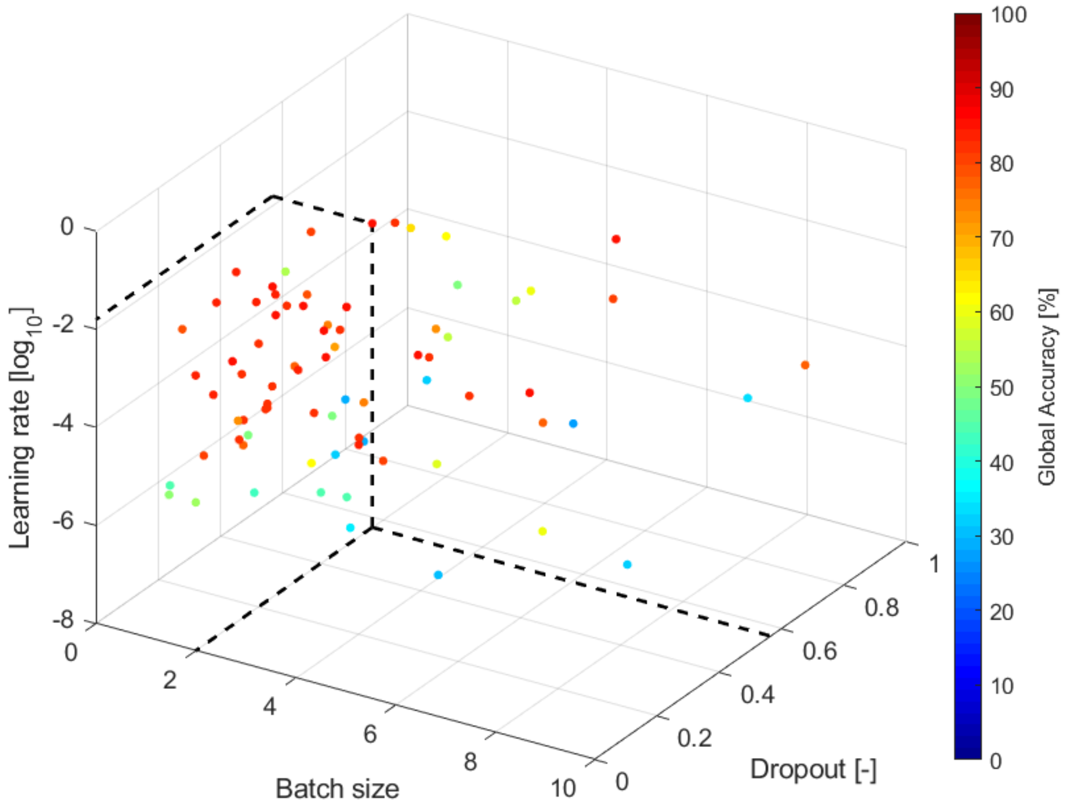
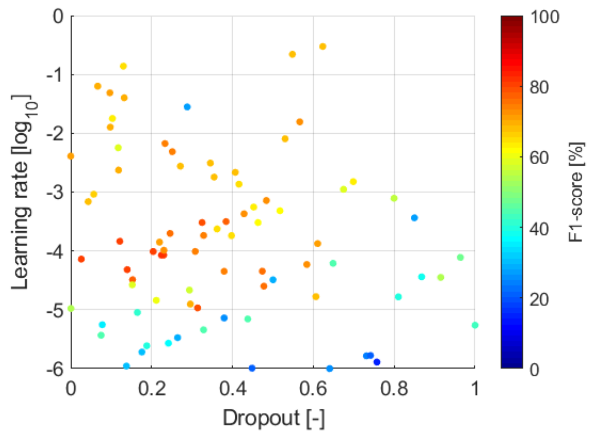
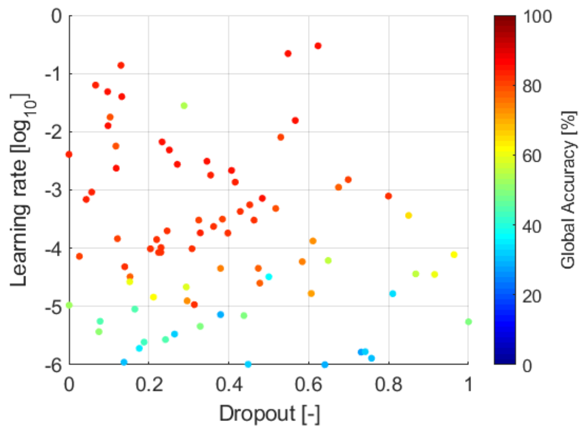


Figure 4 Batch size, droput probaility and learning rate. The dots are colorized based on global accuracy.



(a)



(b)

Figure 5 Droput vs. learning rate for (a) F1-score and (b) global accuracy.

Table 2 Numerical results.

		@ Best F1-score	@ Best Global Accuracy
F1-score	[%]	**81.7**	72.9
Global Accuracy	[%]	82.3	**85.6**
Learning rate	[-]	0.000085 (~= 1e-5)	0.015607 (~= 1e-2)
Weight Decay	[-]	0.000001 (~= 1e-5)	0.000003 (~= 1e-6)
Batch Size	[-]	2	2
Dropout	[%]	22.4	56.6

CONCLUSION

The paper presents our investigation on the hyper-parameters of the SegNet convolutional neural network on aerial images for image segmentation and classification. The encoder-decoder network structure allows for pixel-wise segmentation of an image, where all pixels are annotated with a label by the network. The performance analysis is conducted on the ISPRS's Vaihingen aerial dataset. We found that the hyper-parameters are more flexible for maximizing the overall (global) accuracy, and careful hyper-parameter tuning is required to maximize the segmentation accuracy within the classes. We reported the optimal learning-rate, weight decay, batch size and dropout settings that maximized the F1-score on our dataset.

References

Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla, 2015 SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. ArXiv:1511.00561 [Cs]. http://arxiv.org/abs/1511.00561, accessed January 22, 2019.

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 2546-2554.

Chen, L., G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, 2018 DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. IEEE Transactions on Pattern Analysis and Machine Intelligence 40(4): 834–848.

Dey, Vivek, Yonghui Zhang, and Maosheng Zhong, 2010 A Review on Image Segmentation Techniques with Remote Sensing Perspective. *In* .

Pal, Nikhil R, and Sankar K Pal, 1993 A Review on Image Segmentation Techniques. Pattern Recognition 26(9): 1277–1294.

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox, 2015 U-Net: Convolutional Networks for Biomedical Image Segmentation. ArXiv:1505.04597 [Cs]. http://arxiv.org/abs/1505.04597, accessed January 22, 2019.

Simonyan, Karen, and Andrew Zisserman, 2014 Very Deep Convolutional Networks for Large-Scale Image Recognition. ArXiv:1409.1556 [Cs]. http://arxiv.org/abs/1409.1556, accessed January 22, 2019.