Generalizing satellite maps with neural networks

Adam IWANIAK
Institute of Geodesy and Photogrammetry, Wroclaw Academy of Agriculture
ul.Grunwaldzka 53, 50-357 Wroclaw, Poland
iwaniak@ar.wroc.pl

Witold PALUSZYNSKI
Institute of Engineering Cybernetics, Wroclaw University of Technology
Wybrzeze Wyspianskiego 27, 50-370 Wroclaw, Poland
Witold.Paluszynski@ict.pwr.wroc.pl

Abstract

In this paper the possibility of using neural networks to generalize selected elements of satellite maps was investigated. Generalizing raster maps requires a different approach from generalizing vector maps. Most of the existing raster map generalization methods are connected with methods for analysis and processing of images. Several neural networks for aggregating buildings in satellite maps of urban areas have been created. Image examples of maps, generalized both in vector mode and with neural networks, as well as a summary of the main factors affecting the performance of the networks, are presented in the paper. The conclusion is that neural networks can be trained to perform map processing characteristic of urban area aggregation, and the results, while not perfect in the sense used with vector maps, are very encouraging.

Introduction

Contemporary GIS systems need generalization. While much work in the past has been devoted to vector-based generalization, raster-based techniques have received less attention. However, for some types of generalization processes, like area generalization (aggregation, and the like), raster-based techniques seem very natural to apply [Brown, 1999]. On the other hand, there is a growing supply of high-resolution satellite images, and the resulting demand for efficient techniques for processing these.

Artificial neural networks [Rumelhart, 1991] is an approach to constructing highly parallel systems for distributed processing of data. They are patterned after the network of neurons in the brain, and take advantage of high processing capabilities resulting from connecting a large number of very simple processing elements. Neural networks can also learn and so they do not need to be programmed. A network is trained by presenting it training sets consisting of input data and expected output data, and having the network adjust its parameters to minimize the difference.

In this paper we report on a series of experiments we have conducted with constructing and training neural networks to perform aggregation of buildings in raster maps.

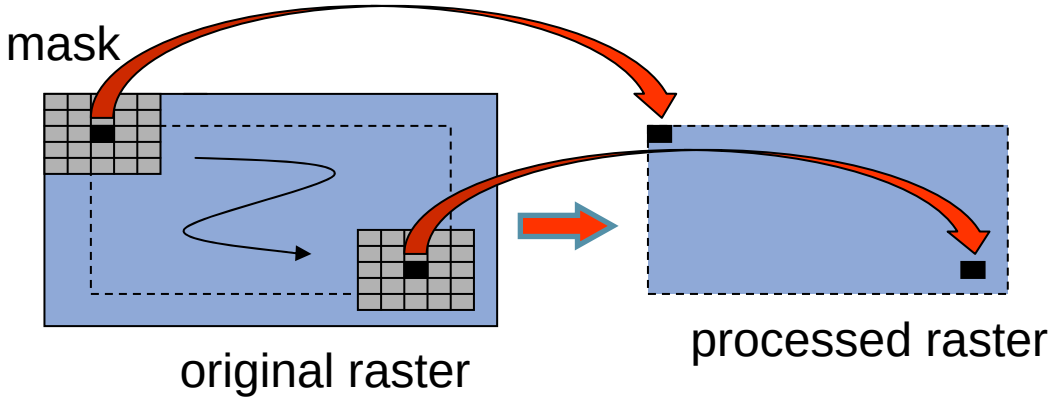
Training neural networks for raster map processing

The goal of the experiments described here was to construct and train a neural network for processing raster maps of arbitrary size. The type of processing was the aggregation

of buildings into built-up areas typical in the process of map generalization when converting to a lower scale. Previous experiments with this type of processing have been described in [Iwaniak, Kubik, Paluszynski, 2001].

Area feature aggregation needs to take into account the relations of nearby objects, so two-dimensional areas of some size have to be considered in making the aggregation decisions. Therefore, the input layer of the neural network should be a two-dimensional array of neurons processing the corresponding elements (pixels) of a map.

It would, however, be hard to create neural networks with input layers the size of raster maps. To solve this problem it was decided that the neural network would work as a filter. More precisely, a network of a constant predefined rectangular size would be fed map data from a likewise-sized area (called a mask) of the map being processed. Then, the output of the network would produce the corresponding area of the result map. In all experiments, the size of this section of the result map was one pixel – the middle



point, **Fig.1**.

Fig.1. Processing a raster using a neural network filter

In this setting, network training would be performed by presenting the network with pairs of original raster maps together with their properly processed versions. The data sets were prepared by taking a large section from a satellite map with buildings marked in a separate layer using the manual procedure. The original map was converted to vector format and area aggregation operator from the MGE Map Generalizer was applied to it, creating the built-up areas aggregated to a larger or lesser extent depending on the distance threshold parameter setting.

Fig. 2 shows a small example of such a training pair (buildings and aggregated areas). The original map size is 500×300 pixels, and the processed map is slightly smaller, 492×292 pixels. (The size difference is due to the filter processing mechanism outlined above and corresponds to the mask size of 9×9 pixels.)

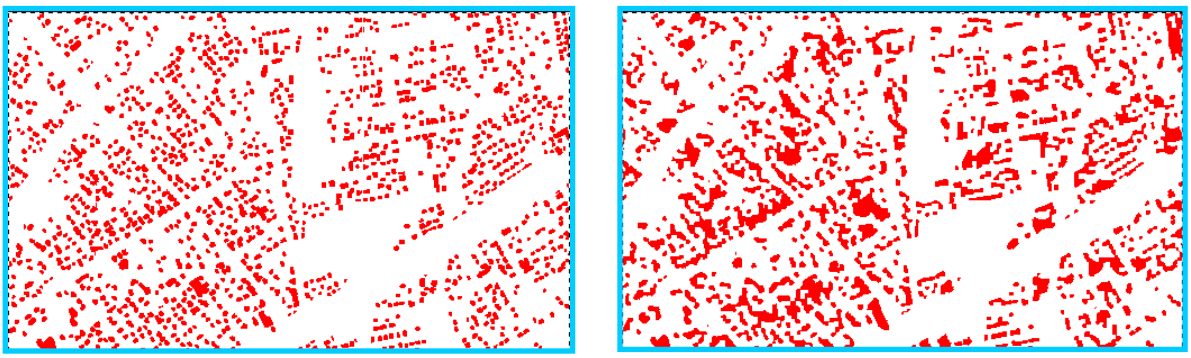


Fig. 2. The first training set: the original raster (left), and the raster aggregated with aggregation extent of 2 pixels (right)

The training proceeded in cycles. In each cycle the mask was moved to all possible positions in the original raster. At each mask position all the network connection weights were adjusted according to the learning algorithm, by comparing the value of the network-computed middle point pixel with the actual value of the corresponding pixel in the processed map. These positions were taken randomly in each cycle, i.e. the mask "jumped" randomly around the map, not taking the same positions twice during each cycle. The training was considered complete when the network calculation error stopped decreasing between successive cycles. This usually occurred after several hundred cycles.

In the second series of experiments the processing of the original map consisted of building aggregation and scale change. The aggregation parameters were set such that corresponded to a two-fold size reduction, and then the whole map size was reduced in half. This process is similar to generalization and the goal of these experiments was to test the ability of the neural network to generalize these elements of raster maps. **Fig. 3** shows the training set. The original raster was the same as before (500×300 pixels), and the processed raster corresponding to the 9×9 mask was 246×146 pixels.

Fig. 3. The second training set: the original raster (left), and the generalized raster (right)

The role of the network architecture

An important decision to make was to select a neural network type and its specific architecture. The network type used was a standard feed-forward network with three layers of neurons, the input layer, which is the processing filter with sizes ranging from 5×5 to 19×19, the middle layer, with several different sizes tried, and the output layer, always the single middle-point neuron. The variations built into the network architecture stemmed from the fact that a greater complexity of the network is necessary for it to be able to learn – and memorize – complex processing patterns. The middle layer of the network and the neuron connection weights would provide this complexity and capacity. Full network connectivity was used, meaning that all neurons of the input layer had direct connections to all the neurons of the internal layer, and all those neurons had connections to the (single) output layer neuron. All these connections had weights constantly adjusted during training.

However, networks too large and complex for a given task exhibit *overfitting*, where the network learns all the patterns perfectly, but on data outside the training set exhibits large errors. In other words, there tends to be the right size and complexity of a neural network for a given task, and aside from some general guidelines, experiments are needed to determine the size of the hidden layer. (There are results in the neural network theory indicating that feed-forward networks with one internal layer are able, given sufficient size, to learn all the same types of patterns as networks with more internal layers.)

On the other hand, the learning function does not affect the final performance of the network, only the learning time, so we chose to use the simple *backpropagation* learning function and each time train the network until it was obvious that no further progress can be made. The initial connection weights were randomly selected in the range of –1.0 to +1.0.

Results

The experiments have been conducted using the Stuttgart Neural Network Simulator [SNNS, 1995]. Several neural networks have been trained on so prepared data, corresponding do different degrees of aggregation and mask sizes. Separate testing data have been then prepared and processed by the networks. The results are presented here. (In theory, testing data only consists of the original, source raster. In actuality, they have been vector-processed in the same way as the training data, for the purpose of error estimation, presented below.)

Fig. 4 shows examples of rasters processed with two networks using 5×5 and 11×11 masks. As can be seen, the neural networks processed the maps generally correctly. Buildings have been merged where they were close together, and left alone where they were far apart. The 5×5 mask is just barely too small for the extent of aggregation used, so the result obtained with the 11×11 mask is more complete.

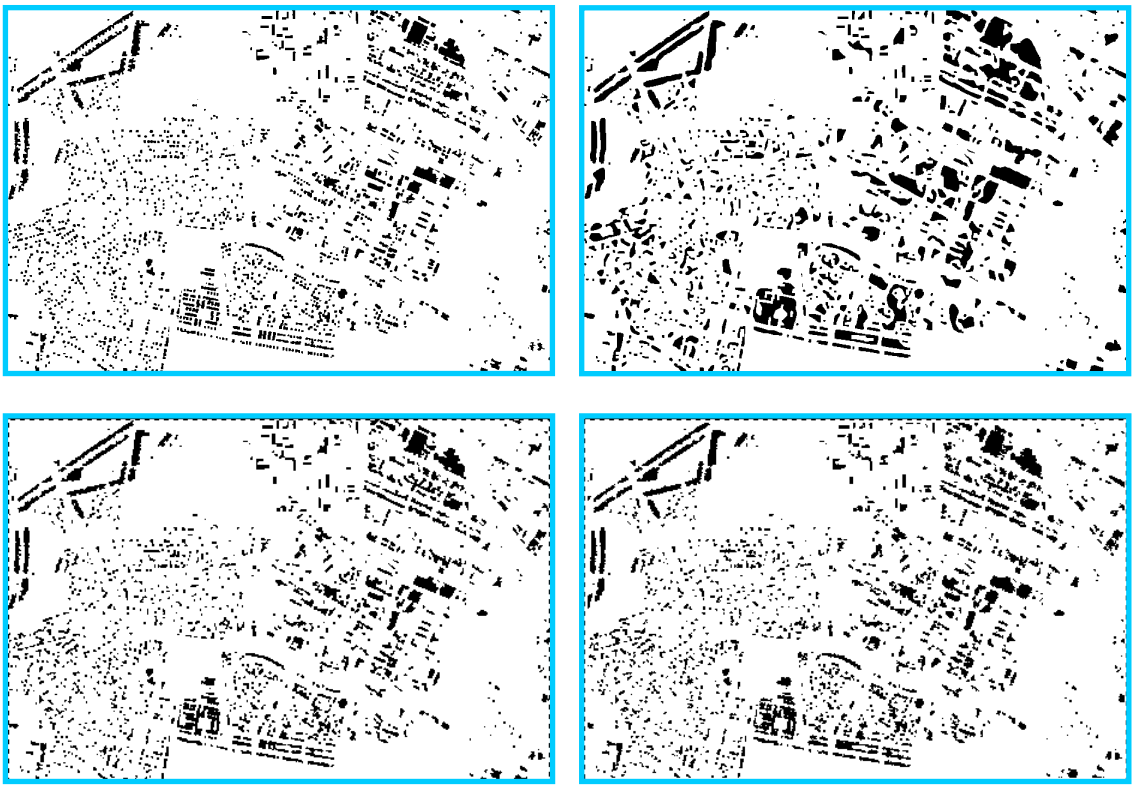


Fig. 4. Top row: original (left) and vector-generalized raster (right); bottom row: neural network results obtained with a 5×5 (left) and 11×11 mask (right); in both cases network output thresholded at 0.5

Fig. 5 shows the results from neural networks trained to perform aggregation as well as scale reduction. This is the setting that this approach is really meant to be applied; namely, to process a map to for its visual appearance accompanying scale change. The appearance of the built-up areas in the smaller map should be similar to the appearance of buildings in the original map.

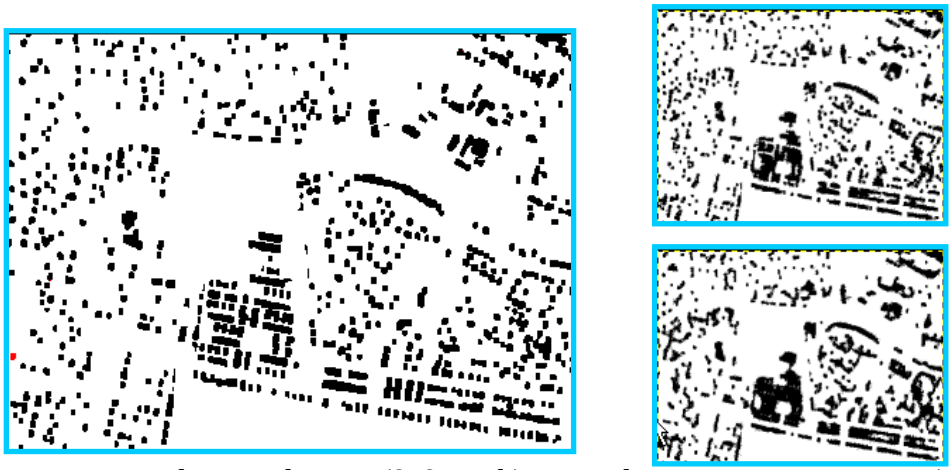


Fig.5. Aggregation and size reduction (9×9 mask): 2-pixel aggregation extension (top right), 4-pixel aggregation extension (bottom right)

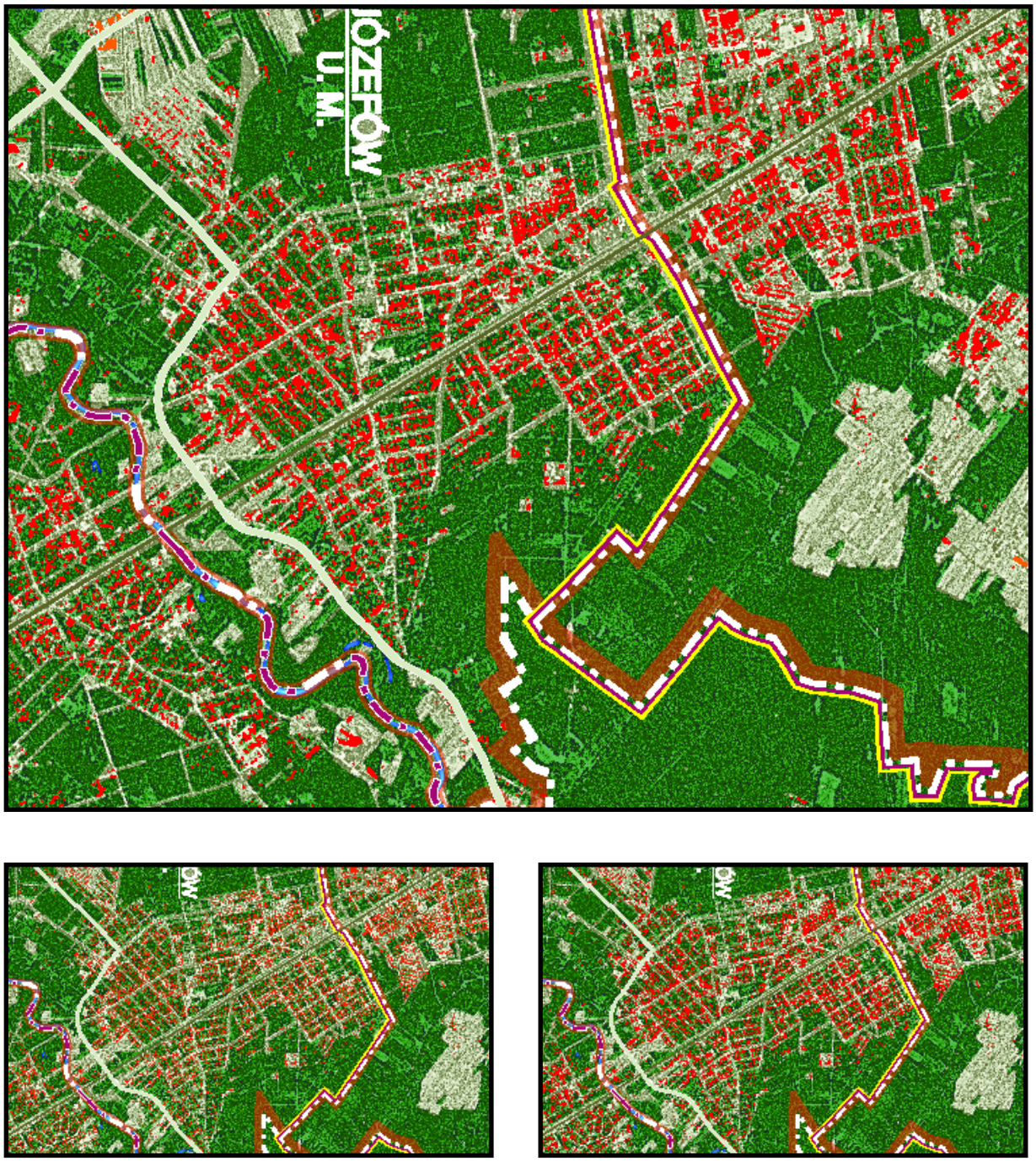


Fig.6. Satellite map example: original raster (top), 2× size reduction (bottom left), aggregated and reduced by neural network (bottom right)

The overall result of the neural network processing of raster maps can be seen in the last example. **Fig.6** shows a section of a satellite image of a suburb area of Warsaw together with (part of) the same image reduced directly, and processed by the aggregating and reducing neural network. The directly reduced image does not reproduce the buildings well while the neural network-processed image shows the built-up areas as expected.

Error estimation

The experiments described above result in different-looking result maps, some of which appear "better" than others. It would be valuable to have some objective measure of quality in order to apply the neural network processing in an optimal way. This, however, turns out to be hard. Aggregated rasters are different from the original, and which of several aggregates are better, or best, is often a matter of subjective judgement. We have not attempted to make such subjective evaluations.

Instead, we have decided to use a simple measure to check which result was the closest to the "standard", by which we mean the same vector-mode aggregation used to create patterns for training the neural network. The measure of processing error is the sum of relative errors of black and white pixels. This error in function of mask size and neural network output threshold was examined in various cases.

The main trend noted is that the mask size tends to have an optimal value, or rather a range of values, outside of which the error goes up. This phenomenon can be explained. A mask of a size too small to encompass the area in which buildings are to be aggregated, may not make correct aggregation decisions. Then, above a certain mask size, the error starts growing rapidly. This is the result of neural network "overfitting" (a network of too high complexity, computing a higher order formula than necessary). The actual values of the optimal mask size range have been found to be related to the aggregation extent used in training the network. This is also the expected effect.

The best results (smallest error values) are generally for the threshold of 0. However, in this case, a visual (subjective) evaluation of the result maps leads to a different observation. Maps obtained with neural network output thresholds higher than 0 have a more pleasing visual appearance, despite higher absolute errors. This effect is due to the nature of neural network computing, which creates some "noise" effects around polygon edges and especially corners. By using higher threshold levels these effects are reduced and lines are smoother, but this happens at the cost of cutting slightly into the contours of buildings.

The same noise effects can also be reduced by some post-processing using such algorithms as median filtration. After using is in some early experiments we have later abandoned it when processing larger satellite images, especially those with size reduction. It turned out that such effects are very apparent in close examination, but when viewing a complete map they do not contribute much to the general impression. On the other hand, the aggregation effects are apparent and unaffected.

The conclusion we make is that neural network processing alone works well for the purpose, for which we intended it, namely, for fast and efficient processing of large images. On the downside, we should note the high processing cost of network training and the necessity of preparing separate networks for different scale changes.

The training of our networks took from a few hours for smaller masks and maps up to many days of computing using computers such as a Sun Microsystems Ultra Enterprise 3500 with 400 MHz processors.

References

Brown, A., Raster-based generalization of polygon data, with special reference to coastlines, Proceedings 19th ICA/ACI, Ottawa, 1999

Iwaniak, A., Kubik, T., and Paluszynski, W., An approach to generalizing raster maps with neural networks, GIS2001, Vancouver, 2001

Rumelhart, D., Neural Networks – a Parallel Distributed Processing Perspective, Proc. of Neural Networks Summer School – Theory, Design, and Applications, Cambridge, 1991

SNNS Stuttgart Neural Network Simulator, User Manual, Version 4.1, University of Stuttgart, Institute for Parallel and Distributed High Performance Systems, Report No.6/95, 1995, http://www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/UserManual/UserManual.html