




Large-Scale Geospatial Data Analysis: Geographic Object-Based Scene Classification in Remote Sensing Images by GIS and Deep Residual Learning

Konstantinos Demertzis¹, Lazaros Iliadis¹ , and Elias Pimenidis²

¹ School of Engineering, Democritus University of Thrace,
67100 Xanthi, PC, Greece

kdemertz@fmenr.duth.gr, liliadis@civil.duth.gr

² University of the West of England Bristol,

FET-Computer Science and Creative Technologies, Bristol, UK

Elias.Pimenidis@uwe.ac.uk

Abstract. Recent advances in optical sensor technologies and Geoinformatics, can support very large scale high definition, used for multispectral and panchromatic images. This capability allows the use of remote sensing for the observation of complex earth ecosystems. Application areas include, sustainability of biodiversity, precision agriculture, land, crops and parasites management. Moreover, it supports advanced quantitative studies of biophysical and biogeochemical cycles, in costal or inland waters. The requirement for precise and effective scene classification, can significantly contribute towards the development of new types of decision support systems. This offers considerable advantages to business, science and engineering. This research paper proposes a novel and effective approach based on geographic *object-based* scene classification in remote sensing images. More specifically, it introduces an important upgrade of the well-known *Residual Neural Network (ResNet)* architecture. The omission of some layers in the early stages of training, achieves an effective simplification of the network, by eliminating the “*Vanishing Gradient Problem*” (VGP) which causes efficiency limitations in other “*Deep Learning*” (DEL) architectures. The use of the *Softmax* activation function instead of the *Sigmoid* in the last layer, is the most important innovation of the proposed system. The *ResNet* has been trained using the novel *AdaBound* algorithm that employs dynamic bounds on the employed learning rates. The result is the employment of a smooth transition of the stochastic gradient descent, tackling the noise dispersed points of misclassification with great precision. This is something that other spectral classification methods cannot handle. The proposed algorithm was successfully tested, in scene identification from remote sensing images. This confirms that it could be further used in advanced level processes for *Large-Scale Geospatial Data Analysis*, such as cross-border classification, recognition and monitoring of certain patterns and multi-sensor data fusion.

Keywords: Geospatial Data · GIS · Remote Sensing · Scene Classification · Residual Neural Networks · Vanishing Gradient Problem

1 Introduction

1.1 Scene Classification and Machine Learning

The rapid advances of digital and communication technologies combined with recent developments in optical sensor technologies have resulted in major changes in the way that we monitor land. This is due to the availability of many different systems that offer high spectral image analysis of the earth's surface (e.g. Multispectral, Hyperspectral [1], Panchromatic and Synthetic Aperture Radar) [2]. Despite such advances, the effective comprehension of the semantic content of such images is still a major challenge and a significant research topic. More specifically, scene classification, aims to achieve automatic semantic tagging of each remote sensing object. This process is content-based and it is one of the most critical problems in *Geoinformatics*. It is very important in a wide range of applications such as, the design and the management of land resources, the applications of precision agriculture, the monitoring of complex ecosystems, the sustainable management of biodiversity and the recording of nature destruction and traffic control.

The past few decades have seen the development of various methods of *Scene Classification* (SCC) based on remote sensing. The early SCC methods were primarily based on low level features or heuristics, which focused on colour, texture, and shape. It is worth mentioning Color Histogram (CH), Histogram of Oriented Gradients (HOG), Local Binary Pattern (LBP), Scale Invariant Feature Transform (SIFT) and Gabor filters Grey Level Co-occurrence Matrix (GLCM) [3]. These methods perform well with images of uniform texture, but their ability to identify more complex scenes is poor. The design of features by humans, affects the effectiveness of the above models considerably. On the other hand, methods based on middle level features, can produce a holistic representation of a scene, which is developed through local visual features such as SIFT, CH, or LBP of local image patches. The deployment of a system capable to develop middle level features, starts with the extraction of the local features of the image, and continues with the codification that can lead to an intermediate representation. The most well-known and widely used model for classifying images of middle level is the *Bag of Visual Words* (BoVW) due to its simplicity and effectiveness [5]. Despite having evolved through considerable improvements in the effectiveness of classification, the techniques based on BoVW have not seen further development and utilisation due to their limitations in representing scenes of high resolution.

Deep Learning (DL) is a branch of computational intelligence that utilises a series of algorithms in attempting to model data of high levels of abstraction. This is achieved by using a multilevel processing architecture, based on consecutive linear and non-linear transformations. It is part of the group of learning techniques that exploit data representations to explain and extract optimal results. In the case of image classification they can produce spatial information such as, edges, shapes and relevant chromatic regions.

Deep Learning Architectures use distributed representation whose main hypothesis is that the observed data are extracted from the interaction of factors grouped in levels.

DL adds the assumption that these levels of factors are either the result of abstraction or synthesis of various scales, depending on their volume and size. Such architectures explore the idea of a hierarchical decomposition of the factors from a high to a low level, with the more abstract concepts drawn from the lowest levels. Thus, they are hierarchically distributed according to levels of abstraction, creating the conditions for selecting the most suitable features of the learning process.

Using the above processes, DL architectures and more specifically Deep Neural Networks (DNN), have achieved impressive performance in many remote sensing applications such as, the accurate representation of features in image classification, the identification of objects and their semantic segmentation. DNN, simulate processes of human vision. Multiple operational levels and intermediate representations are created from image capture in the human eye's retina caused by the reaction of muscles. Such process relies in the transformation of every intermediary unit of input representation into a representation at a higher level. The features at higher level are more generic and less changeable, while those of lower level support the classification of inputs. Their effectiveness can be interpreted based on the *Universal Approximation Theorem* which refers to the potential of a neural structure to approximate continuous functions and the Probabilistic Inference which assumes the activation of non-linearity as a cumulative distribution function [5].

In DNN, every hidden layer trains a discreet group of features resulting from the output of the previous layer. The functioning of such a network allows for the analysis of the most complex features as they recombine and decompose from layer to layer. Such a feature is called a hierarchy. As the decomposition of information increases the complexity of the system hierarchically, it offers the ability to process high level data through non-linear functions. Such networks are suitable for the discovery of non-structured data, for revealing latent structures in unlabelled data and the handling of other problematic structures. Even miniscule similarities or anomalies that these might entail can be identified.

Despite all their important functionality and their advantages, the gradients of the loss function approximate zero, when layers that use activation functions are added neural network architectures. This may cause considerable difficulties in the training of the network, almost to the point of not being capable of training at all, depending on the number of hidden layers that are added. The above is a major vulnerability of deep learning neural networks, called *Vanishing Gradient Problem* (VGP) [6].

2 Theoretical Background

2.1 Facing the Vanishing Gradient Problem

In *Machine Learning* the VGP problem poses a problem in the constriction of neural networks with gradient-based learning methods and backpropagation. In such methods, each of the weights receives, at each iteration, an analogue update with the partial derivative of the error function in relation to the weight it currently uses. In some cases though, the gradient is insignificantly low effectively preventing the weight to change

its value. In the worst case, this could prevent the neural network in completing its training [7]. The following Eq. 1, corresponds to a sigmoid activation function (SGA).

$$S(a) = \frac{1}{1 + e^{-a}} \quad (1)$$

It compresses and projects a large range of inputs to a relative small vector space e.g. [0, 1]. Thus, a large change of the input to the sigmoid activation function, corresponds to a small change in its output. Thus the partial derivative becomes extremely small. It is worth noting that when the inputs to the SGA increase, i.e. when $|x|$ increases or decreases, the partial derivative moves towards zero. The above properties of the sigmoid activation function are ideal for the representation of probabilities and classification procedures. However, both the Sigmoid and the Tangent Hyperbolic (TanH) activation functions, have decreased in popularity recently due to the VGP problem [7].

Let us assume a neural network (NN) with 4 hidden layers with a single neuron at each layer. In the above NN the activity of each neuron depends on the activity of that in the previous layer. More specifically the activity of each neuron depends on that of the previous, multiplied by a given weight. This value is propagated via an activation function (the input is a case of special exemption from the above rule). The margin of error J at the end of the network shows the total error of the system. The backpropagation process is executed to modify the weights via *Gradient Descent* in such a way as to minimise the value of J . To calculate the first weight derivative, the chain rule to backpropagate is used as follows:

$$\frac{\partial error}{\partial w_1} = \frac{\partial error}{\partial output} * \frac{\partial output}{\partial hidden_2} * \frac{\partial hidden_2}{\partial hidden_1} * \frac{\partial hidden_1}{\partial w_1} \quad (2)$$

Subsequently the derivatives are used repetitively until the lowest point has been reached using gradient descent following a specific Learning Rate. The first derivative is used for the activation of the second hidden layer as it is given below. This is performed using the sigmoid activation function (from the output to hidden₂) based on the equations below:

$$z_1 = hidden_2 * w_3 \quad (3)$$

$$\frac{\partial output}{\partial hidden_2} = \frac{\partial Sigmoid(z_1)}{\partial z_1} w_3 \quad (4)$$

Similarly the second derivative is used for the propagation from the hidden₂ to the hidden₁ layer:

$$z_2 = hidden_1 * w_2 \quad (5)$$

$$\frac{\partial hidden_2}{\partial hidden_1} = \frac{\partial Sigmoid(z_2)}{\partial z_2} w_2 \quad (6)$$

In both cases the propagated values contain the derivatives of the sigmoid activation function. This can collectively be expressed as follows:

$$\frac{\partial output}{\partial hidden_2} \frac{\partial hidden_2}{\partial hidden_1} = \frac{\partial Sigmoid(z_1)}{\partial z_1} w_3 * \frac{\partial Sigmoid(z_2)}{\partial z_2} w_2 \tag{7}$$

Both the values of the Sigmoid (z_1) and Sigmoid (z_2), are less than 0.25. The weights w_1, w_2, w_3 , are initialised by the Gauss method so that they can have a mean value equal to 0 and standard deviation equal to 1. Therefore every $\|w_i\|$ is smaller than 1. Thus for the calculation of the derivatives we multiply numbers that are less than 1 and 0.25. Given that two numbers in the range [0, 1] are multiplied, the result will always be a smaller value, e.g. $1/3 * 1/3 = 1/9$. Thus, multiplying such small numbers many times, results in a gradient that is so small that will force the network to stop the learning process. The various terms used in such a multiplication is shown in (8) below:

$$\frac{\partial output}{\partial hidden_2} \frac{\partial hidden_2}{\partial hidden_1} = \frac{<1/4}{\frac{\partial Sigmoid(z_1)}{\partial z_1}} <1 w_3 * \frac{<1/4}{\frac{\partial Sigmoid(z_2)}{\partial z_2}} <1 w_2 \tag{8}$$

In networks with a small number of layers, employing Sigmoid activation functions is not considered a major problem, whereas in multi-layered architectures it could cause major disruption to the normal and effective training of the network.

Totally, four types of solutions for the VGP problem have been proposed in the literature [8]:

- a) Methods that do not use gradients such as *Simulated Annealing*, *Multi-Grid Random Search* and *Random Weight Guessing*. Generally the Global Search Methods (GSM) work well in the case of simple problems with long term dependencies. Simple problems can be resolved with networks that use only a few parameters and do not require complex calculations. Such solutions are characterized as *Flat Minima* situations where the network is attempting to solve the problem using a simple architecture. The weights' interconnection levels, are included in a very specific vector space with the error being almost constant. This can only cover a small range of problems, therefore the use of *Flat Minima* is not recommended as a general solution.
- b) Methods that enforce higher gradients. Higher gradients can be reinforced by using optimization methods. These are considered time consuming and computationally costly. Furthermore, they appear to have problems in learning to store accurate information, related to the real value of classification.
- c) Methods that operate on higher levels. They employ the *Rectified Linear Unit* (ReLU) activation functions that do not yield small derivatives:

$$ReLU(x) = \max(0, x) \quad \text{or} \quad ReLU(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } x \geq 0 \end{cases} \tag{9}$$

In some cases though, the ReLU neurons can be forced to situations in which they become inactive for all inputs. In such cases, none of the gradients backpropagate via

the neuron, resulting in the neuron being stuck in an interminable inactive condition and effectively dies. This category also includes methods that attempt to solve the problem by following an intuitive process, via the use of high level parameters, such as metadata. These cases also fail in yielding solid and generalizing solutions.

- d) Methods that use special architectures. The most important solutions for the VGP problem have been proposed via the use of ResNets [9]. This addresses the stagnation of multiple layer levels even when the size of the network exceeds the 150 layers.

3 The Employed Residual Neural Networks

ResNets, are brain inspired. They are utilizing “skip connections-shortcuts”, in order to jump over some layers. They operate similarly to the Convolutional Neural Networks (CNN), however in ResNets the input is provided sequentially at the exit of the hidden layers, if they are in a certain distance from each other.

In the majority of neural networks, one layer feeds the very next one. In a ResNet that consists of several blocks, every single layer feeds the successive one, but at the same time it provides input to a layer which is located 2 maybe 3 positions far (it skips the order as in Fig. 1). Many linearly interconnected Residual Blocks (REB) constitute a Residual Neural Network. The values of REB parameters are determined based on the network’s structure. They aim at modelling a high level of abstraction of the incoming samples, using multiple stacked non-linear transformations. Their architecture assumes the following characteristics, namely: *Local Receptive Fields* (LRF), *Weight Sharing* (WS), *Spatial Subsampling* (SpaSu), *Feature Combination* (FC) [10].

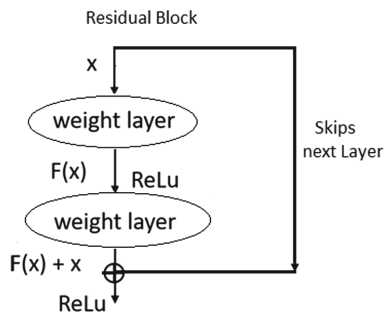


Fig. 1. Image of a Residual Block. Several of them are forming a Residual Neural Net

Essentially, ResNet architecture consists of a set of levels that are characterized by different functionality. They can be configured through their parameters and hyper-parameters. The purpose is to convert an input volume into an output one, through a differential function. Interconnection based on the *Residual Block* parameter,

determines the overall architecture of the network. The levels of neurons commonly used fall into the following basic categories, namely: *Convolutional Layers* (CL), *Fully Connected Layer* (FCL), *Pooling Layers* (PL), *Dropout Layer* (DRL), *Feed-Forward Layer* (FFL). After any level architecture and sequence, all of the top-level neurons are input to an FFL array or any similar forward-current array [10]. The *Loss Function* (LF) aims to attribute the classification error, which is the difference between the prediction output and the known desired output during the training process.

As noted above, the substantial difference between CNNs and ResNets is the sequential addition of the input to the output of the hidden layers, provided that the hidden layers are at certain distance apart. The Residual Block consists of a set of CLs defined by architectural standardization (usually 2). The input vector x of the Block is stored in a local buffer and after passing all levels of the Block, it is added to the output of $F(x) + x$. The increased output is presented as input to the next *Residual Block* and so on [10]. In the case of single skips we can use the notation $l-2$ to l . More specifically, let $W^{l-1,l}$ be the weight vector for connection weights from layer $l-1$ to l and let $W^{l-2,l}$ be the weight vector for weights from layer $l-2$ to l . Thus, the forward propagation through the use of the activation function would be the following [9]:

$$a^l = g(W^{l-1,l} \cdot a^{l-1} + \beta^l + W^{l-2,l} \cdot a^{l-2}) = g(Z^l + W^{l-2,l} \cdot a^{l-2}) \tag{10}$$

Where, a^l is the output of neurons in layer l , g is the activation function for layer l , $W^{l-1,l}$ is the weight matrix for neurons between layer $l-1$ and l . Also, $W^{l-2,l}$ is the weight matrix for neurons between layer $l-2$ to l and

$$Z^l = W^{l-1,l} \cdot a^{l-1} + \beta^l. \tag{11}$$

ResNets, are inspired by the function of pyramidal cells in the cerebral cortex of the brain, where forward skips take place in many layers. This forward propagation process is expressed by the following equation:

$$a^l = g\left(Z^l + \sum_{k=2}^K W^{l-k,l} \cdot a^{l-k}\right) \tag{12}$$

where $k-1$ is the number of skips.

In backward propagation through the activation function, the *Normal Path* is described by Eq. 13 and the *Skip Paths* by Eq. 14:

$$\Delta w^{l-1,l} = -\eta \frac{\partial E^l}{\partial w^{l-1,l}} = -\eta \alpha^{l-1} \cdot \delta^l \tag{13}$$

$$\Delta w^{l-2,l} = -\eta \frac{\partial E^l}{\partial w^{l-2,l}} = -\eta \alpha^{l-2} \cdot \delta^l \tag{14}$$

Where η is the learning rate, δ^l is the error signal of neurons at layer l and α^l is the activation of neurons at layer l . For the case of a ResNet using forward propagation, the above equations are transformed as follows:

$$\Delta w^{l-k,l} = -\eta \frac{\partial E^l}{\partial w^{l-k,l}} = -\eta \alpha^{l-k} \cdot \delta^l \quad (15)$$

This residual connection does not go through activation functions that “squash” the derivatives, resulting in a higher overall derivative of the block.

4 Literature Review

Inspired by the success of deep learning in the field of computer vision, several related studies have been conducted suggesting various DL architectures for the analysis of ultrasonic data, which have admittedly provided new impetus to this field. The authors of [11] use a hybrid method which combines stacked *Autoencoder*, *Principle Component Analysis* (PCA), and *Logistic Regression* to perform hyperspectral data classification. Tao et al. [12] use a sparse stacked *Autoencoder* to efficiently represent features from unmarked spatial data, and then learned features are fed into a linear SVM for hyperspectral data classification. Various 1D [13] and 2D [14] CNN architectures have been proposed from time to time to encode spectral and spatial information. The latest and most sophisticated proposal concerns 3D CNN [15] in which the third dimension refers to the time axis resulting in a spatio-temporal architecture in the spectral classification. In 3D CNNs, the convolution functions are spatial-spectral, whereas in 2D CNNs, they are spatial only. Compared to 1D and 2D CNNs, 3D CNNs can better spectral information thanks to 3D convolution functions.

Trying to exploit the particularities and advantages of unsupervised learning, the authors [16] propose an unsupervised CNN architecture to perform learning of spectral-spatial features. This is performed by using sparse learning to estimate the network’s weights in a greedy layer-wise fashion instead of an end-to-end approach. The algorithm is rooted on sparse representations and enforces both population and lifetime sparsity of the extracted features, simultaneously. They successfully illustrate the expressive power of the extracted representations in several scenarios: classification of aerial scenes, as well as land-use classification in very high resolution or land-cover classification from multi- and hyperspectral images [4].

The proposed algorithm clearly outperforms PCA. The results have shown that single-layer CNNs can extract powerful discriminative features only when the receptive field accounts for neighboring pixels. They are preferred when the classification requires high resolution and detailed results. However, Deep architectures significantly outperform single-layer variants, capturing increasing levels of abstraction and complexity throughout the feature hierarchy.

In [17] authors propose a Deep Recurrent Neural Network model with a new activation function (*parametric rectified Tanh – PRetanh*). The proposed activation function makes it possible to use fairly high learning rates without the risk of divergence during the training procedure. Moreover, a modified gated recurrent unit, which

uses PRetanh for hidden representation, is adopted to construct the recurrent layer in the network to efficiently process hyperspectral data and reduce the total number of parameters.

All of the above architectures have major malfunctions and are hampered by the VGP problem, which generally disrupts deep approaches. He et al. [9] tried to overcome this problem by employing the idea of very deep networks by proposing the 152-layers ResNet, which allowed CNNs to grow much deeper without suffering the problem of vanishing/exploding gradients. The authors provide an in-depth analysis about the degradation problem, i.e., simply increasing the number of layers in plain networks results in higher training and test errors.

It is suggested that it is easier to optimize the residual mapping in the ResNet than to optimize the original, unreferenced mapping in the conventional CNNs. In essence, instead of learning a direct map from low-quality inputs to high-quality outputs, the CNN is tasked with learning the residual, i.e., the difference between the low and high-quality signals, which typically represents missing high-frequency information, at least for the case of super-resolution. To allow networks to capture and extract features from multiple scale, skip connections between different layers have also been considered and are now part of state-of-the-art approaches.

The authors of [18], are proposing a novel network architecture, which is a fully Convolutional/Deconvolutional network, for unsupervised spectral-spatial feature learning of hyperspectral images, which is able to be trained in an end-to-end manner. Specifically, the proposed architecture, is based on the so-called *encoder-decoder* paradigm. The input 3-D hyperspectral patch is first transformed into a typically lower dimensional space via a convolutional sub-network (encoder). Then it is expanded to reproduce the initial data by a *de-convolutional sub-network* (decoder).

However, during the experiments, we have found out, that such a network is not easy to be optimized. Although the proposed network has not been explicitly designed for the task of object detection, we have observed that the target object can be localized by the activated or suppressed pixels in some specific learned feature maps of the first residual block. This makes it possible to achieve the unsupervised object detection in hyperspectral images. Experimental results also demonstrate that the features learned by the proposed unsupervised network can be used for the hyperspectral image classification task, and the obtained classification results are competitive compared with the other supervised approaches.

5 The Introduced Methodology

The methodology that is introduced by this research paper, is based on the Convolutional/Deconvolutional (CN/DC) Network architecture that has been used by Mou et al. [18]. The above authors have proposed a fully CN/DC network approach, in which the desired output is the input data itself. Specifically, the introduced model, consists of two parts as its name typically states, namely the Convolutional and Deconvolutional subsystem. The first subsystem corresponds to an encoder that transforms the input characteristics x_i into an abstract representation of intermediate features h_i .

The corresponding Deconvolutional subset, plays the role of the decoder that reproduces in the original xi input data, the encrypted, intermediate hi features. Essentially the CN/DC network with Residual Learning is a modular network architecture that stacks residual blocks. Similar to Convolutional blocks, a Residual block consists of several Convolutional layers that have the same feature map size and the same number of filters. Their function performs the following calculation.

$$\phi_n = g(\varphi_n) + F(\varphi_n; \Theta_n) \tag{16}$$

$$\varphi_{n+1} = f(\phi_n) \tag{17}$$

Where φ_n refers to feature maps, Θ_n to a collection of weights associated with residual block, F is a residual function and f is the activation function. The corresponding function g, is fixed to an identity mapping:

$$g(\varphi_n) = \varphi_n \tag{18}$$

If f is a linear activation function and $\varphi_{(n+1)} = \varphi_n$, the output for the n^{th} residual block is calculated by Eqs. (16) and (17) so that:

$$\varphi_{n+1} = \varphi_n + F(\varphi_n; \Theta_n) \tag{19}$$

The following Eq. 20 is recursively produced:

$$\varphi_{n+2} = \varphi_{n+1} + F(\varphi_{n+1}; \Theta_{n+1}) = \varphi_n + F(\varphi_n; \Theta_n) + F(\varphi_{n+1}; \Theta_{n+1}) \tag{20}$$

The recurrence formula below, is obtained for any shallower block n and any deeper block L .

$$\varphi_L = \varphi_n + \sum_{i=n}^{L-1} F(\varphi_i; \Theta_i) \tag{21}$$

The way residual learning helps in the effective training of the deep network in question, is found in the rules of backpropagation, where E stands for the loss function:

$$\frac{\partial E}{\partial \varphi_n} = \frac{\partial E}{\partial \varphi_L} \frac{\partial \varphi_L}{\partial \varphi_n} = \frac{\partial E}{\partial \varphi_L} \left(1 + \frac{\partial}{\partial \varphi_n} \sum_{i=1}^{L-1} F(\varphi_i; \Theta_i) \right) \tag{22}$$

In this way, the classification information of a layer in the network does not disappear even when the trainable weights are arbitrarily small, which is the key to make the training in the deep network possible. Also, in order to be able to successfully complete the processes performed by the Convolutional/Deconvolutional subsystems, the need for a pooling layer is imperative. However, the pooling layer leads to a reduced resolution of feature maps. This recreates the original input data to Deconvolutional, through an Unpooling process, in order to separate the feature maps, that is, to increase their spatial range, as opposed to the concentration applied by the Convolutional web.

It should be noted that using *Max-Pooling* indices the system is able to record the position of the maximum value in each local concentration area while concentrating in the Convolutional sub-net. Recently, an advanced version of the *max* and *argmax* functions was presented [19]. It can receive not only the maximum value in the field of indicators of a maximum concentration layer, but the corresponding index of this value as well [19]. Specifically, these two functions can be calculated as follows:

$$\mu = \sum_v z(i,j) \frac{\exp(az(i,j))}{\sum_v \exp(az(i,j))} \approx \max_v z(i,j) \quad (23)$$

$$\mu = \sum_v [i,j]^T \frac{\exp(az(i,j))}{\sum_v \exp(az(i,j))} \approx \operatorname{argmax}_v z(i,j) \quad (24)$$

It is a fact, that with the use of the *max* and *argmax* functions, the *max-pooling* indices can be obtained in any pooling layer. Then, by performing interpolation in the *Unpooling* layers of the *Deconvolutional* sub-network, the interconnected values which are transferred by the *max-pooling* indices can be successfully handled. The use of *max-pooling* indices allows for a more accurate display of location information and allows feature maps to record detailed information about input features.

6 The Geographic Object-Based Scene Classification Algorithm

This research paper suggests an important modification that upgrades the ResNet architecture discussed above, which employs the Softmax activation function, instead of the Sigmoid at its last level. According to the introduced novel approach, the fully Residual Network is trained using the novel AdaBound algorithm that employs dynamic bounds on their learning rates. It achieves a smooth transition to the stochastic Gradient Descent, which accurately addresses the noisy scattered misspellings that other spectral classification methods cannot handle. As it has already been said, the following Softmax activation function which maps the non-normalized output of a neural network to a probability distribution over predicted output classes, was used instead of the Sigmoid, in the last CL [10]:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^k e^{z_j}}, \quad i = 1, \dots, k \quad z = (z_1, \dots, z_k) \in \mathbb{R}^k \quad (25)$$

Where, z_j is every element of the input vector z , $\sigma(z)$ is the output vector and the sum of its components $\sigma(z)_i$ is equal to 1.

The choice of the *Softmax* was based on the fact that it performs better on multi-classification problems, like the one under consideration. On the other hand, the *Sigmoid* is suitable on binary classification tasks. In the case of *Softmax*, the sum of probabilities is equal to 1 whereas in Sigmoid it does not have to be equal to 1. Finally for the *Softmax*, the highest value has the highest probability, whereas in the *Sigmoid*, the

highest value has a high probability but not the highest. The fully residual network was trained using the novel *AdaBound* algorithm [20] that employs dynamic bounds on their learning rates. It achieves a smooth transition to stochastic gradient descent. The following algorithm 1, (in the form of pseudocode) presents the *AdaBound* function [20]:

Algorithm 1. AdaBound

Input: $x_1 \in F$, initial step size α , $\{\beta_{1t}\}_{t=1}^T$, β_2 lower bound function η_l , upper bound function η_u

- 1: Set $m_0 = 0$, $u_0 = 0$
 - 2: **for** $t = 1$ **to** T **do**
 - 3: $g_t = \nabla f_t(x_t)$
 - 4: $m_t = \beta_{1t}m_{t-1} + (1 - \beta_{1t})g_t$
 - 5: $u_t = \beta_{2t}u_{t-1} + (1 - \beta_{2t})g_t^2$ and $V_t = \text{diag}(u_t)$
 - 6: $\hat{\eta}_t = \text{Clip}\left(\frac{\alpha}{\sqrt{V_t}}, \eta_l(t), \eta_u(t)\right)$ and $\eta_t = \frac{\hat{\eta}_t}{\sqrt{t}}$
 - 7: $x_{t+1} = \Pi_{f, \text{diag}(\eta_t^{-1})}(x_t - \eta_t \cdot m_t)$
 - 8: **end for**
-

Compared with other approaches, the *AdaBound* method has two advantages. First, there exists a fixed turning point to distinguish, in the simple ADAM algorithm and the SGD is uncertain. So the *Adabound* addresses this problem with a continuous transforming procedure rather than a “hard” switch. Second, the *AdaBound* introduces an extra hyperparameter to decide the switching time, which is not very easy to fine-tune. In general, the use of the *AdaBound* algorithm has a high convergence speed compared to stochastic gradient descent models and it exceeds the poor generalization ability of adaptive approaches. Moreover, it has dynamic limits on the learning rate, in order to achieve the highest accuracy for the dataset under consideration [20].

7 The Data Experiments and Results

The dataset used in this research, includes images taken from the Reflective Optics System Imaging Spectrometer (ROSIS) covering the Engineering School at the University of Pavia [21]. The available training samples belong to nine categories that are mainly related to land cover items. Each image is 610×340 pixels with a resolution of 1.3 m per pixel. Ultrasound imaging consists of 115 spectral channels ranging from 430 to 860 nm of which only 103 were used in the work as 12 were removed due to noise.

For the network configuration, we leverage convolutional filters with a very small receptive field of 3×3 . In addition, the convolutional stride is fixed to 1 pixel; the spatial padding is also 1 pixel. Max-pooling is performed over 3×3 pixel windows with stride 3. All the convolutional layers are using ReLU as an activation function except for the last layer that uses Softmax. The fully residual network was trained using the novel *AdaBound* algorithm [9] and all the suggested default parameters were used

for all the following experiments. Once the training of the residual network is complete, we can start to fine-tune the network for hyperspectral data classification. We have made use of stochastic gradient descent with a fairly low learning rate equal to 0.0001.

The following criteria were used to evaluate the performance of the geographic object-based scene classification algorithms in remote sensing images [22]:

a) Overall Accuracy (OA): This metric represents the number of samples that are classified correctly, divided by the number of test samples. b) Average Accuracy (AA): This index shows the average accuracy of the classifications of all categories. c) Kappa coefficient: This is a statistical measurement that provides information on the level of agreement between the truth map and the final classification map. It takes into account the percentage of agreement which could be expected only chance. In general, it is considered to be a more robust index than a simple percent agreement calculation, since k takes into account the agreement occurring by chance.

In addition, in order to evaluate the importance of the classification accuracy derived from different approaches, a McNemar statistical test is performed [23]:

$$z_{12} = \frac{f_{12} - f_{21}}{\sqrt{f_{12} + f_{21}}} \quad (26)$$

Where f_{ij} is the number of correctly classified samples in the i^{th} classification and of the incorrectly classified in the j^{th} classification. The McNemar's test is based on the standardized normal test statistic and therefore, the null hypothesis, which is “*no significant difference*,” is rejected at the widely used $p = 0.05 (|z| > 1.96)$ level of significance. This assessment test, determines the significance of the differences between the classification accuracy values obtained by the proposed model, versus the accuracy values obtained by other examined approaches.

To validate the effectiveness of the proposed architecture, the method is compared with the most widely used supervised deep learning models which are summarized as follows:

- a. 1-D CNN: The 1-D CNN network architecture was designed as in [24] and includes an input layer, a convolutional layer, a max-pooling layer, a fully connected layer, and an output layer. The number of convolutional filters is 20, the length of each filter is 11 and the pooling size 3. Finally, 100 hidden units are contained in the fully connected layer.
- b. 2-D CNN: The 2-D CNN architecture was designed as in [15]. It contains three convolutional layers equipped with 4×4 , 5×5 and 4×4 convolutional filters, respectively. The 2-D CNN architecture was designed as in [15]. It contains three convolutional layers equipped with 4×4 , 5×5 and 4×4 convolutional filters, respectively.
- c. The Convolutional layers - except for the latter - are followed by max-pooling layers. In addition, the numbers of convolutional filters for CL are 32, 64 and 128, respectively.
- d. Simple Convolutional/Deconvolutional network: The simple Convolutional/Deconvolutional network with simple convolutional blocks and the Unpooling function are applied in the cases of [25,26].

- e. Residual Convolutional/Deconvolutional network: It is an architecture using residual blocks and a more precise Unpooling function, as presented in [18].

Optimized Residual Convolutional/Deconvolutional (ORCD) network

It is the improved version of the above architecture which was presented in this research paper. It differs in the fact that it uses the *Softmax* activation function in the last convolutional layer and that its fully residual network is trained using the novel AdaBound algorithm. This is achieved by employing dynamic bounds on the learning rates, which results in a smooth transition to stochastic gradient descent. This method shows great efficacy while maintaining advantageous properties of adaptive learning, such as rapid initial progress and hyperparameter insensitivity. Note that, we have used the standard types of training and testing samples in order make the proposed approach fully comparable with other classifiers in the literature.

The ORCD network was trained using the *AdaBound* algorithm, and all the suggested default parameters were used for all the following experiments. The number of Convolutional filters (CF), increases towards deeper layers of the convolutional sub-networks: There were 64 CF in the first residual block, 128 in the following block, and 256 in the last one. This rule is turned over for the *Deconvolutional sub-network*. All of the convolutional layers are with ReLU as activation function except the last layer that uses the *Softmax*.

All weight matrices in the network and the bias vectors are initialized with a uniform distribution, and their values are initialized in the range $[-0.1, 0.1]$. The number of the unlabeled data samples of the Pavia University that were used for training the network is 10000. These unlabeled samples are randomly selected from the whole set of images.

In the *Optimized Residual Convolutional/Deconvolutional* network, the hyperspectral data are normalized in the closed interval $[0, 1]$. Then, all of the weights are updated during the training procedure. Once the training of the network is completed, the *fine-tuning* process for hyperspectral data classification follows. The *stochastic gradient descent* with a fairly low learning rate of 0.0001 has been employed, for the fine tuning of the network. During this process, a percentage equal to 10% of both hyperspectral data sets, has been randomly selected as the validation set.

That is, during fine-tuning, 90% of the dataset has been used for learning and the remaining 10% of the available data samples served as the validation set, to perform tuning of the *hyper-parameters*, such as the numbers of convolutional filters in the convolutional layers. All of the test samples were used to evaluate the final performance of the learned *spectral-spatial feature representations* and the *fine-tuned network* was used to perform the classification.

The following Table 1 shows the classification maps using the *Pavia University* data set and the comparison of the accuracies between the classifiers *1-D CNN*, *2-D CNN*, *Simple Convolutional/Deconvolutional Network (Simple C/D N)*, *Residual C/D N* and the proposed *Optimized R C/D N*.

Trying to evaluate the above algorithms based on the obtained results, it is easy to conclude that the proposed ORCD *Network* outperforms the competing Deep Learning approaches for the *OA*, *AA*, and *Kappa* evaluation indices.

Table 1. Comparison of the accuracies between the classifiers

Class No	Class Name	1D CNN	2D CNN	Simple C/D N	Residual C/D N	Optimized R C/D N*
1	Asphalt	83.73	69.25	82.81	78.99	86.59
2	Meadows	65.70	93.39	97.11	97.16	97.01
3	Gravel	67.03	63.13	60.31	61.46	69.97
4	Trees	94.03	94.39	95.59	95.76	94.91
5	Metal Sheets	99.41	100	97.55	97.77	98.83
6	Bare Soil	96.30	49.06	59.38	59.46	69.87
7	Bitumen	93.83	72.26	78.42	79.50	86.49
8	Bricks	93.56	94.32	96.50	96.82	96.85
9	Shadows	99.79	93.77	92.29	92.40	97.71
OA	–	79.28	82.66	87.82	87.39	90.51
AA	–	88.15	81.06	84.44	84.37	88.69
Kappa	–	0.7423	0.7688	0.8363	0.8308	0.8482
Significance	–	31.362	31.464	23.178	22.232	21.871

The proposed method produces extremely accurate results without repeated problems of undetermined cause, because all of the features in the considered dataset are handled very efficiently. In addition, one of the key advantages gained from the results is the high reliability, resulting from the high kappa values (maximum reliability if $k \geq 0.81$). This can be considered as the result of data processing that allows the most reliable relevant data for the forthcoming forecasts. The above Table 1, also provides information on the results of the *McNemar* test to assess the significance of the difference between the classification accuracy of the proposed network and the other approaches considered. The results of the *McNemar* test, have proven that there exist statistically significant differences among the results obtained by the employed methods. More specifically, the differences between the 1D and 2D CNN with the rest of them are quite high, whereas there are minor but significant differences between the Optimized R C/D N (our proposed approach) and the Residual.

8 Discussion - Conclusions

This research proposes an innovative and highly effective geographic object-based scene classification system in remote sensing images, using an innovative Residual Neural Network (ResNet) architecture. This approach eliminates VGP as it is using Softmax activation function instead of Sigmoid on the last layer of the network. The fully residual network was trained using the novel AdaBound algorithm that employs dynamic bounds on their learning rates. It achieves a smooth transition to stochastic gradient descent, and it precisely addresses the noisy scattering points of misclassification that other spectral classification methods cannot handle properly.

Its implementation is based on the optimal use and combination for the first time in the literature, of two highly efficient and fast learning processes (Softmax activation function and AdaBound algorithm), which create an integrated intelligent system.

This network also remarkably implements a Large-Scale Geospatial Data Analysis approach that attempts to balance latency, throughput, and fault-tolerance using ResNet while simultaneously exploiting learning processes optimally and as efficiently as possible. The reliability of the proposed network has been successfully tested in the recognition of scenes from remote sensing photographs, which suggests that it can be used in higher level Geospatial Data Analysis processes. Such cases are the classification, the recognition and monitoring of specific standards, and the fusion of multi-sensor data.

Suggestions for the evolution and future improvements of this network should focus on further optimizing the parameters of the algorithms used in ResNet architecture. This will be done in order to achieve an even more efficient, more accurate and faster categorization process using a heuristic approach [27] or customization of the algorithm with the use of Spiking Neural Networks [28]. It would also be important to study the extension of this system by implementing the Lamda architecture [29] in an environment of parallel and distributed big data analysis systems (Hadoop). Finally, an additional target that could be considered in the direction of future expansion concerns the operation of the network by methods of self-improvement [30] and redefinition of its parameters in meta-learning. This can fully automate the geographic object-based scene classification process in remote sensing images.

References

1. Plaza, A., Plaza, J., Paz, A., Sanchez, S.: Parallel hyperspectral image and signal processing. *IEEE Sign. Process. Mag.* **28**, 119–126 (2011)
2. Hubert, M.J., Carole, E.: Airborne SAR-efficient signal processing for very high resolution. *Proc. IEEE* **101**, 784–797 (2013)
3. Zhang, W., Tang, P., Zhao, L.: Remote sensing image scene classification using CNN-CapsNet. *Remote Sens. MDPI* **11**(5), 494 (2019). <https://doi.org/10.3390/rs11050494>
4. Yang, Y., Newsam, S.: Bag-of-visual-words and spatial extensions for land-use classification. In: *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, San Jose, CA, USA, pp. 270–279 (2010)
5. Penatti, O.A., Nogueira, K., dos Santos, J. A.: Do deep features generalize from everyday objects to remote sensing and aerial scenes domains? In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Boston, MA, USA, pp. 44–51 (2015)
6. Schmidhuber, J.: Deep learning in neural networks: an overview. *Neu. Netw.* **61**, 85–117 (2015)
7. Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J.: *Gradient Flow in Recurrent Nets: The Difficulty Of Learning Long-term Dependencies*. IEEE Press, New York (2001)
8. Kolen, J.F., Kremer, S.C.: Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In: *A Field Guide to Dynamical Recurrent Networks*, pp. 237–243. IEEE, (2001)

9. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition (2015). [arXiv:1512.03385](https://arxiv.org/abs/1512.03385)
10. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
11. Chen, Y., Lin, Z., Zhao, X., Wang, G., Gu, Y.: Deep learning-based classification of hyperspectral data. *IEEE J. Appl. Earth Obs. Remote Sens.* **7**(6), 2094–2107 (2014)
12. Tao, C., Pan, H., Li, Y., Zou, Z.: Unsupervised spectral-spatial feature learning with stacked sparse autoencoder for hyperspectral imagery classification. *IEEE Explore Geosci. Remote Sens.* **8**(6), 2381–2392 (2015)
13. Kussul, N., Lavreniuk, M., Skakun, S., Shelestov, A.: Deep learning classification of land cover and crop types using remote sensing data. *IEEE Explore Geosci. Remote Sens.* **14**(5), 778–782 (2017)
14. Makantasis, K., Karantzalos, K., Doulamis, A., Doulamis, N.: Deep supervised learning for hyperspectral data classification through convolutional neural networks. In: *Geoscience & Remote Sensing* (2015)
15. Chen, Y., Jiang, H., Li, C., Jia, X., Ghamisi, P.: Deep feature extraction and classification of hyperspectral images based on CNN. *IEEE Trans. Geosci. Remote Sens.* **54**(10), 6232–6251 (2016)
16. Romero, A., Gatta, C., Camps-Valls, G.: Unsupervised deep feature extraction for remote sensing image classification. *IEEE Trans. Geosci. Remote Sens.* **54**(3), 1349–1362 (2016)
17. Mou, L., Ghamisi, P., Zhu, X.: Deep recurrent neural networks for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **55**(7), 3639–3655 (2017)
18. Mou, L., Ghamisi, P., Zhu, X.: Unsupervised spectral-spatial feature learning via deep residual conv–deconv network for hyperspectral image classification. *IEEE Trans. Geosci. Remote Sens.* **56**(1), 391–406 (2018)
19. Glorot X., Bengio Y.: Understanding the difficulty of training deep feedforward neural networks. In: *Proceedings International Conference Artificial Intelligence Statistics*, pp. 249–256 (2010)
20. Luo, L., Xiong, Y., Liu, Y., Sun, X.: Adaptive gradient methods with dynamic bound of learning rate (2019). [arXiv:1902.09843](https://arxiv.org/abs/1902.09843)
21. Grana, M., Veganzons, M.A., Ayerdi, B.: Hyperspectral remote sensing scenes (2020). http://www.ehu.es/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes. Grupo De Inteligencia, Computacional
22. Fawcett, T.: An introduction to ROC analysis. *Pattern Recogn. Lett.* **27**, 861–874 (2006)
23. Agresti, A.: *Categorical Data Analysis*, p. 413. Wiley, Hoboken (2002). ISBN 978-0-471-36093-3
24. Hu, W., Huang, Y., Wei, L., Zhang, F., Li, H.: Deep convolutional neural networks for hyperspectral image classification. *J. Sens. S.I. Deep Learn. Remote Sens. Image Underst.*, art. no. 258619 (2015)
25. Dosovitskiy, A., Springenberg, J. T., Brox, T.: Learning to generate chairs, tables and cars with convolutional networks. In: *Proceedings IEEE Conference Computer Vision Pattern Recognition*, pp. 1538–1546 (2015)
26. Dosovitskiy, A., Fischer, P., Springenberg, J.T., Riedmiller, M., Brox, T.: Discriminative unsupervised feature learning with exemplar convolutional neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **38**(9), 1734–1747 (2016)
27. Demertzis, K., Iliadis, L.: Adaptive elitist differential evolution extreme learning machines on big data: intelligent recognition of invasive species. In: *Proceedings of the INNS Conference Advances in Big Data, Advances in Intelligent Systems and Computing*, vol. 529. Springer, Heidelberg (2016)

28. Demertzis, K., Iliadis, L., Anezakis, V.: A deep spiking machine-hearing system for the case of invasive fish species. In: Proceedings IEEE-SMC Innovations in Intelligent Systems & Applications (INISTA), pp. 23–28 (2017)
29. Demertzis, K., Tziritas, N., Kikiras, P., Sanchez, S.L., Iliadis, L.: The next generation cognitive security operations center: adaptive analytic lambda architecture for efficient defense against adversarial attacks. *Big Data Cogn. Comput.* **3**, 6 (2019). <https://doi.org/10.3390/bdcc3010006>
30. Demertzis, K., Iliadis, L.S., Anezakis, V.D.: Extreme deep learning in biosecurity: the case of machine hearing for marine species identification. *J. Inf. Telecommun.*, 1–19 (2018). Taylor & Francis