Scene Classification of Multispectral Satellite Images Using Convolutional Neural Networks

Anusha Manur
Sri Jayachamarajendra College of Engineering
anusha.manur@gmail.com

P G Diwakar
Indian Space Research Orgnisation
diwaa6@gmail.com

August 3, 2017

Abstract

The availability of high-resolution remote sensing data has opened up the possibility for interesting applications, such as feature classification of satellite images. The aim of this work is to show how a convolutional neural network can be applied to multispectral images to achieve this classification. This exploration uses National Remote Sensing Centre, India (NRSC), obtained remotely sensed data. Various design choices of the CNN architecture are evaluated and analyzed. The ROIs (regions of interest) selected, comprise of four classes i.e. water bodies, vegetation, forest and barren land. The results show that CNNs are a viable tool for solving segmentation tasks in the area of remote sensing.

Nomenclature

CNN Convolutional neural networks

NRSC National Remote Sensing Centre

RELU Rectified linear units

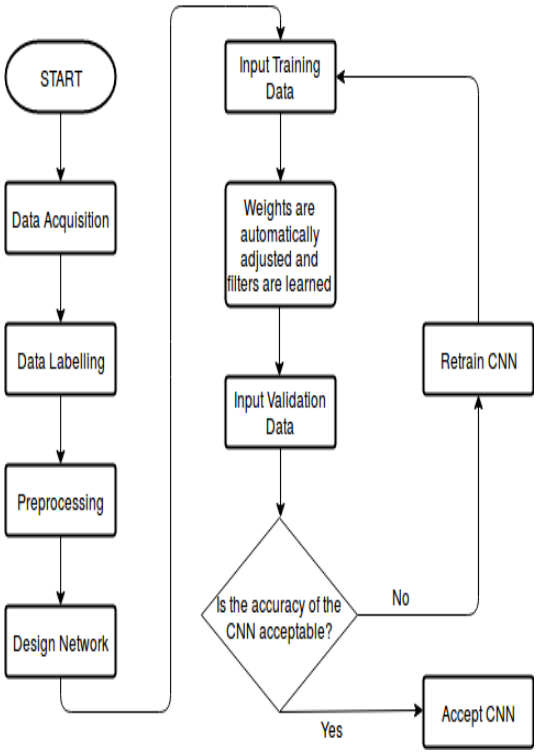
I. Introduction

Aerial and satellite imagery datasets are a vital source of information for observation of the earth's surface and is important for various application domains, such as terrain mapping and vegetation monitoring. An extremely large volume of satellite image data is currently available, but it has little meaning unless it is processed and the required information is retrieved. Traditional applications of satellite remote sensing technology have shown that certain spectral bands can be used effectively in the identification and classification of terrain [1]. Most methods use the gray scale values of a set of corresponding pixels taken from different spectral bands of the same scene to determine the type of terrain present. Fully connected networks are used where every neuron in one layer is connected to every neuron in another layer. However, a single ground cover usually occupies a region of neighboring pixels and improved identification may be obtained by considering an entire region rather than a single pixel. Convolutional neural networks include more spatial information in the neighborhood of the pixel to be classified [2]. CNNs use the concept of receptive fields and shared weights and these properties allow to achieve better generalization and lowers the memory requirements for running the network and thus allows the training of larger, more powerful networks.

In this study, a thorough analysis is done on the application of convolutional neural networks for terrain classification. Remotely sensed data obtained from the NRSC is preprocessed and manually labelled. An unsupervised K-Means algorithm is used for this process. This data is further used to train the CNN model and an in-depth analysis of the model is done in terms of various design choices. The network is fine-tuned till the desired architecture with the best accuracy is obtained.

II. Method

Pattern recognition approaches are commonly deployed to recognize the underlying patterns in remotely sensed data. In this study, the analysis started from data acquisition, data labelling, pre-processing, then supervised classification by convolutional neural networks was carried out with accuracy assessment.



The above diagram shows the adopted method for terrain classification.

i. Hardware and Software Platform

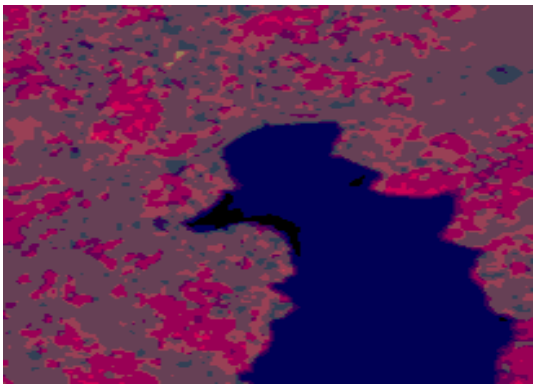
For this study, a computer with an core i7 processor was used. No GPUs were used. The operating system used was Ubuntu 14.04 LTS. The software platform chosen was OpenCV, which is a library of programming functions mainly aimed at real-time computer vision along with Python. For installation in Ubuntu, OpenCV is available in the Ubuntu repository. This was used for image display and manipulations. For implementing neural networks, Keras was used. It is an open source neural network library written in Python. The library contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools which make working with image and text data easier. Keras has a few dependencies which have to be installed beforehand.

- SciPy - It is a Python-based ecosystem of open-source software for mathematics, science, and engineering.
- NumPy - It is the fundamental package for scientific computing in Python. It provides a large assortment of routines for fast operations on arrays. It is one of the core packages of SciPy.
- TensorFlow - An open-source software library for Machine Intelligence.

ii. Data acquisition

Dataset for processing of satellite images was taken from the NRSC repository of remotely sensed data. The data used in this work consists of images of patches of Indian land. They comprise of 3 bands - red, blue, green. The original images obtained are of size 2264*2264 before preprocessing. The biggest challenge is that, though there are plenty of remotely sensed satellite image datasets available, very few are labelled.

This is a major cause of concern as the CNN needs a large amount of labelled data for training to achiever better performance. Therefore, we try to increase the size of the dataset by generating more images from the original data. This is done by splitting the high-resolution images into multiple smaller sized images. Data augmentation is done to boost performance. More images are generated by random rotation, shifts, shear and flips.



The above is a sample image of the dataset. It shows a part of north eastern India captured by the Cartosat satellite launched by ISRO in 2007. It has four classes depicted by the four colors and is majorly covered by water as indicated by the color blue. The hashing of a color to every class allows easier visual classification of the image.

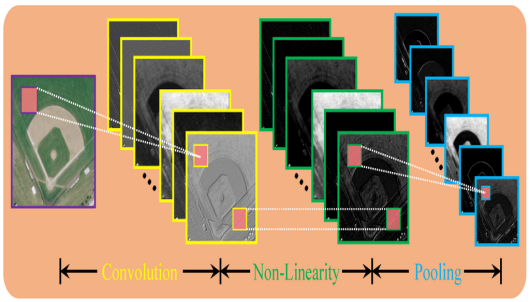
iii. Data preprocessing

In this phase, the obtained images are split to generate more images of size 205*205 each, since the CNN model requires large datasets. The images are then manually labelled using the K-Means method. K-Means clustering algorithm is an unsupervised algorithm and it is used to segment the area of interest from the background. It divides an image into a number of discrete regions such that the pixels have high similarity in each region and high contrast between regions. The pixel values are then cast to floating point values and then normalized.

iv. Implementation of supervised classification using CNN

Convolutional neural networks (CNNs or ConvNets) are designed to process natural signals that come in the form of multiple arrays. They were inspired by biological processes in which the connectivity pattern between neurons is inspired by the organization of the animal visual cortex. A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers are either convolutional, pooling or fully connected. CNNs take advantage of the properties of natural signals by using local connections and tied weights, which makes them easier to train, since they have fewer parameters compared to a fully-connected network and also use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. More specifically, a single CNN layer performs the steps of convolution, non-linear activation and pooling. Firstly, the convolutional layers apply a convolution operation to the input, passing the result to the next layer. Each convolutional neuron processes data only for its receptive field. The convolution operation reduces the number of free parameters and hence resolves the exploding problems in training traditional multi-layer neural network with many layers by back-propagation algorithm where every neuron in one layer to every neuron in another layer. Next, we use activation functions to introduce non-linearity in the network as most real world problems are non-linear. Activation functions cannot be linear because neural networks with a linear activation function are effectively only one layer deep, regardless of how complex their architecture is, because these layers could be summed up to give a linear function. Finally, we add the pooling layers. This combines the outputs of neuron clusters at one layer into a single neuron in the next layer. It is

common to periodically insert a pooling layer between successive convolutional layers in a CNN architecture. The max pooling uses the maximum value from each of the cluster of neurons at the prior layer and the average pooling uses the average value from each of the cluster of neurons at the prior layer.



The above figure [3] shows each step of convolution. The input image is fed to the network. In the convolutional layers, the filters are passed through the entire image to recognize the main features and a feature map is developed. Then, an activation function function is used to introduce non-linearity in the network. A typical choice of activation function for CNNs are hyperbolic tangent or rectified linear units (ReLU). The pooling layer serves to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control overfitting. It downsamples the convolutional layer by computing the maximum (or the mean) over a local non-overlapping spatial region in the feature maps.
The input to the first CNN layer consists of images with 3 spectral bands of of size 205*205. The full architecture consists of four stacked CNN layers , followed by a fully-connected (FC) layer and, finally, a sigmoid classifier. Each convolutional layer uses a kernel size of 3*3 and an ReLU activation function. After every two layers, a 2*2 pooling area is used for maxpooling and a dropout of 0.25 units is used [4]. The preprocessed images are then pushed through the network and the filters are learned by applying them to the entire image and the feature map is assigned to its contextual area. The parameters of the network and the sigmoid classifier are trained with back propagation using an Adam optimiser and a binary crossentropy loss function. The model is then evaluated and the training result is obtained for the predefined metrics. The model is now trained. The testing data is then pushed through the network and the prediction is obtained. The result is a vector of four values consisting of binary digits where 1 indicates the presence and 0 indicates the absence of a class.

III. Results

The optimal design for the model has been derived by analysing the effect of various parameters.

- Number of convoutional layers
- Activation function
- Optimiser function
- Loss function
- Dropout
- Number of epochs

The labelled data consists of 211 images and 25% of it was used for testing. The optimised model gives an accuracy of 92%.

IV. Conclusion

The study shows that CNNs can be used by analysing various architectural parameters and then evaluating them for multilabel classification. It was conluded that an accuracy of 92% could be obtained with given model and labelled input data. Future work includes improvement in the method of manual labelling of input data whch will lead to better learning. The number of input images can be increased by data augmentation. Different kinds of input data will enable the network to train in a more

dynamic way. This model can be improved to obtain a pixel-level classification rather than a feature based classification of images.

References

[1] Marco Castelluccio, Giovanni Poggi, Carlo Sansone, Luisa Verdoliva *Land Use Classification in Remote Sensing Images by Convolutional Neural Networks*.

[2] Y Chen, X Zhao, X Jia *SpectralâĂŤSpatial classification of hyperspectral data based on deep belief network*. IEEE Journal.

[3] Martin LÃd'ngkvist, Andrey Kiselev, Marjan Alirezaie, Amy Loutfi *Classification and Segmentation of Satellite Orthoimagery Using Convolutional Neural Networks*.

[4] N Srivastava, G Hinton,A Krizhevsky, I Sutskever, R Salakhutdinov. *Dropout: A simple way to prevent neural networks from overfitting*.