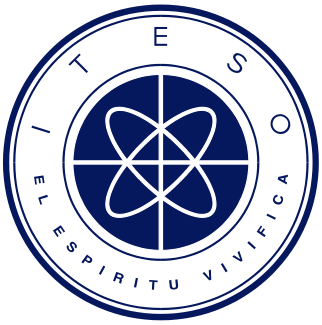

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

DEPARTAMENTO DE ELECTRÓNICA, SISTEMAS E INFORMÁTICA

DEEP LEARNING CURSE

PROFESOR: FRANCISCO CERVANTES

Satellite Image Classification with Deep Learning

PROYECT DEFINITION

PRESENTED BY:

CARLOS ALBERTO CORDERO ROBLES MS197686

11/09/2019

SEPTEMBER, 2019

TLAQUEPAQUE, MÉXICO

1 https://github.com/fMoW/dataset

Contents

1 https://github.com/fMoW/dataset

1 https://github.com/fMoW/dataset

Introduction

Deep Learning is a kind of model of “Machine Learning” that represents different levels of abstractions and layer layers. The last years many amazing achievements in detection and classification when using large Neural Networks CNN and GPUs.
Every year, many contest related with satellite images classification are performed. One example of this kind of contest is “ImageNet” and using CNN has dominated the contest. Many works have done on Deep Learning the progress obtained is huge, although the achievements obtained using satellite images is imitated and the reason is the difficulty to process such images efficiently with CNNs. To maintain the processing reasonable the image are imitated to 299x299. We can consider that the main reason of the scarce results is the lack of labeled datasets.

Problem description

The problem to solve is recognition in high-resolution multi-spectral satellite imagery. To achieve this, we are going to use CNN.
The data base will be IARPA Functional Map of the World (fMoW) that is a data set that gathers information from many telescopes and classify the images in 62 classifications plus false detection. Reference [1] with nominal 0.5 meter per pixel and with more spectral bands as near IR plus the metadata.
The architecture that is going to be used is CNN plus a fully connected network. This doesn’t require any algorithm for feature detection.
Complications:

C.1. One complication is that the satellite images are too big. Normally the images taken are small, otherwise it will take too long to process them. Example for ResNet and DenseNet: 224x224, Inception: 299x299. Normally the dataset (Images) are cropped to fit this size. Satellite images have thousands of pixels and the object to recognize also can have thousands of pixels then it is not useful to crop it.
C.2. Other complication is the orientation of the object, for example in an image of a person the head will be most of the time at the top and the feet at the bottom.
C.3. Another complication are the clouds.
C.4. Datasets.

1 https://github.com/fMoW/dataset

Solutions:
The dataset that is going to be used (IARPA-fMoW) seems to be large and complete enough to cover C.4.
The metadata comes with the bounding box of the image, we can crop the images to such box instead of all the image, that will help to solve C.1. This bounding box can be a little bit enlarged to provide more context of the environment and it also can be adjusted to make it square.
NIR spectrum trespass the clouds that can provide important information in case of C.3. Although, that means that more than the classical three layers might be used and that is quite costly in computer processing. One solution already proposed in papers[2] is to skip all the images with mire that 40% of cloud cover and box sizes smaller than five pixels. For the sequences, the same method is applied and an average value is taken with all the outputs.
The images of the dataset were multiplied by 1 flip and 3 rotations the images 90°,180° and 270°, this also helps to solve point C.2. One image is generating seven more. We are analyzing static objects then image flipping and rotation is not affecting.

Training data

The Database that is going to be used in this experiment is the one provided by IARPA Functional Map of the World (fMoW)[1]. This database has two formats. The full version that contain four or in some cases eight layers. Those images that belong to the full dataset are in TIFF format and all together are close to three and a half terabyte. There is another format compacted in RGB and that has the classical (Red, Green and Blue) layers. This second database is the one that is going to be used. This database is already separated in Training, Test, and Sequences. For this experiment, just Training and Test databases are going to be considered. As mentioned in previous experiments[2] this database will be cropped using the bounding box and all the images with cloud levels bigger than forty percent or images smaller than 5 pixels will be removed.

Training algorithm

Keras is used to create the CNN plus the full connected network, the input is the image and the metadata, four CNNs different are used they are connected to a similar full connected, a mean values of the 63 classifications are taken form the four systems and the maximum value determines the classification.

1 https://github.com/fMoW/dataset

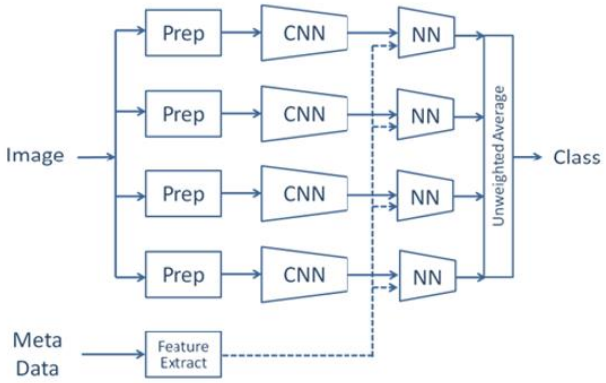

Figure 1Deep Learning Classification System

The CNNs used are going to be DenseNet-161 [7], ResNet-152 [6], Inception-v3 [5] and Xception [8].
The input of the NN is the output of the CNN(63 values) plus 27 values from the metadata but they have to be processed and normalized between -1 and 1.
The NNs layers are 90 size input layer, 1024 size hidden layer with a dropout of 0.6 and a 63 size output layer, softmax is the activation function of the output layer.

Training
90% of false detections for train 10% for testing. The other ones 87% for training and 13% for testing.
The CNN were pre-trained using models of the community [3,4], or ImageNet [9-13]. They are using just one epoch. A second epoch can be executed with a reduced learning rate.

NN were trained for 20 epochs or until the validation loss stopped decreasing. The training can be separated [14].

DenseNet-161
They alleviate the vanishing-gradient problem and reduce the number of parameters. Each layer obtains additional inputs from all preceding layers. ResNets use a similar strategy and have shown that many layers contribute very little and can be randomly dropped without problems.

Compared to Inception networks, which also concatenate feature form different layer, DenseNets are simpler and more efficient.
Normally CNN the last layers have more smaller but more filters, that is expected. That is not a problem, but as proposed here we are going to concatenate all the previous outputs, logically this is not possible if all the filters are not of the same size. Then we apply the strategy mentioned but in blocks. All the filters in the same block have the same size. Although between block and block we have a batch normalization (BN) a 1x1 CNN of one layer and pooling (2x2 average) to reduce the size.

1 https://github.com/fMoW/dataset

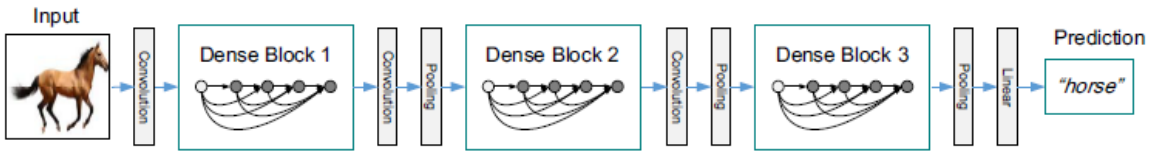

Figure 2DenseNet Blocks diagram

Layers	Output Size	DenseNet-121($k = 32$)
Convolution	112 × 112	
Pooling	56 × 56	
Dense Block (1)	56 × 56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	
	28 × 28	
Dense Block (2)	28 × 28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	
	14 × 14	
Dense Block (3)	14 × 14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$
Transition Layer (3)	14 × 14	
	7 × 7	
Dense Block (4)	7 × 7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$
Classification Layer	1 × 1	

Figure 3DenseNet-121 Architecture

Where k is the number of filters(growth rate). Note that the input image for ImageNet is 224x224.

The filters at each block are a combination of BN-ReLU-Conv(1×1 X 32)-BN-ReLU-Conv(3×3 X 32) for DenseNet 121.

For all the DB excepting ImageNet the first Convolution is of size k=16 or k=2k. Convolutions 3x3 the padding=same. For ImageNet k=32.
The architecture for all de DB excepting imageNet was: {L = 40, k = 12}, {L = 100, k = 12} and {L = 100, k = 24}.

At the end of the last dense block, a global average pooling is performed and a softmax classifier is attached.

Training is with all the networks are trained using stochastic gradient descent (SGD). For imageNet 90 epochs, Initial lr=0.1, 0.01 after epoch 30 and 0.001 after epoch 60.

1 https://github.com/fMoW/dataset

Inception-v3

Designed to perform well even under strict constraints on memory and computational budget.
Design Principles

Principles:
1) The representation size should decrease from the inputs to the outputs.
2) Highly dimensional layers are easier to train in tiles.
3) Reduce the dimensions promotes faster learning.
4) The shall be a balance between the filters per layer and the number of layer, in general the best solution is to increase them in parallel, but the computational power has a limit.

There is a principle of factorization with the deep learning, for example we can reduce one layer 5x5 with two layers 3x3. With this we can reduce from 25 parameters to 2(9) = 18 parameters. We can even factorize in asymmetric filters for example one layer filter 3x3 can be factorized in one layer filters 3x1 and after that one layer filter 1x3. In general we can factorize any nxn layer into a 1xn convolutional followed by a convolutional nx1. It seems not to be good in early layer but seem to have good results with inputs between 12 and 20. Very good results with 7x1 and 1x7.

Following the principle 1) the goal is to reduce the parameters that can be achieved by applying convolution and then pooling but the results have to be concatenated.

Inception-v3 is an improvement of the architecture ILSVRC2012.

The architecture is the following.

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/2	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Figure 4Inception-V3 Architecture

The layer that has padded means that padding=same in keras/Tensorflow. The last layer is for classification and we are going to execute it in a different way. The one highlighted will

1 https://github.com/fMoW/dataset

be executed but using just our 63 lavels that we have and this in how we are going to have our 63 outputs.

Linear layer will execute the following operation:

$$\frac{\exp(z_k)}{\sum_{i=1}^{K} \exp(z_i)}.$$

Figure 5Logits operation

Where k is the number of labels and i the number of inputs. This operation is basically a logistic regression with 63 possible labels.
The inception layers are as the ones showed below:

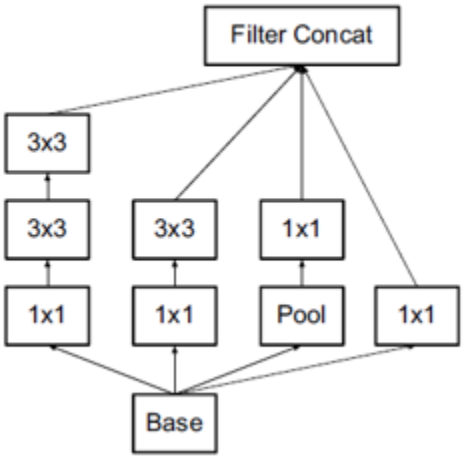

Figure 63xIncepton

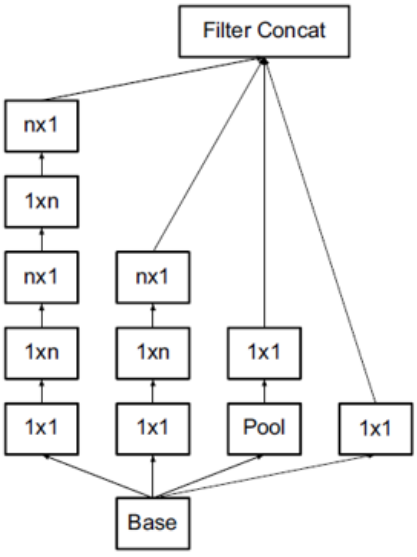

Figure 75xInception

1 https://github.com/fMoW/dataset

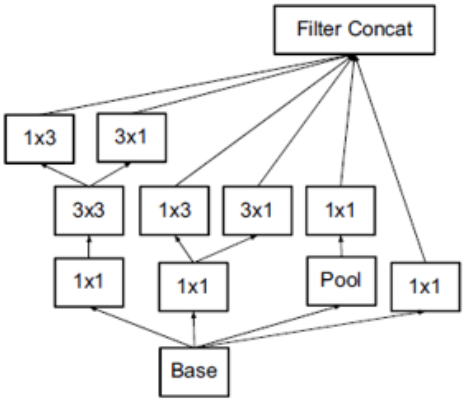

Figure 82xInception

All the layers are 3x3, 3x1 or 1x3 otherwise the filters will not match with the architecture description and inside the inceptions the stride is 1 an padding same excepting the last layer of every inception were the padding is 0 and stride of 2. The concatenation represents that all the output of every path are going to be placed one after the other.

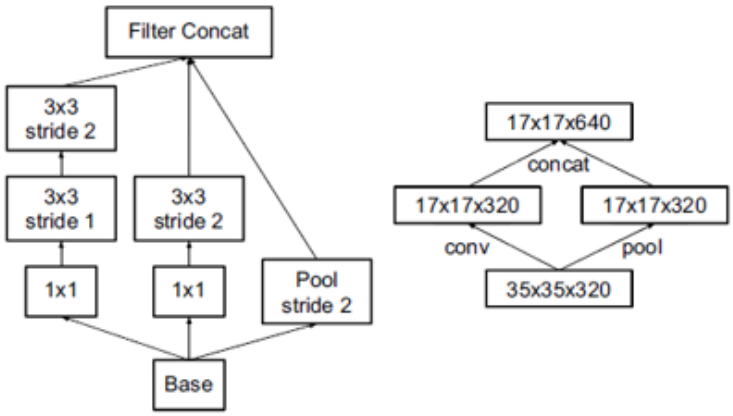

Figure 9Inception Stride and concatenation

Conclusion

The solution that I am trying to replicate consist on four CNN followed by a fully connected network, after that an evaluation of a softmax layer will determine the prediction. There are problems related with satellite images, some of them related with the size of the images, the orientation, the clouds and the datasets. The database that is we are using is large and labeled but the rest of the problems are going to be solved applying a preprocessing to the database.

1 https://github.com/fMoW/dataset

The architectures that are used in this system are specially designed to reduce the memory and processing that is an important milestone when we are working with satellite images if we consider the size that they have.
The goal of this experiment is to replicate the system and try to improve it, unfortunately the scope is not achievable in this curse. Then, the experiment will be bounded just to the DenseNet.
The scope no is to analyze the error of this CNN and see how to improve the results.
The other Networks will be designed in the next semester to complete the system.

1 https://github.com/fMoW/dataset

References

[1] “Earth on AWS: Functional Map of the World,” Amazon.com, https://aws.amazon.com/earth/.
[2] Mark Pritt, and Gary Chern “Satellite Image Classification with Deep Learning” IEEE 978-1-5386-1235-4/17/$31.00 2017
[3] “Applications”, Keras, https://keras.io/applications/.

[4] F. Yu, “CNN Finetune”, Github, 2017, https://github.com/flyyufelix/ cnn_finetune.
[5] C. Szegedy et al., “Rethinking the Inception Architecture for Computer Vision,” IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

[6] K. He et al., “Deep residual learning for image recognition,” arXiv 1512.03385, Dec 2015.

[7] G. Huang, “Dense connected convolutional neural networks,” IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 2017.
[8] F. Chollet, “Xception: deep learning with depthwise separable convolutions,” arXiv 1610.02357, Oct 2016.
[9] Y. Liang, S. Monteiro, and E. Saber, “Transfer learning for high-resolution aerial image classification,” IEEE Workshop Applied Imagery Pattern Recognition (AIPR), Oct 2016.

[10] M. Castelluccio, G. Poggi, and L. Verdoliva, “Land Use Classification in Remote Sensing Images by Convolutional Neural Networks,” arXiv 1508.00092, Aug 2015.

[11] G. Scott, M. England, W. Starms, R. Marcum, and C. Davis, “Training deep convolutional neural networks for land–cover classification of high-resolution imagery”, IEEE Geoscience and Remote Sensing Letters, vol. 14, no. 4, pp. 549-553, Apr 2017.

[12] A. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, “CNN features off-the-shelf: an astounding baseline for recognition,” IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 512-519, 2014.

[13] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, “How transferable are features in deep neural networks?” Proc. 31st Int. Conf. Machine Learning, vol. 32, pp. 647-655, 2014.
[14] G. Christie. N. Fendley, J. Wilson, and R. Mukherjee, “Functional map of the world,” arXiv 1711.07846, 21 Nov 2017.

1 https://github.com/fMoW/dataset