LandUse / LandCover Classification using ResNet50
In search for better test patch size

Vaasudevan Srinivasan[1], Dr. Yun Zhang[1,2] and Mohammad Rezaee[1,3]

[1]Department of Geodesy and Geomatics, University of New Brunswick
[2]Course Instructor [3]Teaching Assistant

Abstract

With the advancements in Deep Learning, particularly Convolutional Neural Networks, LandUse / LandCover classification in Remote Sensing can be thought of as a computer vision task. This study presents a detailed approach to training a well known ResNet50 model based on Sentinel2 EuroSAT dataset and provides results of experiments conducted on different patch sizes and also provides comparisions between those results.

Keywords: Deep Learning, Convolutional Neural Network, Sentinel2, ResNet50, keras

Introduction

This work is hugely inspired from the intuition provided by Mohammad Rezaee and the work done by Abdishakur [1]. Instead of focussing on the accuracy of the model, this work experiments with the patch sizes. The training and testing image size has to be the same in CNN and this work showcases and compares different patch sizes by performing upsampling and then testing. This work also discusses the results of performing enhancement before testing.

Methodology

Order of the work followed in this study is shown in fig. 1.

ResNet

ResNet is the winner of the classification task in the ILSVRC-2015 competition and it is characterized by a very deep network with 50 / 101 / 152 layers. It was developed by He u. a. [5]. The deep ResNet configuration addresses the vanishing gradient problem by employing a deep residual learning module via additive identity transformations. Specifically, the residual module uses a direct path between the input and output and each stacked layer fits a residual mapping rather than directly fitting a desired underlying mapping. A compressed ResNet model is illustrated in fig. 2

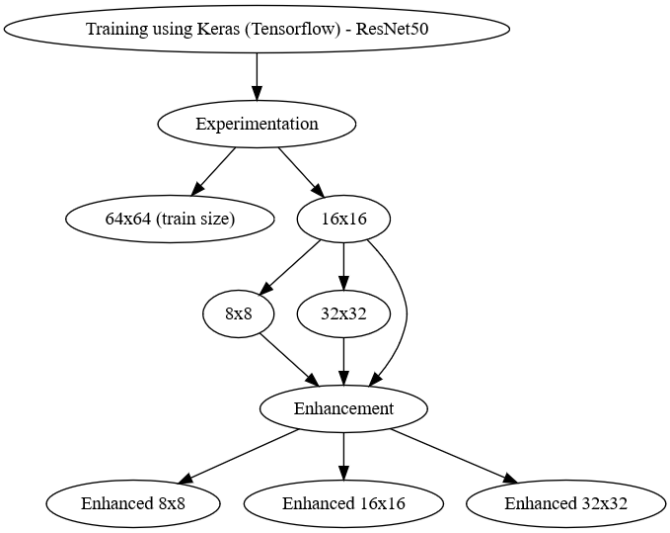


Figure 1: Work order followed in this study

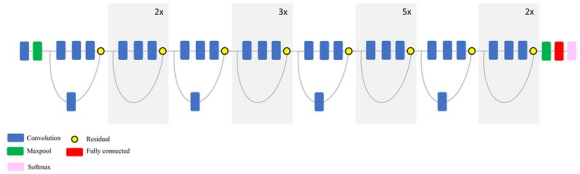


Figure 2: Schematic diagram of ResNet model ([7])

Dataset

EuroSat (A land use and land cover classification dataset based on Sentinel-2 satellite images) dataset was used for training. (Dataset can be downloaded from http://madm.dfki.de/downloads at no cost). Each image has a size of 64 X 64 X 12 where 12 is the number of bands.

The list of bands and its desription can be seen in fig. 3.

Using Matplotlib ([6]), a simple bar plot showing the

Sentinel-2 Bands	Central Wavelength (µm)	Resolution (m)
Band 1 - Coastal aerosol	0.443	60
Band 2 - Blue	0.490	10
Band 3 - Green	0.560	10
Band 4 - Red	0.665	10
Band 5 - Vegetation Red Edge	0.705	20
Band 6 - Vegetation Red Edge	0.740	20
Band 7 - Vegetation Red Edge	0.783	20
Band 8 - NIR	0.842	10
Band 8A - Vegetation Red Edge	0.865	20
Band 9 - Water vapour	0.945	60
Band 10 - SWIR - Cirrus	1.375	60
Band 11 - SWIR	1.610	20
Band 12 - SWIR	2.190	20

Figure 3: Sentinel2 Bands (Satellite-Imaging-Corp)

number of images per class was generated (shown in fig. 4).

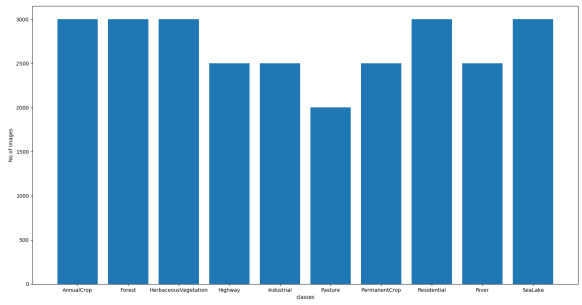


Figure 4: Training images per class

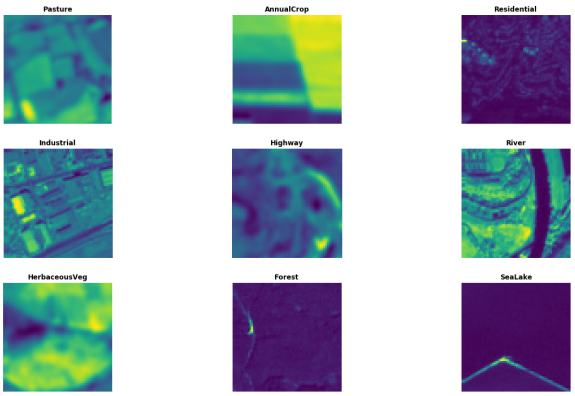


Figure 5: Visualization of classes available in the dataset (Abdishakur)

Implementation

Using Keras.applications module (a high level API for tensorflow) ([3]), ResNet50 model was imported and a dense layer of 10 units (corresponding to 10 classes) with a softmax activation function was added in the end of the model. Rasterio module ([4]) was used to read Green, Blue and Red bands (spatial resolution: 10m). OpenCV module ([2]) was used for upsampling. Adam optimizer was used and the loss function used was sparse_categorical_crossentropy. Below is the partial summary printed when *model.summary()* method is called.

```
_________________________________________________________________________________________________
Layer (type)                 Output Shape              Param #   Connected to
=================================================================================================
input_2 (InputLayer)         [(None, 64, 64, 3)]       0
_________________________________________________________________________________________________
conv1_pad (ZeroPadding2D)    (None, 70, 70, 3)         0         input_2[0][0]
_________________________________________________________________________________________________
conv1_conv (Conv2D)          (None, 32, 32, 64)        9472      conv1_pad[0][0]
_________________________________________________________________________________________________
conv1_bn (BatchNormalization (None, 32, 32, 64)        256       conv1_conv[0][0]
_________________________________________________________________________________________________
conv1_relu (Activation)      (None, 32, 32, 64)        0         conv1_bn[0][0]
_________________________________________________________________________________________________
pool1_pad (ZeroPadding2D)    (None, 34, 34, 64)        0         conv1_relu[0][0]
_________________________________________________________________________________________________
pool1_pool (MaxPooling2D)    (None, 16, 16, 64)        0         pool1_pad[0][0]
_________________________________________________________________________________________________
...
_________________________________________________________________________________________________
conv5_block3_out (Activation (None, 2, 2, 2048)        0         conv5_block3_add[0][0]
_________________________________________________________________________________________________
max_pool (GlobalMaxPooling2D (None, 2048)              0         conv5_block3_out[0][0]
_________________________________________________________________________________________________
dense_1 (Dense)              (None, 10)                20490     max_pool[0][0]
=================================================================================================

Total params: 23,608,202
Trainable params: 23,555,082
Non-trainable params: 53,120
_________________________________________________________________________________________________
```

Training

Training was done for 30 epochs with batch size 500. It was performed in the lab machine with Intel(R) Core(TM) i5-3570S CPU @3.10GHZ; it has 4 cores with 8GB RAM. Training approximately took 18 hours. Below is the minimal log of the training. Accuracy went from 49% to 97% after 30 epochs.

```
Epoch 1/30
27000/27000 [==============================] - 2123s 79ms/sample - loss: 1.7949 - accuracy: 0.4914
Epoch 2/30
27000/27000 [==============================] - 2082s 77ms/sample - loss: 0.7430 - accuracy: 0.7348
Epoch 3/30
27000/27000 [==============================] - 2073s 77ms/sample - loss: 0.5888 - accuracy: 0.7894
...
Epoch 30/30
27000/27000 [==============================] - 2074s 77ms/sample - loss: 0.0580 - accuracy: 0.9784
```

Experiments

For testing (experimenting), 1000X1000 image of Fredericton area was used. Matplotlib was used for plotting and the whole work can be found at https://github.com/VaasuDevanS/DeepLearning

Image size: 1000x1000; Patch size: 64 X 64

First, the test patch size of 64x64 was used (same as the one used for training). The test image was padded on all the sides with null values and 1,000,000 patches were generated and predicted values were logged to a file. This testing for the whole 1000x1000 image took

almost a week and the results did not achieve sufficient accuracy as seen visually in fig. 6. Though the same patch size was used in both training and testing, 64 x 64 contains a huge amount of information for predicting a single pixel (class) value.

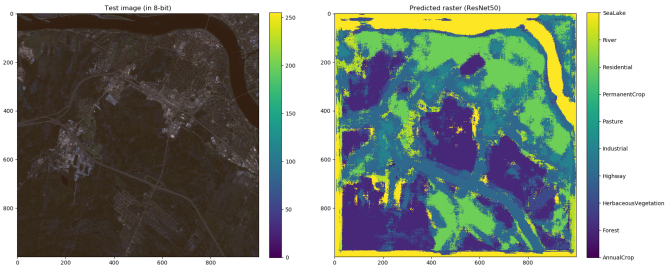


Figure 6: Predictions - 64 x 64 patch size

Image size: 400x400; Patch size: 16 X 16

Next, the test area was reduced to 400x400 due to the excessive computational time for the whole area. This time, a patch size of 16 x 16 was used (this was chosen randomly). Since the model was trained using 64 x 64 images, the testing image should also have the same size. Hence all the 16 x 16 patches were upsampled to match the training image size. The input 400 x 400 image was also padded on all the sides with null values and 160,000 patches were generated and predicted values were logged to a file. This took almost a day and as seen in fig. 7, the results were much clearer than the previous one.

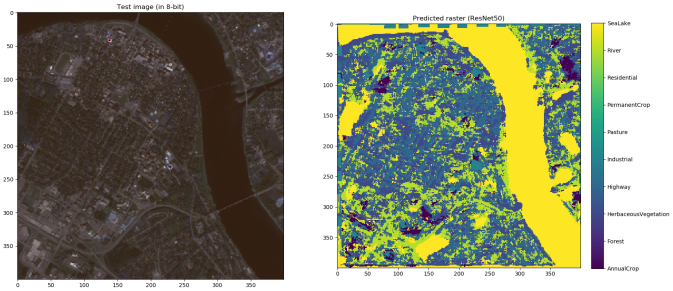


Figure 7: Predictions - 16 x 16 patch size

Comparision: 64x64 vs 16x16

On comparing the results between patch size 64x64 vs patch size 16x16, as shown in fig. 8, 64 x 64 lost to 16 x 16 because of the vague information it holds to predict the class for a single pixel.

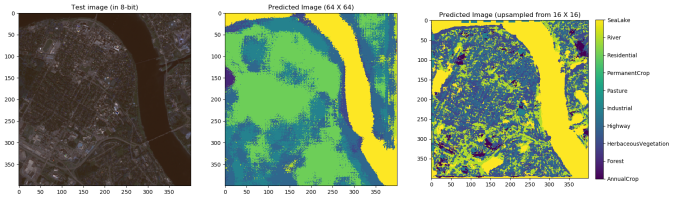


Figure 8: Comparision between 64 x 64 and 16 x 16 patch sizes

Image size: 100x100; Patch size: 8 X 8

Since 16 x 16 results were better than 64 x 64, greedy idea of trying with even lesser patch sizes was considered. For quicker analysis, the test image was even reduced to 100x100 and this time patch size of 8 x 8 was used. After padding the 100x100 image, all the 10,000 patches were upsampled to 64 x 64 and then predictions were saved to a file. As shown in fig. 9, the results were unclear and this was because of the less information available to predict the class for a single pixel.

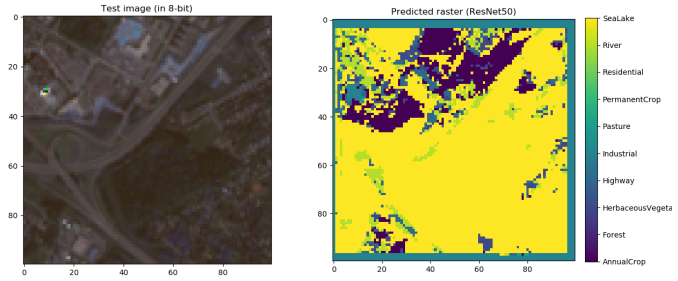


Figure 9: Predictions - 8 x 8 patch size

Image size: 100x100; Patch size: 32 X 32

Next the same approach was followed for 32 x 32 patch size and as shown in fig. 10, the results were clearly unsatisfactory. 32x32 did a good job in predicting pixels in the middle but did a horrible job along the edges.

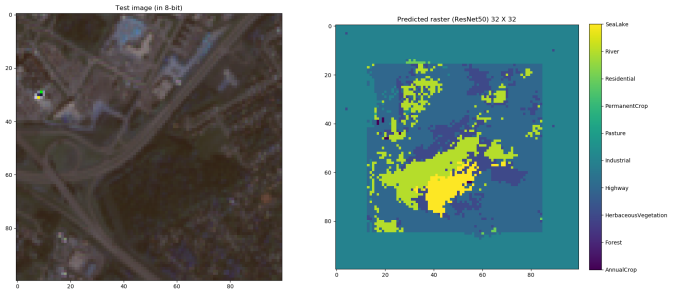


Figure 10: Predictions - 32 x 32 patch size

Comparision: 8x8 vs 16x16 vs 32x32 vs 64x64

The same 100x100 area was extracted from the 1000x1000 - 64x64 precition raster and 400x400 - 16x16 precition raster and then plotted together to get an overall picture of the predicted results. This is shown in fig. 11

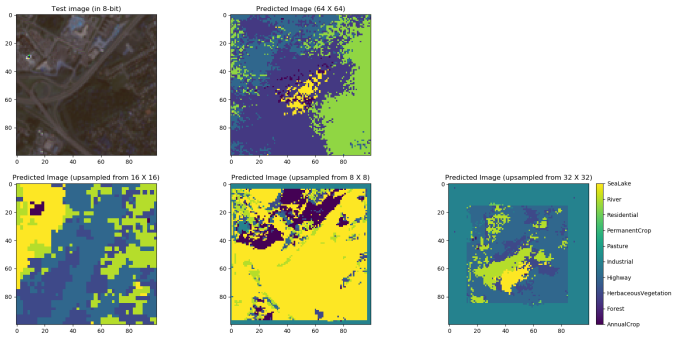


Figure 11: Comparision between 64x64 and 16x16

Comparision: Enhancement

For another set of experimentation, linear enhancement is applied to the test image (100x100) and tested for different patch sizes. The comparison plot is shown in fig. 12. From the results, it is clear that performing enhancement and testing produces even worse results than the predictions without any enhancement.

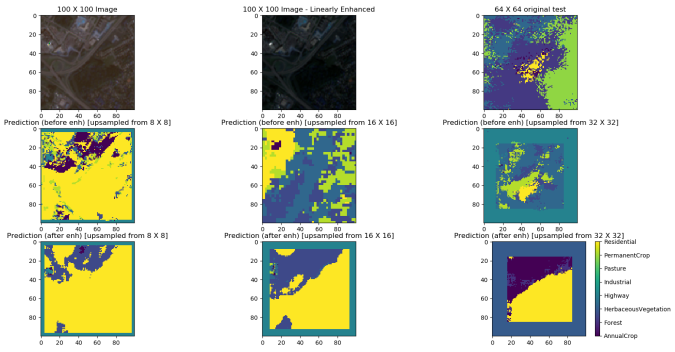


Figure 12: Comparision between predictions (original vs enhanced test image)

Conclusion

This work used various test image sizes because of the computational limitation (Note that the whole work is done on a normal CPU machine. Google Colab was used in the beginning but it was dropped because of the 12-hour running restriction for free accounts). This paper compares and discusses the results of experimenting different patch sizes for testing. This choice actually depends on the dataset and/or the knowledge of the researcher. However to conclude the work, 16x16 patch size (upsampled to 64x64) produces better results than 8x8, 32x32 or 64x64 and performing enhancement only on the test image worsens the result. This is not a strict conclusion to be followed but this work provides a general idea of how results varies for different patch sizes for EuroSat dataset and ResNet50 model.

References

Abdishakur: *Land use/Land cover classification with Deep Learning*. `https://towardsdatascience.com/land-use-land-cover-classification-with-deep-learning-9a5041095ddb`. 2018. – [Online; accessed 24-Aug-2018]

Bradski, G.: The OpenCV Library. In: *Dr. Dobb's Journal of Software Tools* (2000)

Chollet, François u. a.: *Keras*. `https://keras.io`. 2015

Gillies, Sean u. a.: *Rasterio: geospatial raster I/O for Python programmers*. 2013–. – URL `https://github.com/mapbox/rasterio`

He, Kaiming ; Zhang, Xiangyu ; Ren, Shaoqing ; Sun, Jian: Deep Residual Learning for Image Recognition. In: *arXiv preprint arXiv:1512.03385* (2015). – URL `https://arxiv.org/abs/1512.03385`

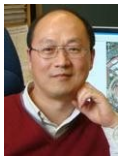
Hunter, J. D.: Matplotlib: A 2D graphics environment. In: *Computing in Science & Engineering* 9 (2007), Nr. 3, S. 90–95. – URL `https://matplotlib.org/`

Mahdianpari, Masoud ; Salehi, Bahram ; Rezaee, Mohammad ; Mohammadimanesh, Fariba ; Zhang, Yun: Very Deep Convolutional Neural Networks for Complex Land Cover Mapping Using Multispectral Remote Sensing Imagery. In: *Remote Sensing* 10 (2018), Jul, Nr. 7, S. 1119. – URL `http://dx.doi.org/10.3390/rs10071119`. – ISSN 2072-4292

Satellite-Imaging-Corp: *Sentinel-2A (10m) Satellite Sensor*. `https://www.satimagingcorp.com/satellite-sensors/other-satellite-sensors/sentinel-2a/`

Contact

Vaasudevan Srinivasan
Graduate Student (course based)
vaasu.devan@unb.ca
Personal Page



Dr. Yun Zhang
Professor
yunzhang@unb.ca
Official Page

Mohammad Rezaee
Post-Doctoral Fellow
Mohammad.Rezaee@unb.ca