A ML/DL Attempt to Identify Changes in Vegetation and Developed Land for Dunhuang City, CN (2016-2020) with EuroSAT and Sentinel-2 Image

MUSA 650 Final Project Report

Group Member: Jiaxuan Lyu, Zhijie Zhou
Github Repository: https://github.com/zjalexzhou/Dunhuang_Sentinel_EuroSat

Introduction

Subjected to dynamic changes triggered by both human and natural factors, the urban ecosystem is vibrant, diverse but at the same time unpredictable and fragile. The collision between residential areas and the natural environment exacerbates at the marginalized regions, like cities and towns which are adjacent to deserts or built on desert oases. Desertification and soil salinization problems are common in those areas in conjunction with water shortage.

In recent years, the improved coverage and resolution of satellite imageries shed lights on the local strategic planning process, as remote sensing offers an informative and reliable way of both qualifying and quantifying land cover changes. Moreover, a novel dataset and deep learning benchmark for land use and land cover classification, EuroSAT, has enriched the pool of ground-truth dataset based on Sentinel-2 images.

Dunhuang is a famous cultural heritage of Buddhism in Gansu Province, China, but it has faced environmental issues like lacking water resources and desertification for long. Several projects started to recover Dunhuang's ecosystems, such as the "Haerteng River-to-Dang River water diversion" program launched in 2015, which aimed to recharge underground water in Dunhuang (Chen et al., 2016). Therefore, in this project, we plan to study the land cover change in Dunhuang as an attempt to help visualize the effects of the ongoing conservation programs in Dunhuang. To be specific, we will emphasize on quantifying the changes of urban and residential areas in Dunhuang within the latest 5-year range using deep learning classification results of Sentinel-2 images with our trained classifier based on the EuroSAT dataset.

Data

Area of Interest

Dunhuang City (Figure 1) features its vast desert landscape surrounding the Dunhuang Oasis, where most of its population, forest canopies, and built-up environments situate. For the scope of this study, the area of interest is restrained to the Dunhuang Oasis, i.e., the rectangle-shape area bounded by (40.00°N, 94.40°E), (40.00°N, 94.95°E), (40.45°N, 94.95°E), and (40.45°N, 94.40°E), as indicated in Figure 2.

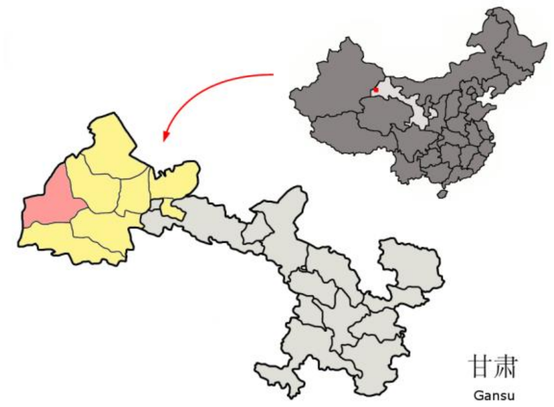


Figure 1. Location of Dunhuang City (pink) within Jiuquan Prefecture (yellow) and Gansu province of China Map drawn in september 2007 using various sources, mainly: Gansu province administrative regions GIS data: 1:1M, County level, 1990 Gansu Counties map from www.hua2.com Jiuquan Prefecture map from gansudaily.com.cn Author: Croquant.

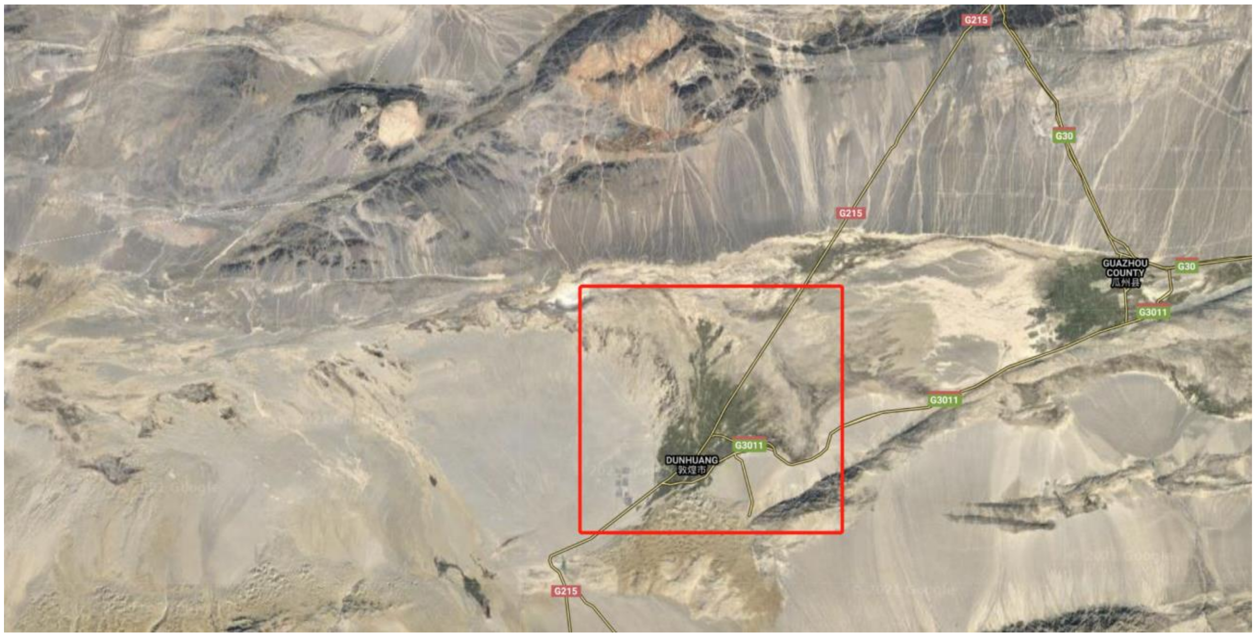


Figure 2. Location of Dunhuang Oasis (within red rectangle). Map. Google Maps. Google, May 2nd, 2021.

Satellite Image

This project involved a land cover classification and change detection workflow for two Sentinel-2 images of Dunhuang Oasis in Summer 2016 and Summer 2020, retrieved using Google Earth Engine scripts at 10-meter resolution. The final image products downloaded were monthly averaged and cloud-masked (using the QA bands, i.e. band 10 and 11) with RGB (visible) bands only. Dimension: 5014 x 6124 x 3 (RGB). Moreover, band 8 (NIR) is downloaded for NDVI calculation.

Ground-truth Image

The project used the RGB-channel version of EuroSAT, a novel dataset based on Sentinel-2 L2A satellite images covering 13 spectral bands and consisting out of 10 classes with in total 27,000 labeled and geo-referenced images. Dimension: 64 x 64 x 3 (RGB).

Method

Preprocessing

Training/Testing Split

The original EuroSAT dataset contains 27,000 images with individual labels out of 10 classes: ['AnnualCrop', 'Forest', 'HerbaceousVegetation', 'Highway', 'Industrial', 'Pasture', 'PermanentCrop', 'Residential', 'River', 'SeaLake']

Then, the dataset is splitted into 50/50 training and testing.

Categories of EuroSAT dataset

Based on the 10 labels that EuroSAT dataset has, we divide them into different categories, in order to improve our result of image classification in the later part. Several categories have been tried, and the results are listed in the result section.

Group	Sub-Categories	EuroSAT Classes Included
Group 1	Green Cover	Forest, HerbaceousVegetation, Pasture
	Built-up Environment	Highway, Industrial, Residential
	Other	AnnualCrop, PermanentCrop, River, SeaLake
Group 2	Non-Pasture Green Cover	Forest, HerbaceousVegetation
	Built-up Environment	Industrial, Residential
	Other	AnnualCrop, PermanentCrop, River, SeaLake, Pasture, Highway
Group 3	Non-Pasture Green Cover	AnnualCrop, HerbaceousVegetation
	Built-up Environment: Residential Only	Residential
	Other	PermanentCrop, River, SeaLake, Pasture, Highway, Forest, Industrial
Group 4	Built-up Environment: Residential Only	Residential
	Other	AnnualCrop, PermanentCrop, HerbaceousVegetation, Highway, Pasture, River, SeaLake, Pasture, Forest, Industrial
Group 5 (Residential & Permanent Crop Removed)	Green Cover	Annual Crop, Forest
	Built-up Environment	Industrial
	Other	HerbaceousVegetation, Highway, Pasture, River, SeaLake

Table 1. Regrouped EuroSAT categories for training and classification.

Image Cutting

For better model performance, it is necessary to cut the Sentinel images into patches matching the size of EuroSAT images used as the training set for this project. Both images were chopped into 7,410 patches and each had a dimension of 64 x 64 x 3 (RGB).

Land Classification

Machine Learning Approach

A SVM Classifier model is trained by using vectorized data with default parameters.

Deep Learning Approach

Our deep learning model in this project is adjusted from the VGG16 model. Based on the VGG 16 model, one flatten layer and two groups of dense and dropout layers are added. The model structure is shown as the following figure.

Moreover, in order to increase our model accuracy, transfer learning approach is also used, as reusing or transferring information from previously learned tasks may facilitate the learning process of a new task (Karimpanal et al., 2019). During the training process, the batch size is set to 128, with 12 epochs and 1 verbose.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 64, 64, 3)]       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 64, 64, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 64, 64, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 32, 32, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 32, 32, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 32, 32, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 16, 16, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 16, 16, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 8, 8, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 8, 8, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 4, 4, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
```

```
block5_conv3 (Conv2D)        (None, 4, 4, 512)        2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 2, 2, 512)        0
_________________________________________________________________
flatten_2 (Flatten)          (None, 2048)             0
_________________________________________________________________
dense_6 (Dense)              (None, 2048)             4196352
_________________________________________________________________
dropout_4 (Dropout)          (None, 2048)             0
_________________________________________________________________
dense_7 (Dense)              (None, 2048)             4196352
_________________________________________________________________
dropout_5 (Dropout)          (None, 2048)             0
_________________________________________________________________
dense_8 (Dense)              (None, 3)                6147
=================================================================
Total params: 23,113,539
Trainable params: 8,398,851
Non-trainable params: 14,714,688
_________________________________________________________________
```

Figure 3. Deep Learning Model Structure

Change Detection

The implementation method of change detection in this project is similar to the idea of map algebra or raster calculator. With the land cover type classification results for both 2016 and 2020 images, we would like to obtain the amount of land changed from natural land cover to a built-up one, or the other way round.

NDVI

Due to the condition that the deep learning model trained by EuroSAT images does not perform well on predicting land categories in Dunhuang, NDVI is also used to detect changes. NDVI, Normalized Difference Vegetation Index, is an approximate index for vegetation existence (Normalized Difference Vegetation Index, Wikipedia, 2021).

NDVI is calculated based on the equation:

NDVI = (NIR - Red) / (NIR + Red)

K-means clustering

4 clusters are set for k-means clustering. After adjusting the output classes (to make sure the clusters match each other between the two images), we calculated the number of transitions happened between classes.

Results

Machine Learning Model - SVM Classifier

```
              precision    recall  f1-score   support

           0       0.64      0.70      0.67      1521
           1       0.62      0.89      0.73      1486
           2       0.39      0.30      0.34      1464
           3       0.74      0.39      0.51      1256
           4       0.91      0.94      0.92      1270
           5       0.36      0.62      0.45       972
           6       0.65      0.50      0.56      1262
           7       0.66      0.79      0.72      1468
           8       0.59      0.54      0.57      1275
           9       0.77      0.48      0.59      1526

    accuracy                           0.62     13500
   macro avg       0.63      0.62      0.61     13500
weighted avg       0.64      0.62      0.61     13500
```

Table 2. Classification Report of SVC on training dataset

```
              precision    recall  f1-score   support

           0       0.46      0.57      0.51      1479
           1       0.62      0.87      0.73      1514
           2       0.24      0.18      0.21      1536
           3       0.47      0.19      0.27      1244
           4       0.74      0.85      0.79      1230
           5       0.34      0.59      0.43      1028
           6       0.35      0.25      0.29      1238
           7       0.50      0.53      0.52      1532
           8       0.44      0.43      0.44      1225
           9       0.73      0.48      0.58      1474

    accuracy                           0.50     13500
   macro avg       0.49      0.49      0.48     13500
weighted avg       0.49      0.50      0.48     13500
```

Table 3. Classification Report of SVC on testing dataset

Deep Learning Model - Training Accuracy

Group	Number of Subcategories	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
1	3	0.9275	0.2109	0.9078	0.3180
2	3	0.9373	0.2030	0.9393	0.3616
3	3	0.8296	0.4286	0.4076	0.8450
4	2	0.9862	0.0517	0.1447	0.9740
5	3	0.9755	0.0785	0.2436	0.9508

Table 4. Summarized Training Accuracy of Deep Learning Model

Deep Learning Model - Prediction Results

(1)There are 1560 predicted results of Green Cover, and there are 1041 predicted results of Built up.

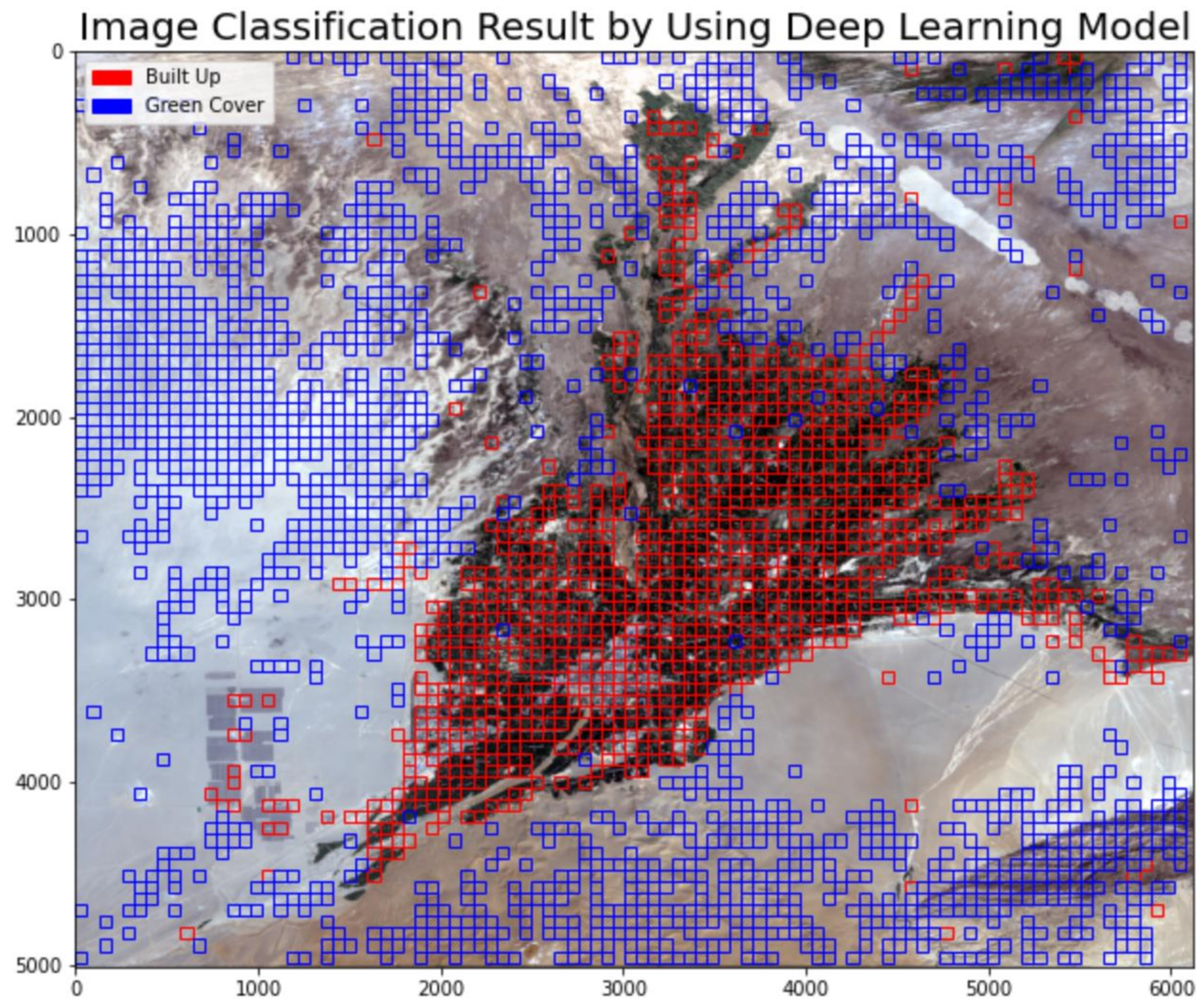


Figure 4. Image Classification Result from Deep Learning Model

(2) There are 1332 predicted results of Green Cover, and 983 predicted results of Built Up.

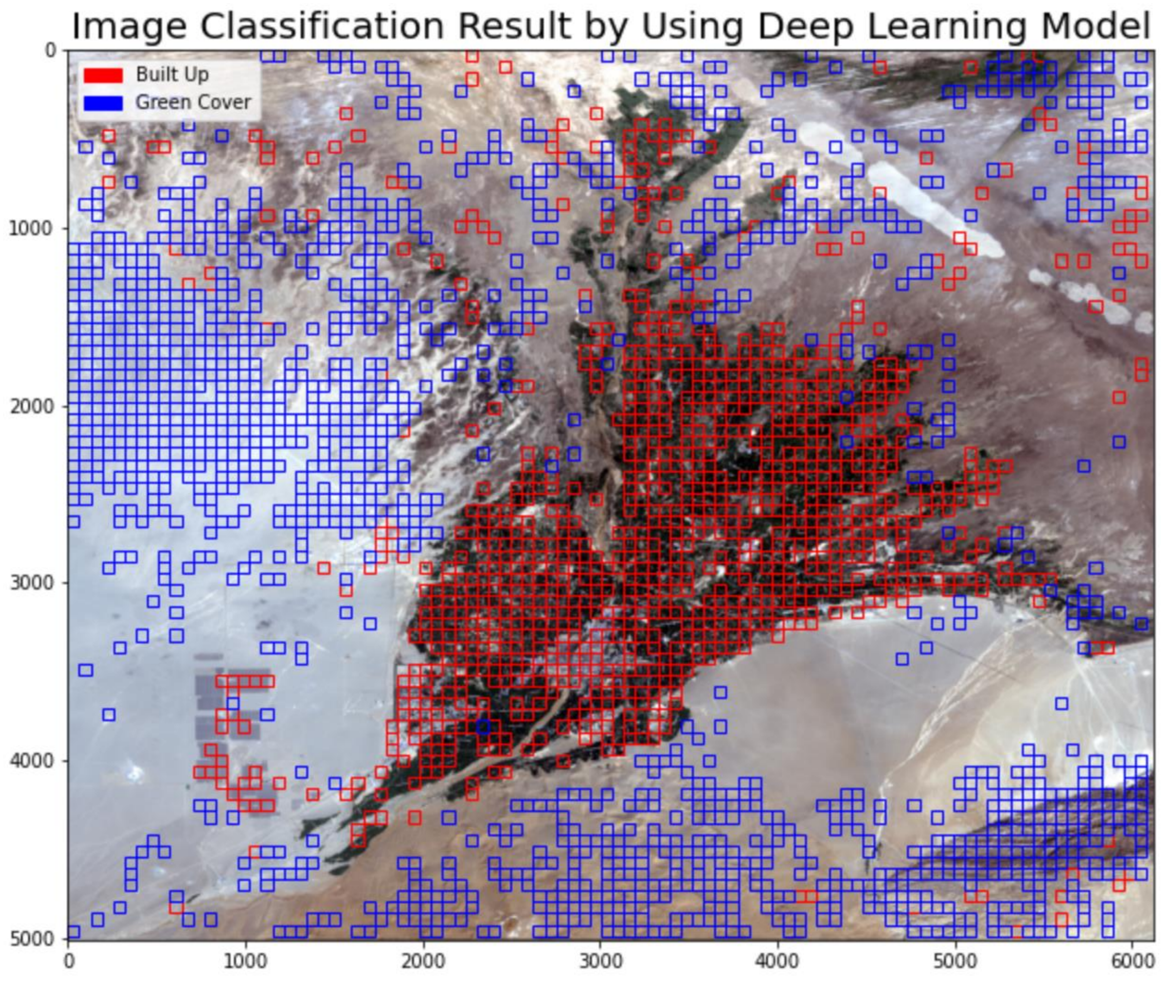


Figure 5. Image Classification Result from Deep Learning Model

(3) There are 1497 predicted results of Green Cover, and 0 predicted result of Built Up

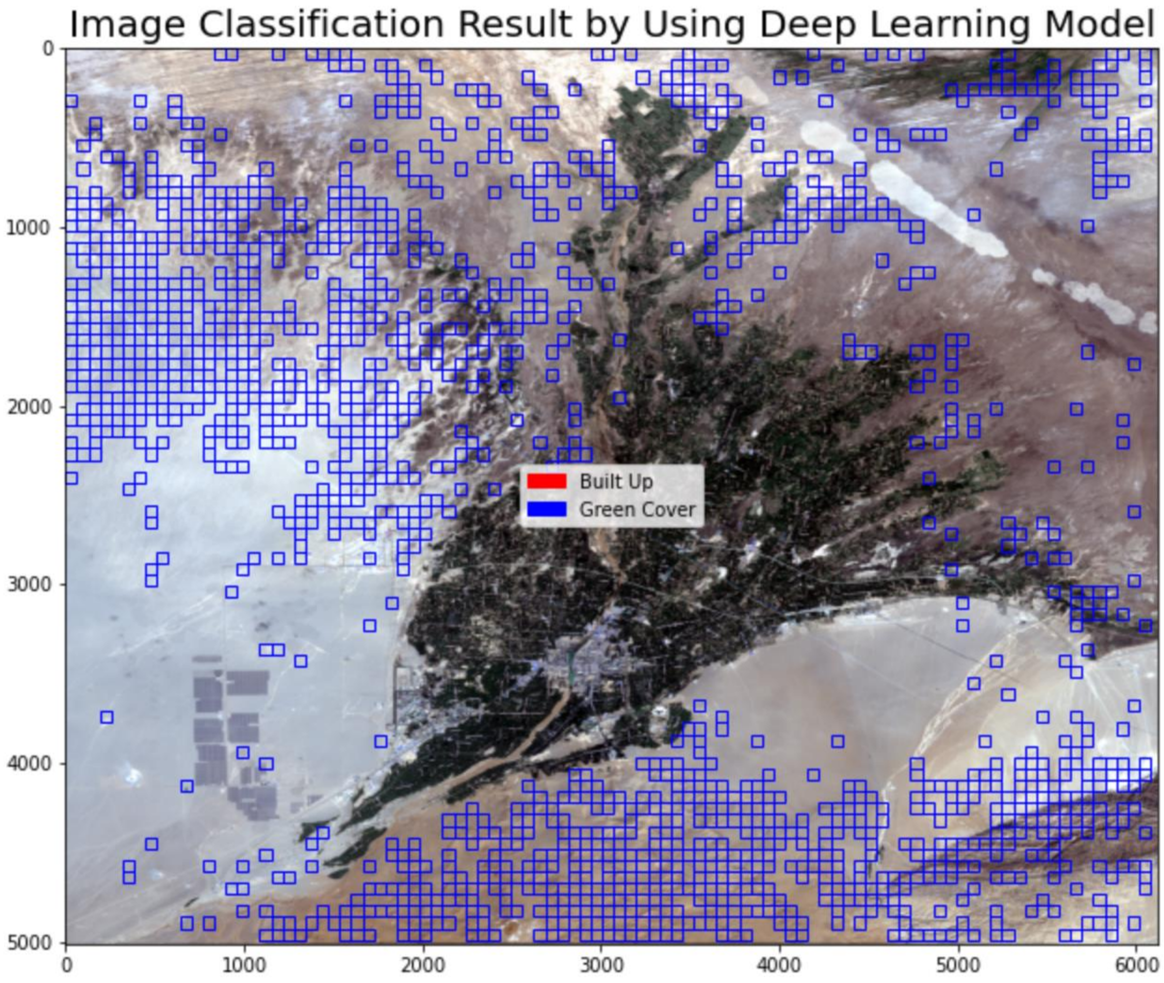


Figure 6. Image Classification Result from Deep Learning Model

(4)There are 13 predicted results of Built Up

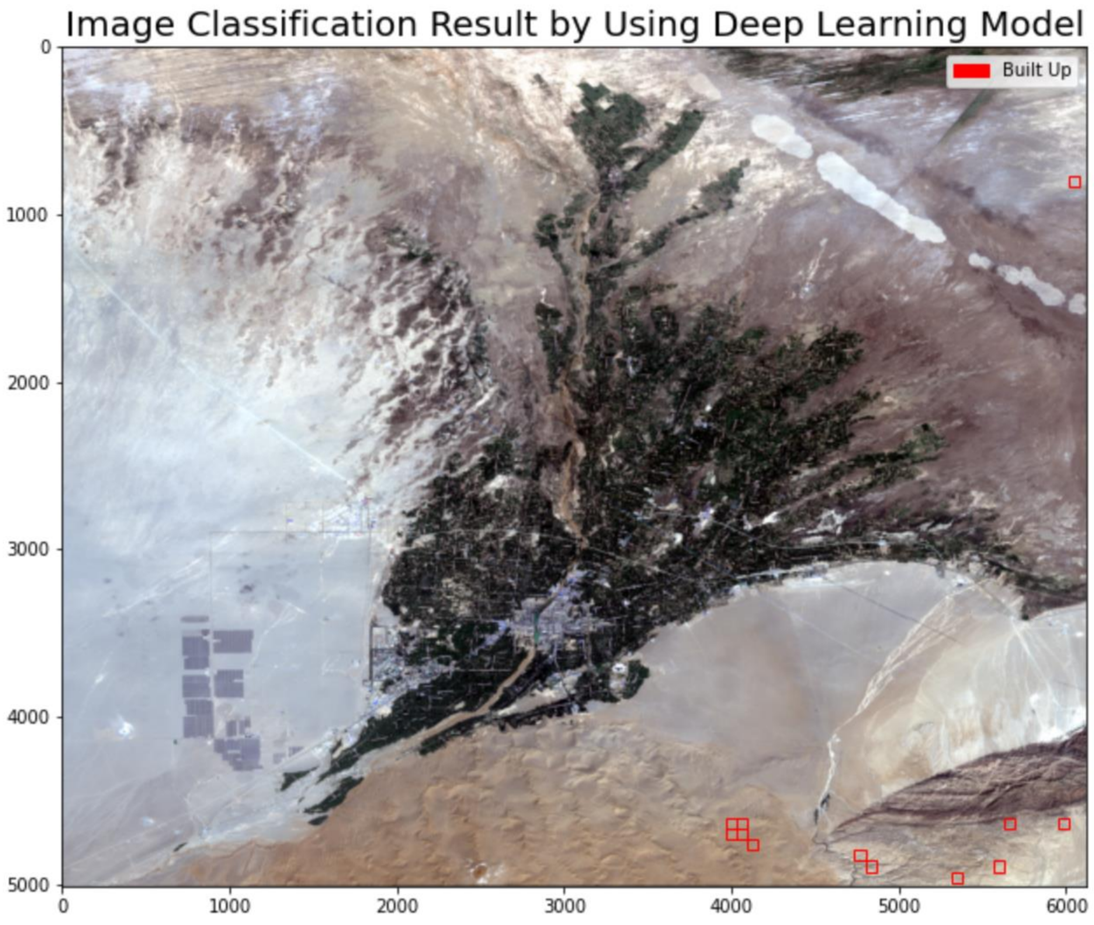


Figure 7. Image Classification Result from Deep Learning Model

(5) There are 310 predicted results of green cover, and 1551 predicted results of built-up

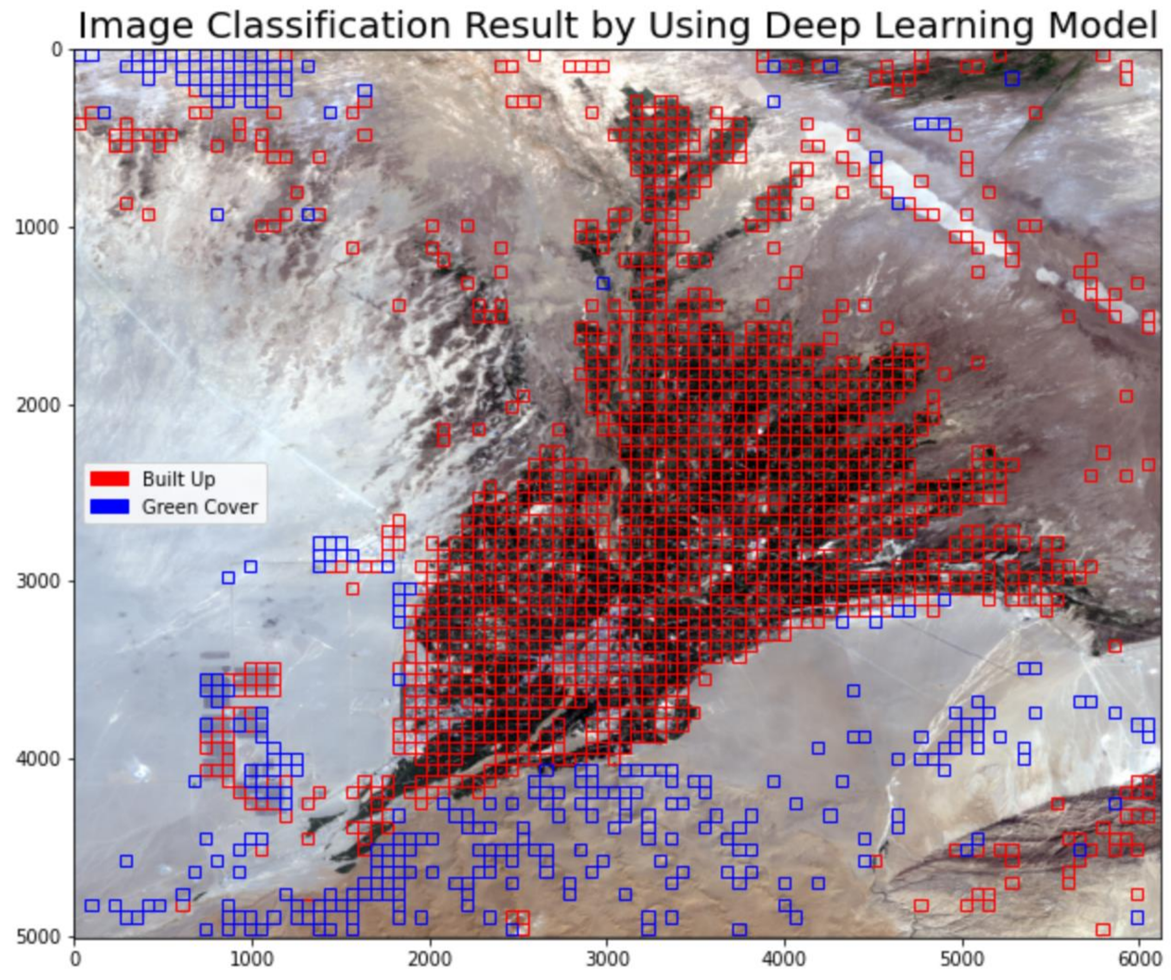


Figure 8. Image Classification Result from Deep Learning Model

Detected Changes

NDVI

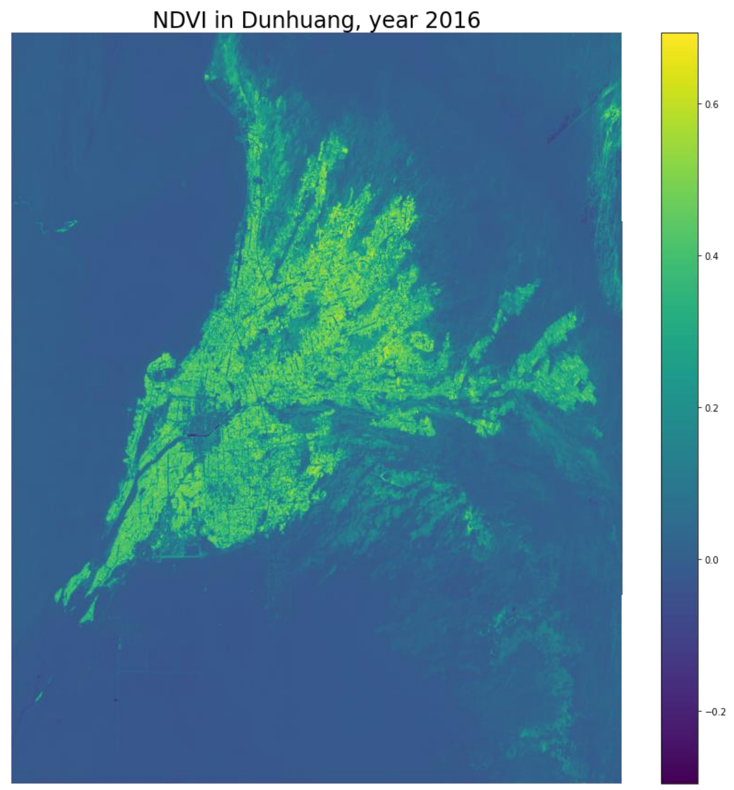


Figure 9. NDVI image of Dunhuang in Year 2016

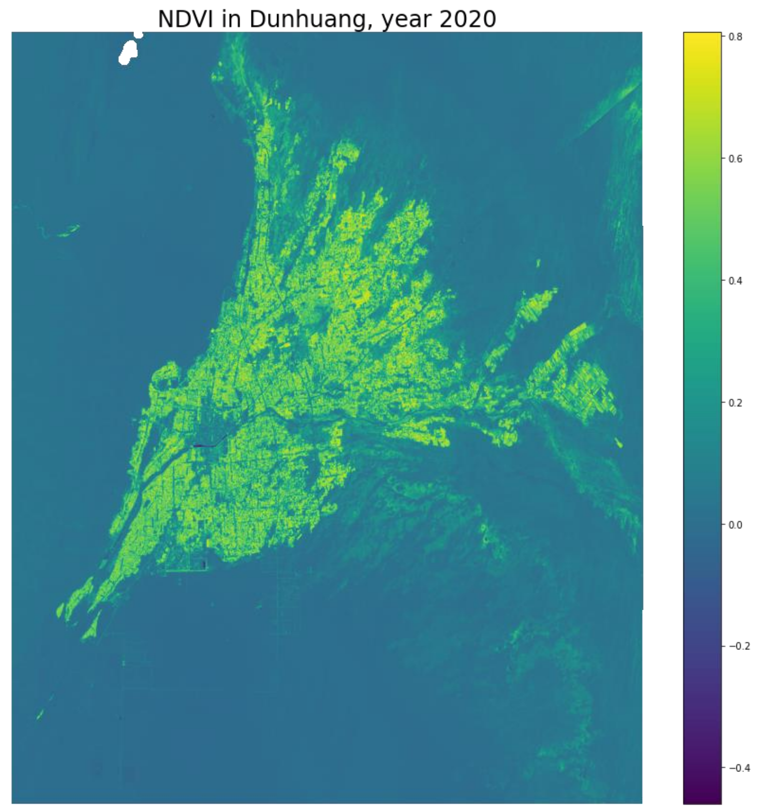


Figure 10. NDVI image of Dunhuang in year 2020

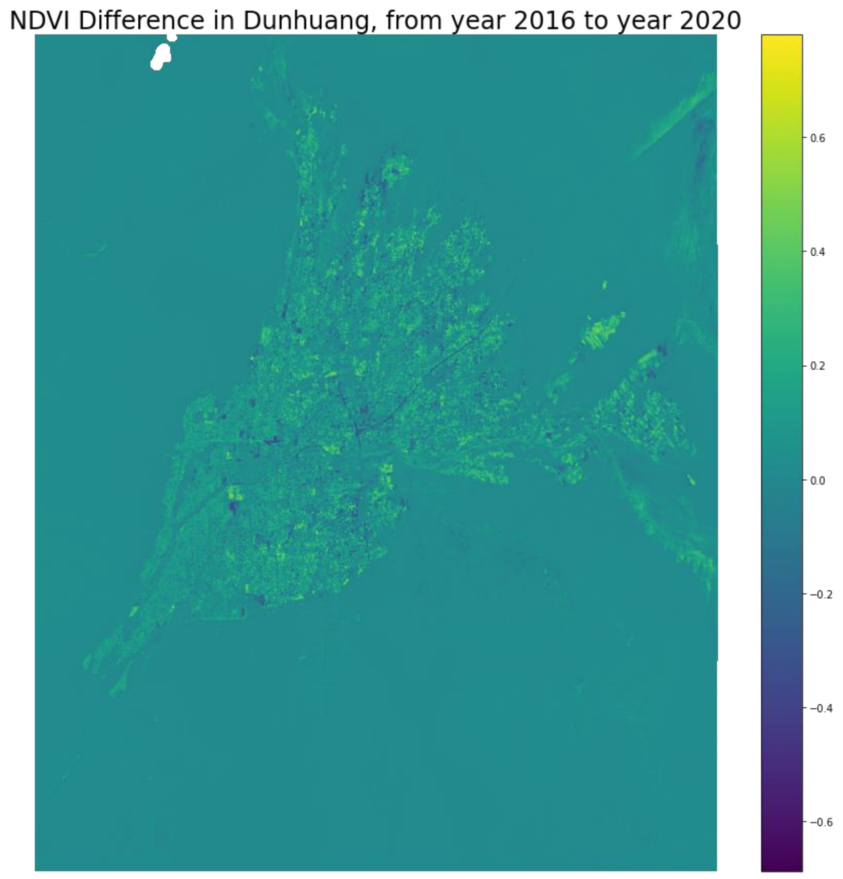


Figure 11. NDVI difference Image of Dunhuang between 2016 and 2020

K-mean clustering

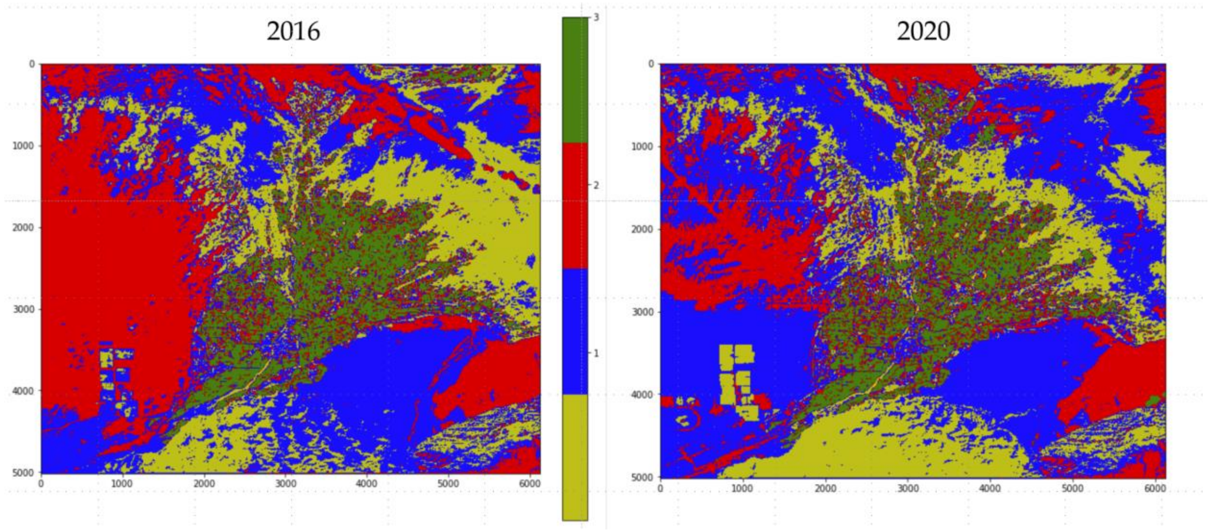


Figure 12. 4-Class Kmeans clustering results for 2016 and 2020 satellite images. Guesses for clusters: Class 0 (dark yellow): Sand/Barren land/Rock; Class 1 (blue): Built-up; Class 2 (red): Sand; Class 3 (green): Greenery Cover.

The change detection results are shown here (calculated number of pair-wise transitions between classes). Out of 30705736 pixels (5014 X 6024) in total, we have seen:

- 3352533 pixels (10.918%) changed from Greenery Cover to Built-up;
- 1961945 pixels (6.39%) changed from Built-up to Greenery Cover.

- 109568 pixels (0.357%) changed from Greenery Cover to Other (without Built-up);
- 4240505 pixels (13.81%) changed from Other (without Built-up) to Greenery Cover.

Discussion

Image Classification is started by using SVM Classifier. Table 2 and Table 3 show the results of accuracy on training and testing datasets. The accuracy reaches around 0.6, and different classes have different accuracies. Therefore, we decide to shift to a deep learning method, and decrease the number of classes in the dataset, as shown in Table 1.

Table 2 summarizes the training accuracy of the deep learning model on different groups of categories. Overall, the model performs well on predicting the training dataset, EuroSAT, and the accuracies are all around 0.9. However, the model does not perform well on predicting Dunhuang's land categories. Of the predicted green cover, the predictions are mixed with sand and mountains. Meanwhile, of the predicted built-up environments, the results cover the whole region of cities and vegetation.

One possible reason for the poor performance of classification is that the EuroSAT dataset does not include a class of sands, mountains or barren lands. Because Dunhuang is a city surrounded by sand dunes, such special land characteristics are not able to be explained by the EuroSAT data. Moreover, the special surrounding environment of Dunhuang fosters vegetations that need to live under water-deficient and long sunshine duration environments. Hence, those characteristics of green cover are not interpreted by the EuroSAT dataset.

Due to the fact that our deep learning model does not perform well on predicting land categories in Dunhuang, we use alternative approaches to do the change detection, which are NDVI and unsupervised clustering.

From the results of NDVI calculations, overall, year 2020 has a higher NDVI value than year 2016. Figure 11 shows the NDVI difference between year 2020 and year 2016. Some road-like patterns of NDVI losses could be observed in the middle of the study area, but sparsely distributed regions with increased NDVI are also being observed.

From the perspective of unsupervised clustering, although the K-means clustering method is only a rough measurement of land cover types, the results shown above seem to justify the conservation efforts for turning unoccupied land into greens, but indicate warnings for urban sprawling / farm expansion patterns (i.e., turning green coverage to built-up land) as well.

Both methods work as references to see the changes over years, but they could not give us very precise results to quantify the changes. After all, those unsupervised results needed to be verified by local knowledge or ground-truth images, which we could not cope with at this time. But at least the clustering and NDVI changes could offer some good starting points for further research, such as measuring land cover changes in different scales with both data-driven and field research methods at those locations with questionable changes. A recent case of the investigation of tree canopy loss of Yangguan forestry station of Dunhuang relied on satellite image and manual examination was an example of how visual and data-driven analysis can be combined into the maintenance and land monitoring efforts.

Reference

Chen, W., Wang, Y., Li, X. *et al.* (2016). Land use/land cover change and driving effects of water environment system in Dunhuang Basin, northwestern China. *Environ Earth Sci* 75, 1027. https://doi-org.proxy.library.upenn.edu/10.1007/s12665-016-5809-9

Dunhuang City Govt. Basic City Information of Dunhuang. http://www.dunhuang.gov.cn/lidunhuang/dunhuanggaikuang/20170104/232905694d7857.htm

Thommen, G.; Bouffanais, R. (2019). "Self-organizing maps for storage and transfer of knowledge in reinforcement learning". Adaptive Behavior. 27 (2): 111–126. arXiv:1811.08318. doi:10.1177/1059712318818568. ISSN 1059-7123. S2CID 53774629.

Helber, P., Bischke, B., Dengel, A. *et al.* (2019). Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification., Andreas Dengel, Damian Borth. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing.

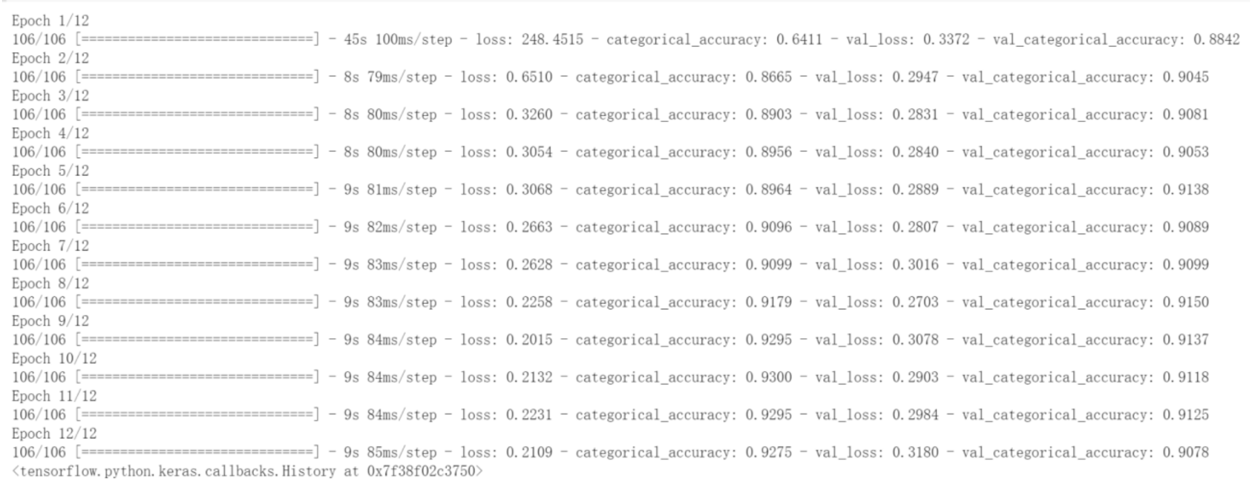
Helber, P., Bischke, B., Dengel, A. (2018). Introducing EuroSAT: A Novel Dataset and Deep Learning Benchmark for Land Use and Land Cover Classification. 2018 IEEE International Geoscience and Remote Sensing Symposium.

Normalized difference vegetation index, "Measuring Vegetation". NASA Earth Observatory. 2000-08-30.

Appendix

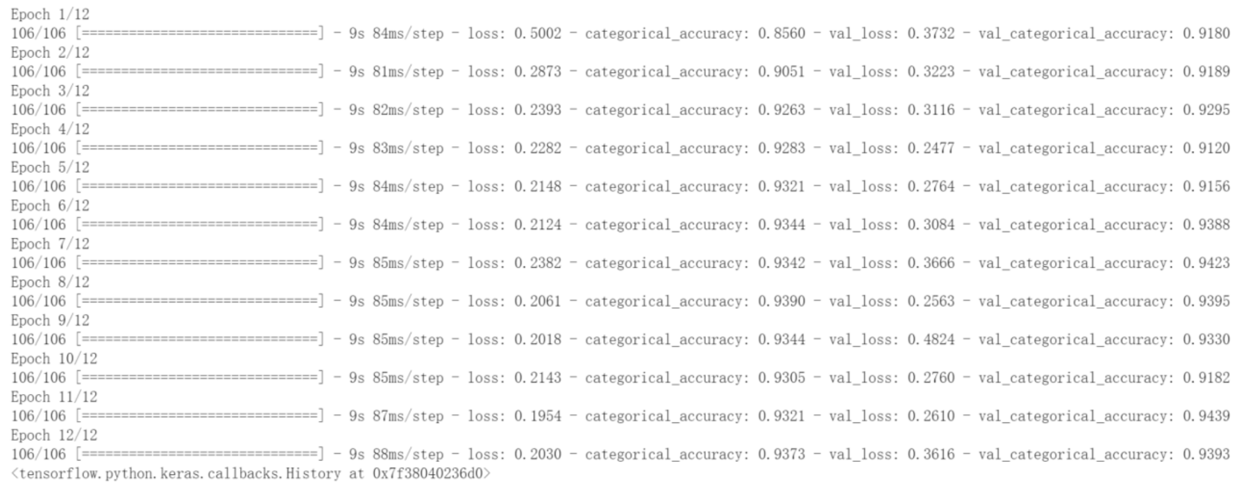
Training Accuracy:

(1)Training Accuracy for Group 1:



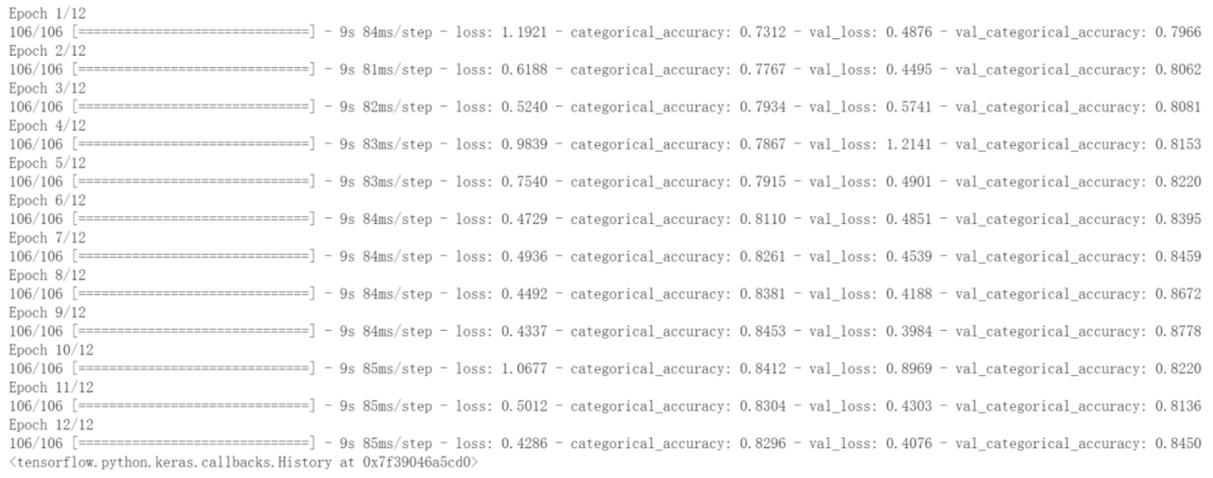
```
Epoch 1/12
106/106 [==============================] - 45s 100ms/step - loss: 248.4515 - categorical_accuracy: 0.6411 - val_loss: 0.3372 - val_categorical_accuracy: 0.8842
Epoch 2/12
106/106 [==============================] - 8s 79ms/step - loss: 0.6510 - categorical_accuracy: 0.8665 - val_loss: 0.2947 - val_categorical_accuracy: 0.9045
Epoch 3/12
106/106 [==============================] - 8s 80ms/step - loss: 0.3260 - categorical_accuracy: 0.8903 - val_loss: 0.2831 - val_categorical_accuracy: 0.9081
Epoch 4/12
106/106 [==============================] - 8s 80ms/step - loss: 0.3054 - categorical_accuracy: 0.8956 - val_loss: 0.2840 - val_categorical_accuracy: 0.9053
Epoch 5/12
106/106 [==============================] - 9s 81ms/step - loss: 0.3068 - categorical_accuracy: 0.8964 - val_loss: 0.2889 - val_categorical_accuracy: 0.9138
Epoch 6/12
106/106 [==============================] - 9s 82ms/step - loss: 0.2663 - categorical_accuracy: 0.9096 - val_loss: 0.2807 - val_categorical_accuracy: 0.9089
Epoch 7/12
106/106 [==============================] - 9s 83ms/step - loss: 0.2628 - categorical_accuracy: 0.9099 - val_loss: 0.3016 - val_categorical_accuracy: 0.9099
Epoch 8/12
106/106 [==============================] - 9s 83ms/step - loss: 0.2258 - categorical_accuracy: 0.9179 - val_loss: 0.2703 - val_categorical_accuracy: 0.9150
Epoch 9/12
106/106 [==============================] - 9s 84ms/step - loss: 0.2015 - categorical_accuracy: 0.9295 - val_loss: 0.3078 - val_categorical_accuracy: 0.9137
Epoch 10/12
106/106 [==============================] - 9s 84ms/step - loss: 0.2132 - categorical_accuracy: 0.9300 - val_loss: 0.2903 - val_categorical_accuracy: 0.9118
Epoch 11/12
106/106 [==============================] - 9s 84ms/step - loss: 0.2231 - categorical_accuracy: 0.9295 - val_loss: 0.2984 - val_categorical_accuracy: 0.9125
Epoch 12/12
106/106 [==============================] - 9s 85ms/step - loss: 0.2109 - categorical_accuracy: 0.9275 - val_loss: 0.3180 - val_categorical_accuracy: 0.9078
<tensorflow.python.keras.callbacks.History at 0x7f38f02c3750>
```

(2)Training Accuracy for Group 2:



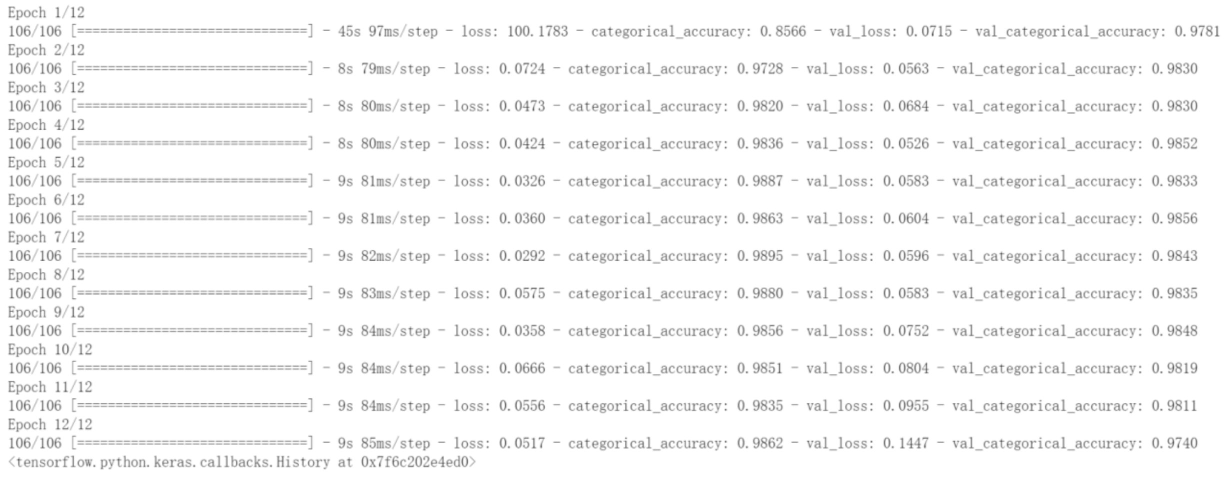
```
Epoch 1/12
106/106 [==============================] - 9s 84ms/step - loss: 0.5002 - categorical_accuracy: 0.8560 - val_loss: 0.3732 - val_categorical_accuracy: 0.9180
Epoch 2/12
106/106 [==============================] - 9s 81ms/step - loss: 0.2873 - categorical_accuracy: 0.9051 - val_loss: 0.3223 - val_categorical_accuracy: 0.9189
Epoch 3/12
106/106 [==============================] - 9s 82ms/step - loss: 0.2393 - categorical_accuracy: 0.9263 - val_loss: 0.3116 - val_categorical_accuracy: 0.9295
Epoch 4/12
106/106 [==============================] - 9s 83ms/step - loss: 0.2282 - categorical_accuracy: 0.9283 - val_loss: 0.2477 - val_categorical_accuracy: 0.9120
Epoch 5/12
106/106 [==============================] - 9s 84ms/step - loss: 0.2148 - categorical_accuracy: 0.9321 - val_loss: 0.2764 - val_categorical_accuracy: 0.9156
Epoch 6/12
106/106 [==============================] - 9s 84ms/step - loss: 0.2124 - categorical_accuracy: 0.9344 - val_loss: 0.3084 - val_categorical_accuracy: 0.9388
Epoch 7/12
106/106 [==============================] - 9s 85ms/step - loss: 0.2382 - categorical_accuracy: 0.9342 - val_loss: 0.3666 - val_categorical_accuracy: 0.9423
Epoch 8/12
106/106 [==============================] - 9s 85ms/step - loss: 0.2061 - categorical_accuracy: 0.9390 - val_loss: 0.2563 - val_categorical_accuracy: 0.9395
Epoch 9/12
106/106 [==============================] - 9s 85ms/step - loss: 0.2018 - categorical_accuracy: 0.9344 - val_loss: 0.4824 - val_categorical_accuracy: 0.9330
Epoch 10/12
106/106 [==============================] - 9s 85ms/step - loss: 0.2143 - categorical_accuracy: 0.9305 - val_loss: 0.2760 - val_categorical_accuracy: 0.9182
Epoch 11/12
106/106 [==============================] - 9s 87ms/step - loss: 0.1954 - categorical_accuracy: 0.9321 - val_loss: 0.2610 - val_categorical_accuracy: 0.9439
Epoch 12/12
106/106 [==============================] - 9s 88ms/step - loss: 0.2030 - categorical_accuracy: 0.9373 - val_loss: 0.3616 - val_categorical_accuracy: 0.9393
<tensorflow.python.keras.callbacks.History at 0x7f38040236d0>
```

(3)Training Accuracy for Group 3:



```
Epoch 1/12
106/106 [==============================] - 9s 84ms/step - loss: 1.1921 - categorical_accuracy: 0.7312 - val_loss: 0.4876 - val_categorical_accuracy: 0.7966
Epoch 2/12
106/106 [==============================] - 9s 81ms/step - loss: 0.6188 - categorical_accuracy: 0.7767 - val_loss: 0.4495 - val_categorical_accuracy: 0.8062
Epoch 3/12
106/106 [==============================] - 9s 82ms/step - loss: 0.5240 - categorical_accuracy: 0.7934 - val_loss: 0.5741 - val_categorical_accuracy: 0.8081
Epoch 4/12
106/106 [==============================] - 9s 83ms/step - loss: 0.9839 - categorical_accuracy: 0.7867 - val_loss: 1.2141 - val_categorical_accuracy: 0.8153
Epoch 5/12
106/106 [==============================] - 9s 83ms/step - loss: 0.7540 - categorical_accuracy: 0.7915 - val_loss: 0.4901 - val_categorical_accuracy: 0.8220
Epoch 6/12
106/106 [==============================] - 9s 84ms/step - loss: 0.4729 - categorical_accuracy: 0.8110 - val_loss: 0.4851 - val_categorical_accuracy: 0.8395
Epoch 7/12
106/106 [==============================] - 9s 84ms/step - loss: 0.4936 - categorical_accuracy: 0.8261 - val_loss: 0.4539 - val_categorical_accuracy: 0.8459
Epoch 8/12
106/106 [==============================] - 9s 84ms/step - loss: 0.4492 - categorical_accuracy: 0.8381 - val_loss: 0.4188 - val_categorical_accuracy: 0.8672
Epoch 9/12
106/106 [==============================] - 9s 84ms/step - loss: 0.4337 - categorical_accuracy: 0.8453 - val_loss: 0.3984 - val_categorical_accuracy: 0.8778
Epoch 10/12
106/106 [==============================] - 9s 85ms/step - loss: 1.0677 - categorical_accuracy: 0.8412 - val_loss: 0.8969 - val_categorical_accuracy: 0.8220
Epoch 11/12
106/106 [==============================] - 9s 85ms/step - loss: 0.5012 - categorical_accuracy: 0.8304 - val_loss: 0.4303 - val_categorical_accuracy: 0.8136
Epoch 12/12
106/106 [==============================] - 9s 85ms/step - loss: 0.4286 - categorical_accuracy: 0.8296 - val_loss: 0.4076 - val_categorical_accuracy: 0.8450
<tensorflow.python.keras.callbacks.History at 0x7f39046a5cd0>
```

(4) Training Accuracy for Group 4:



```
Epoch 1/12
106/106 [==============================] - 45s 97ms/step - loss: 100.1783 - categorical_accuracy: 0.8566 - val_loss: 0.0715 - val_categorical_accuracy: 0.9781
Epoch 2/12
106/106 [==============================] - 8s 79ms/step - loss: 0.0724 - categorical_accuracy: 0.9728 - val_loss: 0.0563 - val_categorical_accuracy: 0.9830
Epoch 3/12
106/106 [==============================] - 8s 80ms/step - loss: 0.0473 - categorical_accuracy: 0.9820 - val_loss: 0.0684 - val_categorical_accuracy: 0.9830
Epoch 4/12
106/106 [==============================] - 8s 80ms/step - loss: 0.0424 - categorical_accuracy: 0.9836 - val_loss: 0.0526 - val_categorical_accuracy: 0.9852
Epoch 5/12
106/106 [==============================] - 9s 81ms/step - loss: 0.0326 - categorical_accuracy: 0.9887 - val_loss: 0.0583 - val_categorical_accuracy: 0.9833
Epoch 6/12
106/106 [==============================] - 9s 81ms/step - loss: 0.0360 - categorical_accuracy: 0.9863 - val_loss: 0.0604 - val_categorical_accuracy: 0.9856
Epoch 7/12
106/106 [==============================] - 9s 82ms/step - loss: 0.0292 - categorical_accuracy: 0.9895 - val_loss: 0.0596 - val_categorical_accuracy: 0.9843
Epoch 8/12
106/106 [==============================] - 9s 83ms/step - loss: 0.0575 - categorical_accuracy: 0.9880 - val_loss: 0.0583 - val_categorical_accuracy: 0.9835
Epoch 9/12
106/106 [==============================] - 9s 84ms/step - loss: 0.0358 - categorical_accuracy: 0.9856 - val_loss: 0.0752 - val_categorical_accuracy: 0.9848
Epoch 10/12
106/106 [==============================] - 9s 84ms/step - loss: 0.0666 - categorical_accuracy: 0.9851 - val_loss: 0.0804 - val_categorical_accuracy: 0.9819
Epoch 11/12
106/106 [==============================] - 9s 84ms/step - loss: 0.0556 - categorical_accuracy: 0.9835 - val_loss: 0.0955 - val_categorical_accuracy: 0.9811
Epoch 12/12
106/106 [==============================] - 9s 85ms/step - loss: 0.0517 - categorical_accuracy: 0.9862 - val_loss: 0.1447 - val_categorical_accuracy: 0.9740
<tensorflow.python.keras.callbacks.History at 0x7f6c202e4ed0>
```

(5) Training Accuracy for Group 5:

```
Epoch 1/12
84/84 [==============================] - 45s 119ms/step - loss: 130.7068 - categorical_accuracy: 0.7236 - val_loss: 0.3664 - val_categorical_accuracy: 0.9339
Epoch 2/12
84/84 [==============================] - 7s 80ms/step - loss: 0.2657 - categorical_accuracy: 0.9310 - val_loss: 0.1820 - val_categorical_accuracy: 0.9463
Epoch 3/12
84/84 [==============================] - 7s 80ms/step - loss: 0.2063 - categorical_accuracy: 0.9414 - val_loss: 0.1895 - val_categorical_accuracy: 0.9444
Epoch 4/12
84/84 [==============================] - 7s 81ms/step - loss: 0.1463 - categorical_accuracy: 0.9507 - val_loss: 0.2431 - val_categorical_accuracy: 0.9439
Epoch 5/12
84/84 [==============================] - 7s 81ms/step - loss: 0.1436 - categorical_accuracy: 0.9576 - val_loss: 0.1927 - val_categorical_accuracy: 0.9487
Epoch 6/12
84/84 [==============================] - 7s 82ms/step - loss: 0.1295 - categorical_accuracy: 0.9587 - val_loss: 0.1893 - val_categorical_accuracy: 0.9512
Epoch 7/12
84/84 [==============================] - 7s 83ms/step - loss: 0.0893 - categorical_accuracy: 0.9688 - val_loss: 0.1911 - val_categorical_accuracy: 0.9576
Epoch 8/12
84/84 [==============================] - 7s 83ms/step - loss: 0.0770 - categorical_accuracy: 0.9722 - val_loss: 0.2160 - val_categorical_accuracy: 0.9552
Epoch 9/12
84/84 [==============================] - 7s 84ms/step - loss: 0.0733 - categorical_accuracy: 0.9759 - val_loss: 0.2252 - val_categorical_accuracy: 0.9548
Epoch 10/12
84/84 [==============================] - 7s 84ms/step - loss: 0.0942 - categorical_accuracy: 0.9686 - val_loss: 0.2068 - val_categorical_accuracy: 0.9540
Epoch 11/12
84/84 [==============================] - 7s 84ms/step - loss: 0.0816 - categorical_accuracy: 0.9750 - val_loss: 0.2146 - val_categorical_accuracy: 0.9577
Epoch 12/12
84/84 [==============================] - 7s 85ms/step - loss: 0.0785 - categorical_accuracy: 0.9755 - val_loss: 0.2436 - val_categorical_accuracy: 0.9508
<tensorflow.python.keras.callbacks.History at 0x7efce481c550>
```