# Amazon Rainforest Satellite Image Labelling Challenge

Yixin Cai
Stanford University
yixincai@stanford.edu

## Abstract

*This project explored different convolutional neural network (CNN) architectures for the multilabel classification challenge of Amazon rainforest satellite images. In this project, I examined how different layers and hyperparameters change the performance of neural network, including training time, training f2 score and test f2 score. The best model I have found is a 10-layer convolutional neural network, which has 0.89 training f2 score and 0.88 test f2 score.*

## 1. Introduction

Every minute, the world loses an area of forest the size of 48 football field, most of which are caused by deforestation in Amazon Basin. Thus, to help governments and local stakeholders understand the location of deforestation and human encroachment on forests, Planet, the designer, and builder of world's largest constellation of Earth-imaging satellites, has launched a Kaggle competition – *Planet: Understanding Amazon from Space* – to analyze small-scale deforestation and human activity influences from Amazon satellite images. The goal of this challenge is to correctly label images with atmospheric conditions, common land cover/land use phenomena, and rare land cover/land use phenomena.

For this project, the input is 256*256 color satellite images, and the algorithm uses a convolutional network to output a set of predicted labels. Training data and test data in the challenge is provided by Kaggle. In this project, I will explore and analyze how different convolutional neural network models perform in terms of training time, f2 score and generalization ability.

## 2. Related work

Research on satellite image processing has over 20 years' history. Simple neural network structure with only fully connected layers has been used as a satellite image classifier early in 1995[1]. In recent years, especially since 2015, training a deep neural network becomes easier with larger computational power, bigger training datasets and better image quality. Thus, convolutional neural network has become a popular tool for satellite image analysis in many areas of application, including image classification [2], land use classification [3], pattern detection in urban environment [4], solar power plant detection [5], orthoimagery segmentation [6], nighttime sky/cloud segmentation [7] and face-like structure detection [8].

Also, with the emergence of large dataset with high-resolution remote sensing (HRRS) imagery, different neural network models have been created and tuned for higher accuracy. Examples models such as traditional neural network [9], deep convolutional neural network [10], multi-channel pulse coupled neural network (m-PCNN) [5] and recurrent neural network [11], have all been used to achieve a high accuracy on high-resolution satellite images.

In addition, methods other than neural networks are still being used for specific tasks. Statistical methods like contrast statistical analysis (CSA) [12], singular value decomposition (SVD) [13], temporally contiguous robust matrix completion (TECROMAC) [14] and non-additive entropy [15] has been used for quality evaluation, image de-noising, matrix completion, image classification and segmentation.

For this project, I will mainly evaluate the performance of deep convolutional neural networks on high-resolution Amazon satellite images.

## 3. Problem Statement and Challenges

The Kaggle challenge is a multilabel classification problem. The input is colored satellite images with 256*256 resolution. The output can be one or multiple labels from 17 possible classes – agriculture, artisinal_mine, bare_ground, blooming, blow_down, clear, cloudy, conventional_mine, cultivation, habitation, haze, partly_cloudy, primary, road, selective_logging, slash_burn, and water. Figure 1 has some sample images with corresponding labels.

The evaluation metric used for this competition is the average f2 score, which basically takes a weighted geometric average between precision and recall. In the case of f2 score, recall has a much higher importance than precision. In other words, false negatives are more


Figure 1: Sample training images with labels.

detrimental to the score than false positives – it is worse to miss a label than to add an incorrect label to images. In later sections, we will explore how hyperparameters and decision boundaries can be adjusted accordingly to achieve a higher f2 score.

Another challenge is that there could be noise in the labels for the training data. Since the training images are labeled manually, the labels are not guaranteed to be 100 percent correct, and the challenger needs to be aware of possible wrong labels in training process. I have not examined incorrect labels in this project, but there are many existing statistical methods, such as principal component analysis (PCA), that can be used for image preprocessing to reduce the noise and further improve correctness.

## 4. Data

There are 40479 training images and 61191 test images. Each image is a 3-channel color image with 256*256 pixels. For training, I used 30000 (75%) images as training set and 10479 (25%) images as validation set. Test set is composed of all test images. Since Kaggle provided an easy interface to check the test performance of 66% (40386) test images, I moved all training images that used to be in test set into validation set, so that all training images are used for training and validation.

One thing to note about the data set is that most labels are highly skewed. As shown in Figure 2, most labels are only assigned to less than 20% of training data. In particular, labels like artisinal_mine, blooming, blow_down, conventional_mine, selective_logging and slash_burn, have proportion less than 1% of all data. On one side, correct classification of these labels becomes a particularly hard and challenging part of the competition. On the other hand, misclassification of these labels will have a very small effect on average score.

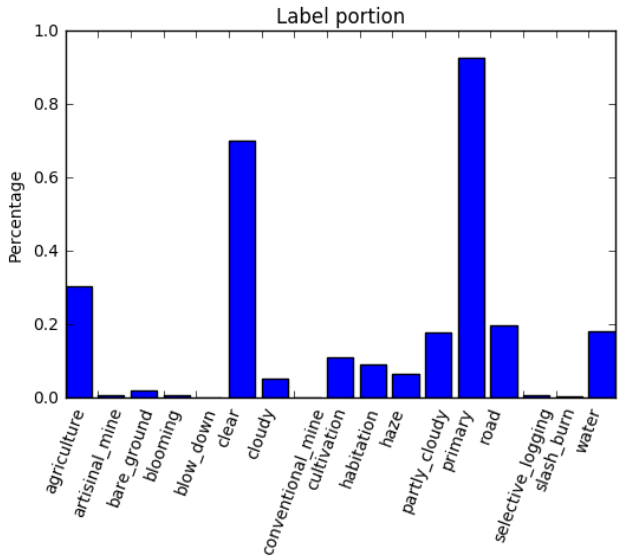


Figure 2: Class label distribution of training data.

## 5. Methods

### 5.1 Initial 4-layer CNN

The first model I tried for this project is the CNN model I used in assignment 2, which is a 4-layer model. The structure is as follows.

Conv (32 3*3 filters, stride 2) – batch norm – ReLu – dropout – max pooling

Conv (32 3*3 filters, stride 1) – batch norm – ReLu – dropout – max pooling

Affine (32768*1024) – batch norm – ReLu

Affine (1024*17)

The loss function is changed from SoftMax cross entropy to sigmoid cross entropy. The initial accuracy is 89%, which seems reasonable, but the training f2 score is very low – only 0.59. Upon further inspection, I realized that the cause is incorrect choice of loss function.

5.2 Classification method and loss function

There are two common loss functions for multilabel classification – sigmoid cross entropy loss and support vector machine (SVM) with hinge loss.

As I mentioned, I started with binary classification and sigmoid cross entropy loss, but the result is not very good. Upon looking at the prediction labels, I realized that the classifier tends to classify all images as clear and primary, which are the two most dominant class labels in training data. It seems like all images are classified in the same way – they only have labels that appear in more than 50% training images. This is not ideal.

As a result, I changed the last layer to SVM and hinge loss, but keeping the other CNN structure the same. Immediately the training accuracy becomes 98%, and average training f2 score grows to 0.92 after 20 epochs.

This behavior explains that happened with the previous loss function. For classification of highly skewed labels, SVM with hinge loss seems to be a much better choice because it only looks at decision boundary and support vector, thus the number of positive examples and negative examples used for training is roughly equal. On the other hand, sigmoid cross entropy loss might weigh all images equally. When the class proportion is skewed, the positive examples with low proportion will have almost no effect on training.

5.3 Batch normalization

Up on deciding to use SVM with hinge loss, I encounter another problem – the result of training data differs greatly during training and test phase. In training time, I can see 0.92 training f2 score, but in test time, the f2 score drops to 0.75 for both training set and validation set. Later I realized that batch normalization caused this weird behavior. Since batch normalization normalizes the input data differently during training and test time, the output of CNN is scaled and shifted differently in test time, and this would completely change the decision boundary of SVM classifier.

I solved the mismatch by getting rid of batch normalization after affine layer and changing the mode of batch normalization after convolution layer to always be training. In this way, I can achieve the same training f2 score during training and test time, and the corresponding validation f2 score is 0.81.

5.4 Initial 6-layer model

After a working 4-layer model, I added two more convolutional layers for image down sampling, hoping to achieve a better result and faster training time. The model is as follows.

Conv (3 2*2 filters, stride 2) – ReLu – dropout – batch norm

Conv (3 2*2 filters, stride 2) – ReLu – dropout – batch norm – max pooling

Conv (32 3*3 filters, stride 1) – ReLu – dropout– batch norm

Conv (32 3*3 filters, stride 1) – ReLu – dropout – batch norm – max pooling

Affine (8192*1024) – ReLu – dropout

Affine (1024*17)

The result of this network is not good. First, the model trains slowly. After 40 epochs, the training f2 score stops at 0.86, which is much lower than the previous model. The validation f2 score is 0.81 – the same as the previous 4-layer model.

In summary, this new model trains slowly, has better generalization ability, but has not improved validation score. The cause of such behavior is that there are too much regularization layers – there is no need to put a dropout layer after each convolution/affine. Thus, I come up with a new 6-layer model with less regularization.

5.5 Fast 6-layer model

To increase training speed, I removed 3 dropout layers – two layers after the first two convolutional layers and one layer after the affine layer. Also, I added more filters since the training speed in significantly increase. The new model is:

Conv (16 2*2 filters, stride 2) – ReLu – batch norm

Conv (26 2*2 filters, stride 2) – ReLu – batch norm – max pooling

Conv (64 3*3 filters, stride 1) – ReLu – dropout– batch norm

Conv (64 3*3 filters, stride 1) – ReLu – dropout – batch norm – max pooling

Affine (16384*1024) – ReLu

Affine (1024*17)

The result of this new mode is much better than the original model. In 30 epochs, I can get f2 score of 0.9 on training set and 0.846 on test set.

5.6 8-layer model

After figuring out a good 6-layer model, I put most effort in training models with more layers. Here is the 8-layer model.

Conv (16 2*2 filters, stride 2) – ReLu – batch norm

Conv (26 2*2 filters, stride 2) – ReLu – batch norm – max pooling

Conv (64 3*3 filters, stride 1) – ReLu – dropout– batch norm

Conv (64 3*3 filters, stride 1) – ReLu – dropout – batch norm – max pooling

Conv (64 3*3 filters, stride 1) – ReLu – dropout– batch norm

Conv (64 3*3 filters, stride 1) – ReLu – dropout – batch norm – max pooling

Affine (4096*1024) – ReLu
Affine (1024*17)

With 2 more layers, validation f2 score can be improved to 0.866 and the corresponding test score is 0.865. The test score would drop to 0.855 if the model is further trained with all training data for 10 epochs, which shows the model starts overfitting.

5.7 10-layer model

2 more layers are added to the model and more filters are used. The model is as following.

Conv (32 3*3 filters, stride 2) – ReLu – batch norm
Conv (32 3*3 filters, stride 2) – ReLu – batch norm – max pooling - dropout
Conv (64 3*3 filters, stride 1) – ReLu – batch norm
Conv (64 3*3 filters, stride 1) – ReLu – batch norm – max pooling - dropout
Conv (64 3*3 filters, stride 1) – ReLu – batch norm
Conv (128 3*3 filters, stride 1) – ReLu – batch norm – max pooling - dropout
Conv (128 3*3 filters, stride 1) – ReLu – batch norm
Conv (128 3*3 filters, stride 1) – ReLu – batch norm – max pooling - dropout
Affine (4096*1024) – ReLu
Affine (1024*17)

This is the last CNN I trained. After 30 epochs, the training f2 score is 0.89, the validation f2 score is 0.875 and the test f2 score is 0.874.

5.8 Existing models

Besides custom models I build I have used a VGG model by changing the size of output to 17, and tried to replace the SoftMax cross entropy loss to either sigmoid cross entropy loss or hinge loss. The result is also not good – all images are classified as clear and primary no matter how I change learning rate, loss function or variable initialization. Given that training a VGG net takes a long time, I decided to spend more time on building my own deep CNN rather than wasting time with the existing model.

5.9 Other feature extraction methods

I tried training a traditional model which uses HOG features and SVM classifier on a tiny subset of training images with 3000 training images and 1000 validation images. The f2 score for validation is 0.77.

The reason for the bad performance of HOG feature is that it only contains orientation information. Since many labels are related to weather information, and HOG cannot capture the related color information, it is not a good model for feature extraction in this project.

6. Experiment Results

6.1 Parameter tuning

| Parameters | Initial value | After tuning |
|---|---|---|
| Dropout keep probability | 0.6 | 0.8 |
| C for SVM | 1 | 1 |
| L2 regularization | 0.5 | 0.05 |
| Training f2 score | 0.89 | 0.895 |
| Test f2 score | 0.874 | 0.878 |

Table 1: Parameters for the final 10-layer model.

The largest problem throughout this project is that CNN models are hard to train. As the number of layers increase, the generalization ability of the model becomes better, but training time also increases. As a result, parameter tuning is done to make sure that training error can decrease faster and a higher training f2 score can be achieved.

Table 1 shows the parameters before and after tuning. We can see an increase in keep probability and decrease in L2 regularization. By having less regularization, the model can achieve higher training and test f2 score.

The optimizer I used is Adam with learning rate as 0.001. I did not spend much time testing with different optimizers because Adam learns very fast – the training f2 score can usually reach 0.74 after one epoch. In practice, dropout layer and l2 regularization in CNN have much more influence on learning speed than the choice of optimizer.

6.2 F2 scores for training, validation, and test sets

| Model | Training F2 | Validation F2 | Test F2 |
|---|---|---|---|
| 4-layer | 0.92 | 0.81 | n/a |
| 6-layer slow | 0.86 | 0.81 | n/a |
| 6-layer fast | 0.9 | 0.846 | n/a |
| 8-layer | 0.895 | 0.866 | 0.865 |
| 10-layer | 0.89 | 0.875 | 0.874 |

Table 2: Result analysis for different results.

Table 2 shows how training f2 score and test f2 score changes with different model depths. The trend is very clear – as the number of layers gets larger, the training score tends to decrease and validation score tends to increase. This is a result of better generalizability and harder trainability. If I have time to train on a CNN with more layers and more epochs, I believe I can get a model that has test score higher than 0.9.

6.3 Precision and recall analysis

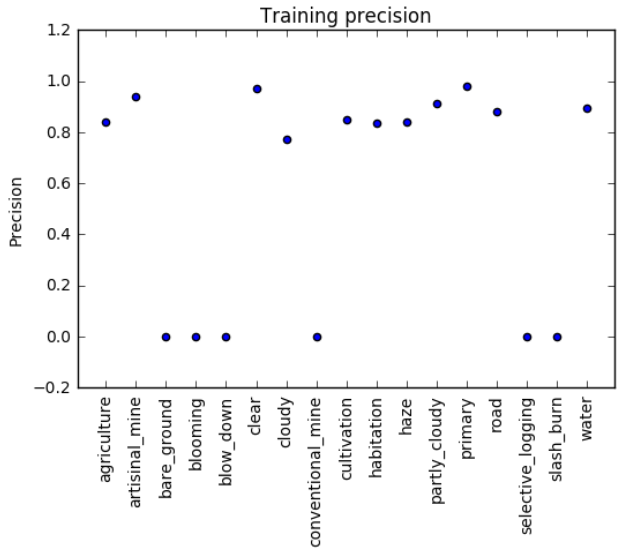


Figure 3: Training precision for all labels.

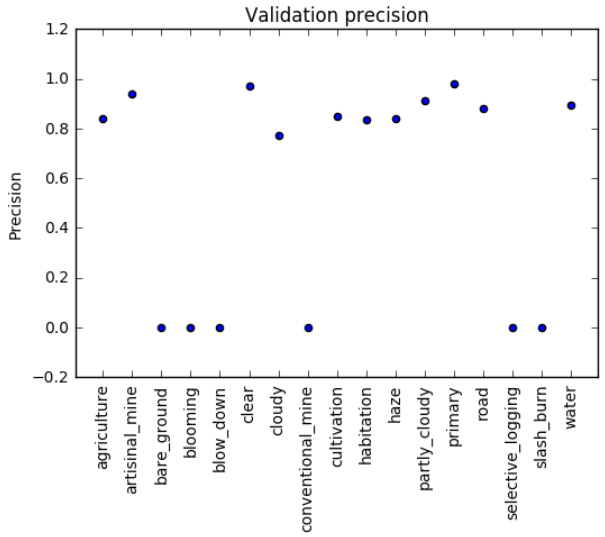


Figure 4: Validation precision for all labels.

Figure 3 and 4 shows the training and validation precision for all labels using the final 10-layer CNN model. The structure of two graphs are very close, indicating that the model has good generalization ability. One interesting thing to note is that 6 labels has 0 precision. On the other hand, the number of labels with 0 precision with the initial 4-layer model is 2. We see an increase in the number of imprecise labels as the model becomes more complicated. This phenomenon has two explanations. One is that those classes appear too rare in data so that ignoring them does not have much influence on the score. The second explanation is that the model is under-trained and requires more training epochs.

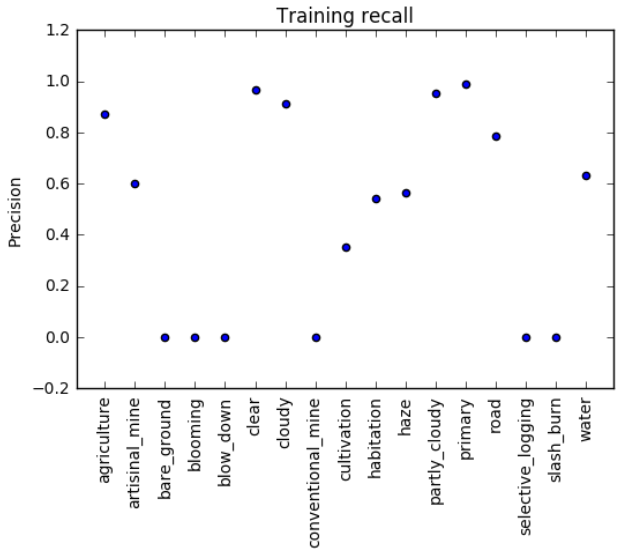


Figure 5: Training recall for all labels.

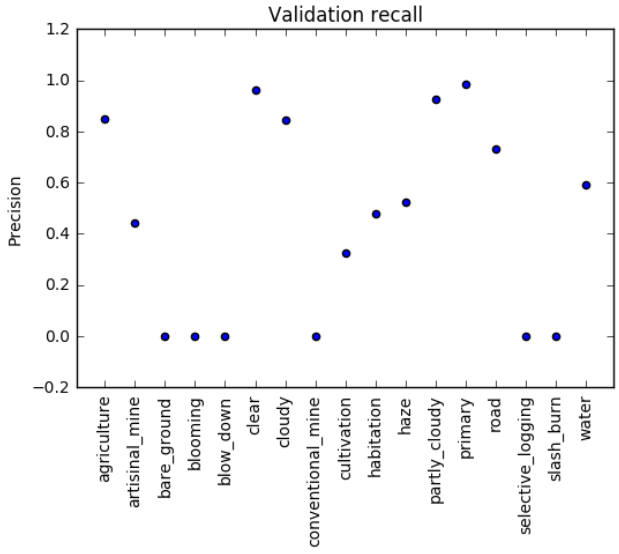


Figure 6: Validation recall for all labels.

Figure 5 and 6 shows the training and validation recall of the 10-layer model. Again, the two graphs share similar structures. However, for some labels, like cultivation, habitation and haze, the recall is very low even though precision is high. This suggests the possibility to lower the decision boundary, and get a higher recall in exchange of a lower precision. Since recall is about 4 times more important than precision in the evaluation of f2 score, a lower threshold will give us a better overall score.

6.4 Error case analysis

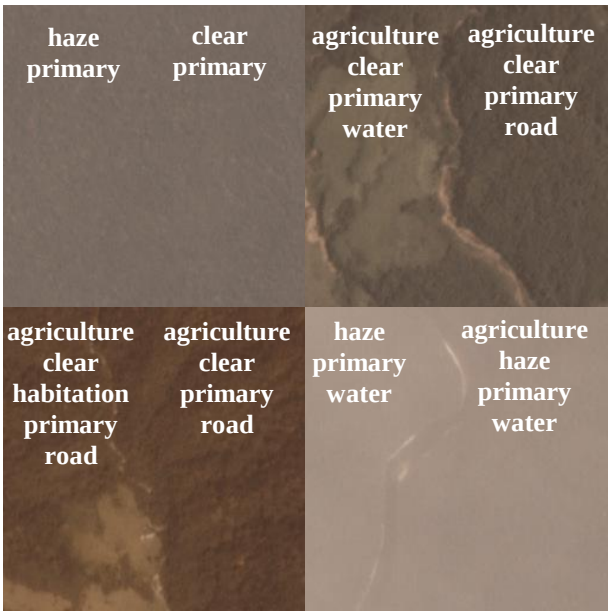


Figure 7: Example misclassified images. For each image, top left corner contains the correct labels, and top right corner contains the predicted labels.

Figure 7 contains 4 examples of misclassified images, and they represent some of the most common mistakes my classifier makes.

The image on the top left is the misclassification of haze and clear. The mistake is caused by a low threshold for clear and a high threshold for haze. One solution might be having a lower bar for haze and a higher bar for clear, given that clear has a recall close to 1 while haze has a recall of only 0.5.

The image on the top right has a misclassification of road rather than water. In fact, this is a common mistake even for human because a road visually looks similar to a river. Adding more higher resolution filters on the first layer of convolutional network might be able to solve this problem.

The image on the bottom left corner is missing a habitation label. This is a typical problem of the model as it has high precision but low recall. One proposal is to have different penalty in hinge loss function for positive and negative labels, but the result of it would need further experiments.

The image on the bottom right has an additional agriculture label. This is in fact the case that CNN is inherently bad at – the best neural network still makes mistakes. Even though precision and recall for agriculture are both high, CNN would still have classification errors.

In theory, most the misclassification case above might be solved by a recurrent neural network structure given that many of the labels are correlated or mutually exclusive. Another possible solution is to do preprocessing on the label of images, such as PCA, and present them as a lower dimension vector for prediction.

6.5 Result and competition standing

The best test f2 score I achieved is 0.877, and the current leading score is 0.933. I believe with deep neural network and more training, I would be able to get above 0.9.

7. Conclusion and future work

At the beginning of this project, I spend a large amount of time to find correct architecture. As a result, with the limited time to test and train models, I was not able to train CNN deeper than 10 layers. But after this project, here are a few more things I could continue to try.

7.1. Add model complexity

For CNNs, it is always good to add more layers to the existing model. I can gradually get better results with deeper neural network and more training time.

7.2. Different CNN architecture

During poster session, I realized that there are other groups working on other CNN architectures and achieve a score over 0.93. Such structures include CNN-RNN and hierarchical CNN. Also, I have not tried using ResNet, which is possible to produce a much better result.

7.3 Modification of loss function

Since precision is less important than recall, the hinge loss function of SVM may not be the best loss function to use. A modified hinge loss could be used so that misclassification of positive labels and negative labels are penalized differently.

7.4 Different CNN for different labels

Since a single CNN always return 0 for some labels, it is possible to train several simpler models for some labels on top of the complex model. In this way, the training and test accuracy may be able to increase more without the interference of other labels.

7.5 Recruit other team members

Training a deep neural network is a hard task for one individual. If I had more teammates testing different models at the same time, I would be able to spend less time on the wrong architecture and achieve a better score.

## 8. Appendix

The base code for this project is based on assignment2 – Tensorflow.ipynb (with bugs fixed) and Piazza post – *Tutorial for data loading and fine-tuning*.

## References


[1] M. F. Augusteijn, L. E. Clemens and K. A. Shaw, "Performance evaluation of texture measures for ground cover identification in satellite images by means of a neural network classifier," in IEEE Transactions on Geoscience and Remote Sensing, vol. 33, no. 3, pp. 616-626, May 1995. doi: 10.1109/36.387577.

[2] Qingshan Liu, Renlong Hang, Huihui Song: "Learning Multi-Scale Deep Features for High-Resolution Satellite Image Classification", 2016.

[3] Marco Castelluccio, Giovanni Poggi, Carlo Sansone, and Luisa Verdoliva. 2015. Land Use Classification in Remote Sensing Images by Convolutional Neural Networks. CoRR abs/1508.00092 (2015). http://arxiv.org/abs/1508.00092.

[4] Adrian Albert, Jasleen Kaur: "Using convolutional networks and satellite imagery to identify patterns in urban environments at a large scale", 2017; arXiv:1704.02965.

[5] Nevrez Imamoglu, Motoki Kimura, Hiroki Miyamoto, Aito Fujita: "Solar Power Plant Detection on Multi-Spectral Satellite Imagery using Convolutional Neural Networks with Feedback Model and m-PCNN Fusion", 2017. http://arxiv.org/abs/1704.06410 arXiv:1704.06410.

[6] Langkvist, M.; Kiselev, A.; Alirezaie, M.; Loutfi, A. Classification and Segmentation of Satellite Orthoimagery Using Convolutional Neural Networks. Remote Sens. 2016, 8, 329.

[7] Soumyabrata Dev, Florian M. Savoy, Yee Hui Lee: "Nighttime sky/cloud image segmentation", 2017. http://arxiv.org/abs/1705.10583 arXiv:1705.10583.

[8] Kazutaka Kurihara, Masakazu Takasu, Kazuhiro Sasao, Hal Seki, Takayuki Narabu, Mitsuo Yamamoto, Satoshi Iida: "A Face-like Structure Detection on Planet and Satellite Surfaces using Image Processing", 2013, ACE 2013, LNCS 8253, Springer, pp. 564-567, 2013.

[9] Mnih V., Hinton G.E. (2010) Learning to Detect Roads in High-Resolution Aerial Images. In: Daniilidis K., Maragos P., Paragios N. (eds) Computer Vision – ECCV 2010. ECCV 2010. Lecture Notes in Computer Science, vol 6316. Springer, Berlin, Heidelberg.

[10] Hu, F.; Xia, G.-S.; Hu, J.; Zhang, L. Transferring Deep Convolutional Neural Networks for the Scene Classification of High-Resolution Remote Sensing Imagery. Remote Sens. 2015, 7, 14680-14707.

[11] Emmanuel Maggiori, Guillaume Charpiat, Yuliya Tarabalka: "Recurrent Neural Networks to Correct Satellite Image Classification Maps", 2016; arXiv:1608.03440. DOI: 10.1109/TGRS.2017.2697453.

[12] Firouz Abdullah Al-Wassai, N. V. Kalyankar: "Spatial And Spectral Quality Evaluation Based On Edges Regions Of Satellite Image Fusion", 2012, International Journal of Latest Technology in Engineering,Management & Applied Science (IJLTEMAS),Vol. I, Issue V, 2012, 124-138; [http://arxiv.org/abs/1207.1922 arXiv:1207.1922].

[13] Jialei Wang, Peder A. Olsen, Andrew R. Conn: "Removing Clouds and Recovering Ground Observations in Satellite Image Sequences via Temporally Contiguous Robust Matrix Completion", 2016; [http://arxiv.org/abs/1604.03915 arXiv:1604.03915].

[14] Prajakta P. Khairnar: "Image Resolution and Contrast Enhancement of Satellite Geographical Images with Removal of Noise using Wavelet Transforms", 2014, International Journal of Engineering Trends and Technology (IJETT),Volume 10, Number 12,Apr-2014 International Conference of Recent Trends in Engineering and Technology (ICRTET-2014),paper code 223; [http://arxiv.org/abs/1405.1967 arXiv:1405.1967].

[15] Lucas Assirati, Alexandre Souto Martinez: "Satellite image classification and segmentation using non-additive entropy", 2014; [http://arxiv.org/abs/1401.2416 arXiv:1401.2416]. DOI: [http://dx.doi.org/10.1088/1742-6596/490/1/012086 10.1088/1742-6596/490/1/012086].