Areal Team project proposal

David Biard david.biard@u-psud.fr
Robin Duraz robin.duraz@u-psud.fr
Samuel Berrien samuel.berrien@u-psud.fr
Hao Liu hao.liu@u-psud.fr
Trung Vu-Thanh trung.vu-thanh@u-psud.fr
Théo Cornille theocornille3@gmail.com
Team's github https://github.com/ArealTeamM2AIC/
Areal Team areal@chalearn.org
Guillaume Charpiat Mentor, guillaume.charpiat@inria.fr
Nicolas Girard Mentor, nicolas.girard@inria.fr
Yuliya Tarabalka Mentor, yuliya.tarabalka@inria.fr

The data set NWPU-RESISC45 we use part of, is a new data set (in 2016), bigger than most in its number of representatives for each class. Hence, it allows a better training for classifiers that use training to learn how to extract new features. (link to paper)

1 Background

Since aerial imagery services and high resolution appeared, aerial imagery has become of the most important components of various industries. Energy, mining, military situation, disaster management, urban planning and more industries as well as other organizations in emergency situations can make use of aerial images to enhance their productivity and quality of work. There exists many applications in which classifying aerial images can prove to be useful. For example, detecting objects like icebergs or even ships in the sea.

Recent breakthroughs in image understanding techniques using deep learning methods and improvements in hardware like GPUs have opened the way for people to experiment with different approaches and techniques.

Most recent approaches make use of deep learning and convolutional neural networks (CNN) to learn new representations of features, allowing a simpler classification task. Those CNN will try to use imagery techniques like convolutions and pooling to transform the image. The main problem using this approach is that the smaller the data set, the worse and the less informative the new representation of the features.

We chose to use only part of the original data, for simplicity purposes. We kept 13 classes, but with all available images for the classes we kept. Each class represents a kind of natural scenery, which even for a knowledgeable human, may not necessarily be easy to classify.

Our project tackles the issue of multi-class classification with uncommon classes like wetland, meadow or snowberg as can be seen in 1. We used a pre-trained CNN, and trained the last layer to conform to our data. We trained it on three dimensional images being 256 x 256 pixels x RGB, with labels being a word (string) describing the class. With a satellite image as input, our network then will be able to output its class. We will also use it to extract a new representation of images, which will be used to train other models, and should be simpler tu use for most models, compared to satellite images.

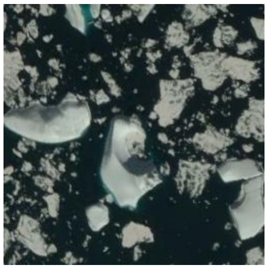
2 Material & Method

In order to create this project, we used the NWPU-RESISC45, which has 45 classes of each 700 images. Each image is a 256x256 RGB image. We selected 13 classes in order to not have too many classes.

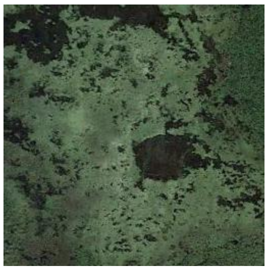
We used a known CNN pre-trained (AlexNet) for image classification, which was a good solution for the biggest imagery classification challenge, ImageNet. We kept all its original layers, which we didn't train, but replaced the last layer in order to conform to our data. We then trained the last layer in order to learn with our data how to better represent original images.

In order to extract features, we used the representation of the image at the last layer, just before classification. We then created a new data set from that, with each image having a new representation.

We will have two different data sets, one being the original images, and the other being their new representation. The size of the original data set is about 120MB, whereas the second data set's size is .



(a) Sea_ice (b) Snowberg



(c) Wetland (d) Meadow

Fig. 1: Examples of 4 different classes

Considering that we have balanced data sets, we will take a simple metric, which is accuracy to determine the classification's quality. It is computed as Number of true positive in class divided by Number in class.

To divide our data set in train, validation and test, we first read the original paper using this data set. In this paper, it was divided with only 10 or 20 % of the whole as train. Considering we spent less time trying to tune our CNN, we decided to keep more images as train. In the end, we have 150 images for each class as train, the same for validation, and the remaining, i.e. 400 images as test.
In the end, data will be of two kinds: one being the original processed images, and the other will be the representation of these images by a trained CNN.

Maybe have a ROC-AUC curve here

3 Preliminary Results

In the paper [1], only considering the data set NWPU-RESISC45, authors have first used state-of-the-art pre-trained CNN, such as AlexNet and VGGNet-16. Then, they fine-tuned them in order to have them adapt to the data.

Models	Training ratios	
	10%	20%
AlexNet	76.69	79.85
VGGNet-16	76.47	79.79
GoogLeNet	76.19	78.48

Table 1: Un-tuned models' accuracy

Models	Training ratios	
	10%	20%
AlexNet	81.22	85.16
VGGNet-16	87.15	90.36
GoogLeNet	82.57	86.02

Table 2: Fine-tuned models' accuracy

Results obtained can be seen in Tables 1 and 2. Training ratios represent how much images they took to train their models.

In order to see how much of these results we would be able to replicate, we performed our own tests on the data, training on 90% of it, and using the rest as validation. We compared accuracy and loss of the AlexNet model, for which we trained the last layer, and a simple CNN having only a linear layer. Our results are visible in Figure **??**.
Seeing the results obtained by the AlexNet model, we were satisfied enough to use it in order to output pre-processed data. We chose to take the data available to the last layer and take it as our data. Then, we had to choose how much data we should take as train, validation and test. For that end, we ran tests by dividing the whole data in a train set and a validation set, with different division values. We trained four untrained classifiers from sklearn in order to have an idea of the division we should make. The results of that are visible in Figure 2. Seeing this, we chose almost the same division as with images, i.e. 1950 examples as train, also 1950 as validation, and the rest, 5200 as test.

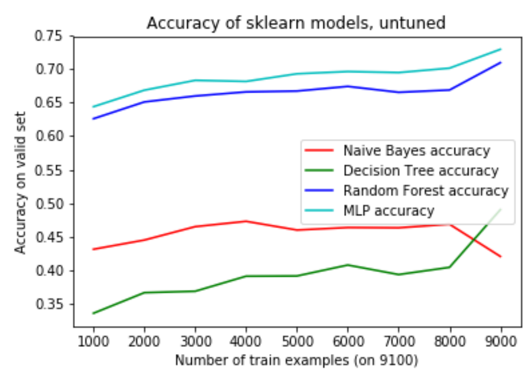
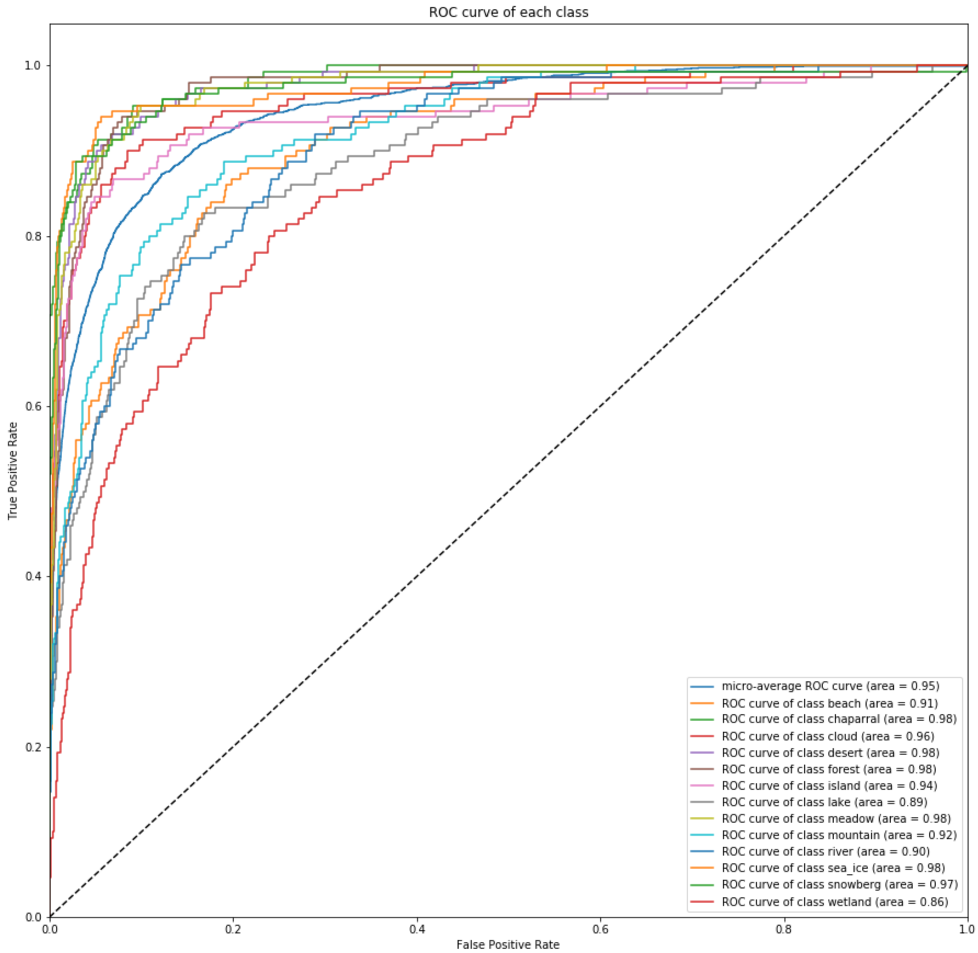
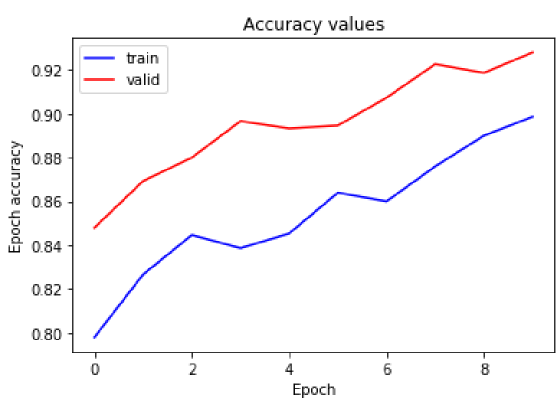


Fig. 2: Accuracy of different models according to size of train set

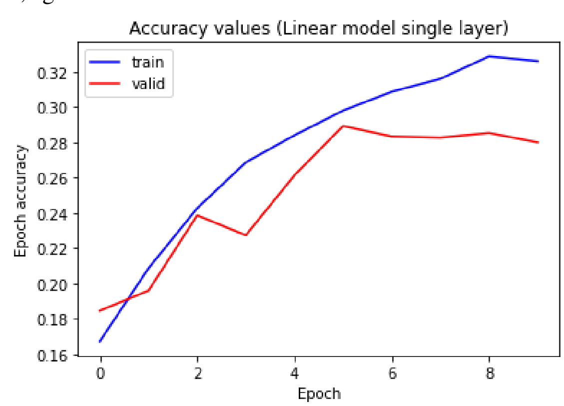
As a last visualization on pre-processed data, we computed the ROC curve area for each class, individually against all the other classes [3a]. We can see in this visualization that, globally, this classification task should not have any big trouble with any of the given classes.



(a) Roc curve for each class, against all others



(b) AlexNet CNN accuracy



(c) Simple Linear CNN accuracy

References

[1] Gong Cheng, Junwei Han, and Xiaoqiang Lu, Remote Sensing Image Scene Classification: Benchmark and State of the Art. IEEE International.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Proc. Conf. Adv. Neural Inform. Process. Syst., 2012, pp. 1097-1105.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in Proc. Int. Conf. Learn. Represent., 2015, pp. 1-13.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in Proc. IEEE Int. Conf. Comput. Vision Pattern Recognit., 2015, pp. 1-9.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE Int. Conf. Comput. Vision Pattern Recognit., 2016, pp. 770-778.