BORIS FLUCKIGER
EPFL - PERSONAL INTERACTION STUDIO 2018

Spotted Indigenous

Technical Report

Introduction

The main objective of this project is to be able to do some image classification to localize villages of indigenous people. It thus follows a usual pipeline of data gathering and machine learning model training. However, another aim was to be able to confront 2 visions. The human perception of the images and the way they are seen by computers. For the latter, the project focus on deep learning analysis and more precisely on transfer learning and a convolutional neural network (CNN), built using the Keras Framework. Without getting too much into details we will try to have a global overview of these technologies in order to give some intuition on their mechanism. The commented and detailed pipeline is also available on the project GitHub.

Data Gathering

Positively labeled images:

To be able to label images, an interface was implemented using JavaScript, through which a user can freely explore a defined zone on a Google Map. After following a few instructions, he can then add a marker on places where settlements are presents. Coordinates from this collaborative map are saved on a database from the website www.Firebase.com which offers a free back-end service.
The images from the saved JSON are then downloaded using the Google Map API and labeled as positive.

Negatively labeled images:

To be able to collect data that do not contain settlements (mostly forest images) snapshots were grid extracted from free marked zones and again downloaded using the Google Map API.

Collaborative map for the users to spot the settlements

At this point of the project, a total of 1700 images were extracted for the training set (1500 positively labeled and 200 negatively labeled) and 360 for the validation set (260 negatively labeled and 100 positively labeled). To be able to use them afterward as input for the neural network they were

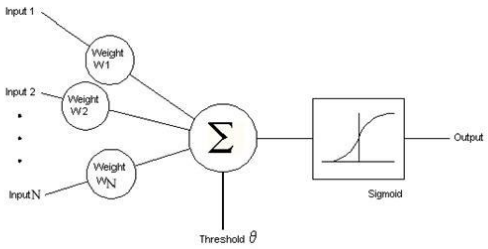
Spotted Indigenous

downloaded with a size of 224x224 and cropped to remove the unnecessary information. Lastly, normalization of the color was also applied.

Model Training

A human is very efficient in labeling images. He can indeed in most of the case label (recognize) 20 elements per second. However, this task can quickly become subjective and time-consuming. This is the reason why we decided to implement an automatic labeling process to facilitate this task, keeping, however, a human factor.

Below, is the representation of an artificial neural network (perceptron) derived from the motivation of willing to model the biological interaction between human neurons.



Source: http://www.cs.ucc.ie

As we can see, there is an input (in our case represented later by our images), transmitted through a processing unit composed from a transfer function which is a weighted combination of the input features and then and an activation function that will produce a decision for the output.

Transfer function:

$$\Sigma(x_1, x_2, ..., x_n) = \sum_{i=1}^{i=n} w_i \cdot x_i + b_i$$

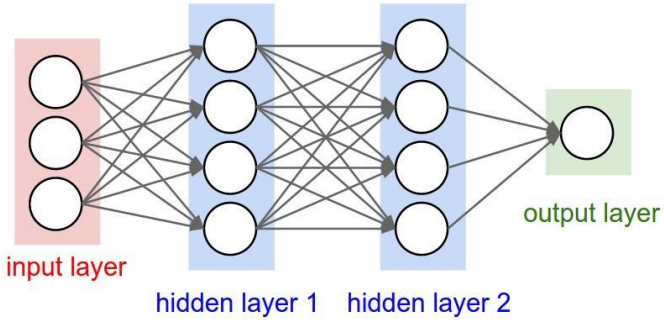
Output:

$$z = \phi\Big(\sum_{i=1}^{i=n} w_i \cdot x_i + b_i\Big)$$

The objective is to find the weights "w_i", that are optimal for a given task, in this case, classifying our images.

Spotted Indigenous

The artificial neurons can then be assembled to create the network.



Source: https://towardsdatascience.com

Two good points of neural networks are that we:

1) Can approximate any continuous function for a given interval with one hidden layer and a sigmoid-like activation function.
2) Can learn and build good feature representation of our data. That do not need to be handcrafted before training.

We can find many architectures of artificial neural networks, and amongst them the convolutional neural network:

Convolutions are operations that given an input matrix [n x n] and a kernel matrix [k x k], will yield another matrix [p x p] that is dependent on the dot-product between a sliding window of [k x k] and the initial input matrix.
In the CNN case, learning the features of an image is equivalent to learning the resulting Kernels. Moreover, searchers observed that usually, the first layers allow the model to construct feature maps that look for simple patterns (edges, diagonals, ...) while deeper layers will start looking for curves, textures, and eventually animals, then species on images. (*Visualizing and Understanding Convolutional Neural Networks* (Zeiler & Fergus 2014))

There exist several already trained neural network on different sets of labeled images. ImageNet, for example, is an agglomeration of millions of images which allowed searchers to compute weights on predefined architectures and make them freely available.

This project is based on the VGG-16 architecture which is detailed on the first annexed page. Based on the ImageNet dataset it was originally trained to classify 1000 type of classes. As our task in our case is binary (containing or not a settlement) we thus re-used its shape detection knowledge from its first layers. Below we can see that without tuning the last layers, an image containing several huts is labeled for VGG-16 as a honeycomb (the top five predicted classes are depicted next to the image).

Spotted Indigenous

```
honeycomb (18.86%)

[[('n03530642', 'honeycomb', 0.18862844),
  ('n03207743', 'dishrag', 0.03881475),
  ('n09246464', 'cliff', 0.029975755),
  ('n01917289', 'brain_coral', 0.020864176),
  ('n03980874', 'poncho', 0.01800093)]]
```

The knowledge stored and computed by the (deep) training can then be used to:

- Re-train some of the layers and update the weights of the model
- Add new layers and train them
- Extract features and use different machine learning regression/classification methods

In this project, we used a combination of these three methods. More precisely, we started by computing the output (features) from the VGG-16 model at a given point. Then we created a smaller architecture taking as input these features in order to predict our specific output, namely predicting if our image contains (or not) a settlement.

Constructed architecture:

```
BatchNormalization(axis=1, input_shape=conv_layers[-1].output_shape[1:]),
Convolution2D(128,3,3, activation='relu', border_mode='same'),
BatchNormalization(axis=1),
Convolution2D(128,3,3, activation='relu', border_mode='same'),
BatchNormalization(axis=1),
Convolution2D(128,3,3, activation='relu', border_mode='same'),
BatchNormalization(axis=1),
Convolution2D(17,3,3, border_mode='same'),
GlobalAveragePooling2D(),
Activation('softmax')
```

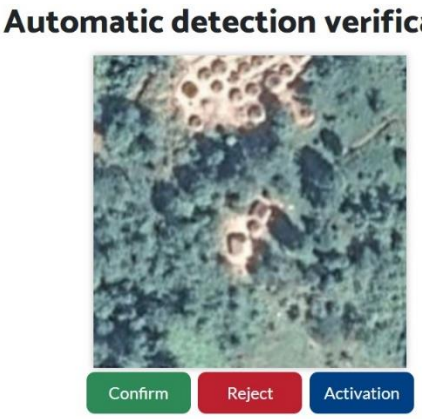
The architecture created is fully convolutional (figure above). This choice comes from the fact that a fully convolutional neural network gives us the opportunity to analyze the regions where the last convolutional layer activated for a given label. These regions under the form of a heatmap, could give us some intuitions and help us understand the way the computer classifies these images. Moreover, this also gives some interesting insights for the second part of this project, which consider the human verification. Indeed, having both the positively predicted image and its corresponding activation heatmap could help the user to understand why it was positively labeled.

Model improvement and human verification

On the project website, a verification part was afterward implemented to be able to refine the training of the model. The user has now the choice to gather more positively labeled data doing dome some free exploration, or to verify the output of the positively classified images (grid extracted from unexplored zones). A feature was also added showing the activation from the last layer to add

Spotted Indigenous

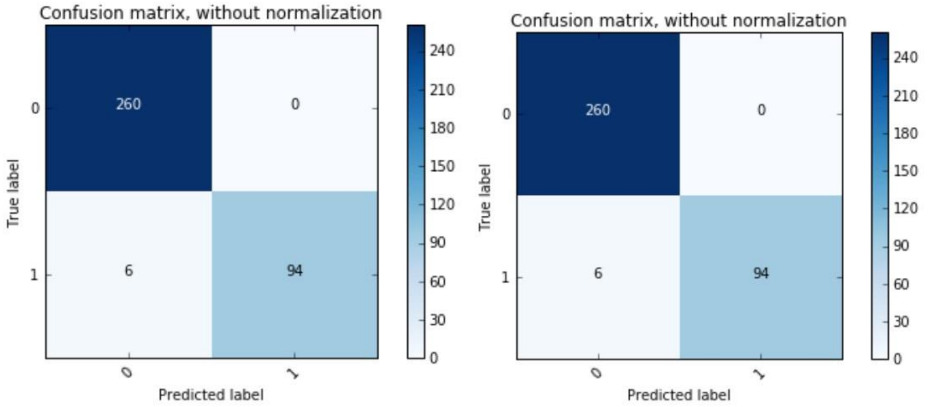
some insights. The results of these verifications are saved and depending on the user means, used to improve the model.



Verification interface (the user can confirm, reject or activate the heatmap layer)

Results

After training the model, the accuracy of the validation set is close to 98% which is better than the original ratio between the positive and negative images. We can see these results on the confusions matrix below.



Moreover, even if sometimes the activation layer can be difficult to interpret it gives for some images interesting feedback.

Spotted Indigenous

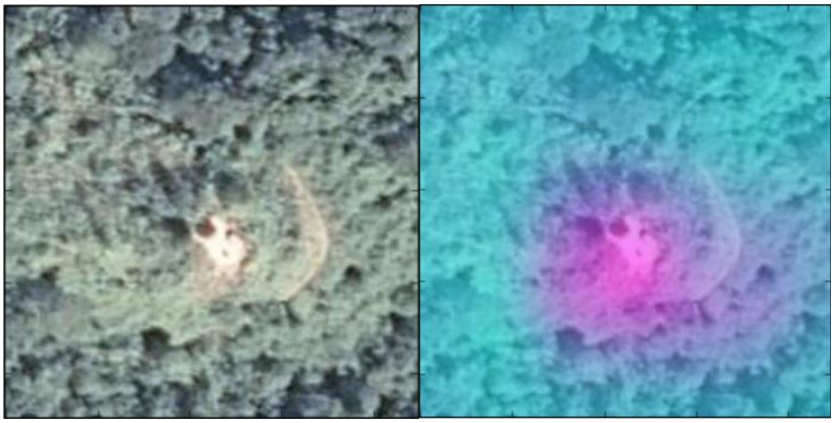


Image with its respective activation heatmap from the last convolutional layer

Conclusion

These results are motivating but we must consider the fact, that the model was trained on a small dataset and that the negatively labeled images are very similar to each other. Moreover, the extraction of the features from the already trained Vgg-16 model took almost 2 hours for 1700 images with a standard computer. As this step will have to be performed (on the back-end) for each new image extracted, it could lead to a substantial amount of time or require a better computational power.

The training of the fully convolutional network for the moment, do not necessitate a big amount of time either a lot of computational power but as for the feature computations, it can lead for a lot more data, to much more complex computations.

Spotted Indigenous

Annexes

VGG- 16 fully described

Layer (type)	Output Shape	Param #
lambda_1 (Lambda)	(None, 3, 224, 224)	0
zero_padding2d_1 (ZeroPaddin	(None, 3, 226, 226)	0
conv2d_1 (Conv2D)	(None, 64, 224, 224)	1792
zero_padding2d_2 (ZeroPaddin	(None, 64, 226, 226)	0
conv2d_2 (Conv2D)	(None, 64, 224, 224)	36928
max_pooling2d_1 (MaxPooling2	(None, 64, 112, 112)	0
zero_padding2d_3 (ZeroPaddin	(None, 64, 114, 114)	0
conv2d_3 (Conv2D)	(None, 128, 112, 112)	73856
zero_padding2d_4 (ZeroPaddin	(None, 128, 114, 114)	0
conv2d_4 (Conv2D)	(None, 128, 112, 112)	147584
max_pooling2d_2 (MaxPooling2	(None, 128, 56, 56)	0
zero_padding2d_5 (ZeroPaddin	(None, 128, 58, 58)	0
conv2d_5 (Conv2D)	(None, 256, 56, 56)	295168
zero_padding2d_6 (ZeroPaddin	(None, 256, 58, 58)	0
conv2d_6 (Conv2D)	(None, 256, 56, 56)	590080
zero_padding2d_7 (ZeroPaddin	(None, 256, 58, 58)	0
conv2d_7 (Conv2D)	(None, 256, 56, 56)	590080
max_pooling2d_3 (MaxPooling2	(None, 256, 28, 28)	0
zero_padding2d_8 (ZeroPaddin	(None, 256, 30, 30)	0
conv2d_8 (Conv2D)	(None, 512, 28, 28)	1180160
zero_padding2d_9 (ZeroPaddin	(None, 512, 30, 30)	0
conv2d_9 (Conv2D)	(None, 512, 28, 28)	2359808
zero_padding2d_10 (ZeroPaddi	(None, 512, 30, 30)	0
conv2d_10 (Conv2D)	(None, 512, 28, 28)	2359808

Spotted Indigenous

```
max_pooling2d_4 (MaxPooling2 (None, 512, 14, 14)       0
_________________________________________________________________
zero_padding2d_11 (ZeroPaddi (None, 512, 16, 16)       0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 512, 14, 14)       2359808
_________________________________________________________________
zero_padding2d_12 (ZeroPaddi (None, 512, 16, 16)       0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 512, 14, 14)       2359808
_________________________________________________________________
zero_padding2d_13 (ZeroPaddi (None, 512, 16, 16)       0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 512, 14, 14)       2359808
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 512, 7, 7)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              102764544
_________________________________________________________________
batch_normalization_1 (Batch (None, 4096)              16384
_________________________________________________________________
dropout_1 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_2 (Dense)              (None, 4096)              16781312
_________________________________________________________________
batch_normalization_2 (Batch (None, 4096)              16384
_________________________________________________________________
dropout_2 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_3 (Dense)              (None, 1000)              4097000
=================================================================
Total params: 138,390,312
Trainable params: 138,373,928
Non-trainable params: 16,384
_________________________________________________________________
```

Spotted Indigenous

VGG-16 without top layers (without activation layers)

Layer (type)	Output Shape	Param #
lambda_2 (Lambda)	(None, 3, 224, 224)	0
zero_padding2d_14 (ZeroPaddi	(None, 3, 226, 226)	0
conv2d_14 (Conv2D)	(None, 64, 224, 224)	1792
zero_padding2d_15 (ZeroPaddi	(None, 64, 226, 226)	0
conv2d_15 (Conv2D)	(None, 64, 224, 224)	36928
max_pooling2d_6 (MaxPooling2	(None, 64, 112, 112)	0
zero_padding2d_16 (ZeroPaddi	(None, 64, 114, 114)	0
conv2d_16 (Conv2D)	(None, 128, 112, 112)	73856
zero_padding2d_17 (ZeroPaddi	(None, 128, 114, 114)	0
conv2d_17 (Conv2D)	(None, 128, 112, 112)	147584
max_pooling2d_7 (MaxPooling2	(None, 128, 56, 56)	0
zero_padding2d_18 (ZeroPaddi	(None, 128, 58, 58)	0
conv2d_18 (Conv2D)	(None, 256, 56, 56)	295168
zero_padding2d_19 (ZeroPaddi	(None, 256, 58, 58)	0
conv2d_19 (Conv2D)	(None, 256, 56, 56)	590080
zero_padding2d_20 (ZeroPaddi	(None, 256, 58, 58)	0
conv2d_20 (Conv2D)	(None, 256, 56, 56)	590080
max_pooling2d_8 (MaxPooling2	(None, 256, 28, 28)	0
zero_padding2d_21 (ZeroPaddi	(None, 256, 30, 30)	0
conv2d_21 (Conv2D)	(None, 512, 28, 28)	1180160
zero_padding2d_22 (ZeroPaddi	(None, 512, 30, 30)	0
conv2d_22 (Conv2D)	(None, 512, 28, 28)	2359808
zero_padding2d_23 (ZeroPaddi	(None, 512, 30, 30)	0
conv2d_23 (Conv2D)	(None, 512, 28, 28)	2359808

Spotted Indigenous

```
max_pooling2d_9 (MaxPooling2 (None, 512, 14, 14)       0
_________________________________________________________________
zero_padding2d_24 (ZeroPaddi (None, 512, 16, 16)       0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 512, 14, 14)       2359808
_________________________________________________________________
zero_padding2d_25 (ZeroPaddi (None, 512, 16, 16)       0
_________________________________________________________________
conv2d_25 (Conv2D)           (None, 512, 14, 14)       2359808
_________________________________________________________________
zero_padding2d_26 (ZeroPaddi (None, 512, 16, 16)       0
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 512, 14, 14)       2359808
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 512, 7, 7)         0
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
```

VGG-16 without last convolutional layers:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lambda_1 (Lambda)            (None, 3, 224, 224)       0
_________________________________________________________________
zero_padding2d_1 (ZeroPaddin (None, 3, 226, 226)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 224, 224)      1792
_________________________________________________________________
zero_padding2d_2 (ZeroPaddin (None, 64, 226, 226)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 224, 224)      36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 64, 112, 112)      0
_________________________________________________________________
zero_padding2d_3 (ZeroPaddin (None, 64, 114, 114)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 128, 112, 112)     73856
_________________________________________________________________
zero_padding2d_4 (ZeroPaddin (None, 128, 114, 114)     0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 128, 112, 112)     147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 128, 56, 56)       0
_________________________________________________________________
zero_padding2d_5 (ZeroPaddin (None, 128, 58, 58)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 256, 56, 56)       295168
_________________________________________________________________
zero_padding2d_6 (ZeroPaddin (None, 256, 58, 58)       0
_________________________________________________________________
```

Spotted Indigenous

```
conv2d_6 (Conv2D)            (None, 256, 56, 56)       590080
_________________________________________________________________
zero_padding2d_7 (ZeroPaddin (None, 256, 58, 58)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 256, 56, 56)       590080
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 256, 28, 28)       0
_________________________________________________________________
zero_padding2d_8 (ZeroPaddin (None, 256, 30, 30)       0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 512, 28, 28)       1180160
_________________________________________________________________
zero_padding2d_9 (ZeroPaddin (None, 512, 30, 30)       0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 512, 28, 28)       2359808
_________________________________________________________________
zero_padding2d_10 (ZeroPaddi (None, 512, 30, 30)       0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 512, 28, 28)       2359808
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 512, 14, 14)       0
_________________________________________________________________
zero_padding2d_11 (ZeroPaddi (None, 512, 16, 16)       0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 512, 14, 14)       2359808
_________________________________________________________________
zero_padding2d_12 (ZeroPaddi (None, 512, 16, 16)       0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 512, 14, 14)       2359808
_________________________________________________________________
zero_padding2d_13 (ZeroPaddi (None, 512, 16, 16)       0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 512, 14, 14)       2359808
=================================================================
Total params: 14,714,688
Trainable params: 14,714,688
Non-trainable params: 0
_________________________________________________________________
```

Fully convolutional network for fine tuning following the one on top (from vgg-16)

```
BatchNormalization(axis=1, input_shape=conv_layers[-1].output_shape[1:]),
Convolution2D(128,3,3, activation='relu', border_mode='same'),
BatchNormalization(axis=1),
Convolution2D(128,3,3, activation='relu', border_mode='same'),
BatchNormalization(axis=1),
Convolution2D(128,3,3, activation='relu', border_mode='same'),
BatchNormalization(axis=1),
Convolution2D(17,3,3, border_mode='same'),
GlobalAveragePooling2D(),
Activation('softmax')
```