420

The first CNN for cannabis category recognition

Introduction

Why this project ?

In France, cannabis isn't legal yet, which means that **you aren't able to know the quality or the category of the cannabis you're buying**. I used to smoke a lot, and I've always been angry about the fact that I wasn't able to know in advance what will be the effects of the cannabis I was smoking. Even though I don't smoke anymore, this problem is still annoying me.

What do you mean by cannabis category ?

There are **3 main cannabis categories** : Indica, Sativa and Hybrid. Each of these category has different effect on the body. As found on Leafly :

- **Indicas** strains are believed to be physically sedating, perfect for relaxing with a movie or as a nightcap before bed.
- **Sativas** tend to provide more invigorating, uplifting cerebral effects that pair well with physical activity, social gatherings, and creative projects.

- **Hybrids** are thought to fall somewhere in between the indica-sativa spectrum, depending on the traits they inherit from their parent strains.

You understood... Let's say it's sunday night, and you want to rest to be in shape for the coming week, so you smoke some cannabis. Sadly, you bought some sativa without knowing it. Resting will be complicated.

How to solve the problem ?

Some firm have already address this problem by creating high-tech cannabis analyzer.

But the price of these analyzer is way to high for the common mortal, the prices range from $15,000 to $25,000. Even if these device are very accurate (thanks god, in view of the price), nobody will pay these prices to gain a better understanding over the cannabis they're buying.

What if a simple picture was enough ?

Imagine taking a picture of your cannabis and getting back an analysis of it, for free. With today technologies, it isn't a dream anymore. That's why I've built the first CNN for cannabis category recognition. My CNN, called 420, is able to take an image of cannabis as input and gives back its category (sativa / indica / hybrid) as output. Welcome to deep learning.

Dataset

Data Preprocessing

I found all of the images directly on Leafly, this website is incredible ! There are a lot of public resources on this website such as informations, images or reviews about cannabis. I've gathered every cannabis images URL in a CSV file called ***images-url.csv***.

I've downloaded all of these images using the Requests Python library. The code can be found in the ***data-extraction.ipynb*** notebook.

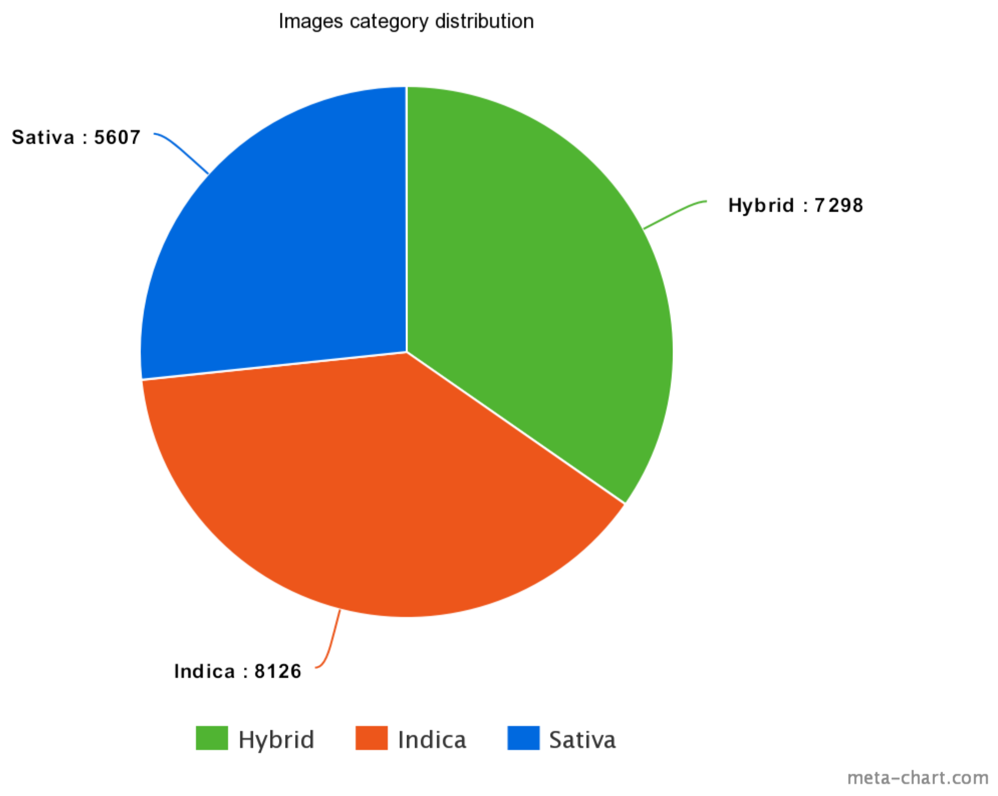
Once I've downloaded all the images I've reorganized them by training, validation and testing set. The code can be found in the ***data-segmentation.ipynb*** notebook.

Except the download and the organization of the data (***data-segmentation.ipynb***), no other data preprocessing method were applied. The images were already labeled and correctly sized (every images were 100px X 100px)

Data Exploration

At the end I've gathered 21,031 images of cannabis. 7,298 of hybrid, 8,126 of indica and 5,607 of sativa.

Here is a graph of the distribution of my image dataset :



All the images has the same size : 100px X 100px and all the images are colored.

Here are 3 samples of the images from my dataset :

One example of hybrid cannabis :

One example of indica cannabis :

And one example of sativa cannabis :

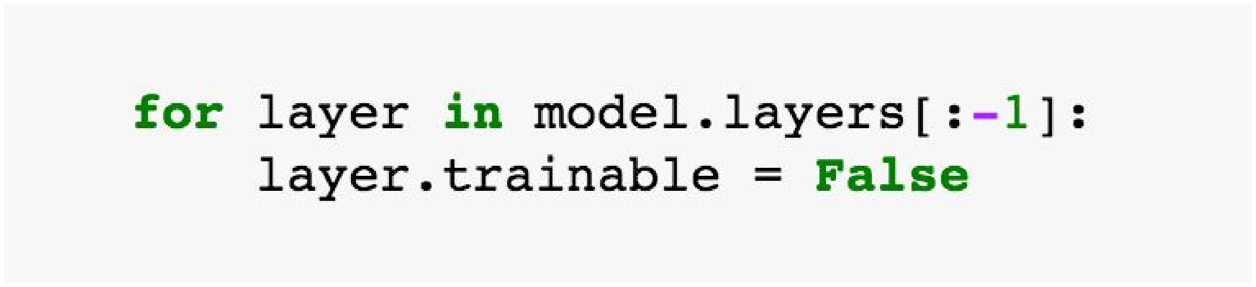
The principal differences are between indica and sativa. As you can see sativa has greater leafs than the indica. The hybrid category is hard to distinguish.

CNN Architecture

Which architecture did you chose ?

In order to get the best results I decided to use transfer learning. I've tried all the greatest architecture including Xception, VGG16, VGG19, ResNet50 and InceptionV3

I've frozen all of the layers except the last one in order to preserve the knowledge gained on imagenet.



```
for layer in model.layers[:-1]:
    layer.trainable = False
```

I've then added 2 dense layer of 1024 nodes and a Dropout layer between them, in order to don't get the weights stuck in a certain part of the network. I've then added an output layer with a softmax function in order to get an estimation of the cannabis category. This way we will be able to tell our users how much they can trust our predictions.

```
x = model.output
x = Flatten()(x)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.5)(x)
x = Dense(1024, activation="relu")(x)
predictions = Dense(3, activation="softmax")(x)
```

I've chosen this architecture after crash testing others. I've found that by adding two dense layer, the output predictions are better. Because more layer our model has, more it become good at predicting. Warning : adding too much layer may ending in a model not very good using the test set.

I've then used the Stochastic Gradient Descent as the optimizer of my model and I used a small learning rate (0.0001).

I've chosen this optimizer because Stochastic Gradient Descent is because of its noisy updates that can allow the model to avoid local minima.

```
model_final.compile(
    loss = "categorical_crossentropy",
    optimizer = optimizers.SGD(lr=0.0001, momentum=0.9),
    metrics=["accuracy"]
)
```

CNN Training

To train my model I first decided to use data augmentation. Data augmentation helps our algorithms to be more adaptive to new images in the future. By rotating, distorting, or rescaling our images, the algorithms gain in invariance.

```
train_datagen = ImageDataGenerator(
    rescale = 1./255,
    horizontal_flip = True,
    fill_mode = "nearest",
    zoom_range = 0.3,
    width_shift_range = 0.3,
    height_shift_range=0.3,
    rotation_range=30
)

test_datagen = ImageDataGenerator(
    rescale = 1./255,
    horizontal_flip = True,
    fill_mode = "nearest",
    zoom_range = 0.3,
    width_shift_range = 0.3,
    height_shift_range=0.3,
    rotation_range=30
)
```

I then used these generator in order to generate more images from my existing images.

```
train_generator = train_datagen.flow_from_directory(
    'datasets/train',
    target_size = (img_height, img_width),
    batch_size = batch_size,
    class_mode = "categorical"
)

validation_generator = test_datagen.flow_from_directory(
    'datasets/valid',
    target_size = (img_height, img_width),
    class_mode = "categorical"
)
```

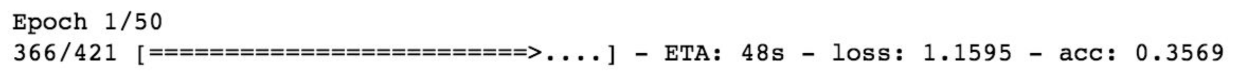
I started first with 50 epochs and a batch size of 421 images. I used that to benchmark the different architecture. I found that ResNet50 was the best architecture for my model.

To decide which architecture was the best for my classification problem I’ve tried different architectures : Xception, VGG16, VGG19, ResNet50 and InceptionV3 and chose the one with the best performance (every performances were evaluated using the accuracy metric).

The metrics used **to track the evolution of my model is the accuracy metric**. I’ve passed it as a metrics argument when I was compiling my model. I then used the validation accuracy (accuracy on the validation set) to validate the effectiveness of my model.

The accuracy metric can be defined as the fraction of predictions our model got right. It’s formula can be written as follow :

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}$$



```
Epoch 1/50
366/421 [=========================>....] - ETA: 48s - loss: 1.1595 - acc: 0.3569
```

Despite all my efforts and different architecture my accuracy was not excellent.

How does the model work ?

So my CNN model use the pre-trained ResNet50 model and add it 2 other layers. The benefits of using a pre-trained model is that we can leverage the knowledge gained by this model and we specialize it in our task by freezing all of the weights except the last one. Than the output is processed by a small neural networks in order to gain more specificity.

CNN Improvement

To improve my model I used different batch size for the learning process (16 and 32).

I've then look at different learning rate from 0.01 to 0.0001 and I've found that the best one was 0.0001.

I've didn't implement Grid search because of the computation cost that were already too high. My AWS instance is not that good and the price goes up quickly so I had to make some trade off.

After the improvements the validation accuracy of my modal was around 41%. Before the improvement the model validation accuracy was around 39%.

I think that my model is quite robust but the problem comes from the data. Some of them are very different from the other (cannabis looks very differently while flowering than after flowering). The validation set comes very handy when trying to avoid variance in the model. Each stochastic gradient step is checked using a validation set (different from the set that the step was trained on) to see the performance of the improve algorithm on other data.

CNN Results

The accuracy of my model is only of 41%, at this point I think that the problem comes from the dataset. The image are very hard to labeled because it's very hard to recognize cannabis category even for humans.

So it made the process of assessing the quality of the images very hard.

As described in the proposal I had no benchmark model to compare from as this problem has never been explore by Data Science before (at list not publicly). So I can't say that my model is under the benchmark.

Nonetheless I can clearly affirm that my model is not good enough to solve the problem. With only 41% of accuracy, it's not enough to draw conclusion from its predictions.

I think one of the main reasons was the noise in the dataset and also the hybrid class. The hybrid class creates noise because it looks a lot like the indica and make the task of classifying very hard.

Challenges encountered

One of the main problem I had was with my AWS instance. The instance was not able to train multiple CNN at the same time (too high memory consumption) so I had to run them one after the other.

The other problem I faced was with the organization of the dataset. In every previous Computer Vision project I've always use an open-source dataset that was already organized in train / validation / test folder for me. Writing the algorithm of folder organization (***data-segmentation.ipynb)*** was a little bit difficult but quite satisfying.

Conclusion

I'm very happy of this project. I hope you'll find it cool too. Even if the conclusion is not very enthusiastic I hope it gave you the passion to go forward in computer vision. Even if the problems are not always solvable, it worth trying.

To conclude we saw the differences in cannabis category, define the problem of recognizing cannabis category from a simple picture instead of buying a $20,000 analyzer.

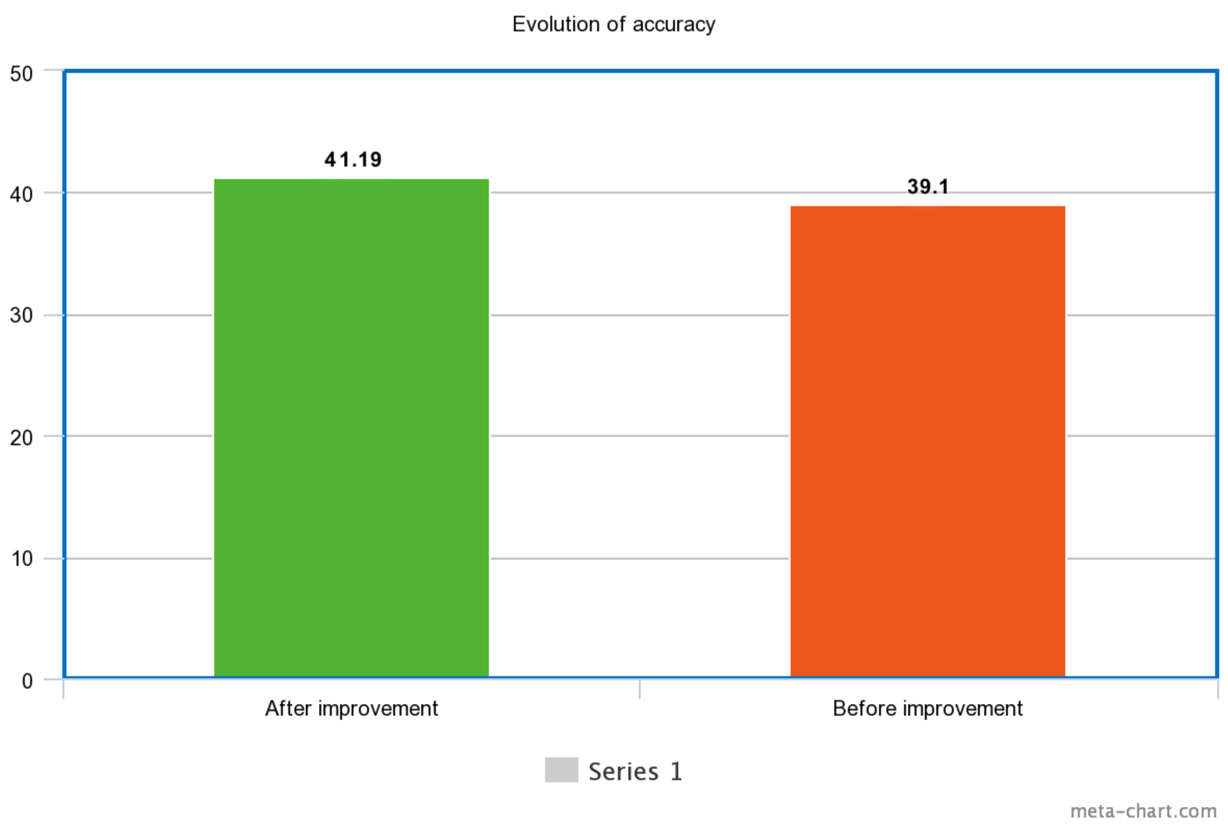
We then saw that the best solution for this was to train a CNN classifier able to recognize cannabis category. So we decided to leverage the knowledge acquired by pre-trained model such as ResNet50 in order to gain time. We also add our custom layers at the end of this pre-trained model in order to gain specificity on the topic.

We tried different hyperparameters and architecture in order to choose the one with the best performance. We also assessed performance using the accuracy metric.

As we saw in the CNN results section the results were not very encouraging (only 41% of validation accuracy).

In the next section we'll see different suggestions to improve our model.

Here you can find a graph modeling the different accuracy after and before improvement :



Possible Improvement

As I mentioned earlier, I think that the bad performance metric come from the hybrid category which is very similar to the indica category. One simple improvement could be to try with only two classes and see how it performs.

Another possible improvement would be to use a larger AWS server and start training with grid search and higher epochs number.

References

- https://www.leafly.com/
- https://keras.io/applications/
- https://medium.com/@14prakash/transfer-learning-using-keras-d804b2e04ef8