Projet 8 : Déployez un modèle dans le cloud

Lancelot LECLERCQ

10 juin 2022

Sommaire

Introduction

Problématique

- Préservation de la biodiversité des fruits
  - traitements spécifiques pour chaque espèce de fruits par des robots cueilleurs intelligents

- 1ère étape développer une application mobile
  - Permettre à l'utilisateur d'obtenir des informations sur un fruit à partir d'une photo
  - Sensibiliser le grand public à la biodiversité des fruits
  - Mettre en place une première version du moteur de classification des images de fruits
  - Construire une première version de l'architecture Big Data

Données

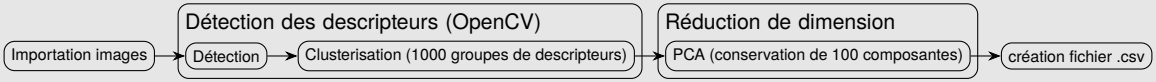
- Utilisation du jeu de données Kaggle : `https://www.kaggle.com/datasets/moltean/fruits`
- Nombre total d'images : 90483
- Taille du jeu d'entrainement : 67692 images
- Taille du jeu de test : 22688 images
- Nombre de fruits : 131

- Exemple d'image de fruit :

BigData

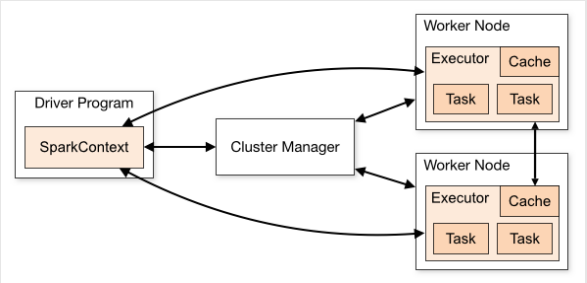
Spark

- Parallélisation des calculs
  - Utilisation de plusieurs machines
  - Distribution des calculs
- Coût d'utilisation des machines à surveiller
- Utilisation de PySpark une interface Spark en python



Spark

- SparkContext supervise la répartition des taches
- Différentes taches sont réparties sur les différents nœuds



PySpark : Importation des données

- Importation des images au format binaire :

```
ImgData = spark.read.format('binaryFile') \
                    .option('pathGlobFilter', '*.jpg') \
                    .option('recursiveFileLookup', 'true') \
                    .load(path) \
                    .select('path', 'content')
```

- Récupération d'un label (nom du fruit et espèce) à partir du chemin d'accès au fichier image :

```
ImgData = ImgData.withColumn('label',
                             F.element_at(F.split(F.col('path'), '/'), -2))
```

- Création d'un nom d'image du label et du nom de l'image :
- Ex :
  - image : 0_100.jpg
  - label : Apricot
  - nom : Apricot_0_100.jpg

```
ImgData = ImgData.withColumn(
    'imgName',
    F.concat('label', F.lit('_'), F.element_at(F.split(F.col('path'), '/'),
                                               -1)))
```

PySpark : Détection des descripteurs

- Fonction permettant de détecter les descripteurs avec OpenCV :

- Utilisation des données binaires des images grâce à io.BytesIO

- Inspiré de `https://stackoverflow.com/questions/60192589/problem-with-pyspark-udf-to-get-descriptors-with-opencv-problem`

```
def get_desc(content):
    try:
        img = np.array(Image.open(io.BytesIO(content)))
    except:
        print(content)
        img = None
        return img
    if img is None:
        desc = None
    else:
        orb = cv.ORB_create(nfeatures=100)
        keypoints_orb, desc = orb.detectAndCompute(img, None)
    if desc is None:
        desc = [np.array(32 * [0]).astype(np.float64).tolist()]
    else:
        desc = desc.astype(np.float64).tolist()
    return desc
```

- Création d'une colonne avec les descripteurs détectés pas OpenCV :

```
udf_image = F.udf(
    get_desc,
    ArrayType(ArrayType(FloatType(), containsNull=False), containsNull=False))
ImgDesc = ImgData.withColumn("descriptors", F.explode(udf_image("content")))
```

PySpark : Création d'un *bag of visual words*

- Utilisation des fonction de machine learning de pyspark : KMeans
  - Création de groupes de descripteurs

```
kmean = KMeans(k=1000, featuresCol='descriptors', seed=0)
model = kmean.fit(ImgDesc)

Pred = model.transform(ImgDesc)
```

- Utilisation de la fonction groupBy pour compter le nombre d'occurrences de chaques groupes de descripteurs

```
ImgPred = Pred.groupBy('label', 'prediction').count()

BoVW = ImgPred.groupBy('label').pivot('prediction').sum('count').fillna(0)
```

PySpark : Réduction de dimension

- Réduction de dimension par PCA :

```
VA = VectorAssembler(inputCols=BoVW.drop('label').columns,
                     outputCol='features')
pca = PCA(k=100, inputCol='features', outputCol='pca_features')
pipe = Pipeline(stages=[VA, pca])

pipePCA = pipe.fit(BoVW)

pcaData = pipePCA.transform(BoVW)
pcaDataDF = pcaData.select(['label', 'pca_features']).toPandas()
```

Déploiement sur le cloud

Déploiement sur le cloud

- Utilisation d'une machine EC2 hébergée par Amazon Web Services
  - Utilisation de Debian
  - Connexion par SSH
  - Installation de java, git
- Utilisation de S3 comme espace de stockage pour la base de données d'images

Déploiement sur le cloud

- Essais avec t2.micro
  - 1 CPU
  - 1 GB de mémoire RAM
  - 10 Go de mémoire disque
  - Connexion réseau lente
- Utilisation d'un jeu d'entrainement très léger : 2 types de fruits
- Offre gratuite
- Coût très faible, mais peu de ressources de calculs

- Essais avec t3.2xlarge
  - 8 CPU
  - 32 GB de mémoire RAM
  - 10 Go de mémoire disque
  - Connexion réseau jusqu'à 5 Gbit
- Sample d'un dixième de la base originale $\simeq$ 60000 image soit environ 6000 images
- 0,3776 $ par heure
- Coût plus important, mais supporte mieux la charge de calculs

Déploiement sur le cloud
Configuration de l'utilisation des services Amazon

- Téléchargement des packages permettant de communiquer avec le stockage S3

```
os.environ[
    'PYSPARK_SUBMIT_ARGS'] = '--packages com.amazonaws:aws-java-sdk:1.12.230, \
                              org.apache.hadoop:hadoop-aws:3.3.1 pyspark-shell'
```

- Configuration de hadoop pour gérer la connexion au stockage sur S3
  - Utilisation d'un fichier *credentials* dans le dossier /.aws/ afin d'avoir les clés de connexion facilement accessibles

```
spark = SparkSession.builder.master('local').appName(
    'FruitsPreProc').getOrCreate()
sc = spark.sparkContext
sc._jsc.hadoopConfiguration().set('fs.s3a.impl',
                                  'org.apache.hadoop.fs.s3a.S3AFileSystem')
sc._jsc.hadoopConfiguration().set(
    "fs.s3a.aws.credentials.provider",
    "com.amazonaws.auth.profile.ProfileCredentialsProvider")
sc._jsc.hadoopConfiguration().set("fs.s3a.endpoint",
                                  "s3.eu-west-3.amazonaws.com")
spark.sparkContext._conf.getAll()
```

Axes d'amélioration

Axes d'amélioration

- KMeans instable, difficile à débugger :

⇒ Essayer d'autres alternatives comme dask

- Utilisation d'un EMR permettant d'avoir les calculs réalisés par plusieurs machines

- Ajout d'un modèle de classification afin de classer les fruits

- Affiner les catégories avec des maturités différentes afin de cueillir les fruits au meilleur moment