LE MACHINE-LEARNING AVEC PYTHON

Etude de cas : regroupement d'images par similarités

Corentin Martel

Data Scientist Freelance

https://www.corelyo.com

CONTEXTE DU PROJET

Création d’une plateforme d’analyse de positionnement marketing pour des marques de luxe

CONTEXTE DU PROJET

Objectif : mettre en évidence les différences de stratégies sur Instagram

CONTEXTE DU PROJET

Objectif : pouvoir afficher les images de plusieurs marques sur un même écran, en les regroupant par similarité

Problématique : comment comparer des images de manière non-supervisée?

LES DONNÉES

COLLECTER DES DONNÉES

Données disponible : liste d'url d'images Instagram

```python
from PIL import Image
import requests
from io import BytesIO
import numpy as np
```

```python
def download_from_url(url):
    img_path = []
    ret=[]
    first = True
    try :
        if len(url.split('/')[-1]) <=75 :
            response = requests.get(url)
            img = Image.open(BytesIO(response.content))
            img = img.resize((224,224), Image.ANTIALIAS)
            x = np.asarray(img)
            ret = np.expand_dims(x, axis=0)
            img_path = url.split('/')[-1]
            return img_path, ret
    except Exception as e:
        print(e)
```

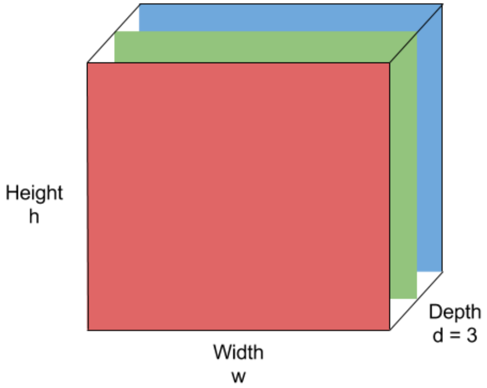
```python
import pandas as pd
from multiprocessing.pool import ThreadPool
```

```python
# download pixels and shape it
%%time

df_url = pd.read_excel('./data/IG_Posts_Brands_prepared.xlsx')
url_list = [_ for _ in df_url.thumbnail[:500] if len(_)<=75]

if __name__ == '__main__':
    num_threads = 7
    p = ThreadPool(num_threads)
    xs = p.map(download_from_url, url_list)
    xs = [elt for elt in xs if elt is not None]
    img = [_[0] for _ in xs]
    X = np.concatenate([_[1] for _ in xs],axis=0)
```

```python
print("Nombre d'images : {}\nX.shape : {}".format(len(img), X.shape))
```

```
Nombre d'images : 10000
X.shape : (10000, 224, 224, 3)
```

COMMENT DÉCRIRE UNE IMAGE ?



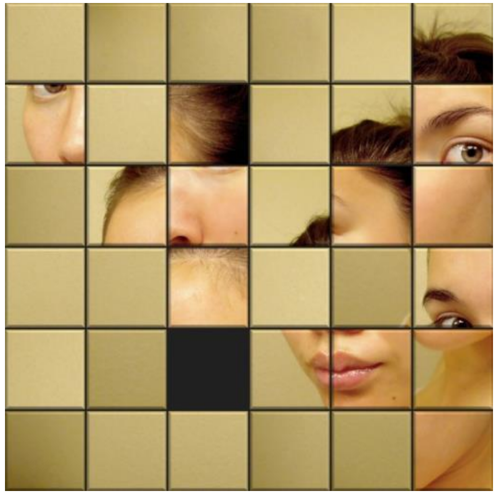
https://blog.datawow.io/interns-explain-cnn-8a669d053f8b

Image = matrice de dimension L * H * 3

Utiliser les pixels ?

- Nombre de variables : 150 528 variables pour 224*224 pixels
- Ces variables sont ordonnées : on ne peut pas les considérer comme un ensemble de variables indépendantes

$\neq$

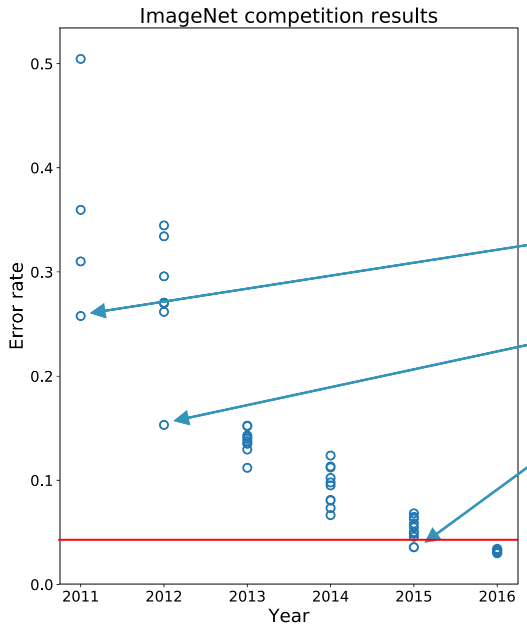


https://unmontonderegalos.com/en/home/16-sliding-puzzle.html

UN PEU D'HISTOIRE : IMAGENET

Base de donnée de 14 millions d'images dans 21 000 catégories

Depuis 2010 : concours international : prédire la catégorie de l'image (1,2 millions d'images, 1000 catégories)



Gkrusze (https://commons.wikimedia.org/wiki/File:ImageNet_error_rate_history_(just_systems).svg), https://creativecommons.org/licenses/by-sa/4.0/legalcode

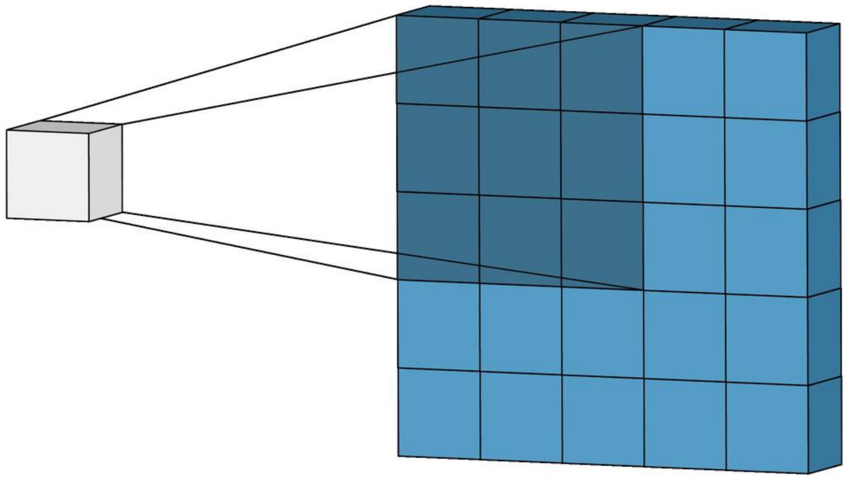
2011 : 25,8%, sans réseaux de neurones

2012 : AlexNet : 15,3 % (2eme place : 26,2%)

2015 : ResNet-152 : 3,6%

Humain (5,1%) : https://cs.stanford.edu/people/karpathy/ilsvrc/

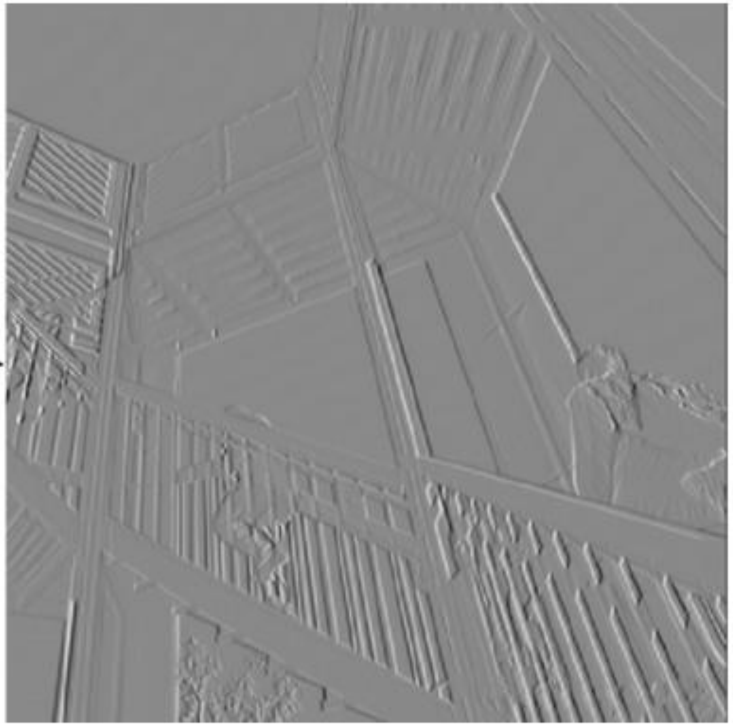
LES CONVOLUTIONS



LES CONVOLUTIONS

Exemple de filtre de détection de lignes verticales :

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

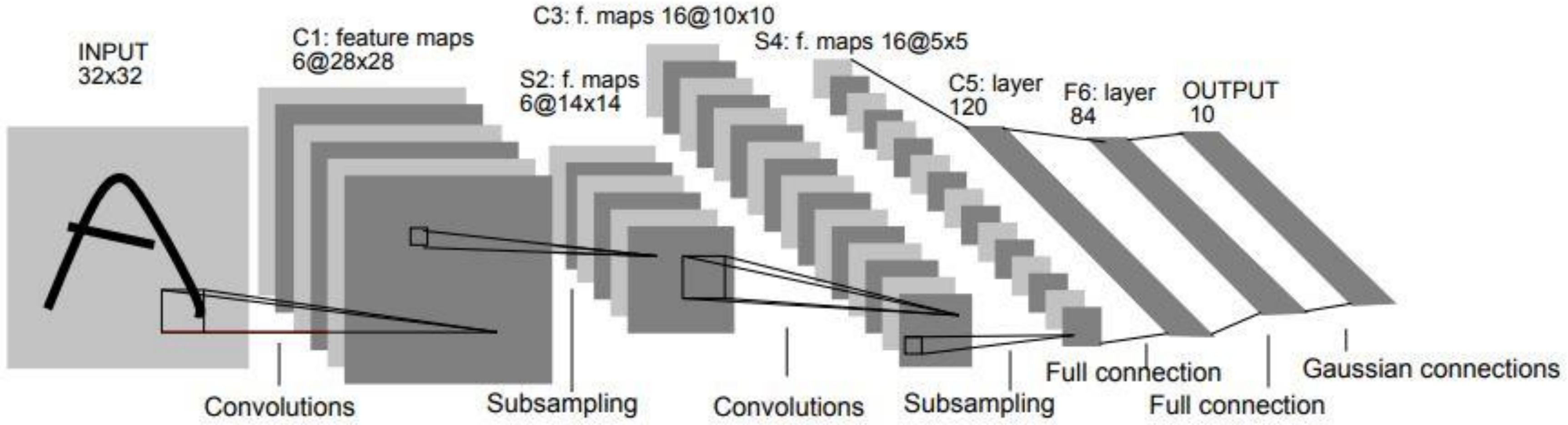


Applying a vertical edge detector kernel

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

LES RÉSEAUX CONVOLUTIFS

1998 : LeNet-5 : reconnaissance de chiffres sur des chèques



Original Image published in [LeCun et al., 1998]

http://scs.ryerson.ca/~aharley/vis/conv/

DESCRIPTION DES IMAGES

CHOIX DU RÉSEAU CONVOLUTIF

ResNet50 : bons résultats

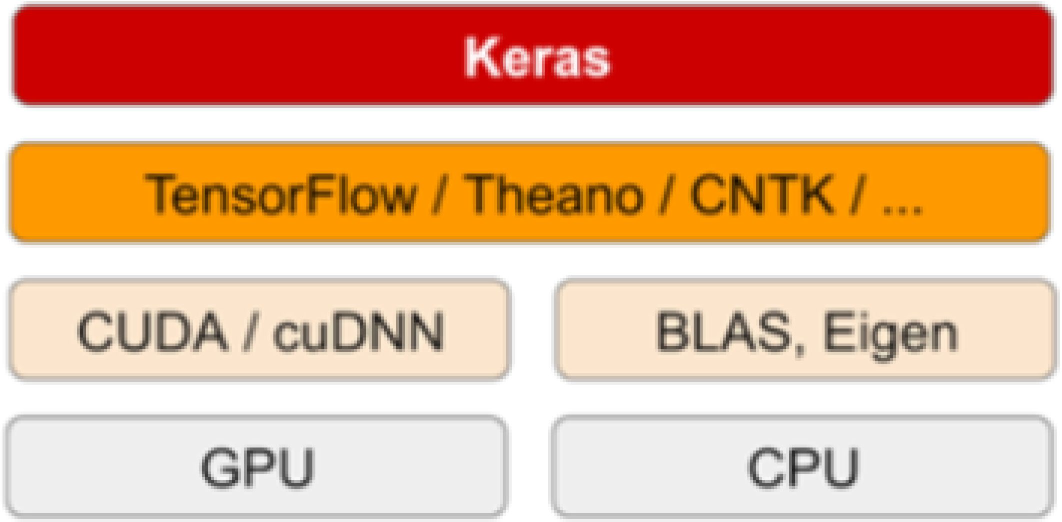
50 couches

25,6 millions de paramètres (61k pour LeNet5)

Durée d'entrainement : 14 jours (Tesla P100 x8)

Utilisation d'un réseau pré-entrainé sur ImageNet pour extraire les variables

IMPLÉMENTATION : KERAS



https://towardsdatascience.com/my-journey-into-deeplearning-using-keras-part-1-67cbb50f65e6

KERAS

```python
import keras
from keras.applications.imagenet_utils import preprocess_input
from keras.preprocessing import image
from keras.preprocessing.image import ImageDataGenerator
def extract_features(img,
                     X,
                     batch_size = 256,
                     feature_save_file = './raw_features.h5'):
    """
    Extract features from an image using resnet and saves in HDF5 files.
    img : list of img names
    X : matrix of shape (Nb of images, 224, 224, 3)
    batch size : for ResNet.
    feature_save_file : HDF5 file where ResNet features of the images are stored
    """

    feat_extractor = keras.applications.resnet50.ResNet50(input_shape = (224,224,3),
                                                          weights='imagenet',
                                                          include_top=False,
                                                          pooling='max')

    datagen = ImageDataGenerator(preprocessing_function=preprocess_input)

    generator = datagen.flow(
                X,
                batch_size=batch_size,
                shuffle=False)

    features = feat_extractor.predict_generator(generator)

    print(features.shape)
    df_features = pd.DataFrame(features,columns=["feat_"+str(i) for i in range((features.shape[1]))])
    df_features['img'] = img
    with pd.HDFStore(feature_save_file) as store:
        store.append('df_features', df_features, data_columns = ['img'], min_itemsize=75)
```

```python
%%time
extract_features(img,
                 X,
                 batch_size = 256,
                 feature_save_file = feature_save_file)
```

```
(10000, 2048)
Wall time: 30.9 s
```

PROJECTION EN 2D

PROJECTION

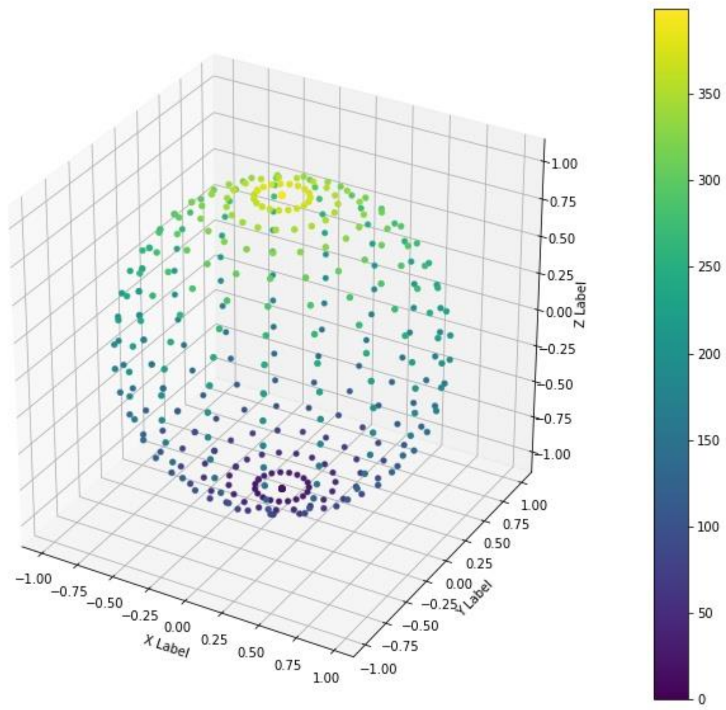
Objectif : créer des coordonnées (x,y) pour chaque image

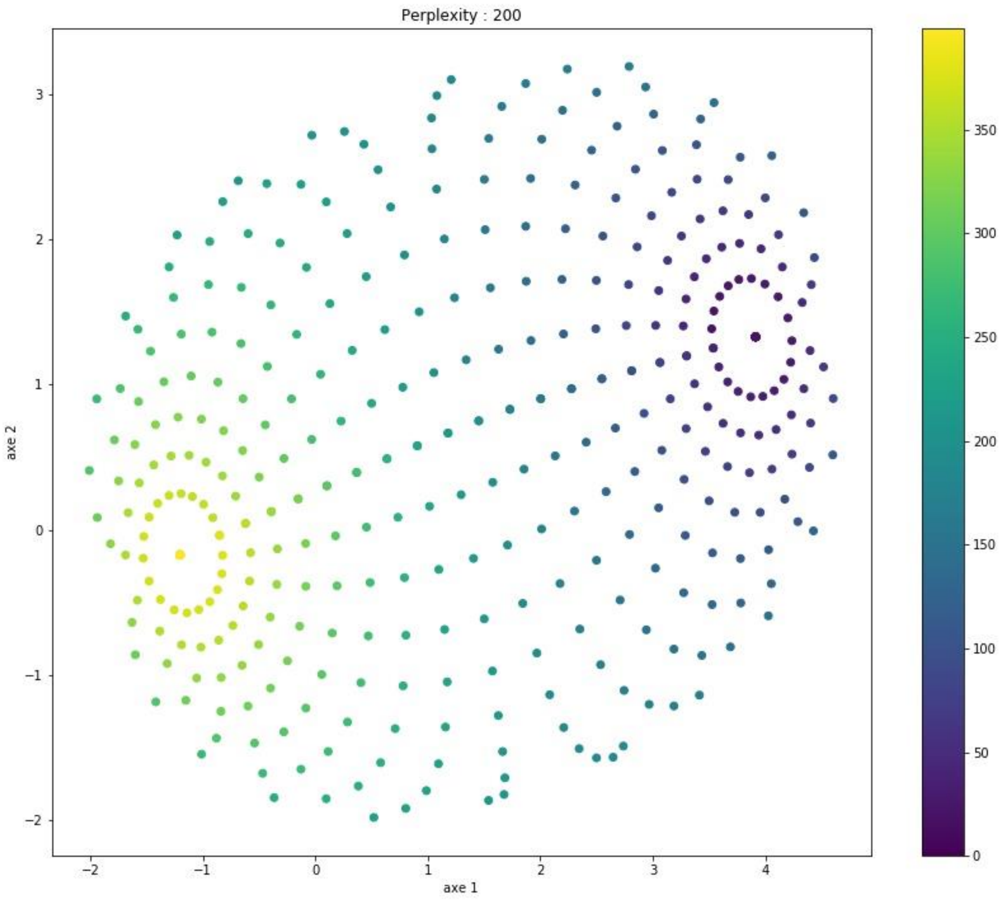
Contrainte : des images proches doivent être similaires

↓

T-SNE : algorithme de réduction de dimensions conservant les relations de proximité

PROJECTION : T-SNE





PROJECTION : T-SNE

```python
def do_tsne(img,
            pca_feature_save_file = './pca_features.h5',
            tSNE_feature_save_file = './tSNE_features.h5',
            perplexity = 10
           ):
    """
    Load PCA features of an image, transform it using parametric t-SNE and save it as HDF5.
    img : list of img names, should be in the pca_feature_save_file.
    pca_feature_save_file : HDF5 file where PCA features of the images are stored
    ptSNE_model_path : path to the ptSNE model file (HDF5)
    tSNE_feature_save_file : HDF5 file to save transformed images
    training : Boolean : if True, trains the ptSNE model and saves it.
    """

    with pd.HDFStore(pca_feature_save_file) as store:
    #     for df in store.select('df_pca_features', where = "img in img", chunksize=chunksize):
        df = store.select('df_pca_features', where = "img in img")

    # load images
    img = df['img'].as_matrix()
    df.drop(['img'],inplace=True,axis=1)
    pca_features = df.as_matrix()

    print(pca_features.shape, img.shape)
    high_dims = pca_features.shape[1]

    output_res = TSNE(n_components=2,perplexity=perplexity).fit_transform(pca_features)

    df_tSNE_features = pd.DataFrame(output_res,columns=["tx","ty"])
    df_tSNE_features['img'] = img

    with pd.HDFStore(tSNE_feature_save_file) as store:
        store.append('df_tSNE_features', df_tSNE_features, data_columns = ['img'], min_itemsize=75)
```

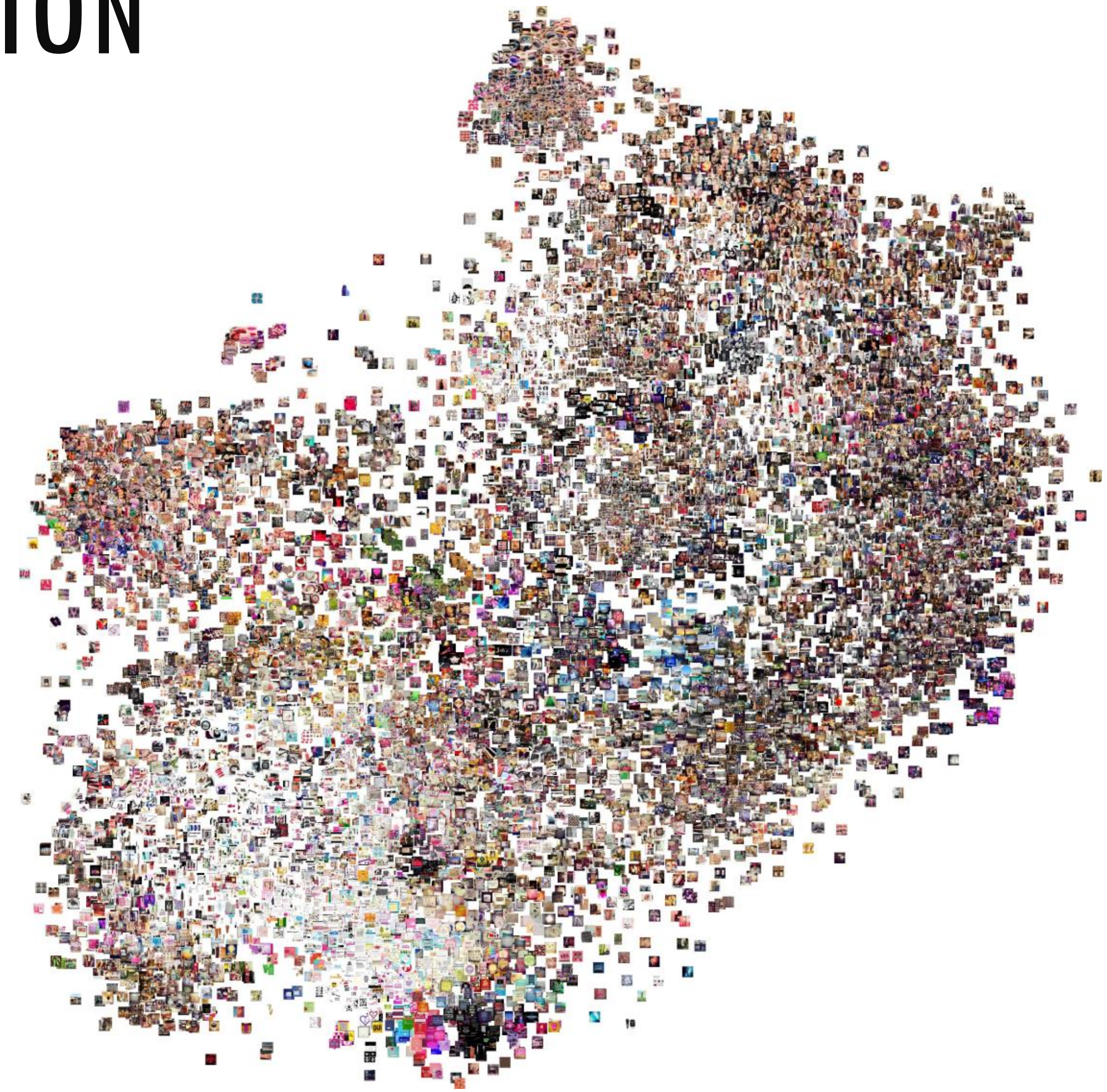
VISUALISATION

VISUALISATION

```python
def visualize(img,
              tSNE_feature_save_file = './tSNE_features.h5',
              save_img_path = "./ptSNE_img.png",
              img_path = './data/img/small_sample/file/',
              width = 5000,
              height = 5000,
              max_dim = 25):

    import matplotlib.pyplot as plt
    with pd.HDFStore(tSNE_feature_save_file) as store:
        df = store.select('df_tSNE_features', where = "img in img")

    tx = df['tx']
    ty = df['ty']
    img = df['img']

    min_x = np.min(tx)
    max_x = np.max(tx)
    min_y = np.min(ty)
    max_y = np.max(ty)
    tx = (tx-min_x) / (max_x - min_x)
    ty = (ty-min_y) / (max_y - min_y)


#     t0 = time.time()
    full_image = Image.new('RGB', (width, height),color=(255,255,255)) # RGBA for png
    i = 0
    for img_, x, y in zip(img[:], tx[:], ty[:]):
        if i%int(len(tx[:])/10.)==0:
            print('img {}/{}'.format(i,len(tx[:])))
        i+=1
        tile = Image.open(img_path + img_)
        rs = max(1, tile.width/max_dim, tile.height/max_dim)
        tile = tile.resize((int(tile.width/rs), int(tile.height/rs)), Image.ANTIALIAS)
        full_image.paste(tile, (int((width-max_dim)*1*x), int((height-max_dim)*1*y)), mask=tile.convert('RGBA'))

#     print("total time : {}".format(time.time()-t0))
    plt.figure(figsize = (16,12))
#     plt.imshow(full_image)
    full_image.save(save_img_path)
```

```python
%%time
save_img_path = "./tests_perplexity/ptSNE_img_train_p{}_V2.jpg".format(perplexity)
######################### below is visualisation############################################
visualize(img[:max_img_number],
          tSNE_feature_save_file = tSNE_feature_save_file,
          save_img_path = save_img_path,
          img_path = img_path,
          width = 5000,
          height = 5000,
          max_dim = 50)
```

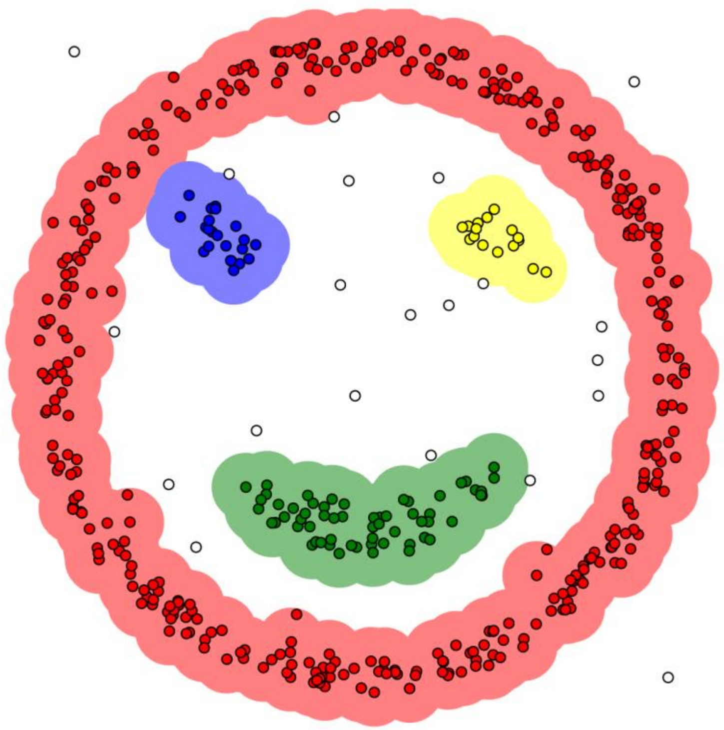
VISUALISATION



CLUSTERING & NETTOYAGE

CLUSTERING : DBSCAN

https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/



CLUSTERING : DBSCAN

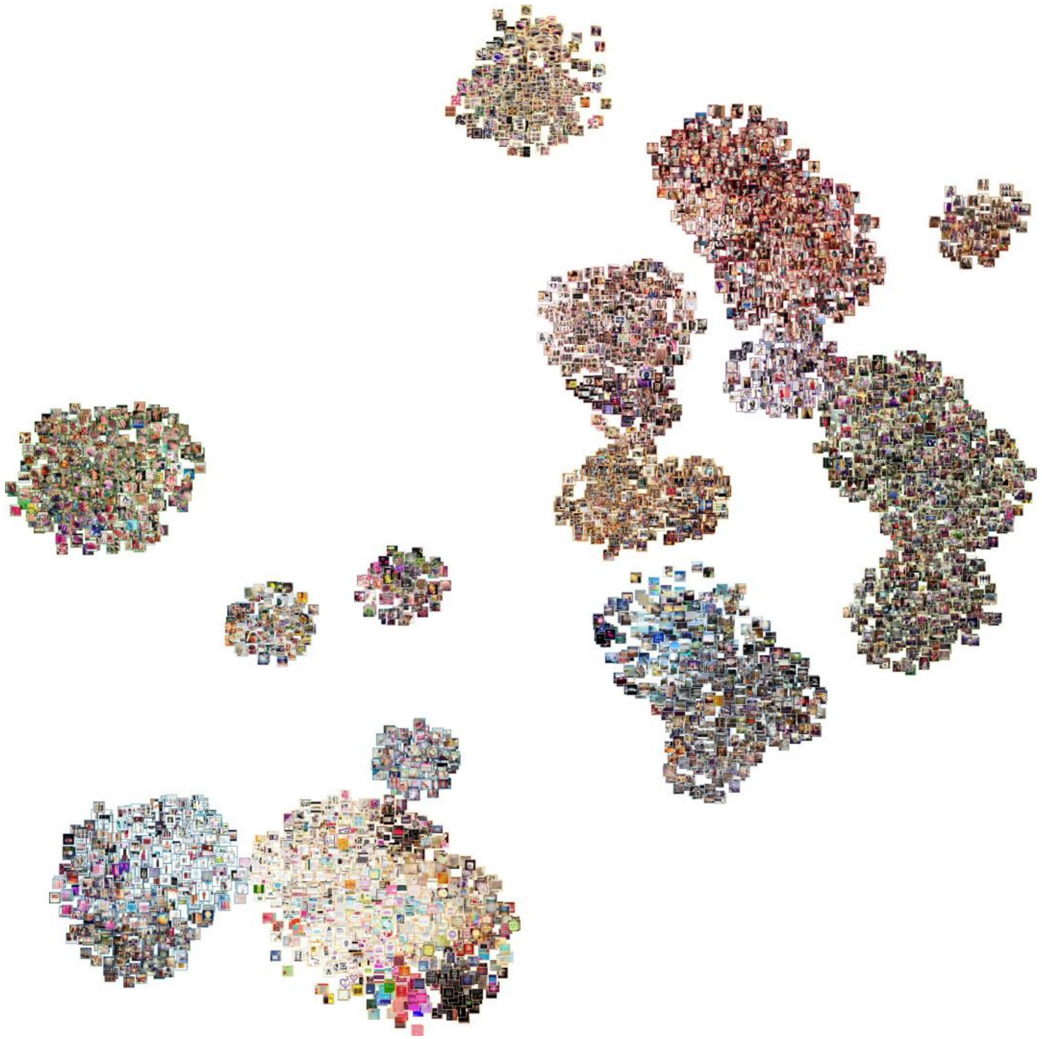
```python
from sklearn.cluster import DBSCAN
```

```python
def visualize_DB(df,
                 save_img_path = "./ptSNE_img_DB.png",
                 img_path = './data/img/small_sample/file/',
                 width = 5000,
                 height = 5000,
                 max_dim = 25):

    import matplotlib.pyplot as plt

    df = df[df['cluster']!=-1] # remove unclustered img
    tx = df['tx']
    ty = df['ty']
    img = df['img']
    cluster = df['cluster']

    min_x = np.min(tx)
    max_x = np.max(tx)
    min_y = np.min(ty)
    max_y = np.max(ty)
    tx = (tx-min_x) / (max_x - min_x) # normalize
    ty = (ty-min_y) / (max_y - min_y)

    color = ["#a6cee3",
             "#1f78b4",
             "#b2df8a",
             "#33a02c",
             "#fb9a99",
             "#e31a1c",
             "#fdbf6f",
             "#ff7f00",
             "#cab2d6",
             "#6a3d9a",
             "#ffff99",
             "#b15928"]
    color.extend(color)

    full_image = Image.new('RGB', (width, height),color=(255,255,255))
    i = 0
    for img_, x, y,cluster in zip(img[:], tx[:], ty[:],cluster[:]):
        if i%int(len(tx[:])/10.)==0:
            print('img {}/{}'.format(i,len(tx[:])))
        i+=1
        tile = Image.open(img_path + img_)
        rs = max(1, tile.width/max_dim, tile.height/max_dim)
        tile = tile.resize((int(tile.width/rs), int(tile.height/rs)), Image.ANTIALIAS)
        clr = color[cluster]
        tile = ImageOps.expand(tile, border=2, fill=clr)
        full_image.paste(tile, (int((width-max_dim)*1*x), int((height-max_dim)*1*y)), mask=tile.convert('RGBA'))

    plt.figure(figsize = (16,12))
    full_image.save(save_img_path)
```

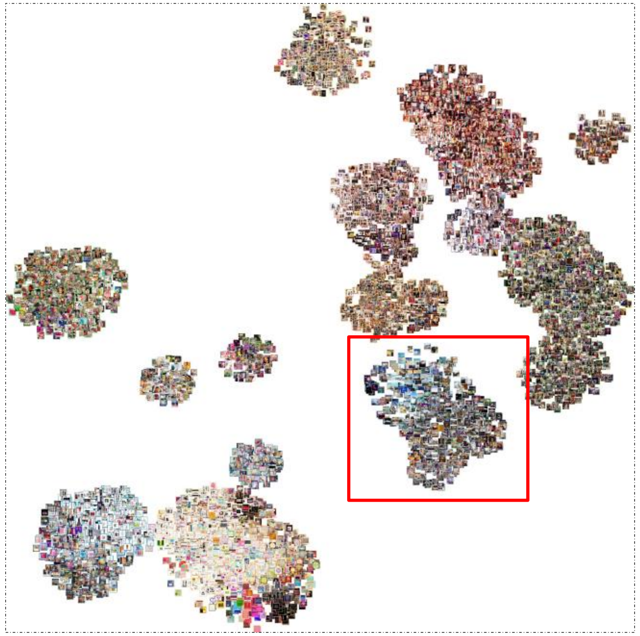
```python
eps = 3
min_sample = 90
clustering = DBSCAN(eps=eps, min_samples=min_sample).fit(df[['tx','ty']])
cnt = Counter(clustering.labels_)
print("eps = {}, n_clusters = {}, unclassified = {} ".format(eps,len(cnt),cnt[-1]))
df['cluster'] = clustering.labels_

visualize_DB(df,
             save_img_path = "./tests_DB/ptSNE_img_DB_eps{}_min_sample{}_v3.1.jpg".format(eps,min_sample),
             img_path = img_path,
             width = 5000,
             height = 5000,
             max_dim = 50)
```

```
eps = 3, n_clusters = 15, unclassified = 4381
```

VISUALISATION

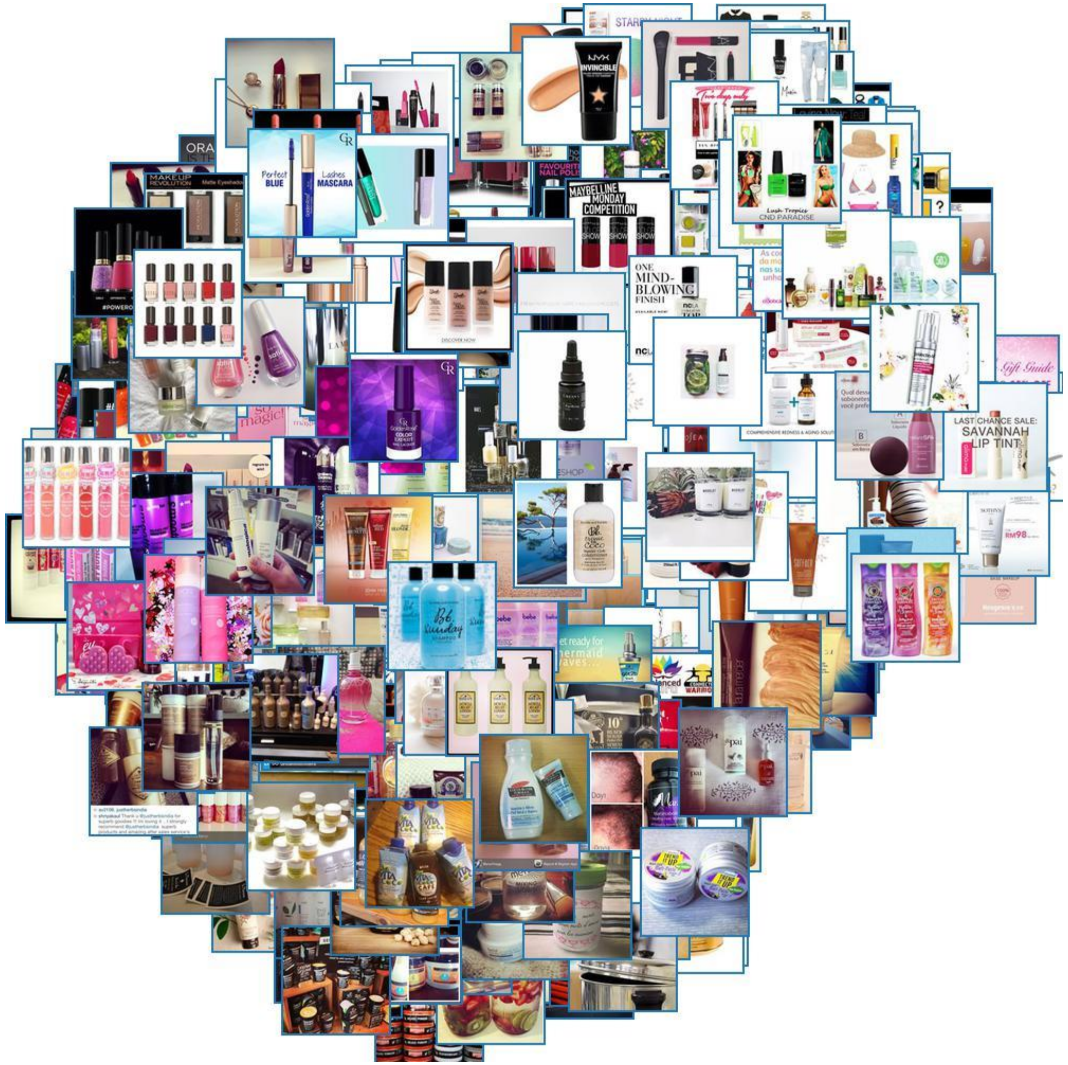


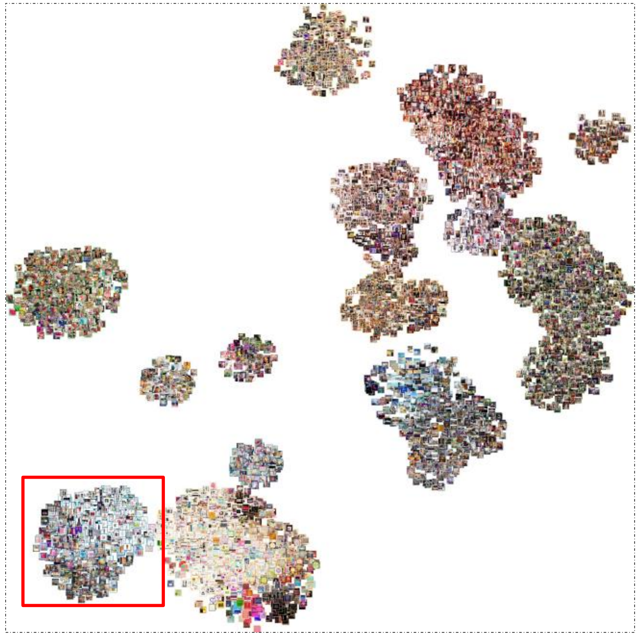
VISUALISATION



Cluster 0

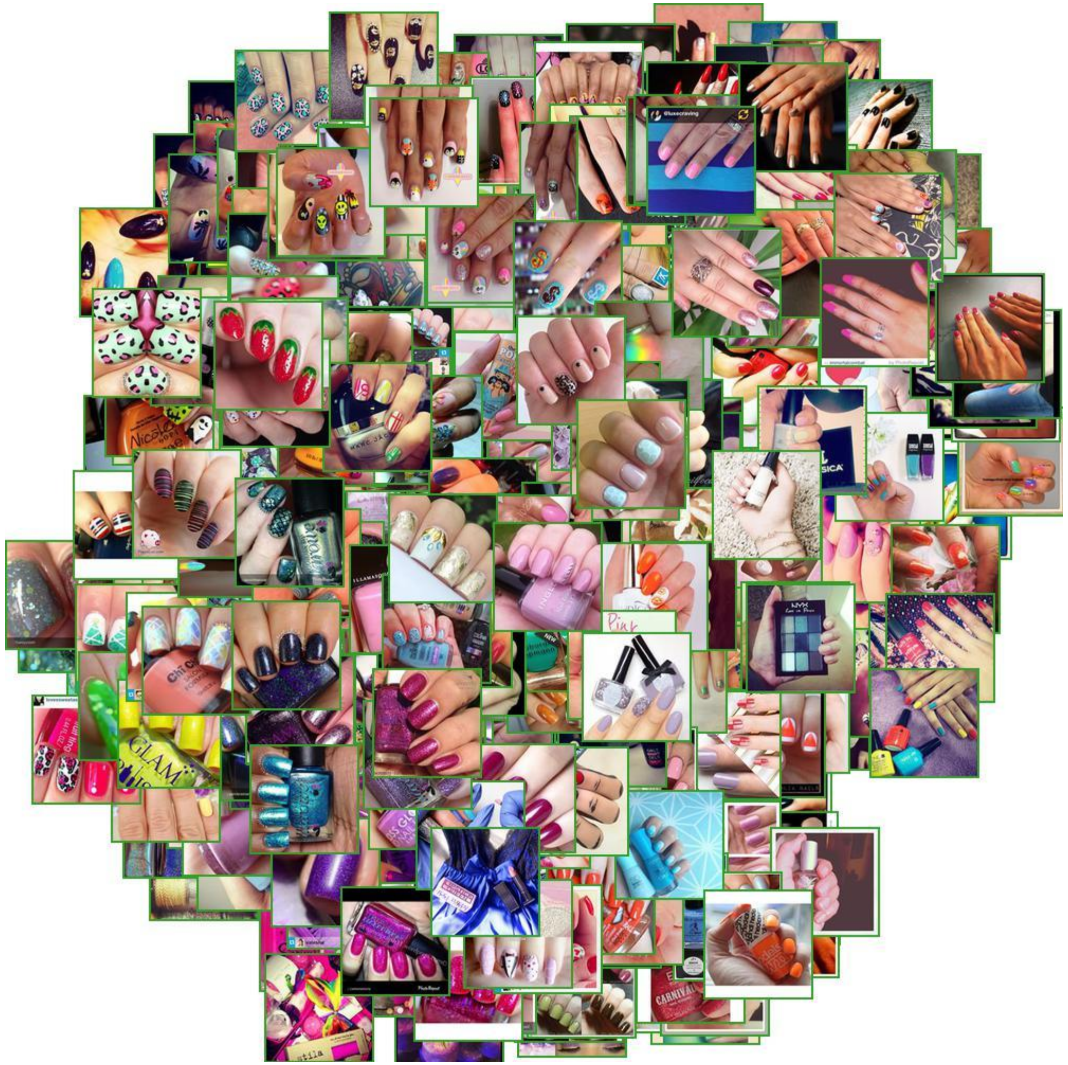
VISUALISATION

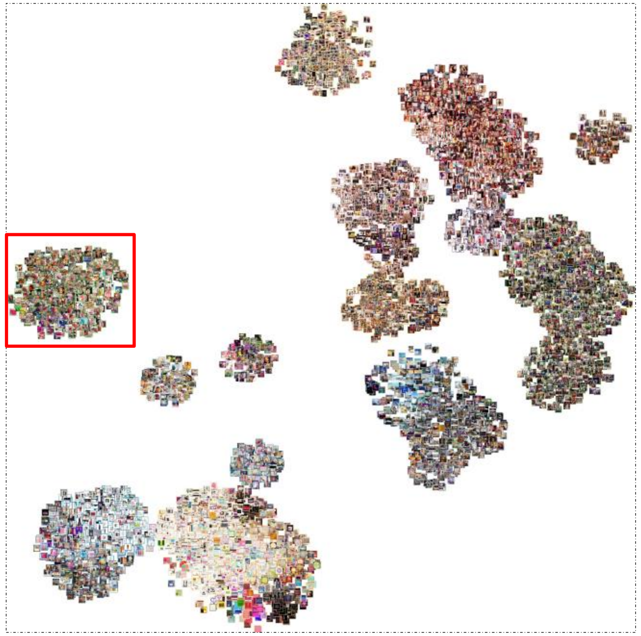




Cluster 1

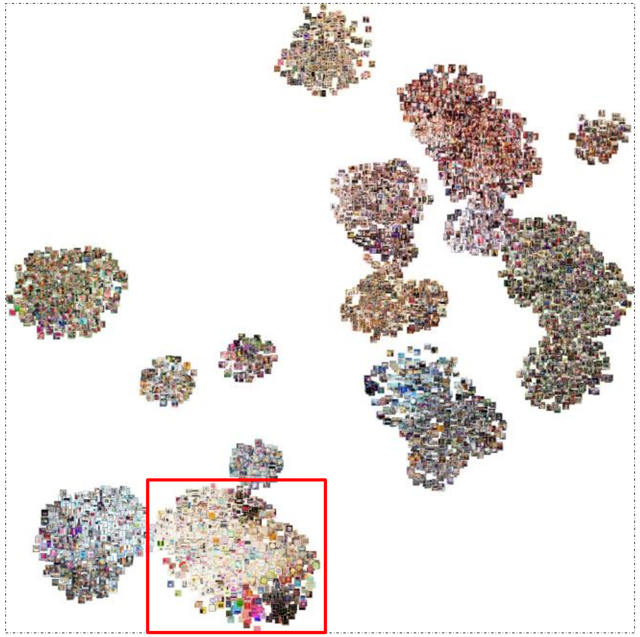
VISUALISATION





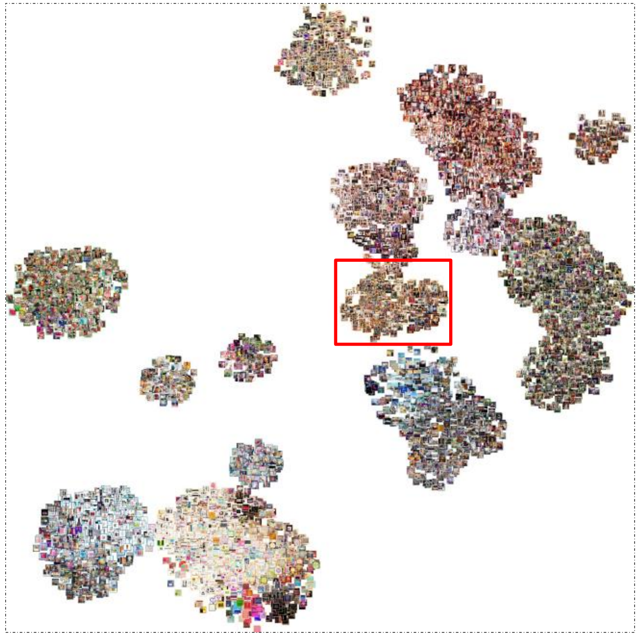
Cluster 3

VISUALISATION



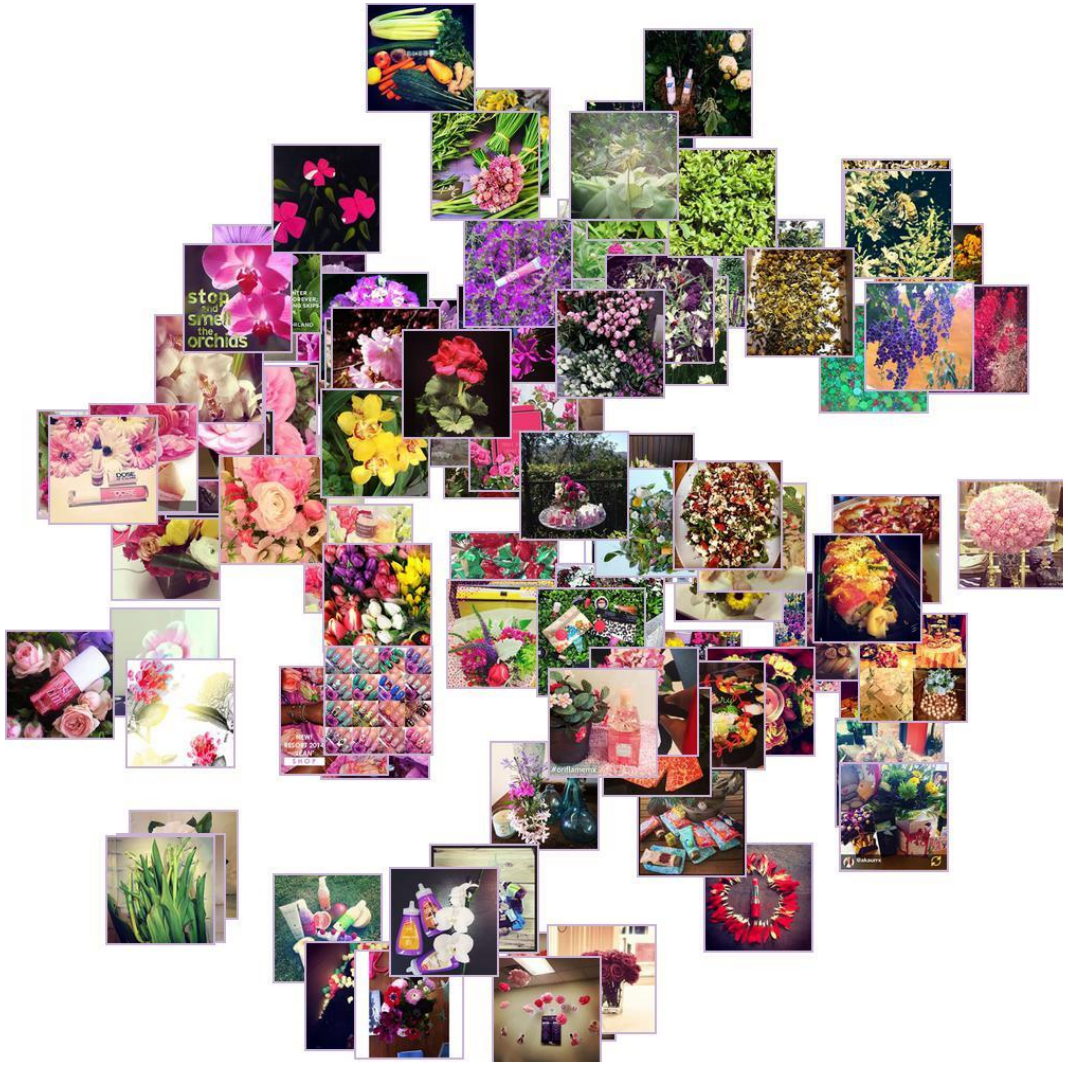
Cluster 6

VISUALISATION



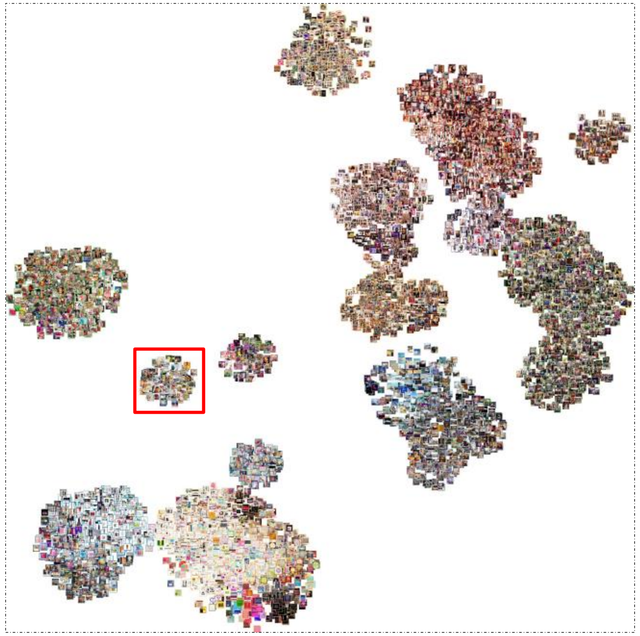
Cluster 7

VISUALISATION



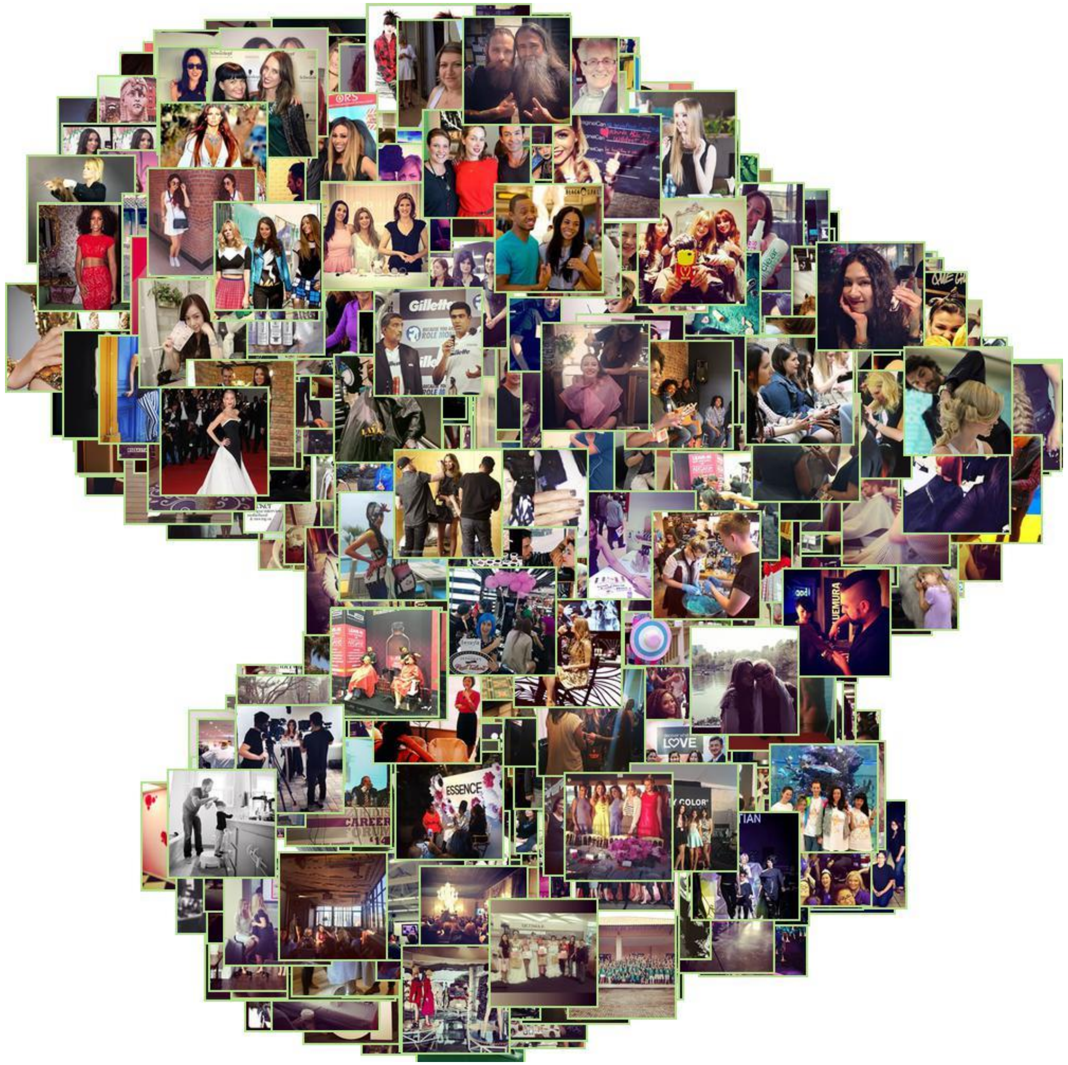
Cluster 8

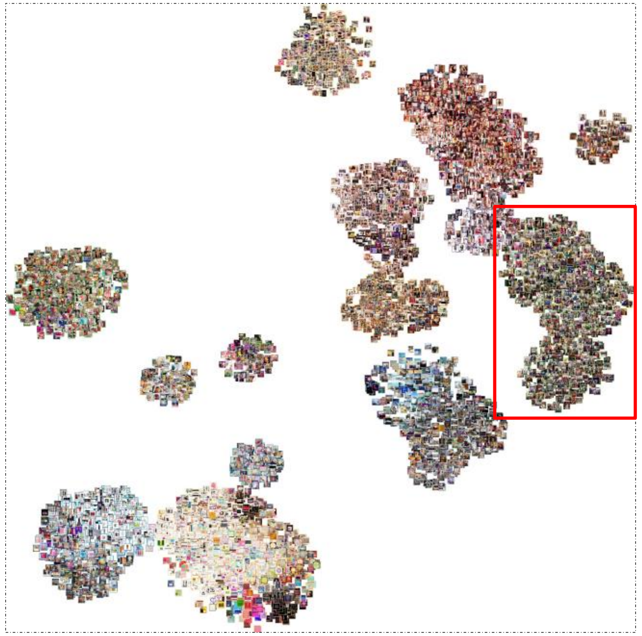
VISUALISATION



Cluster 12

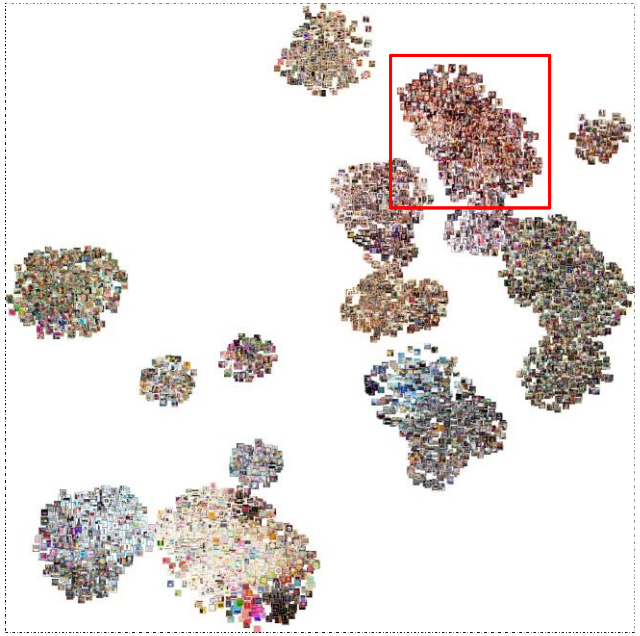
VISUALISATION





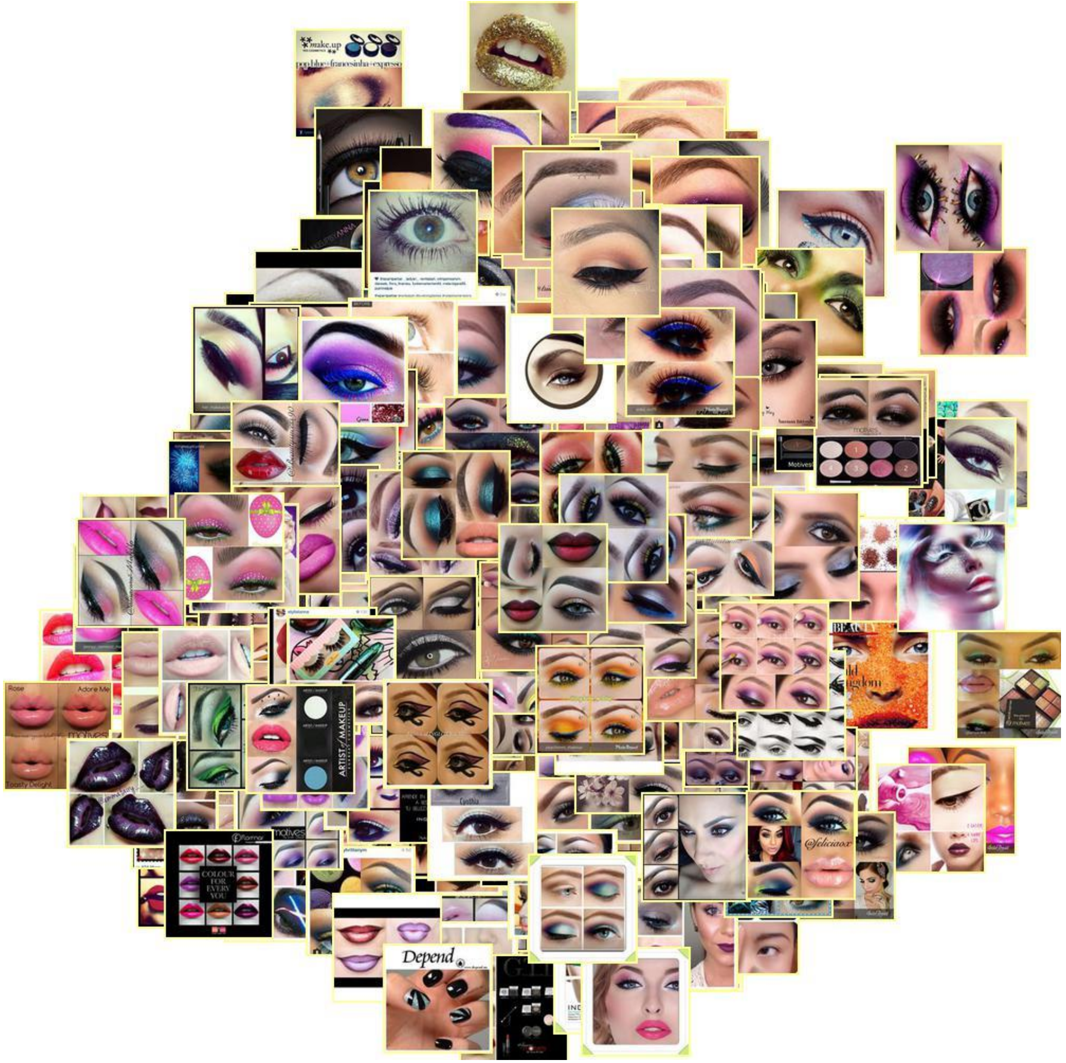
Cluster 2

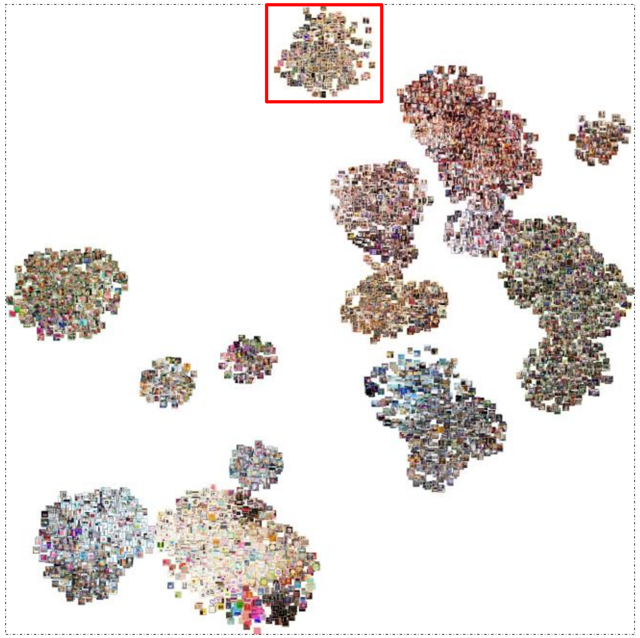
VISUALISATION



Cluster 5

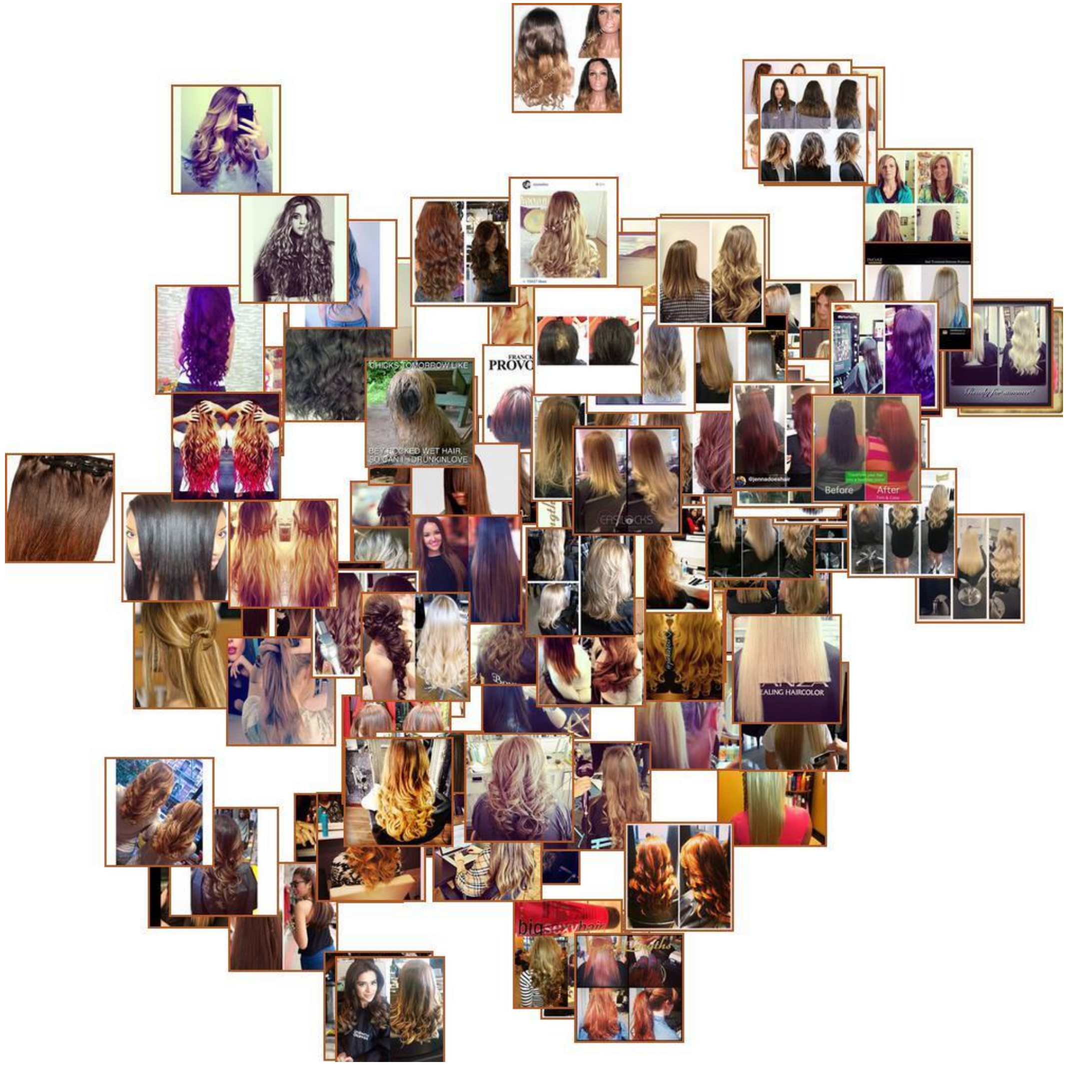
VISUALISATION





Cluster 10

VISUALISATION



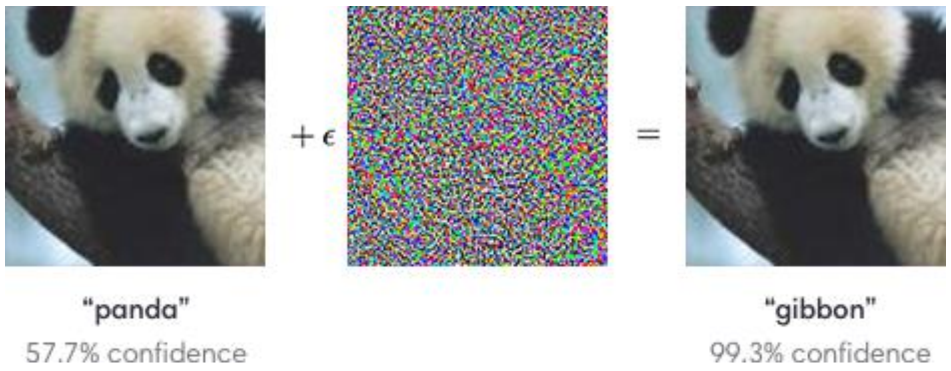
Cluster 11

LIMITES

Images similaires au dataset d'entrainement ImageNet

Pas de contrôle de la définition de similarité : il peut y avoir des « erreurs » par rapport à notre perception d'images similaires

Sensible au bruit :



https://arxiv.org/abs/1412.6572

MERCI !

Corentin Martel

Data Scientist Freelance

https://www.corelyo.com

RESSOURCES

https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

http://huboqiang.cn/2017/02/20/DeepLearningSkinCancer

https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/

https://arxiv.org/pdf/1512.03385.pdf

O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma,Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenetlarge scale visual recognition challenge.arXiv:1409.0575, 2014.

https://nnabla.org/paper/imagenet_in_224sec.pdf

https://towardsdatascience.com/intuitively-understanding-convolutions-for-deep-learning-1f6f42faee1

http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture05.pdf

https://towardsdatascience.com/how-to-visualize-convolutional-features-in-40-lines-of-code-70b7d87b0030

https://distill.pub/2016/misread-tsne/

http://alexhwilliams.info/itsneuronalblog/2016/03/27/pca/

https://www.qwertee.io/blog/deep-learning-with-point-clouds/