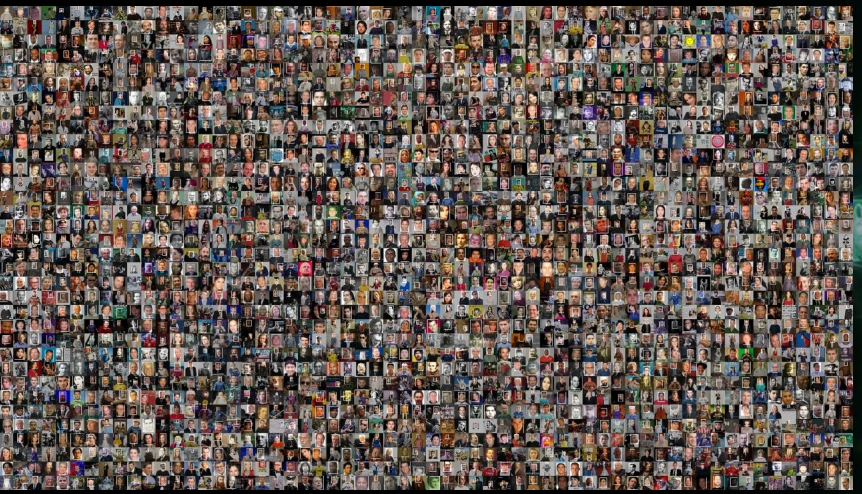
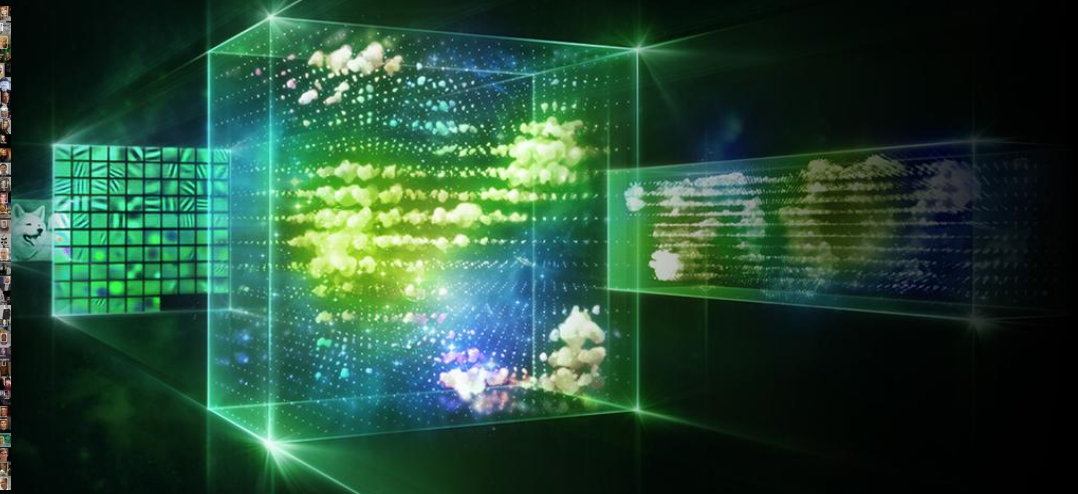
2,000,000 Faces a second with python





Alain Richardt
https://atomictessellator.com

Dataset

Kaggle Face Recognition Dataset

https://storage.googleapis.com/kaggle-data-sets/546691/997012/bundle/archive.zip

52,000 Faces

512x512 px

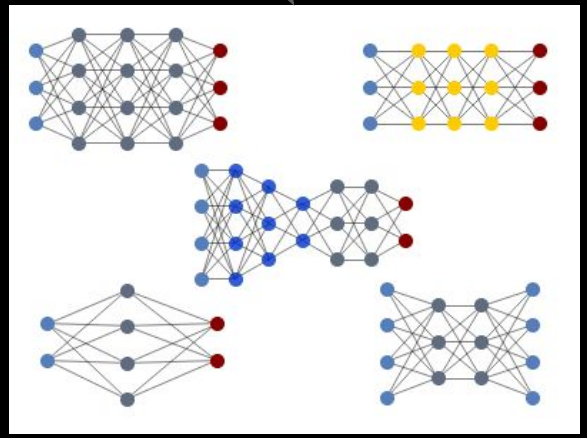
20.9 GB

Faces <–> Encodings

01_generate_encodings.py

We need a data structure that best represents the uniqueness of faces.

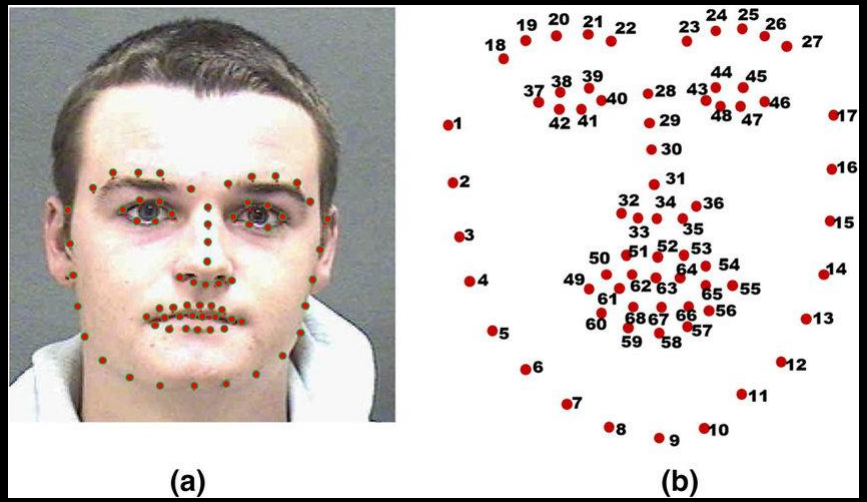
In ML talk that means our feature extraction objective function should maximise the distance between individuals faces



```
[array([-0.11115586,  0.01333801,  0.05365208, -0.09734353, -0.10214046,
       -0.02681433,  0.04466205, -0.05318493,  0.12746717, -0.1147783 ,
        0.1604809 , -0.00199237, -0.30696589,  0.0185116 , -0.05634476,
        0.10986105, -0.05970628, -0.12320047, -0.17041762, -0.1805921 ,
       -0.07820766,  0.03572616, -0.0544406 , -0.03576037, -0.05495634,
       -0.20094359, -0.10276579, -0.03365591,  0.05245824, -0.04228309,
        0.02082079,  0.04047918, -0.19184038, -0.06968617, -0.03838123,
        0.04994577, -0.08131331, -0.0469272 ,  0.24006367,  0.05962883,
       -0.10063579, -0.02934341,  0.09430658,  0.2501975 ,  0.13262746,
        0.02394336,  0.0293446 , -0.06876627,  0.09918878, -0.32048333,
        0.11797751,  0.07502637,  0.11766519,  0.11920492,  0.06608473,
       -0.21826521,  0.0538491 ,  0.09149722, -0.13539639,  0.12855764,
        0.07509172, -0.13878338, -0.10551377, -0.00354823,  0.22146326,
        0.14403334, -0.01378807, -0.15616426,  0.29372475, -0.18254563,
       -0.03574862,  0.08875197, -0.01294353, -0.04464227, -0.27830625,
        0.09086196,  0.35151899,  0.12191065, -0.10077263, -0.00121126,
       -0.1060065 , -0.01243002,  0.10261983, -0.04952341, -0.14896646,
       -0.05863239, -0.12135565,  0.01250695,  0.25890017, -0.02276999,
       -0.03972171,  0.164186  ,  0.10973116,  0.01010293,  0.10477081,
        0.07534386, -0.07627255, -0.0780039 , -0.13776836,  0.03658303,
        0.12615952, -0.19396366,  0.00921777,  0.08279327, -0.16152218,
        0.14457917, -0.02525565, -0.06919114, -0.03003786, -0.0077064 ,
       -0.08291472, -0.0062177 ,  0.19500105, -0.19226897,  0.19748309,
        0.14408699, -0.03837791,  0.1241489 ,  0.09106639,  0.11225412,
       -0.04982456, -0.05624177, -0.12845463, -0.12502889,  0.00210423,
        0.01562229,  0.01961705,  0.14906931])]
```

Faces <–> Encodings

01_generate_encodings.py




```
[array([-0.11115586,  0.01333801,  0.05365208, -0.09734353, -0.10214046,
        -0.02681433,  0.04466205, -0.05318493,  0.12746717, -0.1147783 ,
         0.1604809 , -0.00199237, -0.30696589,  0.0185116 , -0.05634476,
         0.10986105, -0.05970628, -0.12320047, -0.17041762, -0.1805921 ,
        -0.07820766,  0.03572616, -0.0544406 , -0.03576037, -0.05495634,
        -0.20094359, -0.10276579, -0.03365591,  0.05245824, -0.04228309,
         0.02082079,  0.04047918, -0.19184038, -0.06968617, -0.03838123,
         0.04994577, -0.08131331, -0.0469272 ,  0.24006367,  0.05962883,
        -0.10063579, -0.02934341,  0.09430658,  0.2501975 ,  0.13262746,
         0.02394336,  0.0293446 , -0.06876627,  0.09918878, -0.32048333,
         0.11797751,  0.07502637,  0.11766519,  0.11920492,  0.06608473,
        -0.21826521,  0.0538491 ,  0.09149722, -0.13539639,  0.12855764,
         0.07509172, -0.13878338, -0.10551377, -0.00354823,  0.22146326,
         0.14403334, -0.01378807, -0.15616426,  0.29372475, -0.18254563,
        -0.03574862,  0.08875197, -0.01294353, -0.04464227, -0.27830625,
         0.09086196,  0.35151899,  0.12191065, -0.10077263, -0.00121126,
        -0.1060065 , -0.01243002,  0.10261983, -0.04952341, -0.14896646,
        -0.05863239, -0.12135565,  0.01250695,  0.25890017, -0.02276999,
        -0.03972171,  0.164186  ,  0.10973116,  0.01010293,  0.10477081,
         0.07534386, -0.07627255, -0.0780039 , -0.13776836,  0.03658303,
         0.12615952, -0.19396366,  0.00921777,  0.08279327, -0.16152218,
         0.14457917, -0.02525565, -0.06919114, -0.03003786, -0.0077064 ,
        -0.08291472, -0.0062177 ,  0.19500105, -0.19226897,  0.19748309,
         0.14408699, -0.03837791,  0.1241489 ,  0.09106639,  0.11225412,
        -0.04982456, -0.05624177, -0.12845463, -0.12502889,  0.00210423,
         0.01562229,  0.01961705,  0.14906931])]
```

Q: Why does Deep Learning perform better than normal computer vision techniques?

A: Automatic feature optimization

The most simple case - Load and Query

02_load_and_query.py

Load numpy dataset from disk

Build a KDTree so we can query vector similarity quickly

2.2 million faces a second, but that assumes linear query time with dataset growth, this is actually untrue with a KDTree, so the actual rate is higher!

Load encodings : 0.044001 s
Build KDTree : 0.110525 s
Prepare query : 0.06317 s
Query : 0.023375 s Rate: 2224598.930481 /s

Distance : [0.26000397]
Index : [50002]
Image : kaggle_faces/knuth2.png

Query

Closest Match

Scaling: The rate is much higher, 10 m/s easily with KDTree

Preprocessing - holding that structure hot in RAM and query via RPC