Superpixel preprocessing as data reduction in image recognition

Mitan Theodor-Alexandru
Coordonator: Lect. Dr. Benchea Razvan

Outline

We use clustering to **reduce computational effort** and training time required for image recognition with a simpler neural network architecture that **trains very rapidly**.

As such, we use a combination of two fundamental machine learning techniques.

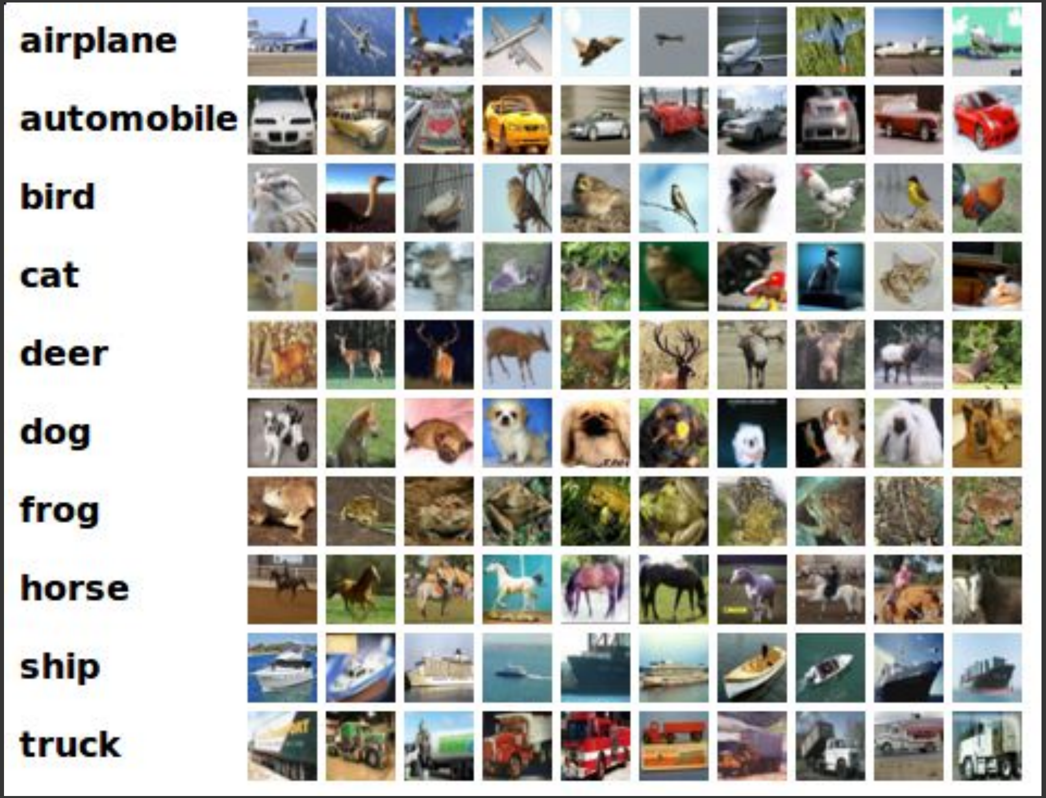
Problems addressed

1. CNNs require GPU use or high-powered machines
2. Long training times vs. rapid innovation / testing
3. Complex architectures
4. Per-image testing time can be too high
5. Fixed input size

The Process

- **Acquire images**
  CIFAR-10 and ImageSoup scraping.
- **K-means on pixels**
  Cluster / segment the pixel
- **Centroids to neural network**
  Use the 5-D centroids as input for a neural network that solves the classification problem

CIFAR-10: 60.000 small images



Scraping: suns and moons

```
images[0] = ImageSoup().search('sun', n_images=200, image_size='medium')
images[1] = ImageSoup().search('moon', n_images=200, image_size='medium')
```

Images to XYRGB

```
[[ 0.          0.          0.49019608  0.58823529  0.74901961]
 [ 0.03125     0.          0.4627451   0.56078431  0.71764706]
 [ 0.0625      0.          0.45490196  0.55294118  0.70980392]
 ...,
 [ 0.90625     0.96875     0.32156863  0.31764706  0.36862745]
 [ 0.9375      0.96875     0.2745098   0.27058824  0.31372549]
 [ 0.96875     0.96875     0.29803922  0.29019608  0.30196078]]
```

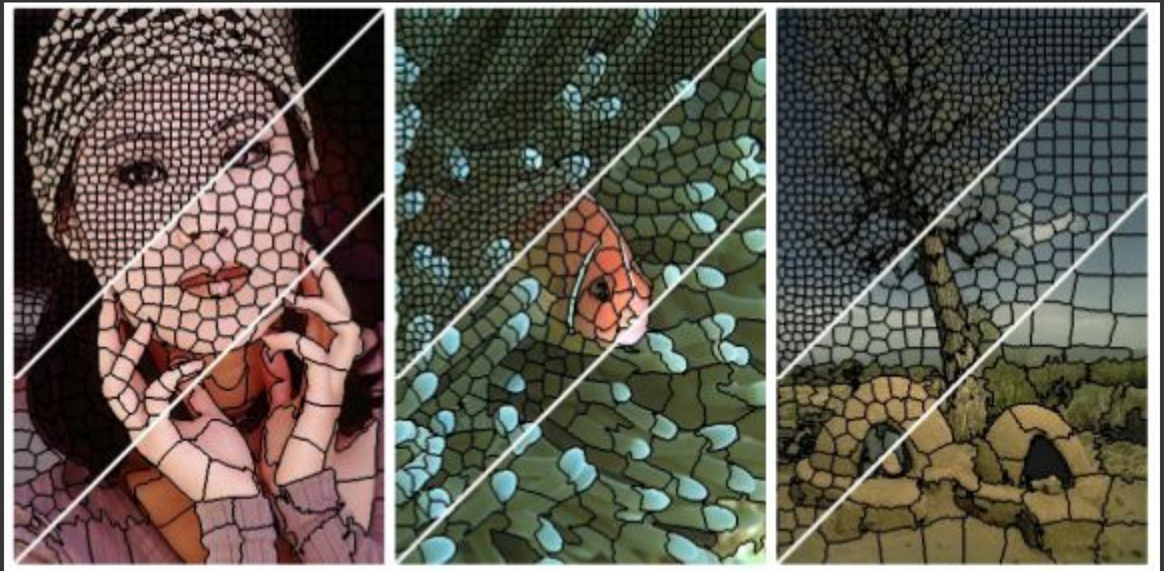
Flattened meshgrid

Each 5-D unit has its coordinates normalised by the shortest side

Normalised RGB

We normalise the RGB components to [0, 1].

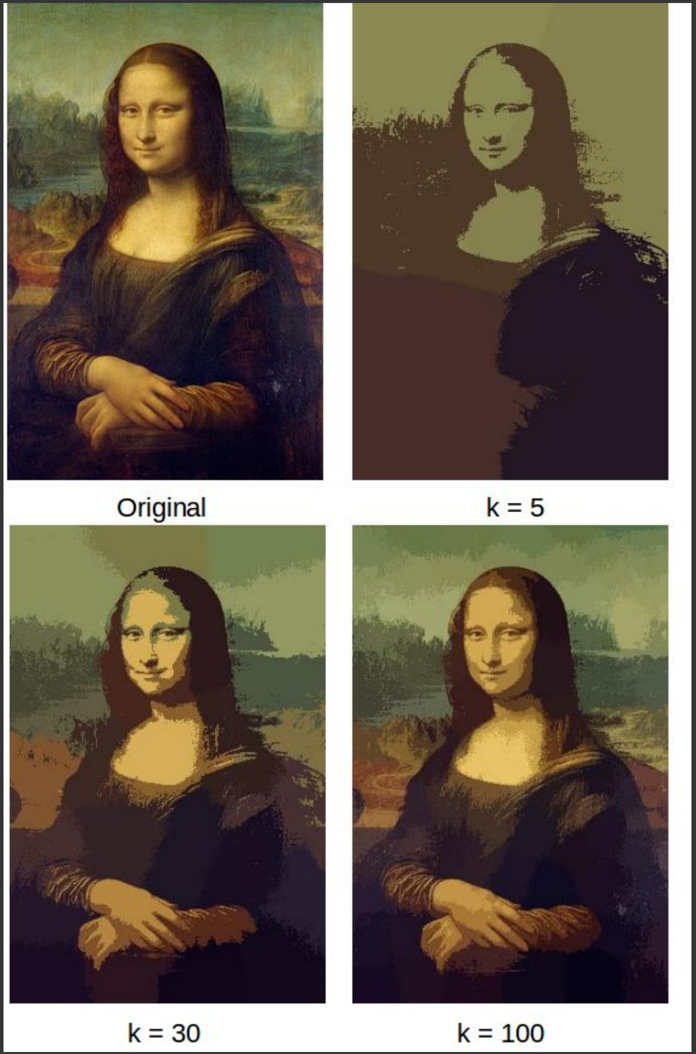
Superpixels (SLIC)



Inspiration: SLIC

SLIC superpixels use a combination of spatial and chromatic distance in the La*b* space

Superpixels: k-means



XYRGB clustering

My model: X, Y, R, G, B are considered similarly, but the first two are "tweaked".

Low dmul

Colour reduction

(space doesn't matter)

dmul = 0

dmul = 0.5

dmul = 1

dmul = 2

High dmul

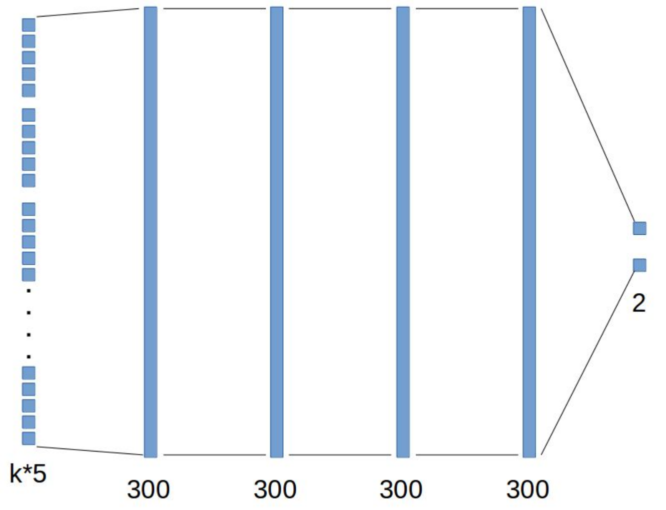
Detail lost, space matters more than colour.

Superpixels to input

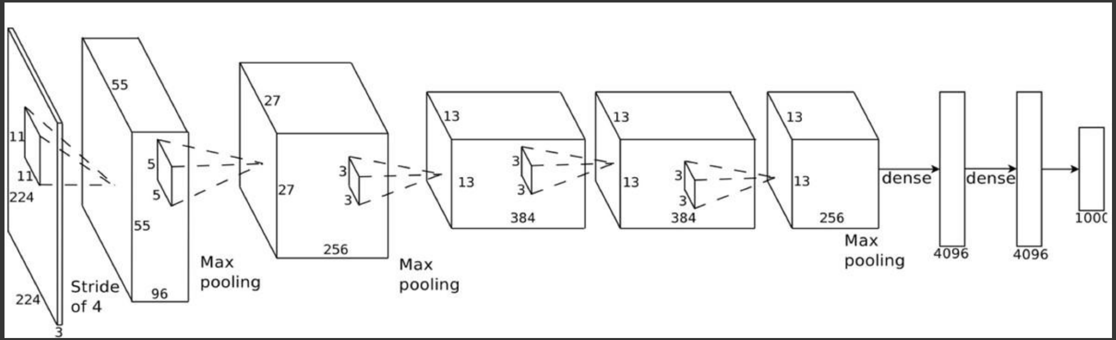
1. “Cellbatch” uses the averages of each superpixel
2. “Relbatch” uses batches of relationships with Euclidean metrics

```
[R1, G1, B1, R2, G2, B2, XY_dist, RGB_dist]
```

The neural network



Cellbatch



CNN

Rules

1. 30 clusters
2. 20 epochs
3. All images
4. We track accuracy and time
5. Baseline: flat, fully-connected network

Other parameters

Each algorithm has its own parameters, and they have been tweaked to roughly similar degrees of complexity

Accuracy for “CIFAR-2”

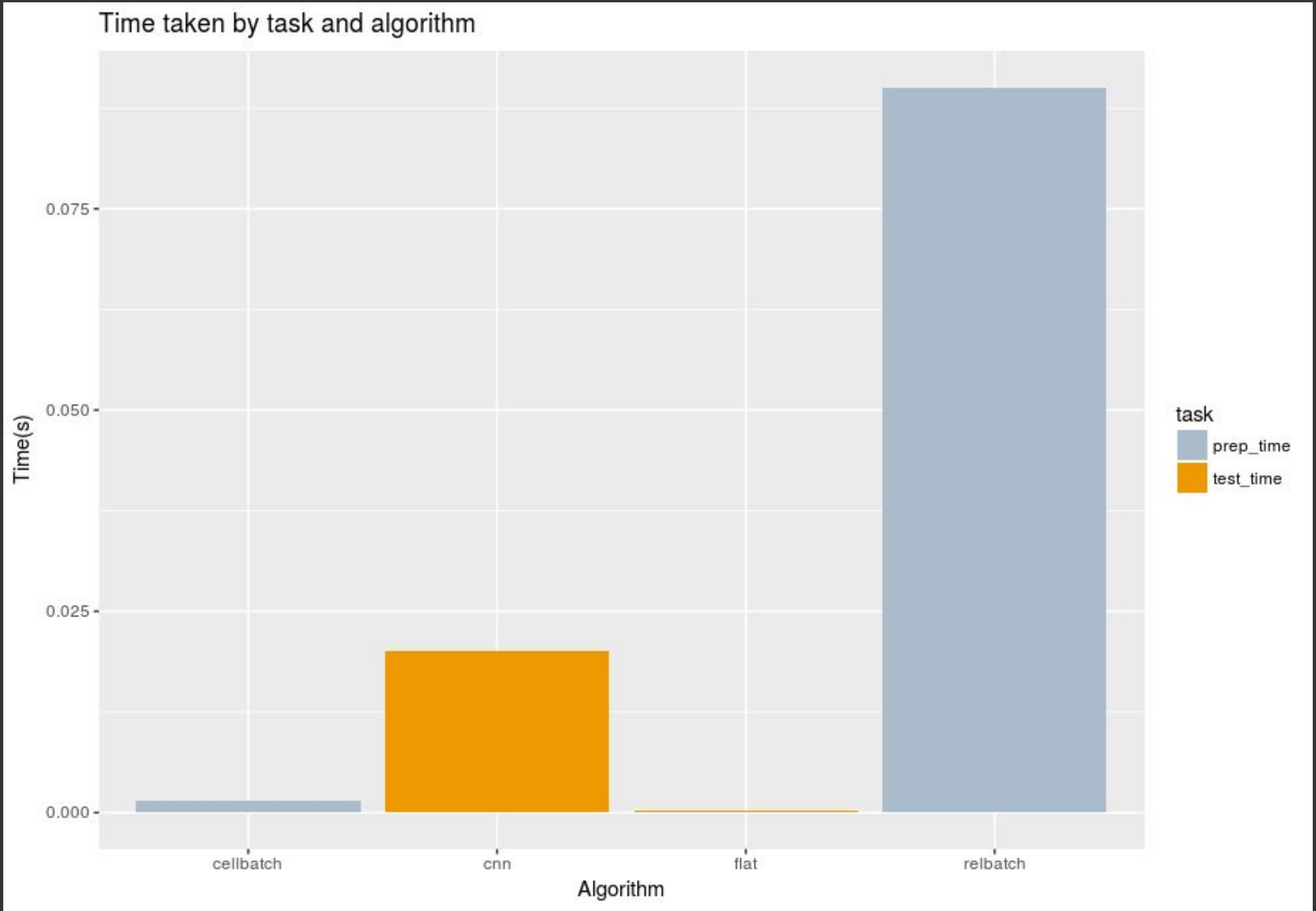
1. Cellbatch: ~85%
2. Relbatch: ~86%
3. Flat: ~50%

Data size

Cellbatch has reduced the input data to under a tenth of its original size.

Relbatch has increased it slightly, depending on number of relationships.

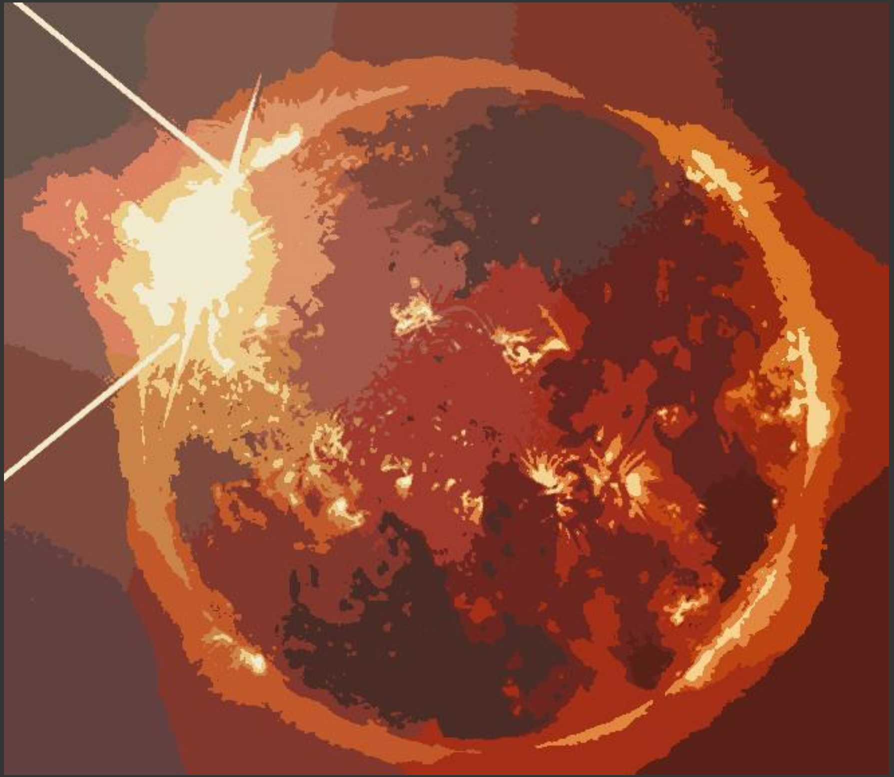
Timing



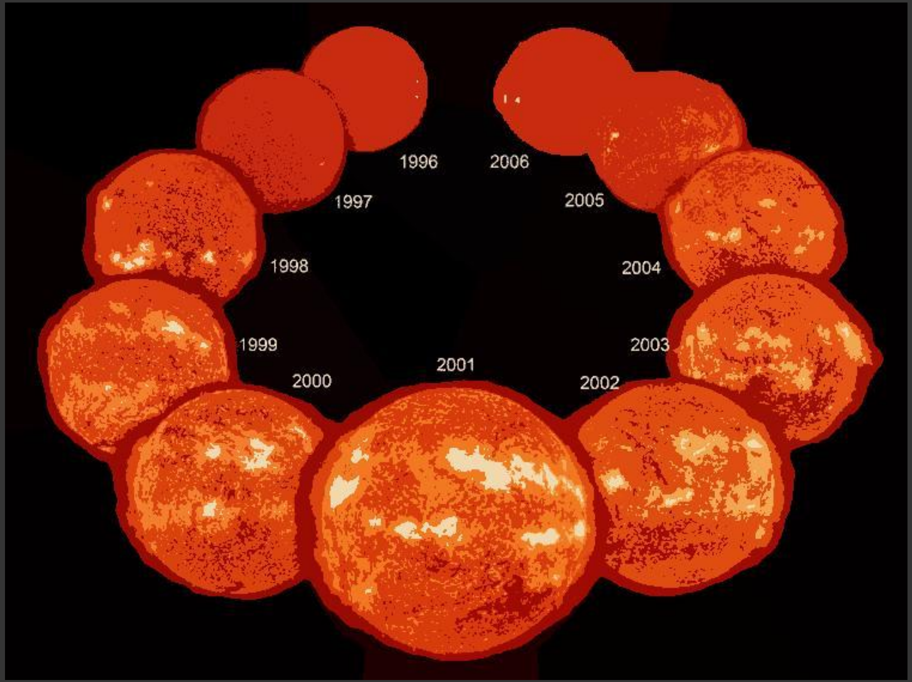
Time taken by task and algorithm
Time(s)
0.075
0.050
0.025
0.000
cellbatch
cnn
flat
relbatch
Algorithm
task
prep_time
test_time

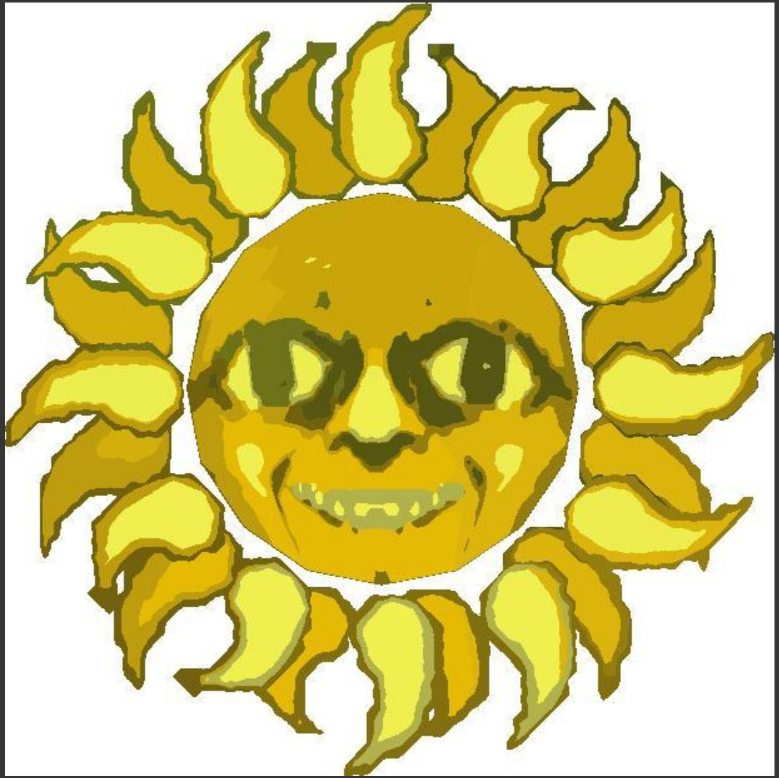
Gif, demos (changing dmul)

Correctly classified suns

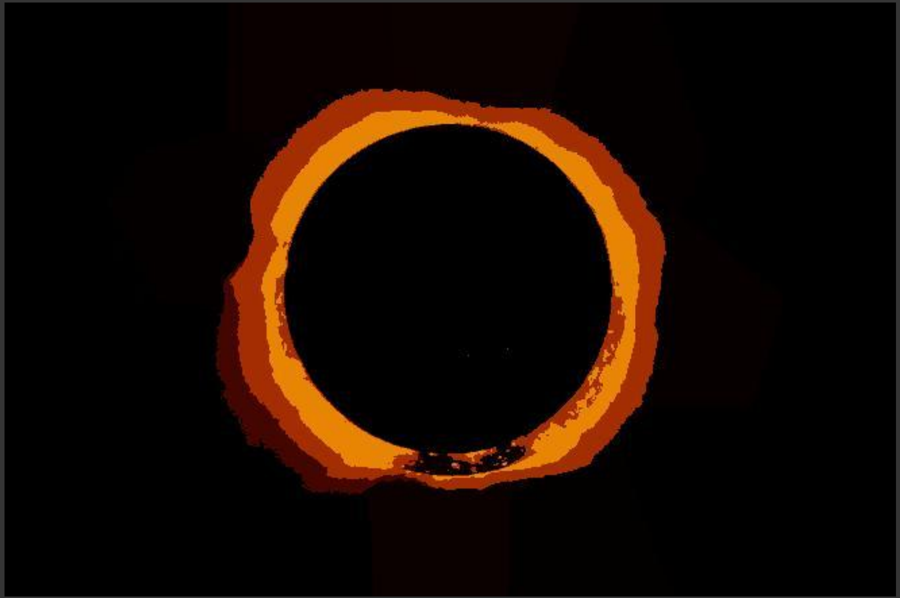


Correctly classified suns

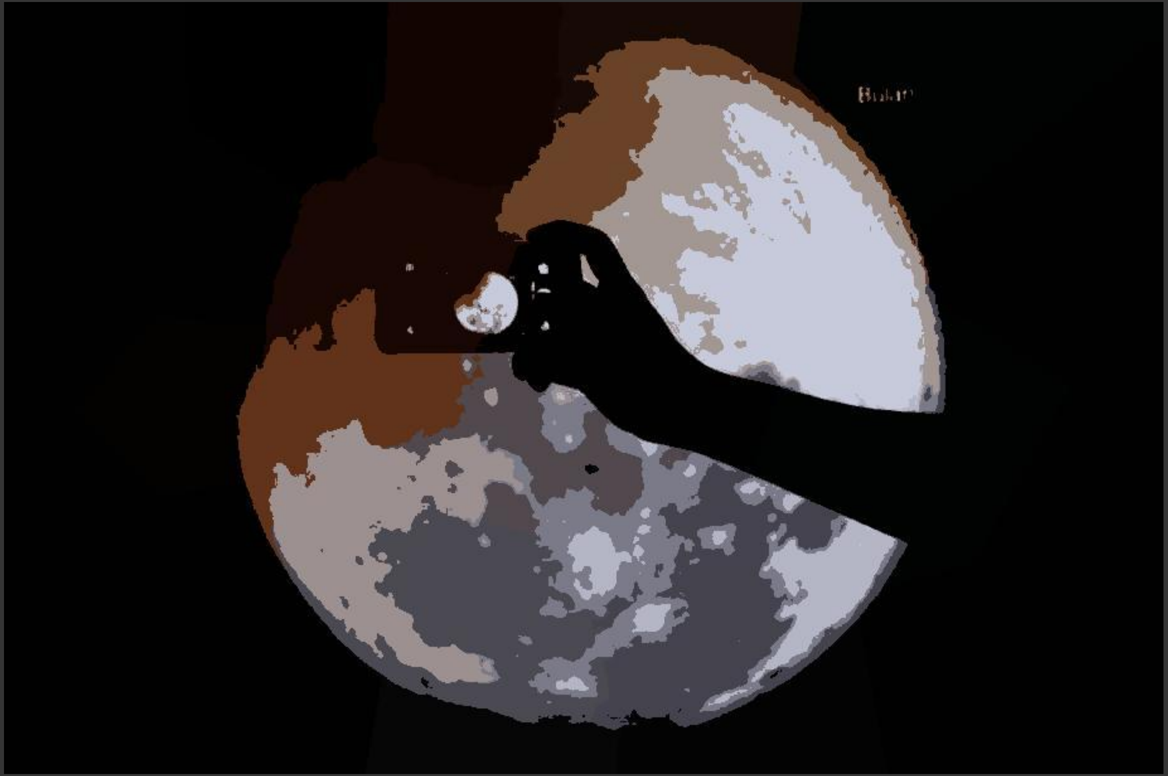




Wrongly classified suns



Correctly classified moons



Correctly classified moons

Wrongly classified moons

Further work

- Better use of the superpixels’ immediate neighbourhood
- Cluster analysis on superpixels
- SLIC-like optimisations (limited search space)

Conclusions

- Reasonable accuracy with very fast training times
- Simple architecture
- Flexible input size and type
- Substantial data reduction
- Effort moved to preprocessing, testing is almost instant: can prepare data regardless of network architecture

Questions