Finding things in images

How does that work?

Davis King

August 27, 2018

Who are you?

- Been writing open source software for 16 years

- I do applied machine learning research in industry

The high level view

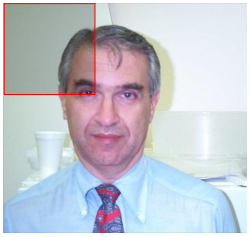
- A lot of variety in object detection applications. Some examples:
  - Finding boxy things in RGB images: e.g. faces, cars, etc.
  - Finding the pose of an object: e.g. face landmarking
  - Finding curves: road and pipeline detection in airborne images or side looking sonar
  - Finding "interest points" for image registration
- Different applications need different approaches.
  - Often machine learning is used: e.g. face detection
  - Often machine learning isn't used. Or is only part of the solution.
- What are the most common mistakes?

Finding Faces

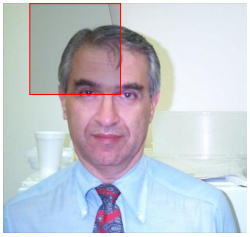
Detecting things: Starting simple

- Everyone loves a sliding window

Sliding Window Detection - 1

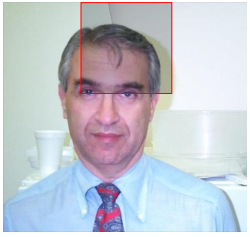


Sliding Window Detection - 2



Sliding Window Detection - 3

Sliding Window Detection - 4

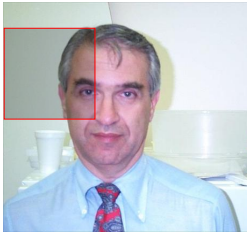


Sliding Window Detection - 5

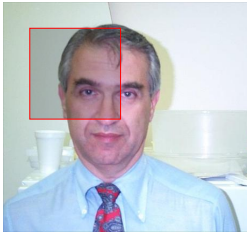
Sliding Window Detection - 6

Sliding Window Detection - 7

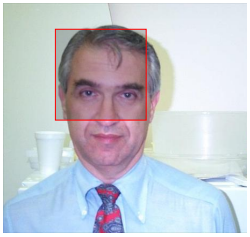
Sliding Window Detection - 8



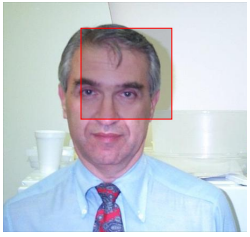
Sliding Window Detection - 9



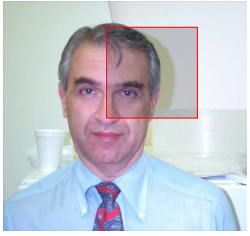
Sliding Window Detection - 10



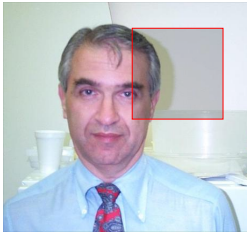
Sliding Window Detection - 11



Sliding Window Detection - 12

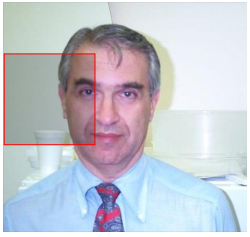


Sliding Window Detection - 13

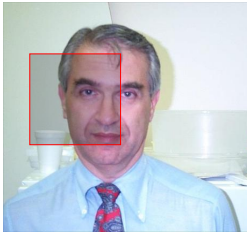


Sliding Window Detection - 14

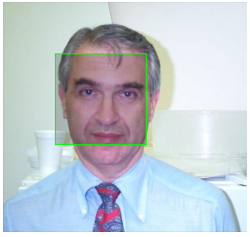
Sliding Window Detection - 15



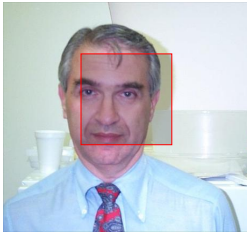
Sliding Window Detection - 16



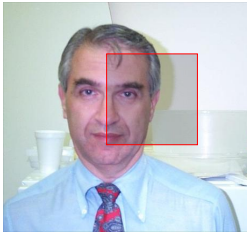
Sliding Window Detection - 17



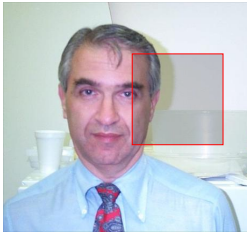
Sliding Window Detection - 18



Sliding Window Detection - 19

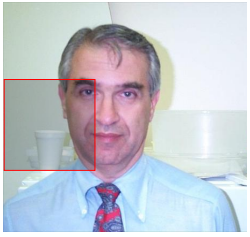


Sliding Window Detection - 20

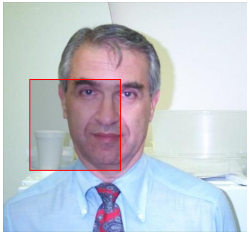


Sliding Window Detection - 21

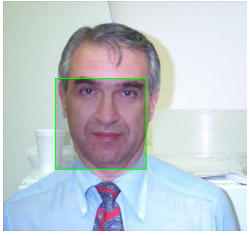
Sliding Window Detection - 22



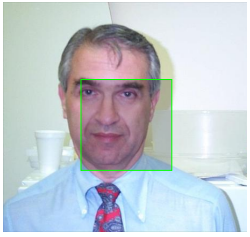
Sliding Window Detection - 23



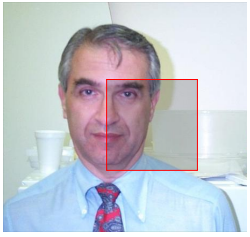
Sliding Window Detection - 24



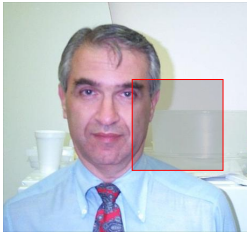
Sliding Window Detection - 25



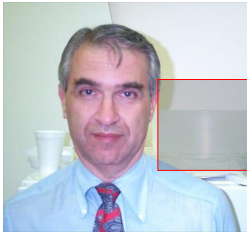
Sliding Window Detection - 26



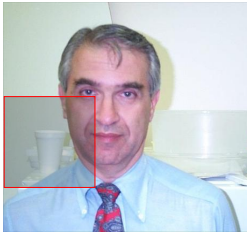
Sliding Window Detection - 27



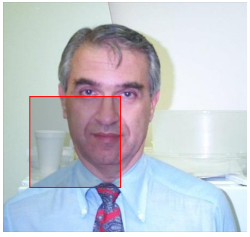
Sliding Window Detection - 28



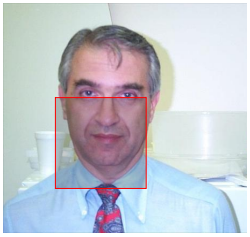
Sliding Window Detection - 29



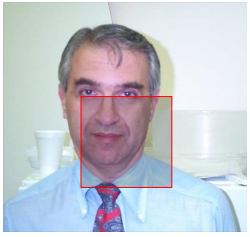
Sliding Window Detection - 30



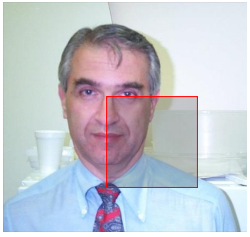
Sliding Window Detection - 31



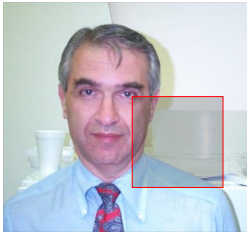
Sliding Window Detection - 32



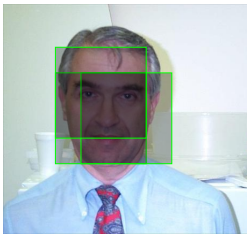
Sliding Window Detection - 33



Sliding Window Detection - 34

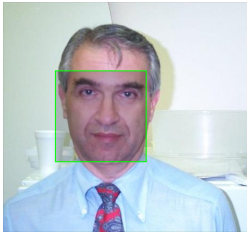


Non-max Suppression



- Non-max suppression: Sort all the detections by their score and greedily scan that list, outputting any detection that doesn't overlap an already output detection.
- Sometimes use some fancy data structure to optimize this, but in most cases sorting and a loop is fine.

Non-max Suppression



- Non-max suppression: Sort all the detections by their score and greedily scan that list, outputting any detection that doesn't overlap an already output detection.
- Sometimes use some fancy data structure to optimize this, but in most cases sorting and a loop is fine.

What if you don't know the object's size?

- What I just described only works if you know the size of the object.

Image Pyramid

Image Pyramid

Image Pyramid

- What are the limitations of this approach?

The venerable HOG detector

- The most textbook detector, the HOG detector.

```
import dlib
img = dlib.load_rgb_image("obama.jpg")
detector = dlib.get_frontal_face_detector()
faces = detector(img)
win = dlib.image_window(img)
win.add_overlay(faces)
```

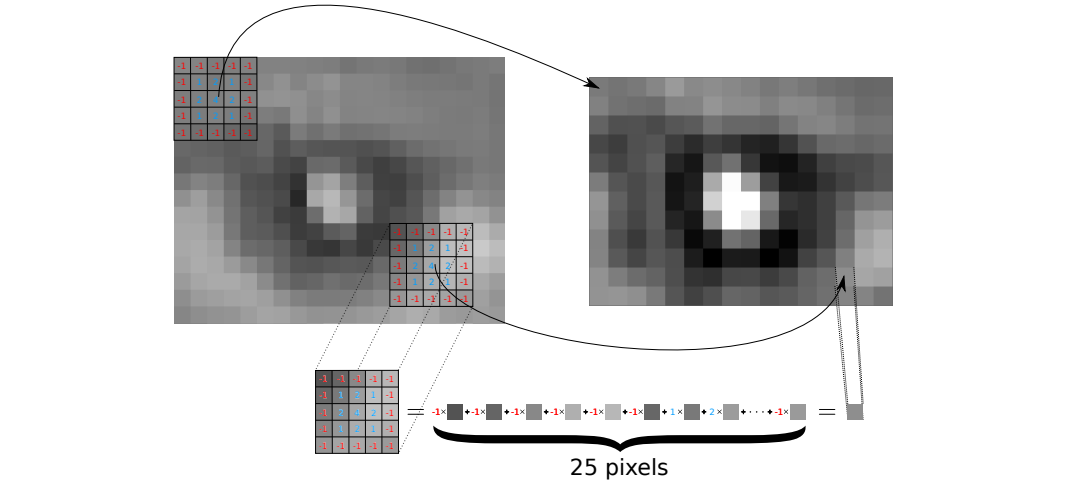
HOG detectors are super easy to train

```
import dlib
options=dlib.simple_object_detector_training_options()
options.C = 5
dlib.train_simple_object_detector("training.xml",
                                  "detector.svm",
                                  options)

detector = dlib.simple_object_detector("detector.svm")
```

- Training often takes just seconds, no local minima, no learning rate. Easy
- How does this work? What kinds of applications is it good for?

What is a spatial filter/convolution/cross-correlation?

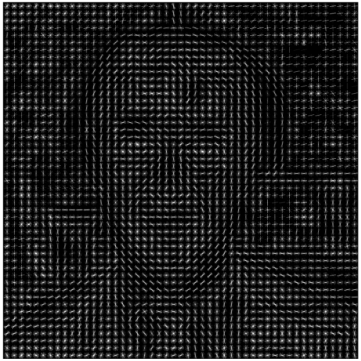


Used to detect or accentuate something in an image. In this case to detect the bright spot.

What is a HOG detector?

600x600x3

HOG

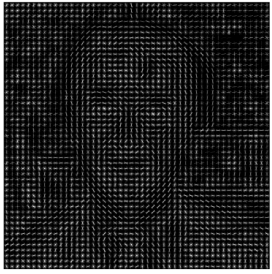


75x75x31

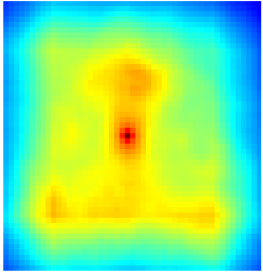
What is a HOG detector?

600x600x3

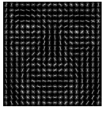
HOG



75x75x31



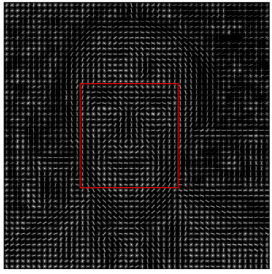
Convolve with above HOG image



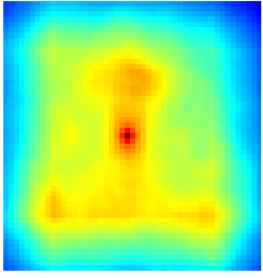
What is a HOG detector?

600x600x3

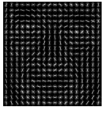
HOG



75x75x31

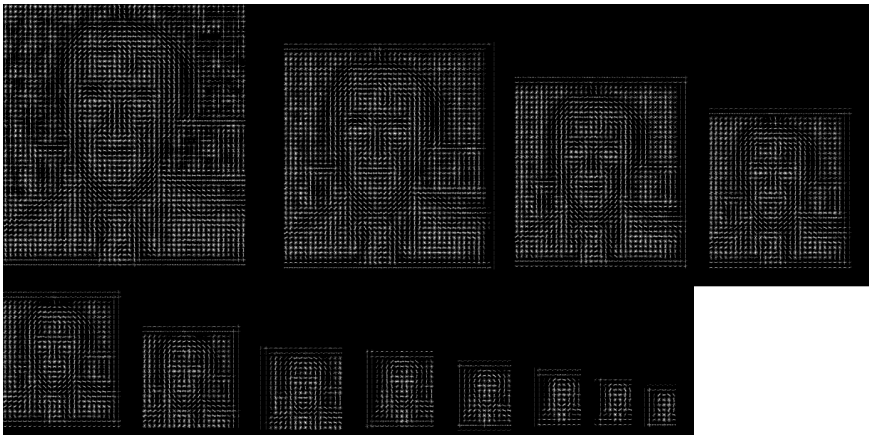


Convolve with above HOG image

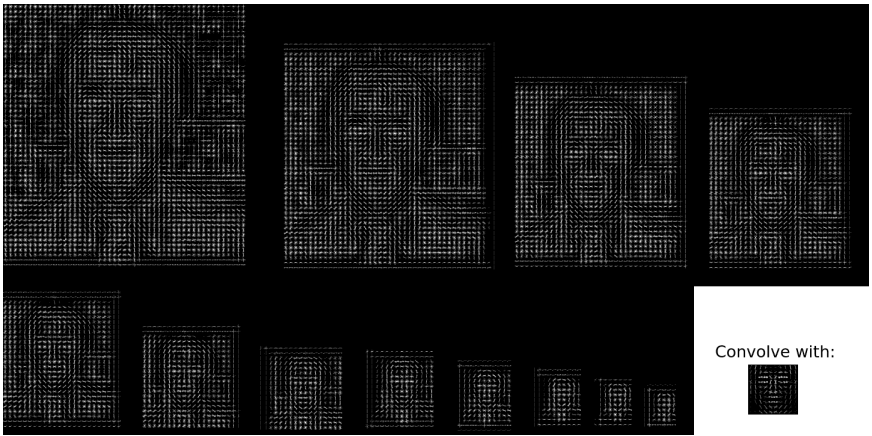


What is a HOG detector?

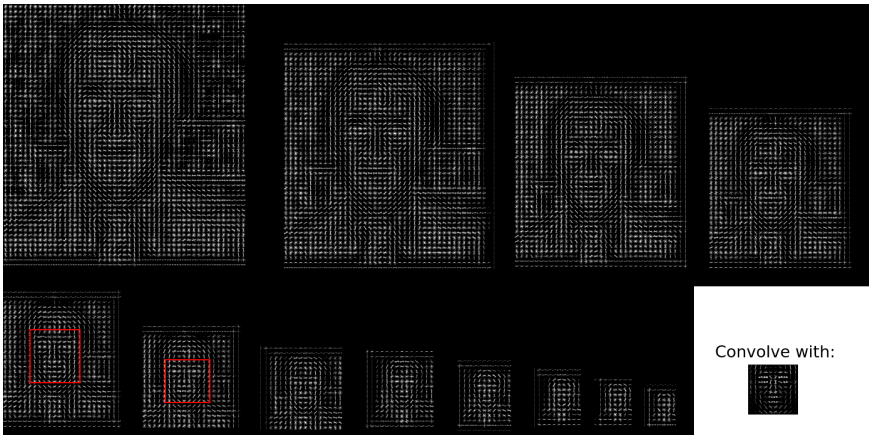
What is a HOG detector?



What is a HOG detector?

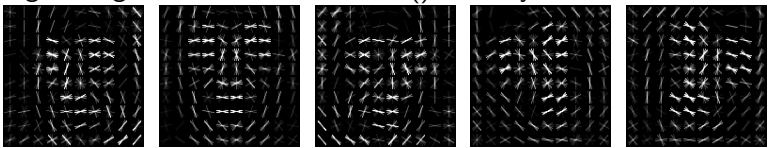


What is a HOG detector?



What is a multi-HOG detector?

- e.g. dlib.get_frontal_face_detector() is really 5 HOG detectors:



- Using 5 filters allows the detector to find a wider range of head poses
- Making a multi-HOG detector just means training multiple separate HOG detectors. Nothing fancy.

HOG Pros and Cons

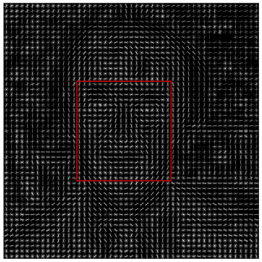
- Pros:
  - Super fast to train, only seconds to minutes.
  - No learning rate parameter and trainer can't get stuck in local minima.
  - Fast to run. Easily real-time on a normal CPU.
- Cons:
  - Is really just a rigid template. Won't work if the object rotates or deforms a lot. Using multiple HOG detectors only somewhat mitigates this.
  - Totally ignores color.

Putting HOG into perspective

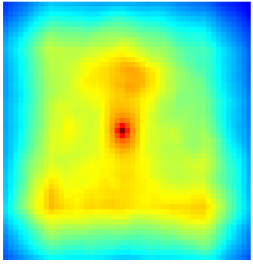
Recall how HOG detectors work:

600x600x3

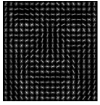
HOG



75x75x31

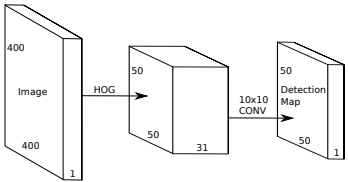


Convolve with above HOG image

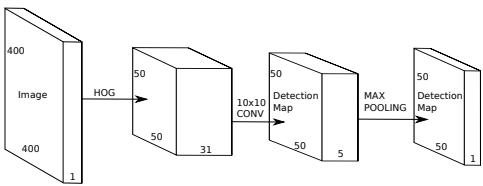


Putting HOG into perspective

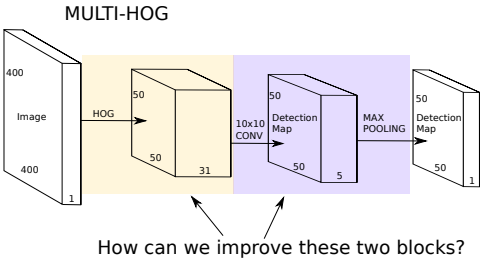
HOG DETECTOR



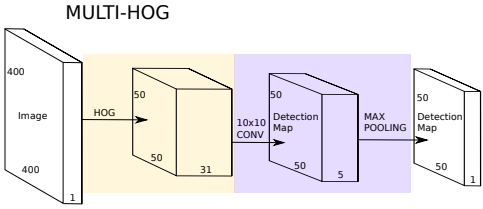
MULTI-HOG DETECTOR

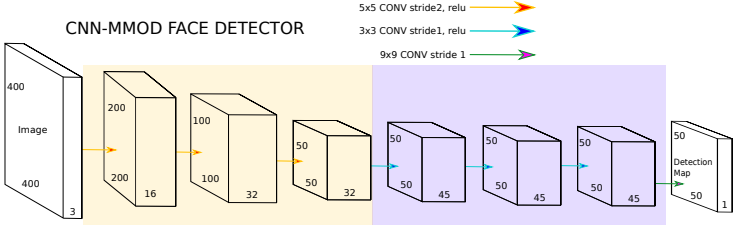


Doing better than HOG

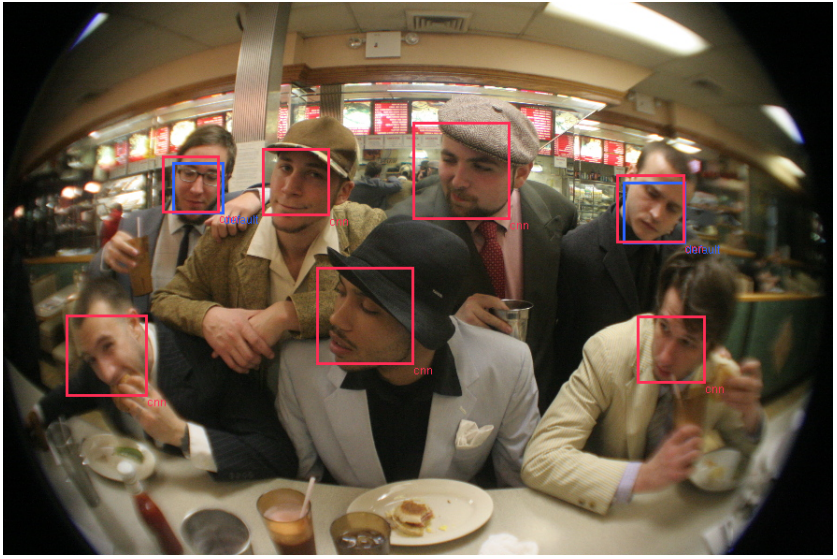


Doing better than HOG

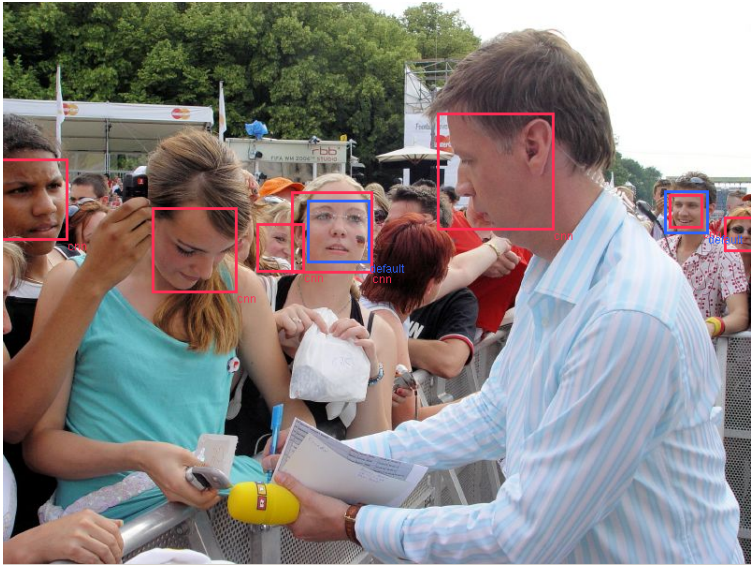




HOG vs CNN

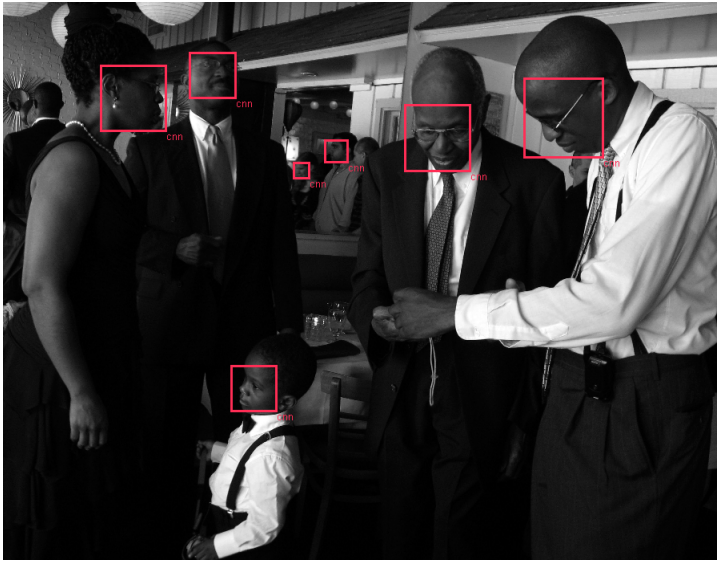


HOG vs CNN

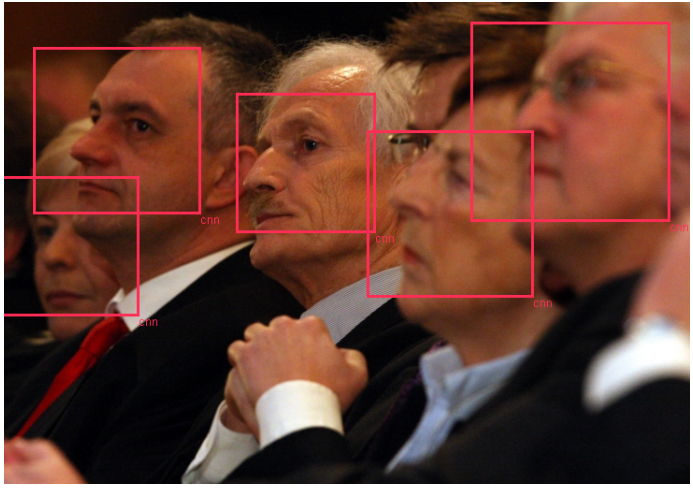


HOG vs CNN

HOG vs CNN



HOG vs CNN

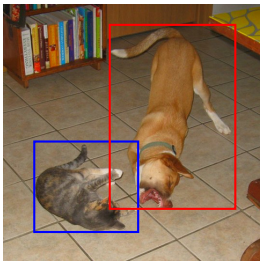


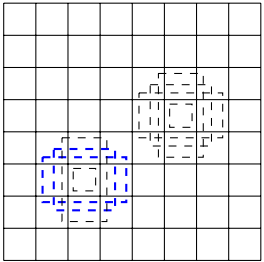
Using image pyramids with CNNs

- We could run the CNN many times, once on each image in a normal image pyramid. Doesn't parallelize well on GPU.
- Better: run CNN once on a tiled pyramid image:

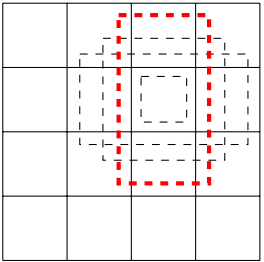
Alternatives to pyramids

SSD's output:





8 × 8 detection maps



4 × 4 detection maps

- CNN examples in dlib use small stride (so large output maps)
- SSD uses large stride (so small output maps)
- Why? What are pros and cons?
  - "Feature Pyramid Networks for Object Detection" by Lin, Et al. has a good discussion

Small stride example, lots of small objects

Small stride CNN from dlib example (19 cars):

Why isn't my detector working?

- You just have a bug in your code

Why isn't my detector working?

- You just have a bug in your code
- Your training data is bad

Why isn't my detector working?

- You just have a bug in your code
- Your training data is bad
- The algorithm just can't do what you are asking it to do.
  - Maybe the objects aren't included in the locations the detector looks. E.g. maybe you are using a sliding window+pyramid that doesn't look at objects as small as your targets.

Why isn't my detector working?

- You just have a bug in your code
- Your training data is bad
- The algorithm just can't do what you are asking it to do.
  - Maybe the objects aren't included in the locations the detector looks. E.g. maybe you are using a sliding window+pyramid that doesn't look at objects as small as your targets.
- What do you do?

Why isn't my detector working?

- You just have a bug in your code
- Your training data is bad
- The algorithm just can't do what you are asking it to do.
  - Maybe the objects aren't included in the locations the detector looks. E.g. maybe you are using a sliding window+pyramid that doesn't look at objects as small as your targets.
- What do you do?
  - Get it working on a small dataset. *Make sure the detector works on the images it was trained on.*

Why isn't my detector working?

- You just have a bug in your code
- Your training data is bad
- The algorithm just can't do what you are asking it to do.
  - Maybe the objects aren't included in the locations the detector looks. E.g. maybe you are using a sliding window+pyramid that doesn't look at objects as small as your targets.
- What do you do?
  - Get it working on a small dataset. *Make sure the detector works on the images it was trained on.*
  - If it works on training data but not on testing data **get more training data!**

Why isn't my detector working?

- You just have a bug in your code
- Your training data is bad
- The algorithm just can't do what you are asking it to do.
  - Maybe the objects aren't included in the locations the detector looks. E.g. maybe you are using a sliding window+pyramid that doesn't look at objects as small as your targets.
- What do you do?
  - Get it working on a small dataset. *Make sure the detector works on the images it was trained on.*
  - If it works on training data but not on testing data **get more training data!**
- It is working but your metric is bad. Always look at the output with your eyes and see what's really going on.

The danger of chasing metrics

- **Do not just look at a single evaluation metric**
- Always visually inspect the outputs to make sure things are working the way you expect.
  - Maybe you aren't calculating the metric correctly
  - The metric might just be inappropriate for your problem
  - There might be weird dataset bias



The danger of chasing metrics

From the wonderful "YOLOv3: An Incremental Improvement" by Joseph Redmon, Ali Farhadi:

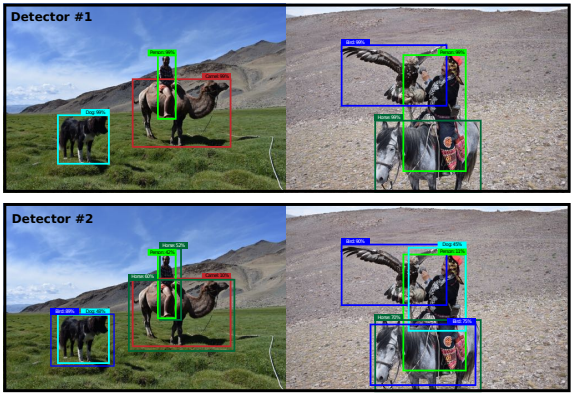


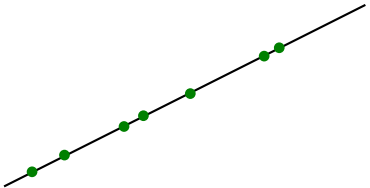
Figure 5. These two hypothetical detectors are perfect according to mAP over these two images. They are both perfect. Totally equal.

What makes a good dataset?

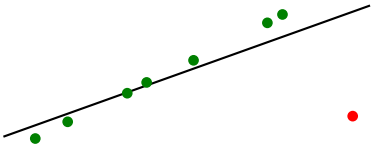
- The training data must look like the test data
- The training data should cover all the cases you care about
- You want at least several thousand training images
- The annotations (e.g. box positions) must be complete, accurate, and consistent
- Don't label things for detection if you know your algorithm can't possibly detect them

More data is not the same as better data

- Adding more training data isn't going to help if it's low quality.
- There are often really hard cases you don't care about. Exclude them from the dataset.



Great Fit



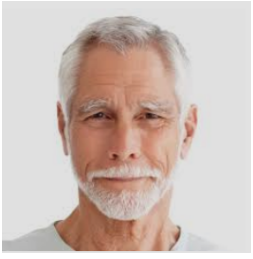
Bad Fit

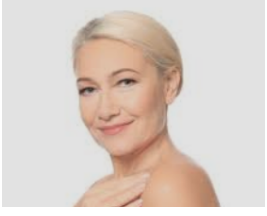
More Bad Training Data

Suppose you want to train a face detector to find faces in images like this:

More Bad Training Data

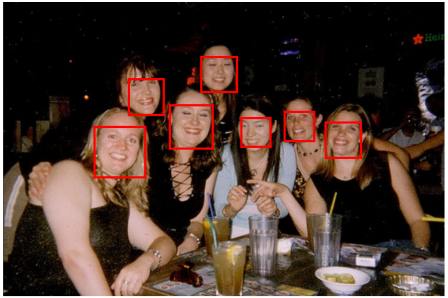
Then this is horrible training data:


Data augmentation is really important

- CNN+MMOD face detector trained with and without data augmentation:

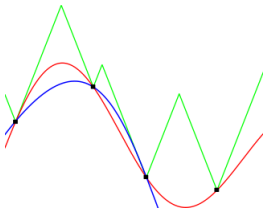


- Done by setting this in examples/dnn_mmod_ex.cpp:

```
cropper.set_translate_amount(0);
cropper.set_randomly_flip(false);
cropper.set_max_object_size(1);
cropper.set_max_rotation_degrees(0);
```

Setting hyperparamters

- Hyperparameters are the parameters machine learning doesn't set for you. E.g. stopping conditions, learning rates, regularization strengths, etc.
- Don't just try default settings. Use cross-validation/hold out testing to find good parameters.
- A great hyperparameter optimization algorithm: LIPO+TR



Tomorrow's workshop

- We are going to make our own training datasets and build working detectors
- Work through 40 python examples
  - What do you do when it doesn't work?
  - How did I know to do that?
  - How to make it run fast?
- Install C++ compiler, CUDA, and cuDNN: we will be making custom python extension modules using C++ and CUDA as part of the workshop
- Get workshop materials from http://dlib.net/files/data/pyimageconf2018.tar.bz2