COMP309 in Week 11, 2024

deep neural nets

some tricks of the trade, and intro to classifying images

Marcus Frean marcus@ecs.vuw.ac.nz

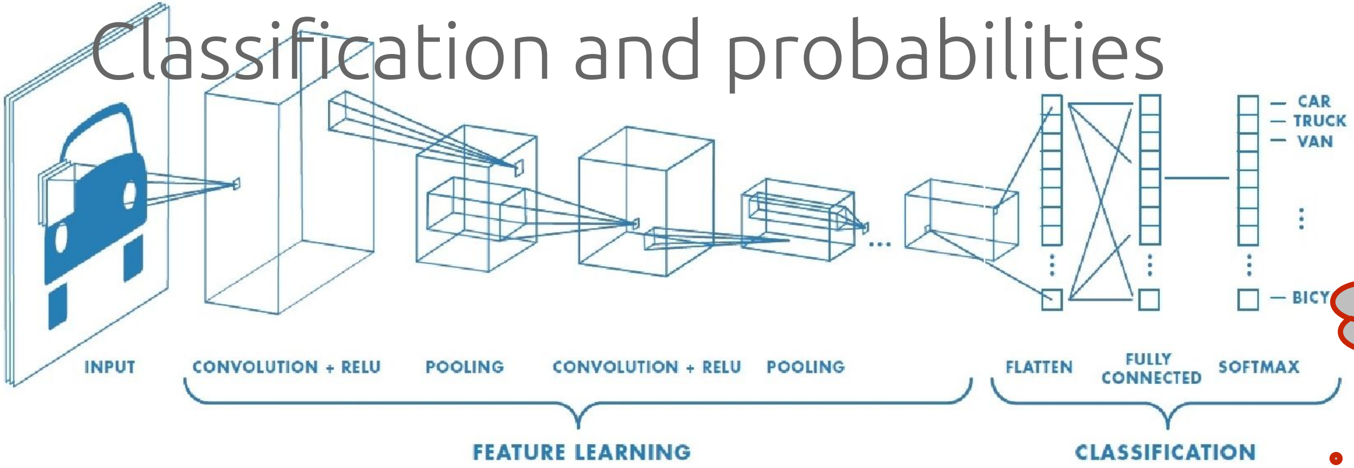
Week 11

- Lecture 1 (lots of review here)
  - softmax
  - cross-entropy
  - regularisation
  - batch norm, dropout, and residual nets

- Lecture 2
  - convolutional nets (image recognition)

- Tutorial:
  - walk-through of a CNN built in PyTorch

Classification and probabilities



probability

1. When we classify anything, it's good to express a <u>*degree of belief*</u> that the input belongs to each plausible class
   - e.g. consider "this image is a car",compared to "this is *most likely* a car, but *could* be a truck, but is almost certainly *not* a bicycle"
2. We want a loss that is *differentiable*, in order to use Gradient Descent

probabilities that depend smoothly on parameters

Both these things suggest:

output <u>floats</u> → softmax function → **probabilities as outputs**

& cross-entropy ("log loss") as the loss function

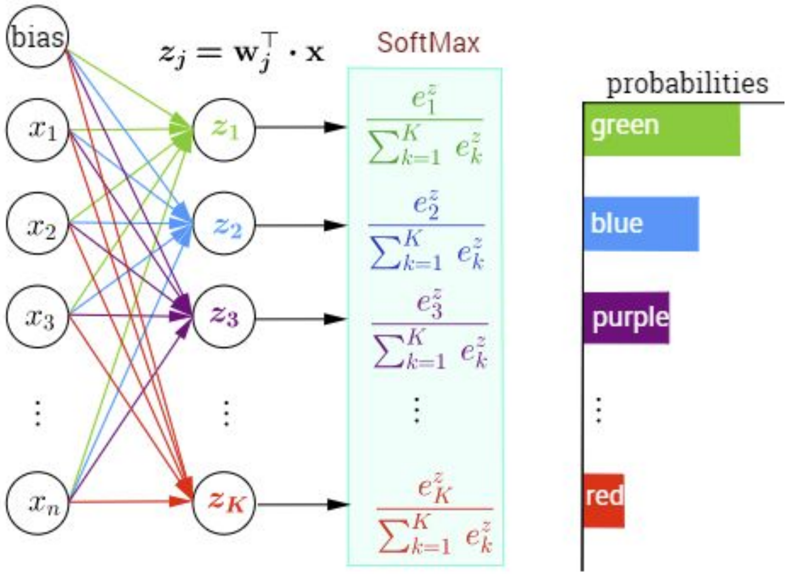
softmax

"logits" are the raw sums, as outputs from a weights layer

I. exponentiate them, to make them all positive

II. rescale them so they sum to 1

We can then call them "probabilities" if we want



Q: but couldn't we just push each z through a sigmoid function, and get probabilities that way?

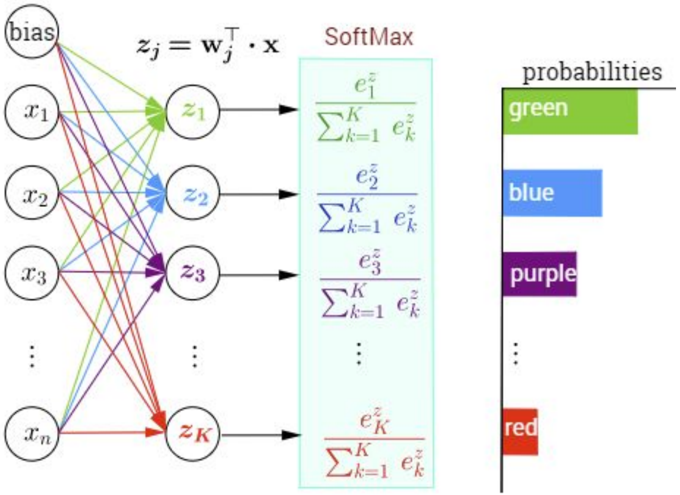
What's the difference?

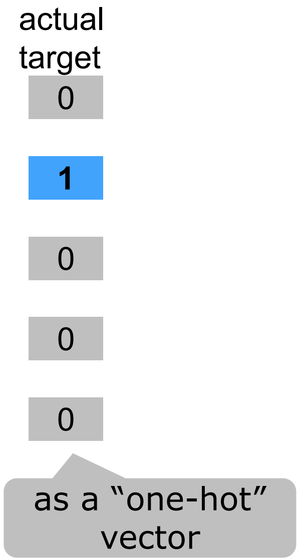
http://rinterested.github.io/statistics/softmax.html

log-loss and cross-entropy

In words: Imagine guessing the class at random from the classifier's distribution: what's the chance you'd guess correctly? We want this likelihood to be high.

Called "log loss" and also known as "cross-entropy": $\log \Pr(\mathit{all}\ \text{correct}) = \sum_{\text{item } i} \sum_{\text{class } j} t_{ij} \log y_{ij}$





REMINDER from tutorial: PyTorch's function `CrossEntropyLoss` actually takes raw logits as argument, *not the probabilities.*

So for learning, you don't need to build in SoftMax. (But to know "doubt" in the predictions, you do).

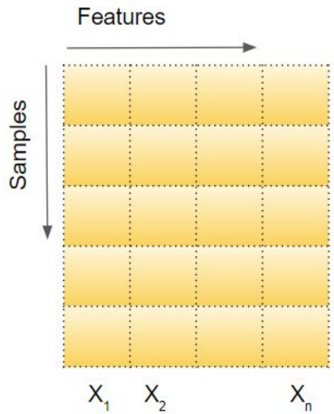
Batch normalisation

Note that "normalisation" has several meanings, e.g.

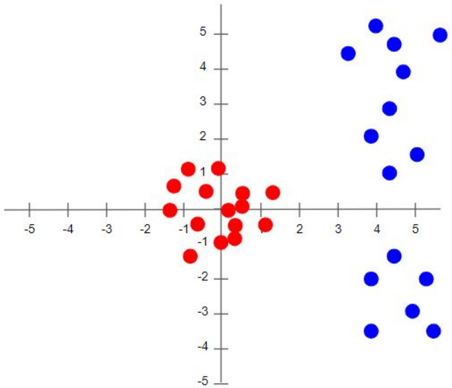
- normalising a probability distribution:
  scaling it so it sums to 1
- normalising input data: (usually)
  shifting it to have mean=0, and scaling it to have variance=1.

  BatchNorm is like doing this with every layer
- normalising a database is something else altogether.

familiar case: normalisation of *inputs*



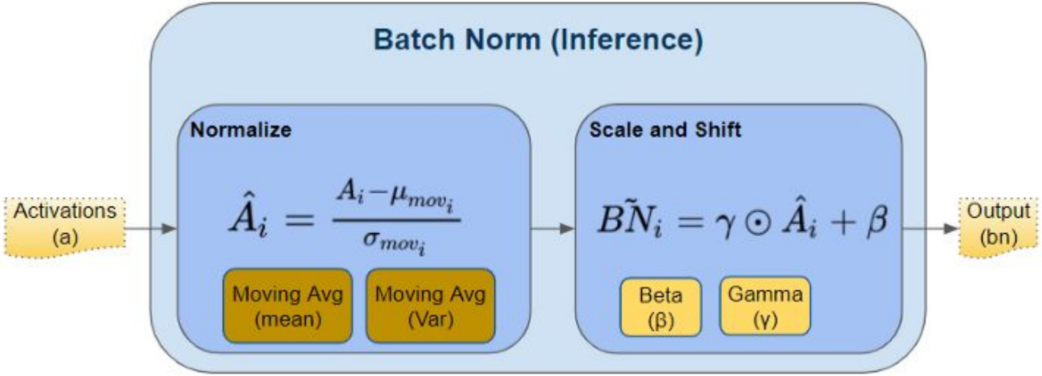
$$X_i = \frac{X_i - Mean_i}{StdDev_i}$$



"Batch normalisation" shifts and scales in the same way, but can be inserted as a processing layer (just like ReLU, etc) *anywhere in the network*.

- Has parameters that are learned (nb. Autograd goes right through)
- Yet another weird hack that helps :)
- Allows higher learning rates, as well as training of deeper nets.

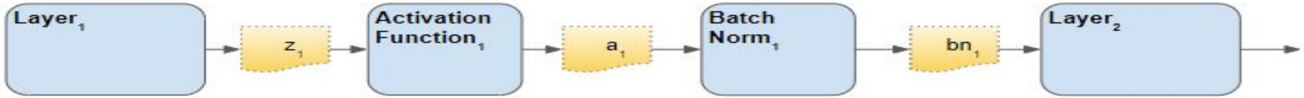
batch normalisation → better behaved gradients

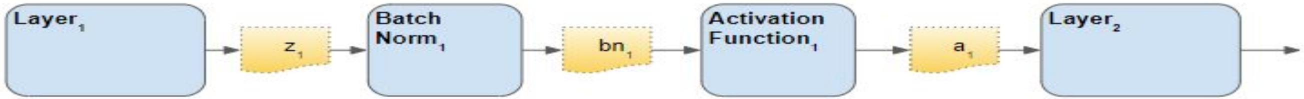


The innovation, compared to normalisation of inputs:
not restricted to mean of *zero* mean and variance of *one* – instead, these become learnable parameters of the system.

Not 100% clear *why* it works! :(
It makes the overall **scale** of weights irrelevant to learning

note we could put it *before*, or *after*, the non-linearity:



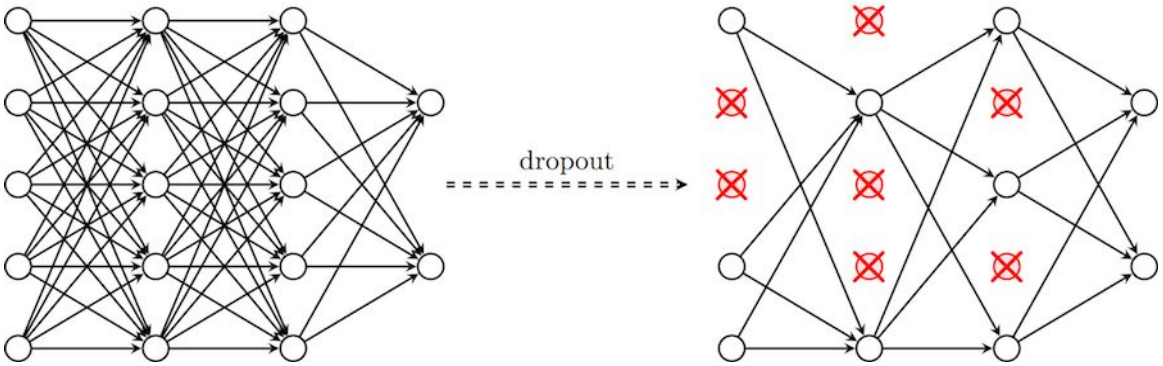


pics source: https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

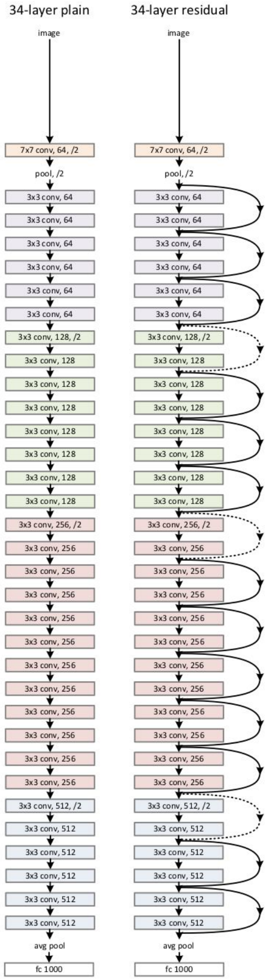
Dropout

Dropout is a regularization technique, used for reducing overfitting in neural networks by *preventing complex co-adaptations* on training data.

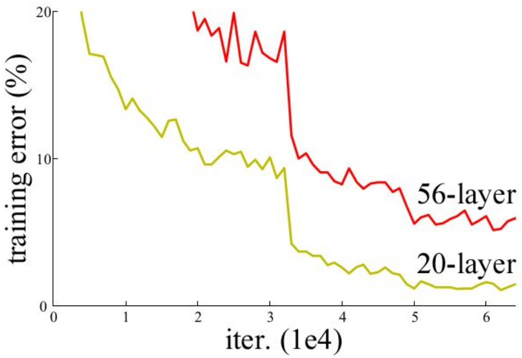
The key idea is to randomly drop units (along with their connections) from the neural network during training. This prevents units from co-adapting/fitting too much.

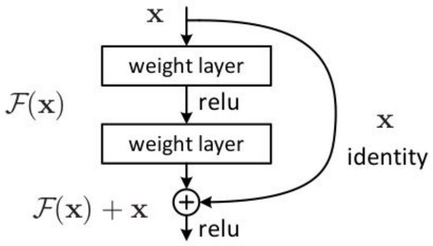


(possibly not a good idea to use Dropout *as well as* Batch Norm)



Residual nets ("resnets")





Training really deep nets was hard.

Resnets added "skip" connections.

In a sense, the network is learning a "residual" now…

pics from the original resnet paper: https://arxiv.org/abs/1512.03385v1

Residual nets (“resnets”)

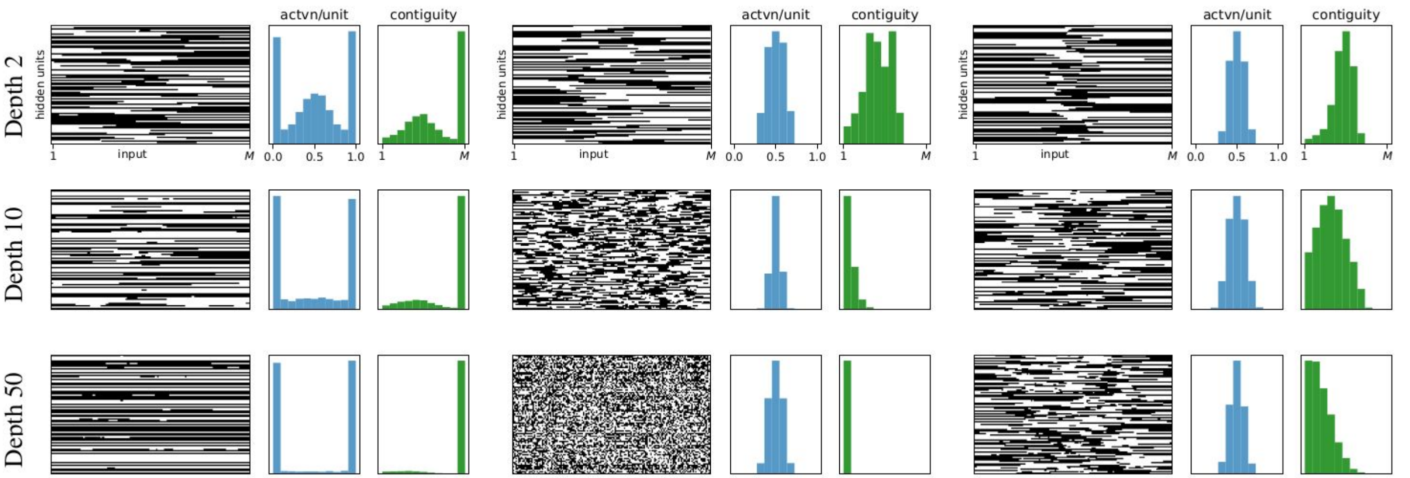
As with BatchNorm, it’s actually not so clear what the reason for the success is 😐

We think it’s because resnets simplify the gradient

**The Shattered Gradients Problem:
If resnets are the answer, then what is the question?**

David Balduzzi [1] Marcus Frean [1] Lennox Leary [1] JP Lewis [1,2] Kurt Wan-Duo Ma [1] Brian McWilliams [3]

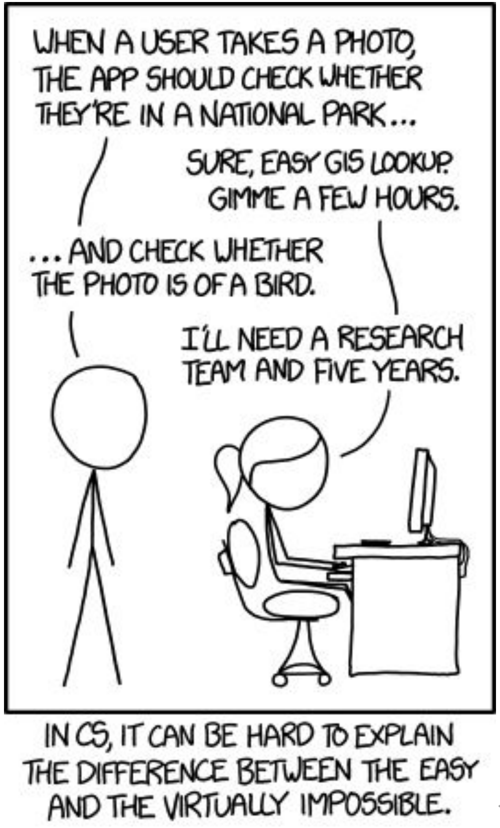
The Shattered Gradients Problem



(a) Feedforward net without batch norm. (b) Feedforward with batch normalization. (c) Resnet with batch normalization.

end of discussion of deep learning in general

image classification - some ideas

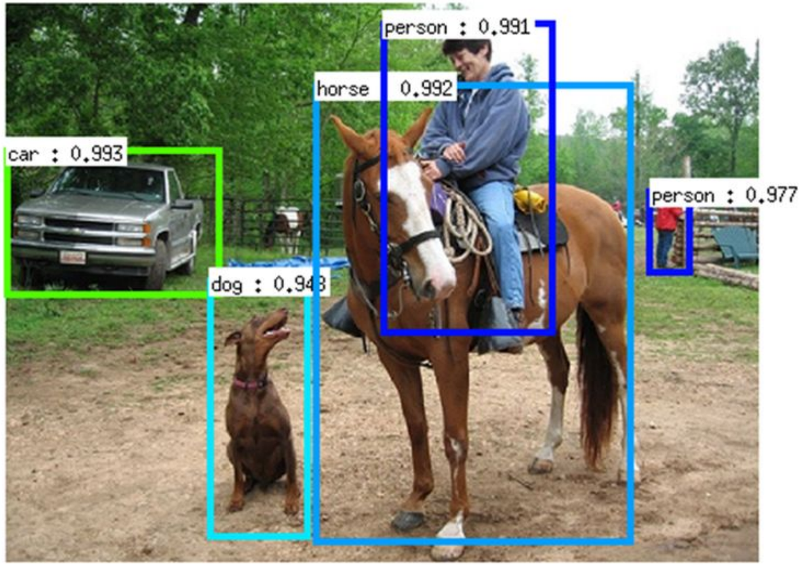


not long ago, detecing an object in a natural image was virtually impossible

http://imgs.xkcd.com/comics/tasks.png

image classification

- a core problem, with a large variety of practical applications.
- Many other seemingly distinct Computer Vision tasks (such as object detection, segmentation) can be reduced to image classification.



Object Detection

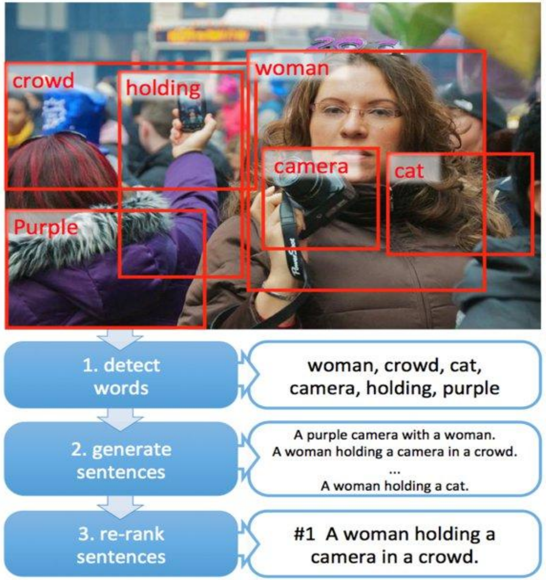
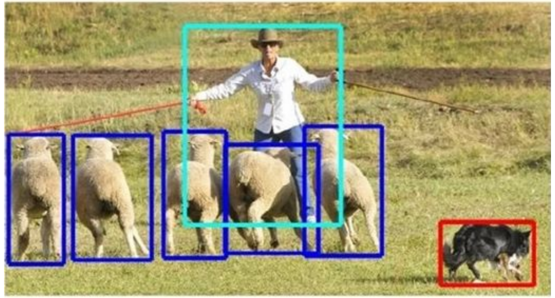
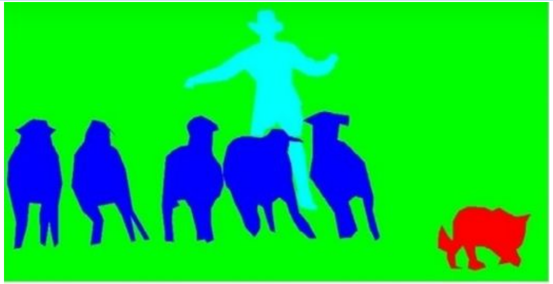


Image Captioning

1) Image Classification



2) Object Localization



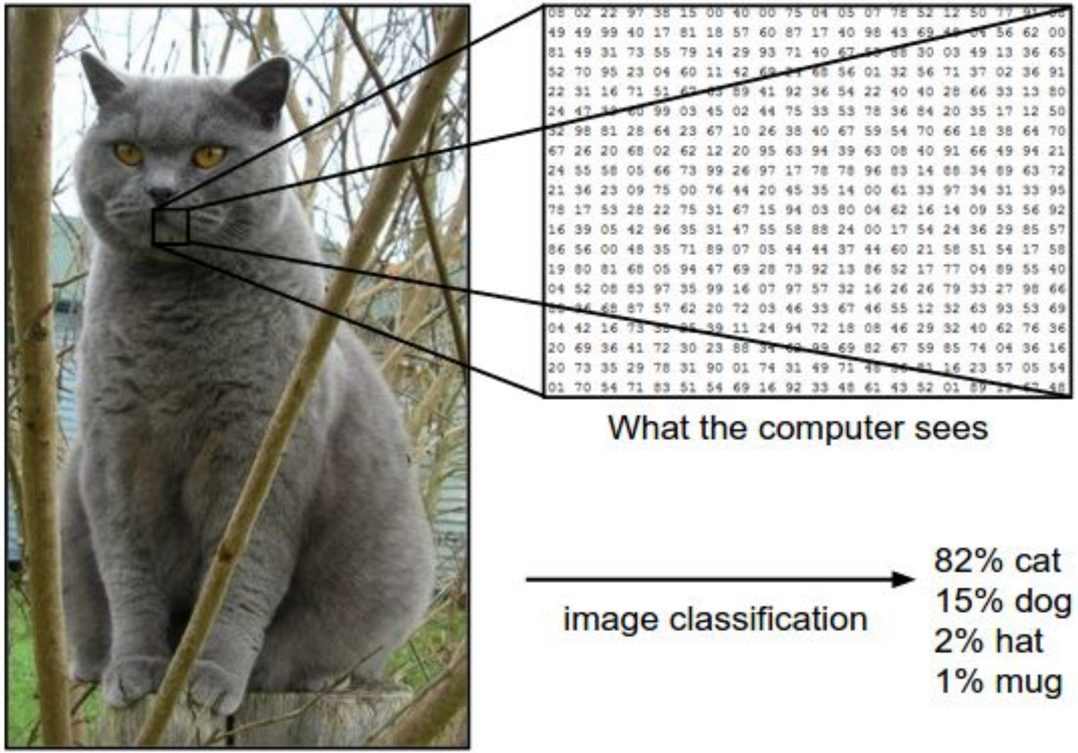
3) Semantic Segmentation

4) Semantic Instance Segmentation

https://openaccess.thecvf.com/content_cvpr_2016/papers/Yang_WIDER_FACE_A_CVPR_2016_paper.pdf

image classification

Obviously, to computers, images are just numbers arranged in a grid:



Challenges in image classification

Scale variation

Visual classes often exhibit variation in their *real* size (not only in terms of their extent in the image).

Deformation

- objects of interest might be very flexible

Challenges in image classification

Occlusion

- the objects of interest could be blocked by other things in the foreground. Sometimes only a small portion of an object could be visible.

Background clutter

- the objects of interest may *blend* into their environment, making them hard to identify.

Challenges in image classification

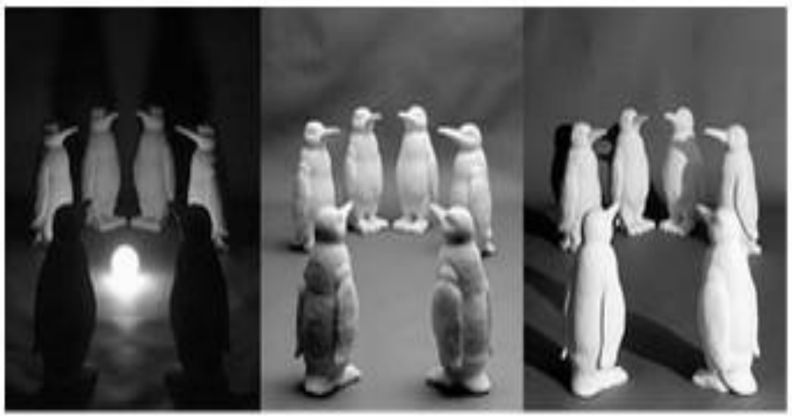
Viewpoint variation

the same object can be oriented in many ways with respect to the camera

Challenges in image classification

Illumination conditions

- the effects of illumination are drastic at the pixel level.



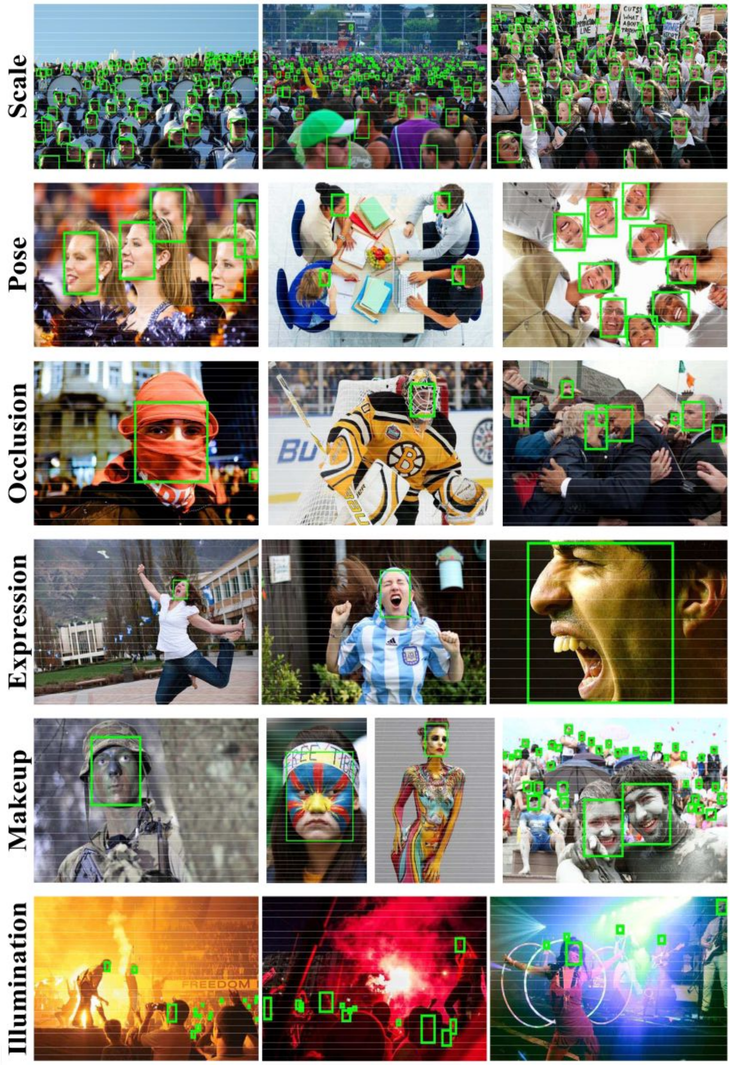
Challenges in image classification

Intra-class variation

the classes of interest can often be relatively broad, such as *chair*.

There are many different types of these objects, each with their own appearance, yet they may be lumped together under the same label

all of these issues, combined!



https://openaccess.thecvf.com/content_cvpr_2016/papers/Yang_WIDER_FACE_A_CVPR_2016_paper.pdf

one reason why it's been so difficult

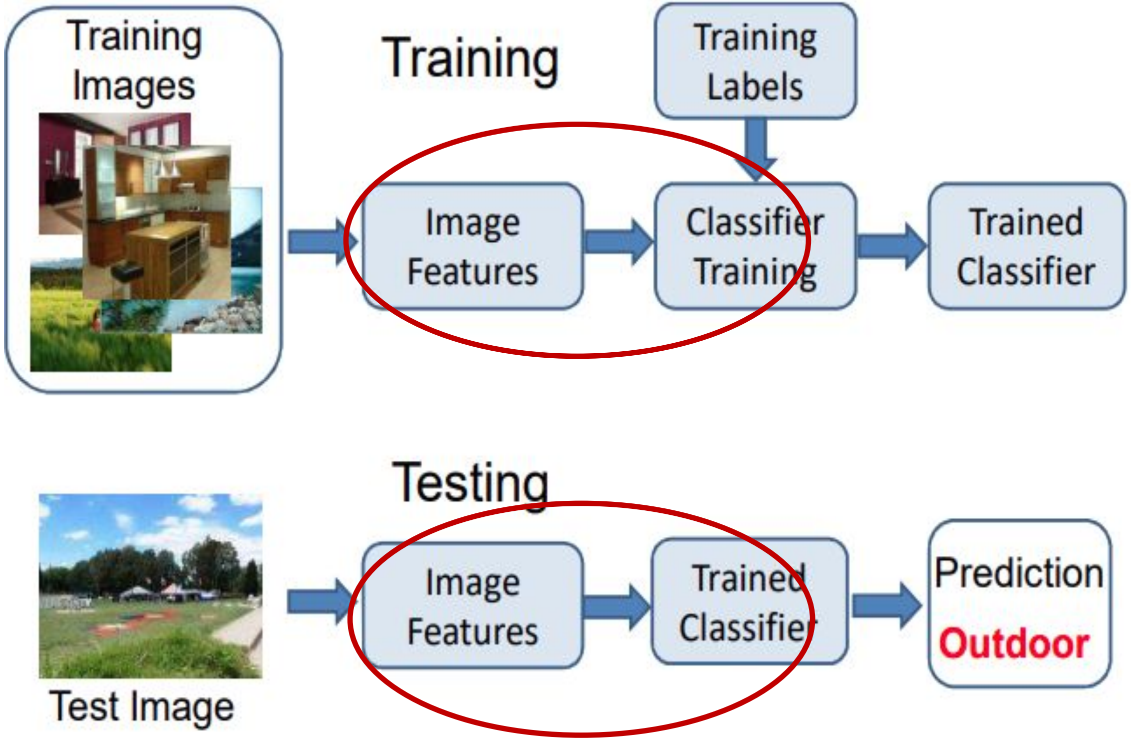
The goal for us is to build a good image classification model that:

- is invariant to the many effects of all these variations
- yet remains sensitive to the critical variations that indicate the class

Actually this is true of *any* classifier

Feature Learning

Traditional Image Classification Process:



Feature Learning

- Many methods have been proposed over the years:
  - Scale-Invariant Feature Transform (SIFT) – Lowe, 2004
  - Speeded Up Robust Features (SURF) – Bay et.al, 2006
  - Many others…

- Implementations:

  OpenCV – Python API

  https://docs.opencv.org/master/

  Scikit-Image – Python

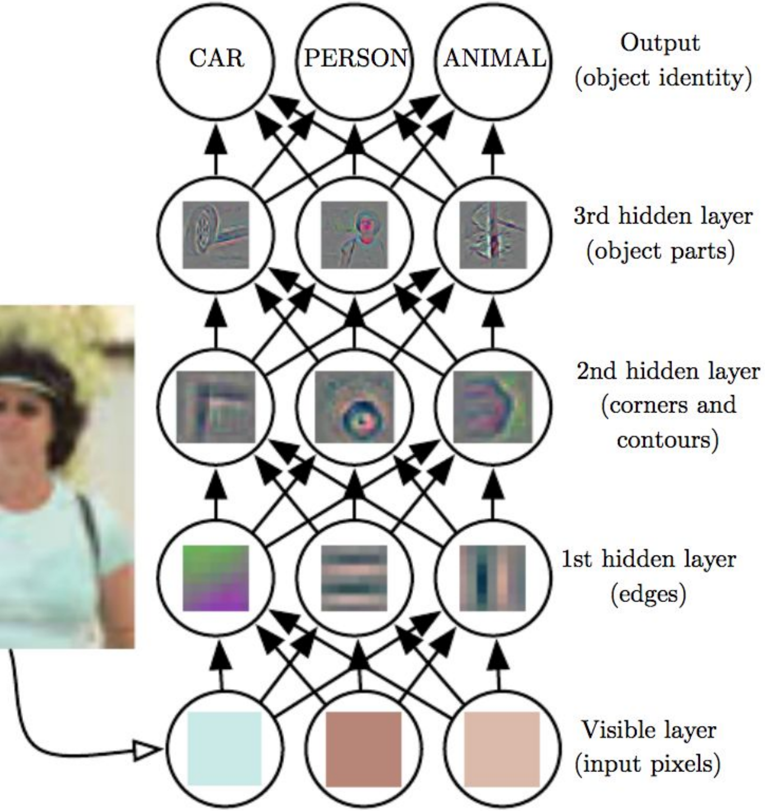
  http://scikit-image.org/docs/stable/api/api.html

We used to dwell on visual features, but it’s become less relevant

Feature Learning “all the way up”?

Learning via a hierarchy of feature extractors

- Each layer extracts features from output of previous layer
  all the way from pixels to classifier
- Layers have the (nearly) the same structure
- We train all layers jointly



Deep Learning

- we can learn the whole hierarchy of features from images