# Image Classification

Aram Karimi

LT2318 H21 Artificial Intelligence: Cognitive Systems

November 15th, 2020

# Outline

- What is "image classification"?
- Why is "image classification" important?
- How does "image classification" work?
- "Image classification" methods Using machine
- The need for AI to understand image data
- Convolutional Neural Network (CNN)
- When do we use pre-trained image features?
- Hands-on Tutorial

## What is "image classification"?

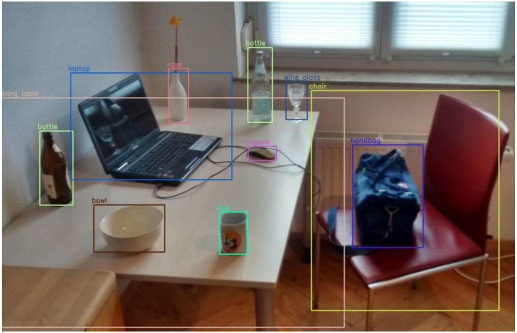
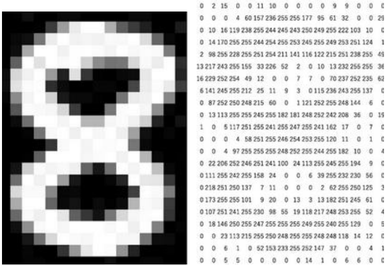


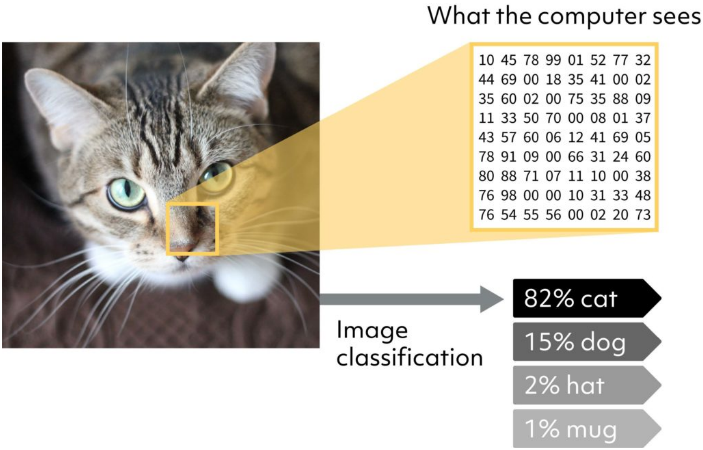

Image classification is a basic task for human, but still one of the most important tasks that computer vision engineers can tackle.

# What makes image classification a very important task?



An image is a large grid of numbers between [0, 255]

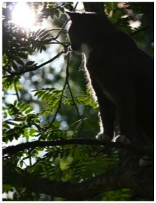
An image can be of size 800 x 600 pixels, each pixel is represented via three numbers, which provide values of RGB (red, green, blue) channels


What the computer sees
10 45 78 99 01 52 77 32
44 69 00 18 35 41 00 02
35 60 02 00 75 35 88 09
11 33 50 70 00 08 01 37
43 57 60 06 12 41 69 05
78 91 09 00 66 31 24 60
80 88 71 07 11 10 00 38
76 98 00 00 10 31 33 48
76 54 55 56 00 02 20 73
Image classification
82% cat
15% dog
2% hat
1% mug

## Challenges

- Viewpoint
- Illumination



## Challenges

- Deformation

Challenges

- Occlusion

## Challenges

- Clutter

## Challenges

- Intraclass variation

How does Image classification work in machines?

- In digital image processing, image classification is done by automatically grouping pixels into specified categories, so-called “classes.”
- The algorithms separate the image into different classes based on their prominent features or specific patterns

Image classification techniques are mainly divided into two categories:

- Supervised
- Unsupervised

Where is it used?

- Robotics, Computer Vision, NLP, information retrieval, etc.

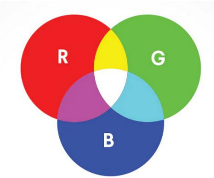
Representing images with “features”

- CV vs NLP features
- Visual features
  - Color, size, center, orientation, etc.
  - Invariant to transformations
- Lexical features / semantic classes
  - Labels, context

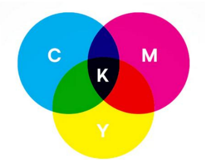
- Learned vs. pre-engineered features
  - We need to choose how to represent an image

# Visual feature: color

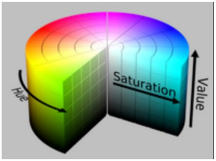
RGB




CMYK



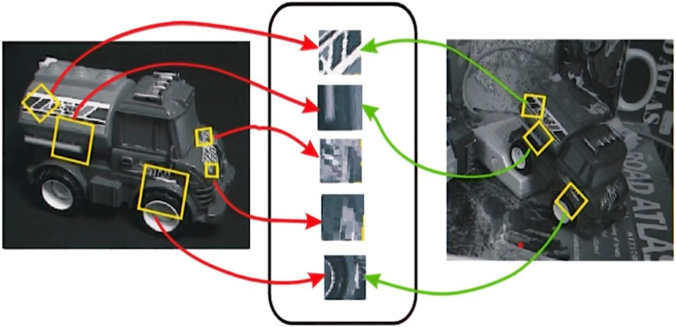

HUE



## Visual features: SIFT, HOG, SURF

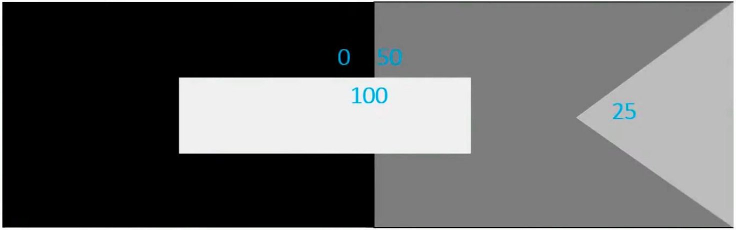
- **SIFT**: Scale-Invariant Image Transform
  - Commonly used in CV
  - Extract invariant (not changeable) image features
  - Applied to grayscale images
  - Mathematically complicated, computationally heavy
  - Based on histogram of gradients, e.g. computing the gradients of each pixel in the image takes a lot of time
  - Quite slow compared to SURF
  - Does not work well with lighting changes and blur

# Visual features: SIFT



Visual features: HOG(Histogram of Oriented Gradients)

- **HOG:**: compute centered horizontal and vertical gradients
  - Tries to extract contrasts in various image parts
  - Computes gradients magnitudes and their directions

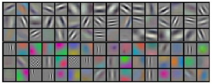


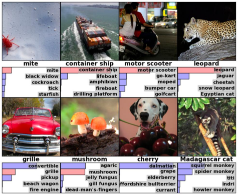
The need for AI to understand image data

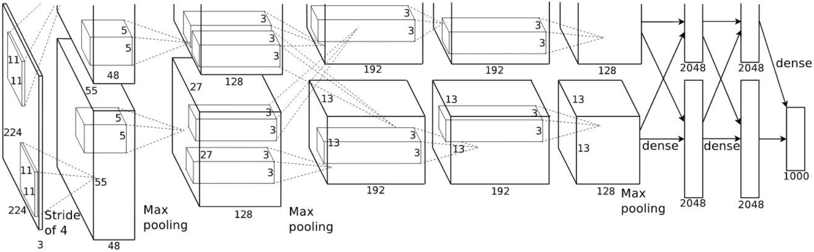
- Supervise learning task
  - **Convolutional Neural Networks** as feature extractors hierarchical layer-wise representation learning
  - Extract invariant (not changeable) image features
  - CNN is a hierarchical deep learning model which is able to model data at more and more abstract representations
  - CNN features are highly adaptive, they are trained end-to-end
  - CNN can learn features similar to SIFT and HOG from training examples alone, which is quite cool. Therefore, **using CNNs minimizes feature engineering**

Convolutional Neural Network (CNN)

- Used for object detection, image classification, image captioning, etc.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12). Curran Associates Inc., Red Hook, NY, USA, 1097–1105.

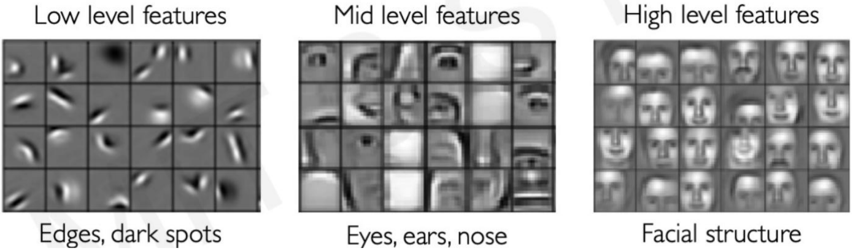




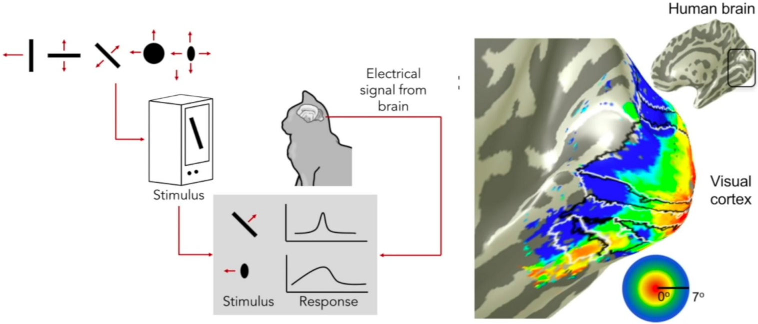
## Learning Feature Representation

Can we learn a hierarchy of features directly from the data insted of hard engineering?

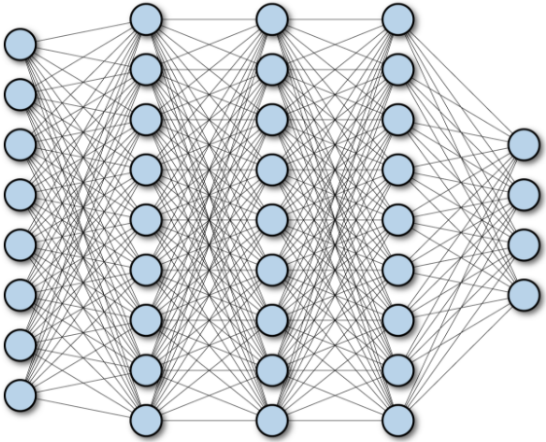


## CNNs: inspiration

- Hubel and Wiesel (1959, 1962, 1968): cat's visual cortex maps information in a structured and hierarchical way



# Fully Connected Layer

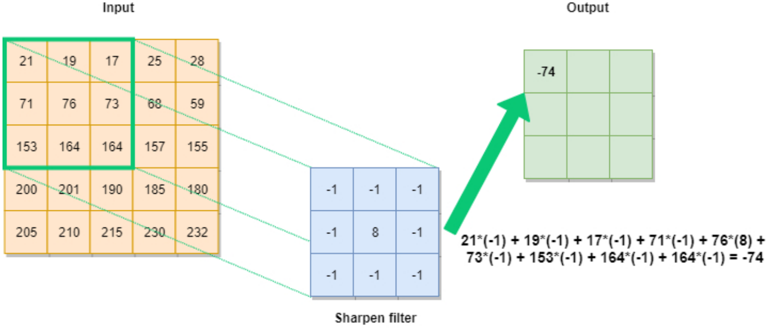


Using Spatial Structure

**Input**: 2D image.

**Idea**: Connect patches of input to neurons in hidden layer.

(Neuron connected to region of input only sees these values)

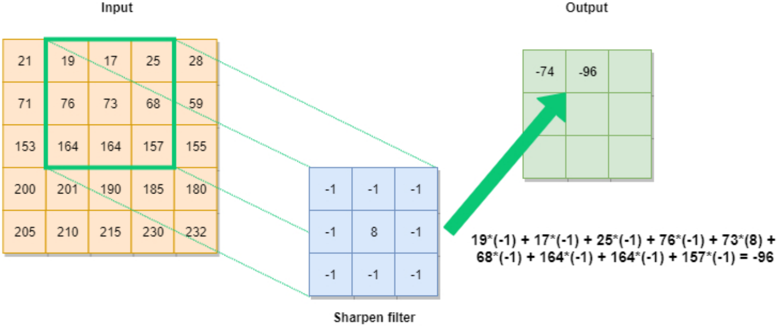


## Using Spatial Structure

**Input**: 2D image.

**Idea**: Connect patches of input to neurons in hidden layer.

(Neuron connected to region of input only sees these values)

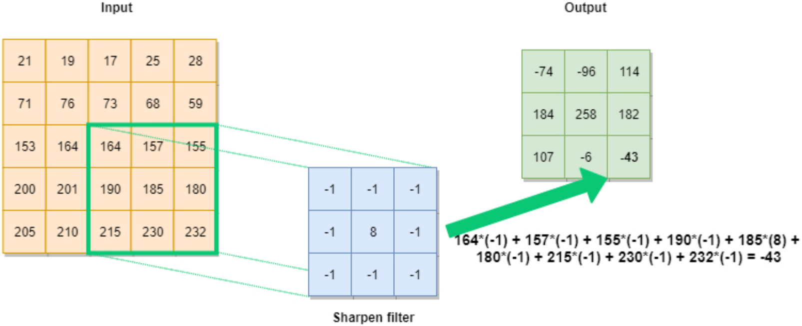


## Using Spatial Structure

**Input**: 2D image.

**Idea**: Connect patches of input to neurons in hidden layer.

(Neuron connected to region of input only sees these values)

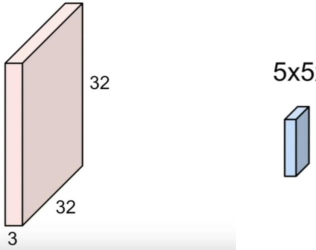


# CNN: we want to preserve spatial structure

## Convolution Layer

32x32x3 image

5x5x3 filter



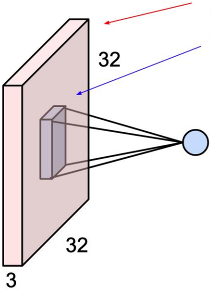

**Convolve** the filter with the image
i.e. "slide over the image spatially,
computing dot products"

# CNN: apply filter to the convolution layer

## Convolution Layer

32x32x3 image

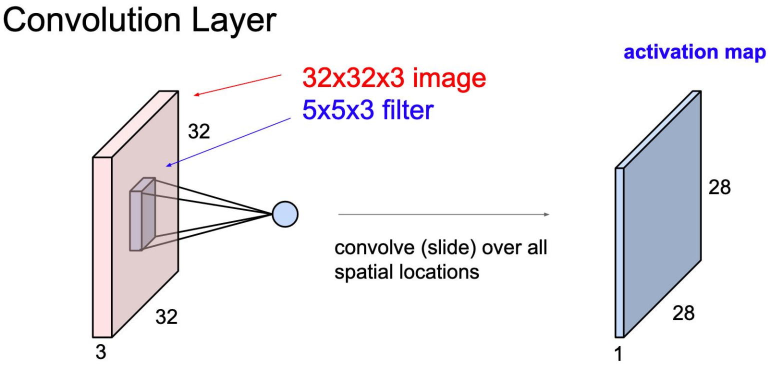
5x5x3 filter $w$



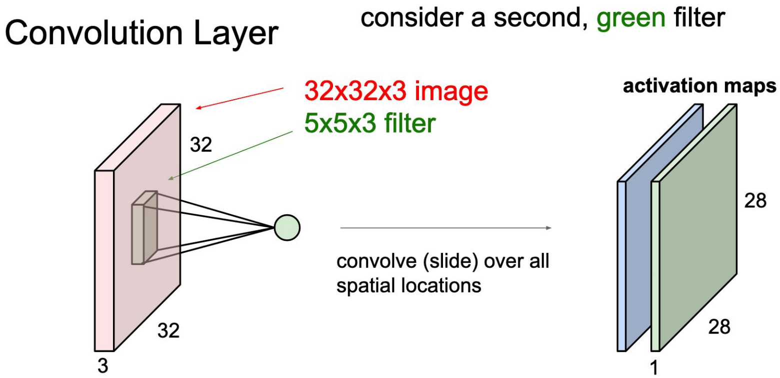

**1 number:**
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

# CNN: use filter to get an activation map

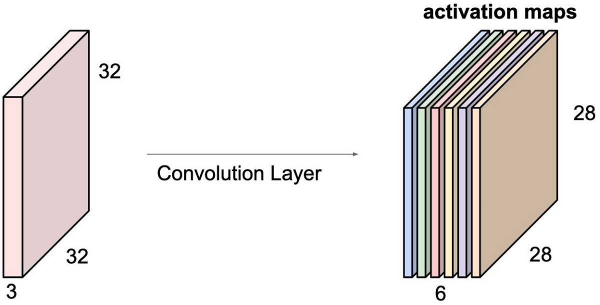


# CNN: maps per filter



# CNN: activation maps

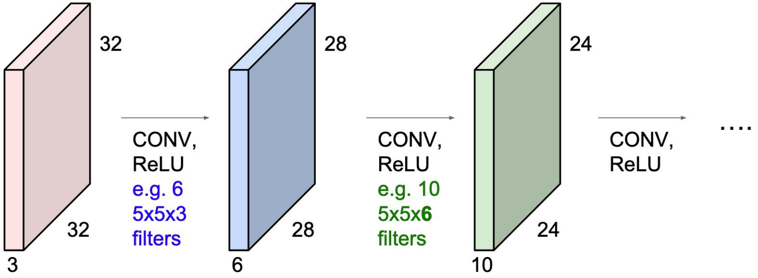
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



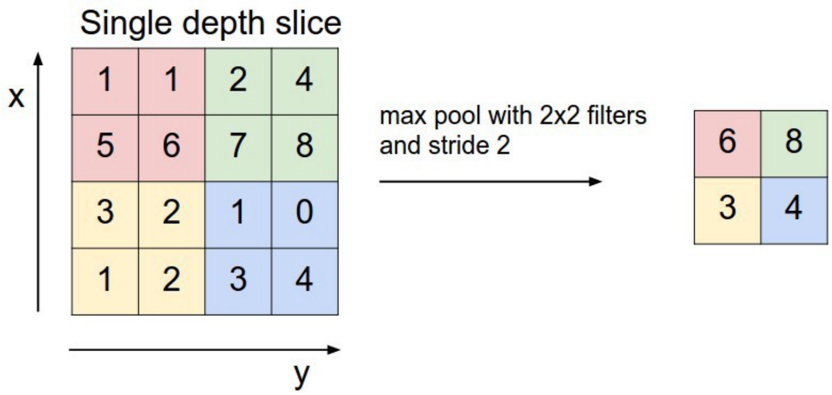

We stack these up to get a "new image" of size 28x28x6!

# CNN: a stack of CONV, FC, POOL + activations

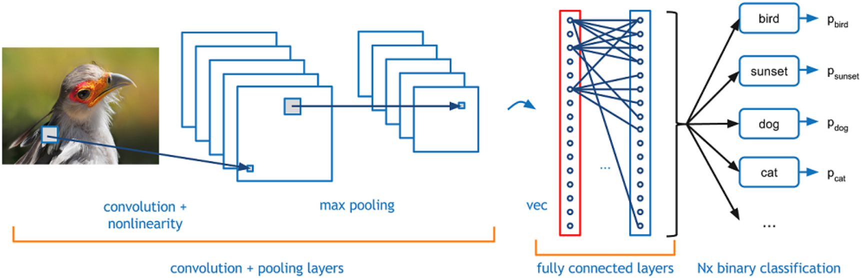
**Preview:** ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



# Pooling layer



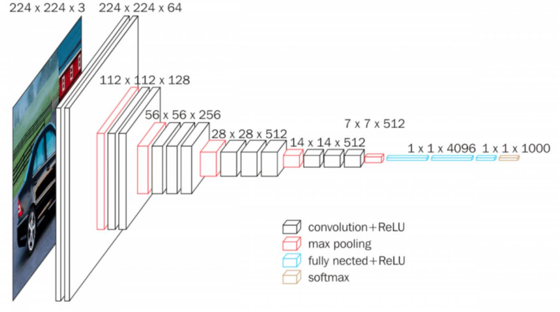
Finally, the raw values which are predicted output by network are converted to probabilistic values with use of soft max function.



## CNN: conclusion

- Smaller filters, deeper architectures
- Tend to remove POOL and FC, keep CONV only

**Example CNN network structure, VGG16**



## Useful Links

- PyTorch Image Classification Tutorial: https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
- TensorFlow Image Classification Tutorial: https://www.tensorflow.org/tutorials/images/classification
- More recent work on CNNs: https://github.com/matterport/Mask_RCNN https://github.com/facebookresearch/detectron2
- Accuracy scores for published CNNs: Accuracy scores for published CNNs:
- Class code: https://github.com/sdobnik/aics/tree/master/tutorials/01-image-classification/2021