Computer Vision

Professor Marie Roch with major contributions from Professor Liu
Chapter 25, Russell & Norvig (we only cover highlights)
Computer vision tasks

- Detection
- Labeling
- Relationships between objects

Most vision tasks are handled as classification problems.
Our focus: Single label image classification

Spanish shawl nudibranch
(Flabellina iodinea)
What makes this hard?

• Perspective and view point changes
Pinhole cameras & perspective

Similar triangles:
\[-\frac{x}{f} = \frac{X}{Z}, \frac{-y}{f} = \frac{Y}{Z}\]

\[
\Rightarrow x = \frac{-fx}{Z}, y = \frac{-fY}{Z}
\]

f – focal distance between pinhole & projection plane

Image: Wikimedia commons
Illumination

Each row is the same person with different lighting conditions

Specular reflections
Deformation

Instances of the same class may be nonrigid and undergo deformation

image credit: Sarah K, Benton County 4-H, MN, https://extension.umn.edu/
Occlusion

Objects can be hidden by other objects
Clutter and noise

Background regions that are similar to the foreground
• Satantic leaf-tailed gecko
• rock wallabees
Variation

Images have same label (dog) although they are quite different
Summary

Computer vision must deal with a variety of complicating factors

• perspective
• illumination
• deformation
• occlusion
• clutter and noise
• variation
Like all supervised machine learning problems

• Need labeled corpus
  • Training data
  • Test data

• Simple k-nearest neighbor classifier
  • Training: Retain database of images
  • Test: Look for most similar images and use plurality class
Measuring distance between images

• Naïve method: pixel by pixel comparison (all we will be doing)

• L1 or L2 distance

\[ d(I, J) = \sqrt{\sum_{x,y} |I_{x,y} - J_{x,y}|} \]
How well is our method going to do with respect to?

- perspective
- illumination
- deformation
- occlusion
- clutter and noise
- variation

(images from CIFAR 100 dataset)
A better way...

What if we compared local regions?
Convolutional neural networks

- Learn *kernels* of recurring subregions of images
- Convolve the *kernels* with a tensor (e.g. image)
- Deeper layers typically learn more complex information
Convolution in 2D

Animation credit: Irhum Shafkat
Convolution in 2D

\[
w = \begin{bmatrix}
0 & 1 & 2 \\
2 & 2 & 0 \\
0 & 1 & 2 \\
\end{bmatrix}
\]
Convolution in 2D

• Convolve (*) image I with kernel K:

\[ S(i, j) = (I * K)(i, j) = \sum_m \sum_n I[m, n]K[i - m, j - n] \]

• Many libraries implement cross-correlation instead of convolution (kernel not reversed, not important in practice):

\[ \sum_m \sum_n I[i + m, j + n]K[i, j] \]
Convolutional layers

- Fixed-size kernel, e.g. (5, 5)
- Convolutional output is passed through the activation function
- Stride parameters cause skips

- Common to pool regions and take some statistic e.g., max
Basic ideas

• Convolution is an operation like any other
• Loss can be backpropagated and kernel weights adjusted
• Loss function guides what is learned
• Downsampling operations called pooling reduce size

• Going from convolution
  • Flatten last layers
  • feed-forward network
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Input, Conv2D, MaxPooling2D

model = Sequential()
model.add(Input(shape=(1000,1000,1)))  # 1000x1000 grayscale
model.add(Conv2D(32, kernel_size=(5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2), strides=(2, 2)))
model.add(Conv2D(64, kernel_size=(5, 5), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
# Convert to vector, then feed forward network
model.add(Flatten())
model.add(Dense(1000, activation='relu'))
model.add(Dense(NumClasses, activation='softmax'))