Computer Vision

Lecture 1

- SIFT / HoG: Histogram of Gradients

  Histogram of intensity does not work due to loss of geospatial information

  ↳ any permutation of original picture will work for the classifier.

  Instead, consider a Sobel Kernel to detect horizontal and vertical edges → compute gradient in each position.

  Then, plot a histogram based on how many times you perceive a specific value of angles between edge and grad.

- Gaussian vs. linear blur: Gaussian has more control through the value of σ.

- Filter banks: looks for particular lines and orientations. → to represent textures

  Can be obtained through different derivatives of Gaussian filter.

- Mathematical morphology: (mostly works only on binary classification) → for restauration, contour detection

  structured element composed by 0's and 1's.

- 4 operations:
  1) Dilation: if the center (white) $\underbrace{\text{interacts}}_{\text{while sliding}}$ with another white pixel ⇒ apply logical sum (OR)
  2) Erosion: only when all whites in mask are matched with whites ⇒ keep the center
  3) Closing: dilate + erote. Equivalent to: if I can put the mask between two unconnected structures and it works as a bridge → connect them.
  4) Opening: erote + dilate. Erote, but keep whole mask.

- Substracting dilation and erosion (Dilate(I) - Erote(I)) works as contour detection.

Lecture 2. Handcrafted Methods

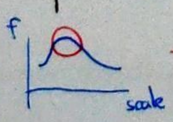
define a descriptor → use machine learning

- Used when not enough data for Deep Learning.
- In some cases, descriptors can be injected into a NN.

- usable also in video.

Occlusion: common problem → but we still may have enough points of interests to detect an object!

To detect points of interest: compute a function that for each region, it computes an output for different scales.



→ the $\underbrace{\text{maximum}}_{\text{minimum also}}$ represents stability for a scale → may be a PoI.

• SIFT: detects PoI and describes what is there.
very simple.

Feature vector with info about $grad_x$, $grad_y$, orientations.

- robust towards global changes in illumination → because it is normalized.
  it may find difficulties with local changes.
- not robust towards orientations: due to the axes of the gradients in the orientation.
  rotate the batches towards the predominant direction always.
  (if match, they will be rotated in the same way → orientation-friendly)
- robust towards affine transformations through affine patches (they use the two principal orthogonal directions)

useful when comparing in the same scene: cameras, robots, ...

• RANSAC: helps when there are too many candidate PoI with similar descriptions.
It will assume they will follow the same homography.
Check if all candidates follow the same transformation.

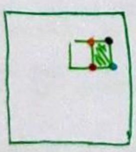
• Highlight features: like what we used to detect faces

→ eyes

Integral images: cummulative sum from top-left.

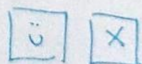
it helps us compute this 'easierly'.

used in



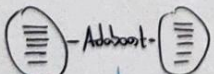
= • − • − • + •
in integral image (4 acceses)

• Adaboost cascade:
- compute a lot of features from each image of faces



10K ∞
unbalanced

- we have



Face desc. – Adaboost – non-face

learns with success of
99 % 90 %

next adaboost
same faces + failed backgrounds → each classifier especializes in the error of the rest of classifiers

...

success end of cascade:
99 % ~99 %

- applied through sliding window:
- few operations (Integral images; most bgs don't go through many layers; only faces do)

Lecture 3. Convolutional Neural Network

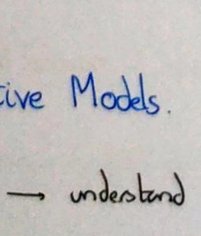
↳ gradient descent to minimize the loss.

- **Stochastic Gradient Descent:** use sampling to optimize (use a batch instead of whole dataset)
  ↳ computational graph
- **Regularization:** way to guide towards no-overfitting.
  We want weights to be regularized
  $\begin{pmatrix} 0.1 & 0.8 & 0.7 & \cdots \\ 0.2 & 0.1 & 0.8 & \cdots \\ 0.1 \\ 0.9 \\ \vdots \end{pmatrix}$ is better than $\begin{pmatrix} 0 & 1 & 1 & \cdots \\ 0 & 0 & 1 & \cdots \\ 0 \\ 1 \\ \vdots \end{pmatrix}$
  ↓
  more robust toward changes in data.
  Dropout has the same effect.

  - small values of λ (please)

- **Data augmentation:** also in testing. crops, scales

- Case studies:
  1) LeNet drawing in "Extra"
  2) AlexNet : duplicated parallel structure (uncommon now)
  3) VGG: 90% params in FC. Simple structure
  4) GoogleNet: "inception" module extracts different levels of detail
     + dimensionality reduction.
     average pooling on output ⇒ no F.C layer at the end.
  5) ResNet:



$H(x) = F(x) + x$

$F(x)$ is the residual
↑
shortcut for the grad → no vanishing grads.

Lecture 5. Generative Models.

- Unsupervised learning → understand the world.
- **Generative models:** generate samples from the same distribution. Many applications/interests.
- **Autoencoders:** do not use labels : input → Encod → feat → Decode → output error?

You can change little values in the encoding to check what each component looks for → interpretability.

• GANs: → it does not 'generalize' well. it is very dependant on the dataset distribution.

= = = = = = = =

• Recurrent models: based in previous states. generated by previous layers, generally.

↓

with input of images: ⚠ high res is way too unfeasible.
use representations (encoder, etc.) → backbone } all trained together.

- backpropagation: keep a small window of the past.
in practice it is adding a new input to the formulation, multiplied by parameters (Wiv)

- attention. n-dimensional vector with activation of how important is each features.
- self-attention: multiply features * itself.

with attention: depending on which part of the captioning we are at, we may be interested in changing where I am looking at.

$v = v^*$ attention
(v + att_mask)
↓
it comes from the previous stages.

Lecture 6. Human Pose Lecture.

• Decomposition in simpler parts ⇒ many false positives. (classical approach)
↳ evaluate the joint probability of combinations.

• Human pose regression: img $\xrightarrow[DL]{}$ 26 outputs (one per point)
↳ with bad regularization → it will converge to the mean of the coordinates.

- cascade: the problem is partially solved → use that as guide and start again
dangers: overfitting
converging to avg. again.

• Heatmaps: we want to regress an image with a gaussian centered on the joint. } ↛ does not converge to mean.
↳ works infinitely better.

- no structure inference in the network. (Geometric Deep Learning). we need context (how to add?)
↳ convolutions?

Context: • Stacked hourglass network: SoA, not efficient. ↺ inject.
• Compositional models: optimize pairwise (between body parts) + bottom-top of hierarch. tree

• 3D: nothing really new → refinement (3D shapes)

↳ the projection in 2D can be generated from several 3D scenes (NP).

sharing problems but harder → convergence to avg again.

- multitask can be too complex

Just adapt what we do in 2D: (most suitable)

1) Geometric Heatmap

2) Predict 2D pose + add a component that learns 'z' coordinate.

== SoA ends here ==

• Problems we may find: occlusions and clothing variations. ⇒ data augmentation with occluders.

(paste random objects on the image. pixels ain't enough)

• Hand pose estimation (accesibility, gesture detection): do the same approach.

- PROBLEM: all parts ~~are~~ look the same. You need hi-res and context.

↓ you do not really need RGB. It still works.

Voxel to voxel works best for 3D hand scenario.

• Bonus 1: human 3D → clothing motion

• Bonus 2 - Face: affective computing → what you feeling (games, neorehabilitation,...)

It is not emotion recognition (we need neouron analysers)

detect faces commonly associated to...

↳ Heatmap with 60+ joints in face.

Lecture 7. Human behavior in video

• Basic approach: sliding window (+ time)

• Action / Gesture recognition: in general, 1 frame ain't enough.

↳ For action understanding: gradients, bg extraction, optical flow

- Optical Flow: computed between 2 consecutive frames. It assumes pixels move slowly around its neighborhood.

pixels don't change 'much'

neighborhoods move in the same direction

not very uses.

equations $\xrightarrow[\text{solver}]{}$ shows translation of pixels.

"if something disappears" → it should disappear smoothly.

The flow can be obtained in more efficient ways.

HoG + time ⇒ HoF (Flow)

against noise: "moving cubes" (like checking if hitbox moves)

- tracking: "following"
- STIPS: detect change in curvature to check changes in trajectories. Wonderful without noise.
- Levenshtein → to compare trajectories.

adapt ↓

length maybe should not be penalized: invariant to time. (remove +1's in Levensh, but keep +diff)

- Dynamic Time Warping: divide in subproblems
- Deep Learning: using 3D Kernels for convolutions → one value. : if 3rd dimension is as big as temporal length

getting rid of temporal dim → compute here.

$\xrightarrow[\text{3rd dim < time}]{}$ 3D Kernel output (spatial-temporal information)

5 strategies:
1) 2 stream ConvNets: one for image, one for optical flow.
2) 3D conv: use 3D Kernels) less params than) more computations than) needs more training data than.
3) ConvNet + LSTM
4) Two-stream I3D convnets: 1) + 2) "if you lose temporal info, add 3D Kernels" → it may work.
5) Transformer-like: encoding spatio-temporal features.

L1: Image processing

- Low level: images → images
  Mid level: images → attributes } img processing
  High level: scene understanding

- Color space (sRGB): $\underbrace{\text{strongly correlated}}_{\ominus}$, $\underbrace{\text{non-perceptual}}_{\ominus}$

- Image: $F(x,y) = z$ → intensity / gray scale (Digital if discrete)
  ↓
  coord.

  - digitalization: ≡ discretization
  - resolution: number of pixels
  - quantization: gray levels associated to b bits: $L = 2^b$
  - brightness: global light intensity value $B = \frac{1}{N} \sum F(x,y)$
  - contrast: perceived intensity difference between two regions $C = \frac{|F_a - F_b|}{F_a + F_b}$
  - acutance edge contrast
  - sharpness: level of detail (acutance + resolution)
  - noise: gaussian, salt & pepper

- Spatial filtering: operations in terms of neighbourhood (sliding window) } linearity $(F(a+b) = F(a) + F(b))$, shift invariant

  box filter $\frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$ → soothing

  sharpening $\begin{bmatrix}0&0&0\\0&2&0\\0&0&0\end{bmatrix} - \frac{1}{9}\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$

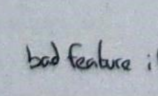
  sobel (edges): $\underset{\text{vertical}}{\begin{bmatrix}1&0&-1\\2&0&-2\\1&0&-1\end{bmatrix}}$ $\underset{\text{horizontal}}{\begin{bmatrix}1&2&1\\0&0&0\\-1&-2&-1\end{bmatrix}}$

  Gaussian / linear noise

- Segmentation: separate object from background
- 2D connectivity: - 4-conn: share a side
  - 8-conn: share a vertex

L2. Handcrafted methods

- Detecting points of interest bad feature if . Invariant to a lot of stuff.
  - scale 'invariant' functions



↓ independent for each region.

$F = \text{Kernel} * \text{image}$

Laplacian
Diff of Gaussians (SIFT)
Affine patches (elliptical. two main eigenvec)

  - Feature matching: describe PoIs. - Ransac
    - SIFT (128 dim: 8 orientations, 4x4 histogram array)

- Human detection:
  - HoG
  - Depth-based
  - Thermal: (intensity + gradient)

- Fuse feature extraction of different modalities:
  - early → concatenate
  - late → weighted
  - middle → include complementary feat. in middle steps of models.

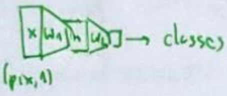
  Iterative ≠ fusion. Multi-modal.

L3. Introduction to CNN

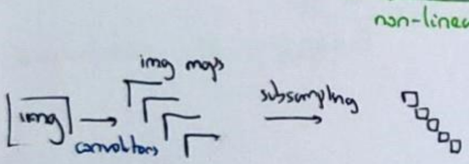
- Linear classifier: $f(x, W) = Wx + b$ → class scores → use softmax (probability dist)
  - x ↓ image
  - x ↓ squeezed ⇒ (nº pixels, 1)

- loss function: how good it currently is. (cross entropy)

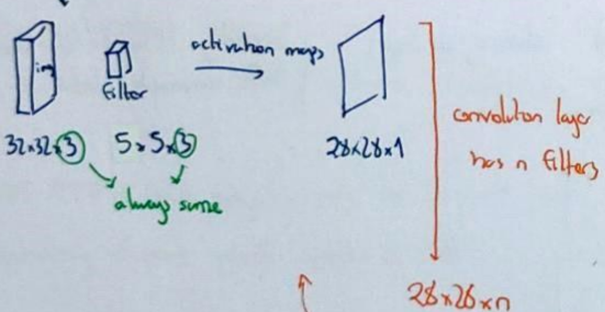
- Neural Networks
  - 1 layer: $F = Wx$
  - 2 layer: $F = \underbrace{W_2 \max(0, W_1 x)}_{\text{non-linear}}$



- CNN:



→ FCC → output → classes (as before)
FCC ↓ stretched x



stride: step taken in convolution's sliding window

zero padding: set borders to 0.

- ConvNet: sequence of convolutional layers:
  - (apply non linearity + pooling in middle)
    - non linearity ↘ activ. functions: relu, sigmoid, tanh, ELU, maxout
  - first layers → low-level features (almost image-like)
  - mid level
  - high level
  - linearly separable classifier

(1x1 conv make sense)

- pooling: down sampling (max, avg)... also sliding window
- weight init:
- batch normalization: after FC, or conv.
- learning rate: start with high and decay with time
- dropout

L4. Detection and segmentation.

- **Semantic segmentation**: label pixels with categories (ignore instances)

idea: use CNN maintaining size

or: ▷◁

↳ unpooling: near. neigh:
$$\begin{matrix}1 & 2\\ 3 & 4\end{matrix} \rightarrow \begin{matrix}1 & 1 & 2 & 2\\ 1 & 1 & 2 & 2\\ 3 & 3 & 4 & 4\\ 3 & 3 & 4 & 4\end{matrix}$$

bed of nails:
$$\begin{matrix}1 & 2\\ 3 & 4\end{matrix} \quad \begin{matrix}1 & 0 & 2 & 0\\ 0 & 0 & 0 & 0\\ 3 & 0 & 4 & 0\\ 0 & 0 & 0 & 0\end{matrix}$$

max unpooling: remember position where max was.

transpose convolution.

- **Object detection**: deep learning applied to $\underbrace{\text{several crops}}_{\text{selective search}}$ of images (to detect of instances) (R-CNN)

  - Single Stage: YOLO, SSD, Retina Net

image $\xrightarrow[\text{div.}]{}$ grid $\xrightarrow[\text{per cell}]{}$ 5 base boxes $\xrightarrow[\text{regress}]{}$ (dx, dy, dh, dw, conf) $\xrightarrow[\text{predict}]{}$ class score ↗ bg included

- **Instance segmentation** Mask RCNN

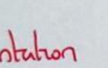
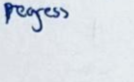
L5. Generative Models

- **Supervised**: learn a function $x \rightarrow y$
- **Unsupervised**: no labels → discover structures (clustering, dim reduction...)

- **PixelRNN**: RNN (LSTM) that computes prob. dist of each pixel ↓→
  CNN: dependency of past pixels modeled by CNN

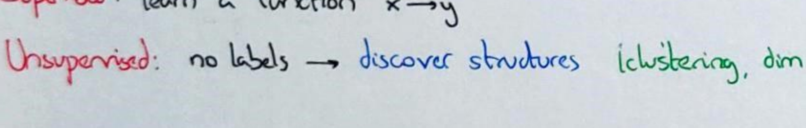
- **Variational Autoencoders**: probabilistic spin to Autoencoders

$$\left.\begin{matrix} & x|z & \\ \mu_{x|z} & & \Sigma_{x|z} \\ & z & \end{matrix}\right\}\text{decoder network}$$

- **GAN**: loss  → train on $\frac{1}{-\text{loss}}$: 

  - can have convolutional architectures

L6. Recurrent Models



img caption — words → sentiment — translation — video class.

idea

hidden state

$h_t = f_w(\underbrace{h_{t-1}}_{\text{past}}, \underbrace{x_t}_{\text{input at time } t})$

f_w → always same

- time can be long: truncate

- LSTM: $\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$ $\quad c_t = f \odot c_{t-1} + i \odot g$, $\quad h_t = o \odot \tanh(c_t)$ $\quad$ input, forget, output, gate

L7. Human pose lecture

L8. Human behaviour in video