11-785

08/06/2020

L10 - CNNs pt II

Story so far

- scanning, redistr., hierarchical.
- deformations in input → max pooling
- 2D scan → convnet
- 1D scan → time delay NN.

(*) CNNs have neurophysiological roots (inspiration)

- Key paper is Hubel & Wiesel paper (1959)

(*) 2 levels of processing:-

i) S-cells (not robust)

ii) C-cells. (fires if majority of S-cells fire)

S-cells → C-cells

S-C layer (simple-complex module)

- Build more complex patterns by composing early neural responses

---

(*) Neocognitron, Fukushima (1980)

- computationalise a revealed vision of H-W.
- Orig issue → position invariance
- grandmother cells fire regardless of position

(*) Neocognitron (v2)

- Hierarchical, successive layers of S-cells and C-cells.
- S-cells respond to signal in previous layer
- C-cells confirm S-cells response.

(*) Neocognitron

S cells → S planes (all cells within S-plane - look resp. for same pattern in a diffent region of input)

- each plane is a detector for one specific pattern

- every one of cells has identical response (within S-plane)

C-cells - behind

C-cells → C-planes

- C planes smaller than S-planes
- Fukushima's original paper
  - S planes examine elliptical regions of input
  - C planes examine elliptical -"- of S-planes
  } used successively
- one C plane per S plane.

(*) Neocognitron (full diagram)

- more complex patterns
- composition/refinement of patterns. (e.g. 2 eyes + nose)

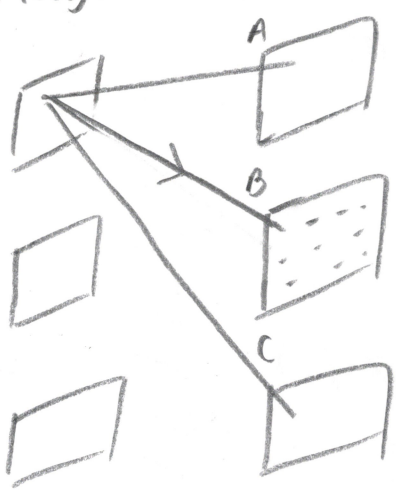
(A1) - Review logic of pattern detection

(*) Neocognitron

- S-cells
- C-cells
  } → complex orig. form.
- But note BR draws similarity of S-cells with RELU
  C-cells with max

(*) Neocognitron (depth, rec. field)

- deeper layer → larger receptive field.



C plane weights S-plane

- C-plane contributes to every S-plane in subs. layer
- Which of A, B, C has highest response
- e.g. B.
- use Hebbian rule
- as B largest
- update weights; but due to fixed response, all weights in other parts of plane updated

(A2) - rev. - slides

(?)

(*) Learning in neocognitron

- unsupervised flavour. → clustering.
- C planes do not need to be estimated

(*) Neocognitron - finale

- Present lots of examples → semantic labels 'automatically learned'

(*) Adding supervision

- Temporal corr.: Homma, Atlas, Marks '88
- TDNN: Lang, Waibel et al. 1989, 1990. (time delay NN)
- CNN: Yann LeCun.

- Addition of supervision
- Final output

(*) Supervising neocognitron

- Add extra layer after final C layer → c-l output.
- BP: Fully feedforward MLP with shared param
  - All S-cells within S-plane share weights.
- Backprop → estimate S-cell weights in every plane of every layer
- C-cells not updated.

(*) Scanning vs multiple filters / layer

i) square recept. fields

ii) S-cells RF → $K_l \times K_l$

iii) C-cells RF → $L_l \times L_l$

(*) Supervising NC

- include form to include exp.

$$u_{S,l,n}(i,j) = \varphi\left(\sum_{f}\sum_{k=1}^{K_l}\sum_{t=1}^{K_l} w_{S,l,n}(p,k,t)\, u_{S,l-1,p}(i+t-1, j+k-1)\right)$$

$$u_{C,l,n} = \max_{k \in (i, i+L_l),\, j \in (l, l+L_l)} \left(u_{S,l,n}(i,j)\right)$$

(*) weights $w_{s,i,n}$ are position independent (between cross-plane)?
and do not change based on which portion of image you are analysing

(*) weights - corresponding region of image
w u
- imposing weights on top of image
- compute pixel by pixel pairwise product
- summing over all places
- Yielding combined response at one particular point
- Put through activation (sigmoid') → S-cell output
- C-cell looks at rectangular patch of S-cells; ~~use~~ applies max
(A3) - Tie this intuition together with the equations

(*) Instead of having many copies of neuron looking at every region of the image, it is possible instead to have one neuron scanning
(*) Scanning with a neuron → 'convolution'

(*) CNN
- ATOT LENET → Turing Award
- Revolutionary → USPS.

(*) Story so far
- Good summary
- visual cortex → neocognitron → external sup. → CNNS

(*) CNNS
- started as a computational model of a biological phenomenon (!)
- (W): A cybernetic development → instantiation of a mechanism

---

(*) General architecture of CNN

- successive convolutional, downsampling layers
(*) convolutional - neurons that scan inputs for patterns.

(*) downsampling → perform max operations on groups of outputs from convolutional layers

- may occur in sequence, but typically alternate.

(*) followed by MLP with one or more layers (final)

(*) doesn't usually make sense for downsample to be followed by downsamp.
↳ one large downsample op.

(*) downsamples (max) reduce size of input (map).

(*) MLP operating on flattened vesion of the outputs of the final layer of neurons.

(*) (*) GA of CNN - learnability

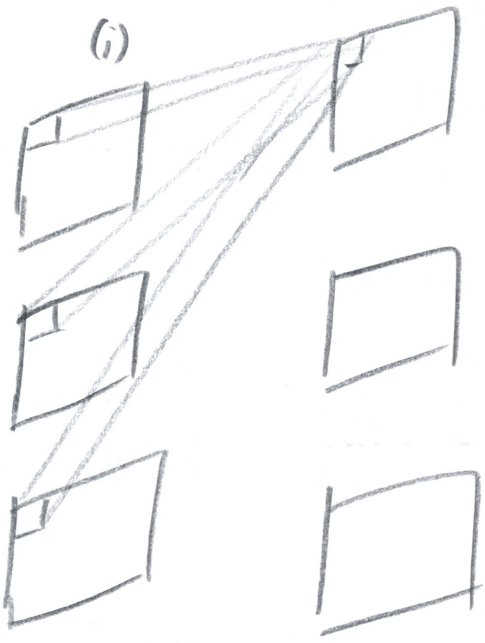
- convolution layers and MLP are learnable
  i.e. params estimated from training for target class. task
- downsampling layers
  - no learnable parameters.

(*) Convolutional layer (CL)

- CL - series of maps
  - analagous to NC S-planes
  - receive/act. maps

(*) CL

- within each map - pair of operations
- i) affine combination
  - A linear map obtained by convolution on maps in the previous layer
  - each affine map assoc. with a learnable filter



ii) Activation that operates on convolution

(**) - see diagram (excellent)

- CL → affine + activation (similar to MLP)

(*) If you want to compute affine combo at any one location, it is done so over all maps from previous layer (97)

(*) All maps in previous layer contribute to each convolution

---

(*) What is a convolution?

- consider contr. of single map (brown patch)

(*) Examine how brown patch contributes to one point in the corresponding map

(*) Weights are binary in pedagogical example (1/0)

- green → patch from previous layer.
- orange → place filter on patch

input map ; filter ; convolved feature

↓ values ↓ values

(*) Take affine combo of filter values with input map values for a particular region

(affine combo)

(*) See slides for a visual demo (!) (97).

(*) Stride > 1 possible → smaller output map

(*) What really happens.

- (*) See slides → small grey patch is a result of computation over multiple input maps simultaneously.

interactive.

- 3x3 convolved feature ⟶ 9 weights, 1 bias.

(94) - match visual display to formulae - crucial

(*) 1st map, 2nd map

- weights for 1st, 2nd map very different.

(57) - weights, activ. formulae

(*) A different view

- stacked arrangement of planes
- generalise to cubes / cuboids

(57): key formula to anchor intuition visually:-

$$z(s,i,j) = \sum_{p} \sum_{k=1}^{k} \sum_{\ell=1}^{k} w(s,p,k,\ell)\, y(p, i+\ell-1, j+k-1) + b(s)$$

(I)

(II)

(I) - pixel by pixel product, then summing over → (has specific weights)

(II) - for every patch (over planes)

- computation of convolutional map at any location sums the convolutional outputs at all planes

(*) filter (cube) - has different values at every single location

- weights at one plane not identical to weights on all other planes

---

(*) CNN - vector not.

pseudocode

- colour convos.
- weight $\underline{W}(\ell,j)$ is a 3D $D_{\ell-1} \times K_\ell \times K_\ell$ tensor (ass. sq RFs)
- prod. in blue → tensor inner prod. with scalar output

$\underline{Y}(0)$ - image

FOR $\ell = 1 : L$

FOR $j = 1 : D_\ell$ # layers op. on vector at (i,j)

yellow convolution

```
FOR x = 1 : W_{ℓ-1} - K_ℓ + 1
  FOR y = 1 : H_{ℓ-1} - K_ℓ + 1
    segment = Y(ℓ-1, :, x : x+K_ℓ-1, y : y+K_ℓ-1)   - 3D tensor
    z(ℓ,j,x,y) = W(ℓ,j) . segment                    # tensor inner prod.
```

$\underline{Y}(l,j,x,y) = \text{activation}(\underline{Z}(l,j,x,y))$

$y = \text{softmax}((\underline{Y}(l,:,:,:))$

(*) Engineering consid.

- stride of conv. can be greater than 1 pixel.
- size of output of convol. op depends on implementation
- no need for it to be same size as input

(**) Essentially examine constraints of problem

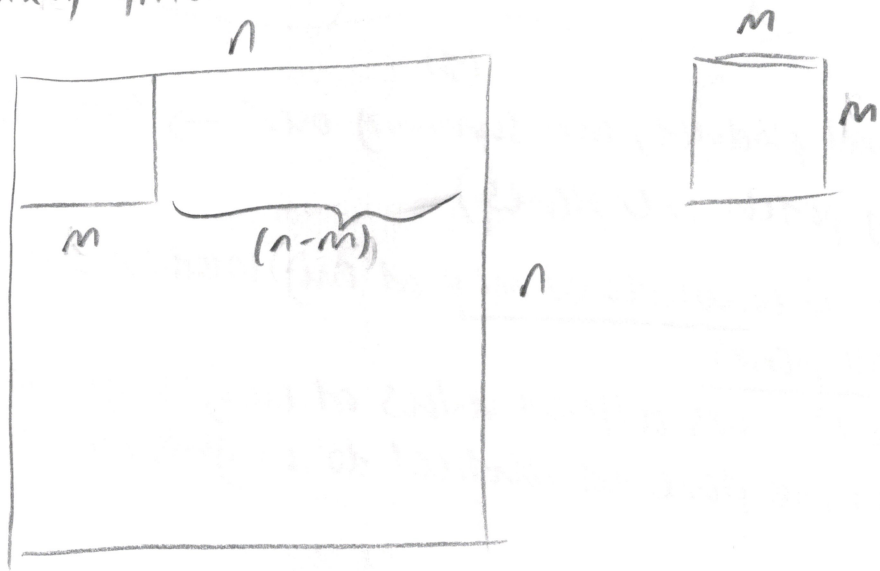
↳ does filter 'fall off edge'?

(*) convolution size

(*) In general for an $(n \times n)$ image / input map

$(m \times m)$ filter

output will be :-



output size: $1 + \left\lfloor \frac{(N-M)}{S} \right\rfloor$ - no. of pixels in output in each dim.

→ Reduction in output size

- even if $S=1$

(*) often; this is not acceptable — want to maintain size to keep track of proportions

(S): Hence allow filter to 'fall off the edge' → systematic way.

- Assume stride $S=1$, how many rows are 'lost' when going from left to right? $N-(N-M+1) = (M-1)$ rows

(AS) review calc.

- Hence $\frac{(M-1)}{2}$ - pad this many columns of 0s to left and right

- output size = input size - regardless of stride
- After 0 padding

(*) Padding (A5)

(A6) review skipped slides

(*) Activation

HW2 → Relu activ.

(*) Other component

(A7) - review.

- downsampling / pooling (follows convolution)
- maxpooling → selects logest from a pool of elements
- pooling performed by 'scanning input' (?)
- (*) looks at groups of pixels from output map (?)
- (*) one downsampling corresp. to every output map
  - one C for every S
  - C is looking at one S; not across all Ss
- (*) max strides by more than one pixel typically. (to account for jitter).

(*) Pooling - size of output

(*) See slides (!)
- interactive. (similar principle)

- NxN picture compressed by a PxP pooling filter with stride D results in an output map of side :- $\left\lfloor \frac{N-P}{D} \right\rfloor + 1$.

- typically do not zero-pad

(*) All - mean pooling

(*) Purpose of max is to insure jitter and variance
- If petal location moves a little, should still detect petal (?) (shift-invariance jitter distinction?)

- compute mean instead.
- or softmax
  or p-norm.

(*) Other options

- pooling may be a learned filter (?) ← (18) - review quick.

(*) 'All convolutional' net

- downsampling - achieved with conv. layer with stride > 1 (?)
- softmax instead of max → standard convolutional

(BR): - these days → convolutions all the way through

(*) Fully convolutional network (no pooling).

pseudocode

(A9) - review.

(*) Setting everything together (engineering / actual task)

- image scaling → often contingent / informed.

(*) CNN -

- parameters to choose: K_1, L, S
  1. No. of filters → K_1
  2. Size of filters → $L \times L \times 3$ + bias
  3. Stride of convolution → S

} total no. param

3 colors

(?)

$K_1(3L^2+1)$

↓ need to be est. as part of training.

↑ design choice (not param.)

(A10) - review equations
'RGB' CNN
- i) convolution
- ii) pooling

- max pooling
  ↳ keep track of which position had highest value
  (since max res picked up from)
  ↳ required for backprop

- i.e. reg. of max.

(*) note CNN jargon - kernels/channels

(*) A11 - CNN's

- diagram, equations review
- 1st pooling 2nd conv. layer

(*) size of the layers.

A12 - review

- summarises no maps, sizes of I/O maps. no.
(*) in general, no. of conv. filters increases with layers no.:.
- going through layers → maps get smaller and smaller, maps ↑, no. filters/layer ↑
- to retain info; want size of them
- e.g. original 3 colour, 1024 × 1024 pixels
  ↳ 3 million values
  - retain all info.
    - no. of channels × outmap of each channel has to be 3 million
    - else info loss
(*) information loss in general a good thing acc. to BR

(*) learnt to choose (design choices)

- lots of design choices, but not daunting

(*) training

- don't be put off
- use one giant MLP
- shared private networks
- pool derivatives

(*) only difference is structure of network

- next class in depth

(*) convolution does not change anything

(*) learning CNN

(B). review

(*) Note connect/refesh

training loss - divergence

persistance

- etc.