CS 182 lecture 6: Convolutional Networks

→ many useful image features are local

convolution (filter + features)

↳ apply a "mini-layer" (aka filter) to process patches of image.
↳ "downsample" activation grid after filtering (aka pooling).
→ typically: MaxPooling2D.

→ historical example: LeNet, handwritten digit recognition.

→ a way of avoiding needing millions of parameters w/ images
each layer is "local", produces an "image" w/ (roughly) the same width and height. # of channels = # of filters.

N-dim (ND) arrays ⟷ tensors

this is the feature @ l,m. channel

Conv: (F_h, F_w, C_i, C_o)

$$z^{(2)}[i,j,k] = \sum_{l=0}^{H_F-1} \sum_{m=0}^{H_W-1} \sum_{n=0}^{C_{in}-1} W^{(2)}[l,m,k,n]\, a^{(1)}\left[i+l+\frac{(H_F-1)}{2},\ j+m-\frac{(H_W-1)}{2},\ n\right]$$

$$z^{(2)}[i,j] = \sum_{l=0}^{H_F-1} \sum_{m=0}^{H_W-1} W^{(2)}[l,m]\, a^{(1)}\left[i+l-\frac{(H_F-1)}{2},\ j+m-\frac{(H_W-1)}{2}\right]$$

$a^{(2)}[i,j,k] = \sigma(z^{(2)}[i,j,k])$ apply non-linearity.

Input: 32 × 32 × 3
Filter: 5 × 5 × 6
Output: 28 × 28 × 6

$$H_{out} = H_{in} - \frac{H_F-1}{2} \times 2 = H_{in} - H_F + 1.$$

A way to tackle padding w/o losing dimensionality: zero-pad

Strided convolutions: skip over some positions w/ our filter. Saves computation.
→ ppl argue: just as good as conv + pooling.

Examples:

→ AlexNet. obtained SOTA results on ILSVRC (work of art), 8 layers
trained on two GPUs!
Conv 1: 11×11×96 Stride 4, maps 224×224×3 → 55×55×96 (no zero pad)
Pool 1: 3×3×96, Stride 2, maps 55×55×96 → 27×27×96

more features, more things could be "going on"

Norm 1: Local normalization layer (not widely used anymore)
Conv 2: 5×5×256, Stride 1, maps 27×27×96 → 27×27×256 (w/ zero pad)
Pool 2: 3×3×256, Stride 2, maps 27×27×256 → 13×13×256
Norm 2: local norm layer
Conv 3: 3×3×384, Stride 1, maps 13×13×256 → 13×13×384 (w/ zero pad)
Conv 4: 3×3×384, Stride 1, maps 13×13×384 → 13×13×384 (w/ zero pad)
Conv 5: 3×3×256, Stride 1, maps 13×13×384 → 13×13×256 (w/ zero pad)
Pool 3: 3×3×256, Stride 2, maps 13×13×256 → 6×6×256
FC6: 6×6×256 → 9216 → 4,096 [matrix is 4096 × 9216]
FC7: 4096 → 4096
FC8: 4096 → 1000
Softmax

tradeoff, features ↑ res ↓

ReLU every conv/FC layer but last

19 layers

→ VGG (work of engineering), still often used today, homogenous stacks of convs.
↳ more layers, more processing, bulk of parameters in FC1...

→ ResNet (152 layers!), longer stacks of convolutions
↳ no large FC layer at end, just average pool last conv. results in one linear layer, reduces parameters
↳ implements residual (skip) connections: $H(x) = F(x) + x$ (add input to output) ← 2 layers
(good gradient flow)

helps stabilize $\frac{dL}{dx}$ neither 0 nor ∞!

CS 182 lecture 7: Initialization, BatchNorm 6/15

$\frac{dL}{dW^{(1)}} = \frac{dz^{(1)}}{dW^{(1)}} \frac{dL}{dz^{(1)}} = \delta x^T$ ← if x is "imbalanced", so are gradients.
Therefore, we really want all entries in x → same scale.

standardization: $\mu = 0$, $\sigma = 1$.

$\bar{x}_i = x_i - E[x]$ $\bar{x}_i = \frac{x_i - E[x]}{\sqrt{E[(x_i - E[x])^2]}}$ ← $\bar{\sigma}_i$ $E[x] \approx \frac{1}{N} \sum_{i=1}^{N} x_i$

Basic idea of Batch Norm: standardize activations at each layer, controlling gradients.
computationally expensive idea, so only perform on batch.

$\mu^{(1)} \approx \frac{1}{B} \sum_{j=1}^{B} a_{ij}^{(1)}$ $\sigma^{(1)} \approx \sqrt{\frac{1}{B} \sum_{j=1}^{B} (a_{ij}^{(1)} - \mu^{(1)})^2}$ $\bar{a}_i^{(1)} = \frac{a_i^{(1)} - \mu^{(1)}}{\sigma^{(1)}} \gamma + \beta$ ← learnable scale and bias, same dim as $\bar{a}_i^{(1)}$

→ Can be trained via backprop since these are differentiable.
→ can be placed either before/after nonlinearity.
→ we can often use a larger learning rate
→ models can train faster
→ generally requires less regularization

Basic initialization methods: ensure activations are on reasonable scale that stays roughly constant
More advanced init. methods involve eigenvalues/Jacobians
"Try" to have well-behaved gradients from the get-go

set $W_{ij} \sim N(0, \sigma^2_W)$, $b_i \approx 0$, $z_i = \sum_j W_{ij} a_j + \cancel{b_i}$ assume $a_j \sim N(0, \sigma_a)$

so, set std of $W_{ij} = 1/\sqrt{D_a}$ $E[z_i^2] = \sum_j E[W_{ij}^2] E[a_j^2] = D_a \sigma^2_W \sigma^2_a$ — D_a: dim of a

"Xavier initialization" if $D_a \sigma^2_W > 1$, magnitude grows w/ each layer
ReLU issue: zeros out activations. If $D_a \sigma^2_W < 1$, magnitude shrinks w/ each layer → $\sigma^2_W = \frac{1}{D_a}$
Ergo, increase std of W_{ij} → $1/\sqrt{\frac{1}{2} D_a}$ (proposed in ResNet).
Alternative init. $b_i = 0.1$ (or small constant) to avoid any zero-out in ReLU.

Reminder: $\frac{dL}{dW^{(1)}} = J_1 J_2 J_3 \cdots J_n \frac{dL}{dz^{(n)}}$ $\forall \lambda_i$, $J_i = U_i \Lambda_i V_i$ (SVD). Λ_i: diag. matrix of λ's of J_i.
By ensuring Λ_i's are scaled, we avoid 0/infinity convergence issue.
Since $J = W^T$ (derivative is W) → $W^{(i)} = U^{(i)} \Lambda^{(i)} V^{(i)}$, force $\Lambda^{(i)} = I$, $W^{(i)} \leftarrow U^{(i)} V^{(i)}$

"Measure of last resort": Gradient clipping, because "monster gradients" can occur
→ per-element clipping: $\bar{g}_i \leftarrow \max(\min(g_i, c_i), -c_i)$
→ norm clipping: $-c_i < g_i < c_i$
$\bar{g}_i \leftarrow g \frac{\min(\|g\|, c)}{\|g\|}$ ← clips length, not dimension; choose c_i thru experimentation, see what "healthy" magnitudes look like (plot).

NNs → high variance (lots of parameters). Set with multiple, more agreement on "right"
Variance $= E_{D \sim p(D)}[\|f_D(x) - \bar{f}(x)\|^2]$ $\bar{f}(x) = E_{D \sim p(D)}[f_D(x)] \approx \frac{1}{M} \sum_{i=1}^{M} f_{D_i}(x)$ ← bootstrap sample

Select? → principled approach: average $(p(y|x) = \frac{1}{M} \sum_{j=1}^{M} p_\theta(y|x)$, often more robust)
→ simple approach: majority vote (analogous to "First Past The Post")

Ensembling in practice: train M models $p_{\theta_j}(y|x)$ on the same D training set, maj. vote.
Even faster ensembles? Share features for all M models. Separate ensemble of final classifier "heads" at end (these are task-specific)
Snapshot ensembles: save our parameters as "snapshots" during SGD, use later as a model.
The bigger the ensemble, the better, the more expensive.
Dropout: randomly set some activations to 0 in forward pass. Creates a "new" network, helps w/ ensembling out of a single NN. 2^{km} diff. models (k layers, m nodes/layer)
At test time: $W^{(i)} = \frac{1}{2} W^{(i)}$, since on average, $\frac{1}{2}$ of dimensions are forced to 0.
Hyperparameters that affect generalization (validation): ensembling, dropout, arch.
General method to pick hyperparameters? coarse-to-fine, broad sweep before "zeroing in"
↳ In practice: random hp. search ≥ grid search.