Lecture 8: Convolutional networks

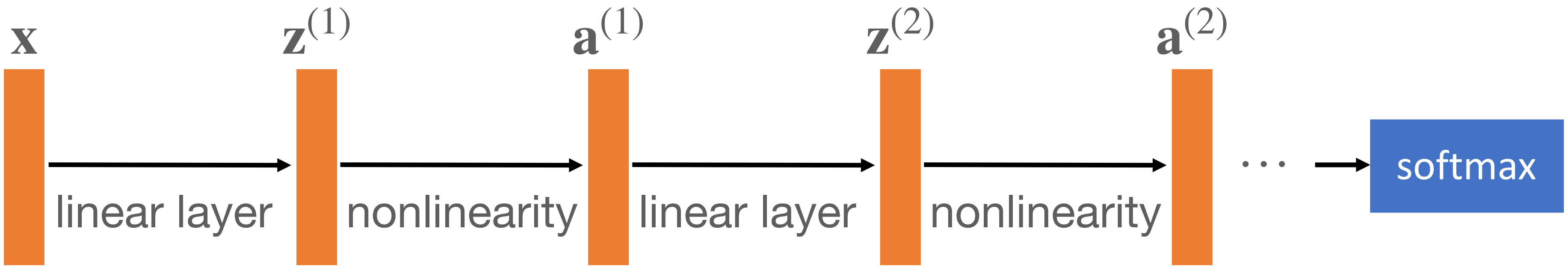
CS 182/282A ("Deep Learning")

2022/02/14

Today’s lecture

- Today, we take a detailed look at the most widely used class of neural network models for image based problems (**computer vision**)
- **Convolutional neural networks (conv nets)** can be used for other applications
  - Conversely, other types of neural networks can be used for computer vision
  - However, more often than not, conv nets and computer vision go “hand in hand”
- We will cover the motivation behind conv nets, detail the mathematical formulation, and cherrypick the last decade of developments in conv net architectures
  - On Wednesday, Prof. Jitendra Malik will give a guest lecture on computer vision!

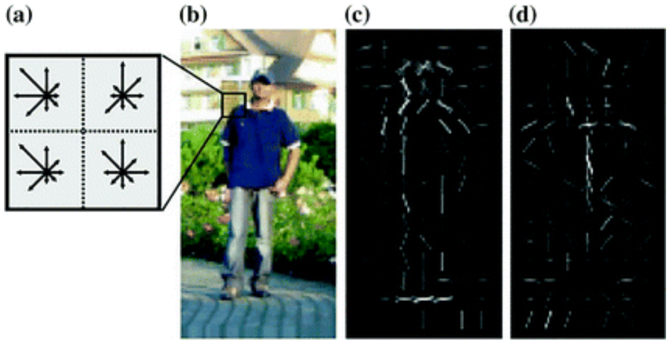
Fully connected layers for processing images?



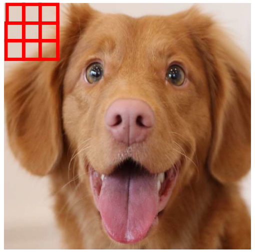
- $\mathbf{x}$ is an image, e.g., 224 (height) $\times$ 224 (width) $\times$ 3 (RGB) $= 150528$ dims
- Let's make $\mathbf{z}^{(l)}$ modest, e.g., 128 dims (in reality, this is probably too small)
    - Then, we have 150528×128 ($\sim$ 20M) parameters in the first layer

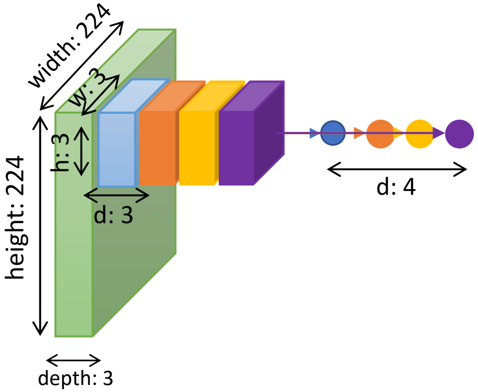
The key idea behind conv nets

- The key idea behind reducing the massive number of parameters is the observation that many useful image features are **local**
    - E.g., edge information, used by many once-popular hand designed features
- We won't go so far as to hand design the features, but we will place limits on the features that can be learned via the architecture
    - Inductive biases at work
- You might be wondering: surely there is useful nonlocal information as well? More on this later…



“Locally connected” layers for processing images

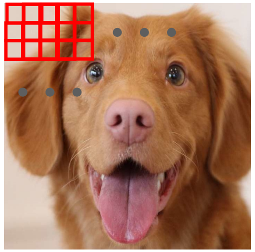


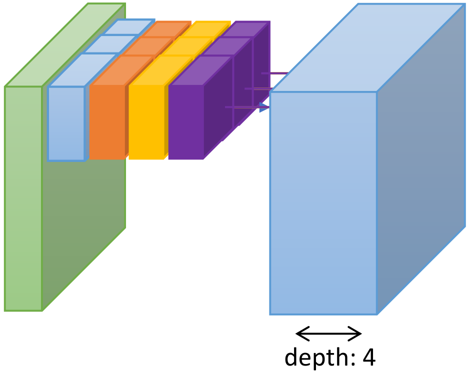


filter

- Before, we had ~ 20M parameters. How many parameters do we have now?
- The filter consists of 4 tensors each with $3 \times 3 \times 3 = 27$ parameters, so we have 108 parameters — if we add a bias term to the output, 112 parameters
- Wait, but, we haven’t yet processed the whole image!

“Sliding” the filter along the image





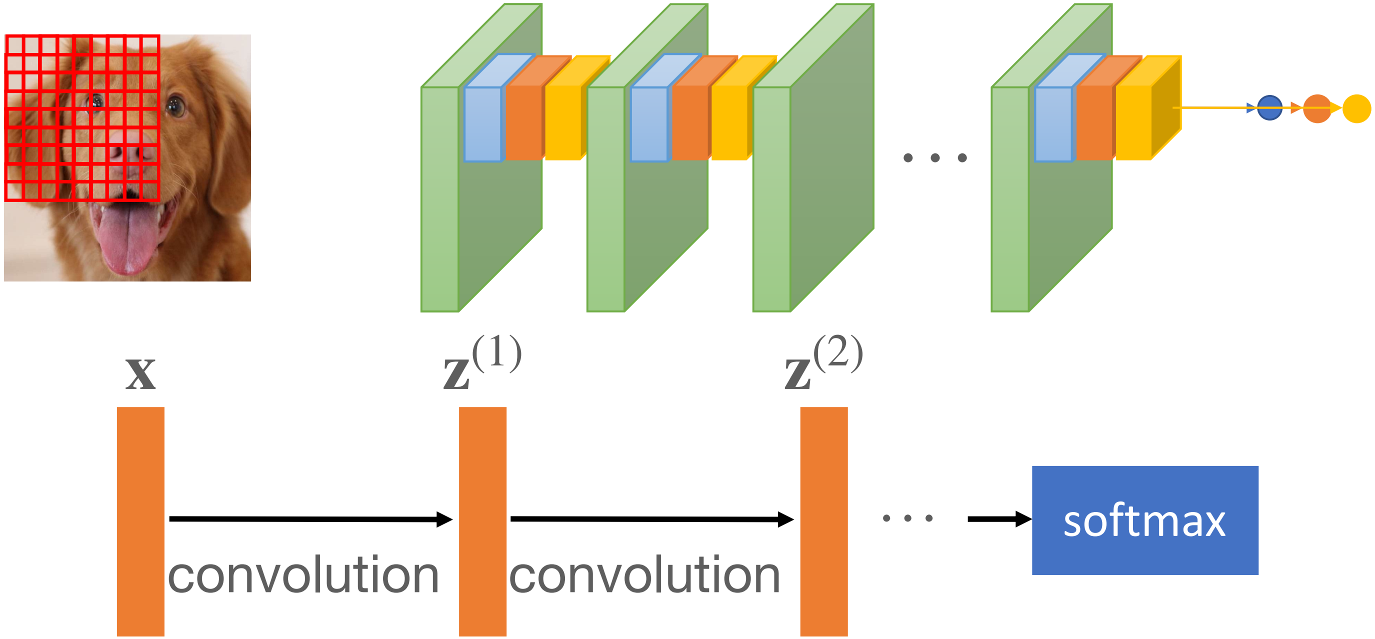
- We process the whole image using the same filter — so, in the end, we will still have only 112 parameters
- What will our output look like?
  - It actually looks quite like an “image” itself… interesting…

The convolution layer

- The processing step we have described is referred to as **convolution**
    - Convolution is performed with a filter — a tensor with dimensions $[K, K, O, I]$ (e.g., $[3, 3, 4, 3]$) — and a O-dimensional bias term
- (2D) convolutions take in an input of size $[I, H, W]$ (or $[H, W, I]$, depending on the convention) and output a tensor of size $[O, H', W']$
    - What are H' and W'? It depends on certain hyperparameter values
- Because the output has similar dimensions, we can stack convolutions on top of each other to make *deep convolutional networks*

Stacking convolutions

- Stacking convolutions increases the *receptive field* the deeper we go

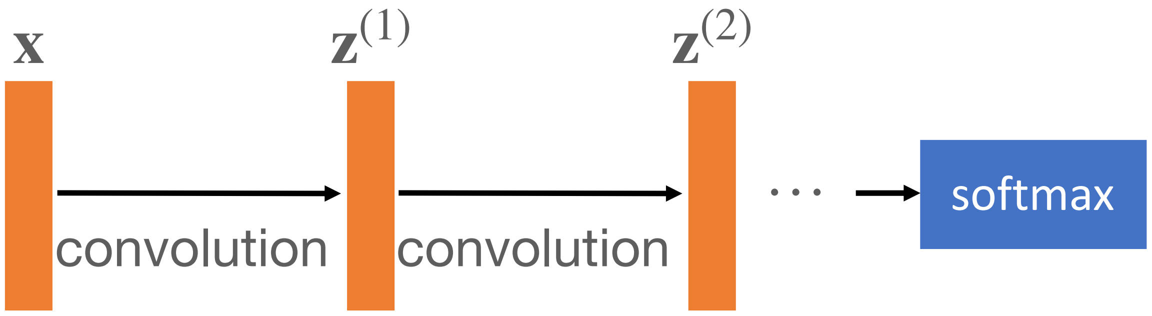


Determining H' and W'

- Two important hyperparameters determine the size of the convolution output
- First, we can choose to **pad** the input by a certain number of “pixels” on all sides
  - Most common choice: pad with zeros (make sure to use normalization)
- Second, we can choose the **stride** that the filter shifts by, i.e., how many “pixels” it moves over every time
- For a $K \times K$ filter, we will have $[H', W'] = 1 + ([H, W] + 2 \times \text{pad} - K)/\text{stride}$
- It is common to choose a stride of 1 and (in total) pad by the size of the filter minus 1 (e.g., 1 on all sides for a 3×3 filter) such that $H' = H$ and $W' = W$

Convolutional networks: attempt #1

- Can we just stack convolutions on top of each other?
  - What's the issue with this?
  - Convolution is a linear operator!



Introducing nonlinearities

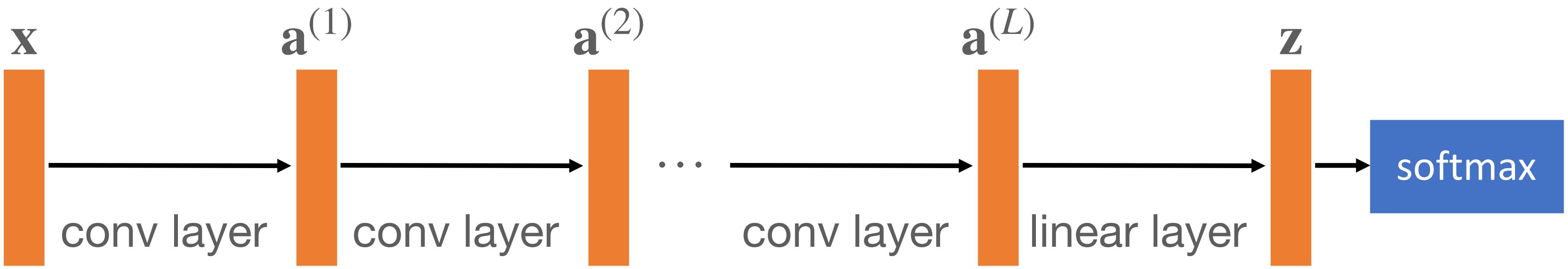
- Just like before, we will interleave our linear layer (convolution) with a nonlinearity, e.g., ReLU, applied element wise to the output of the convolution
- Sometimes, the term "convolution layer" is used to refer to the popular recipe of convolution → BN → ReLU (but it could also refer to just the convolution part)
  - Like input standardization for images, BN on inputs of shape $[N, C, H, W]$ compute statistics on the N, H, and W dimensions, rather than just N
  - If using LN instead, we compute statistics on the C, H, and W dimensions

Pooling

- Another common operation in convolutional networks is **pooling**, which reduces the size of the input and possibly the number of parameters later in the network
- Pooling uses a *window size* (typically, 2×2) and a stride (typically, whatever the window size is) and slides over the input as specified by these hyperparameters
  - **Max pooling** "lets through" only the largest element — a nonlinear operation
  - **Average pooling** averages all the elements in the window — this is linear
- The output of the pooling layer, with a 2×2 window size and stride of 2, will be one quarter the size of the input

Convolutional networks: attempt #2

- A simple convolutional network repeats the convolution $\rightarrow$ BN $\rightarrow$ ReLU recipe L times to process the input image into a representation $\mathbf{a}^{(L)}$
- We *flatten* or pool $\mathbf{a}^{(L)}$ into a one dimensional vector, pass it through one or more linear layers, and then (for classification) get our final probabilities with softmax



Convolutions in math

Convolutions, the "forward" direction

consider processing $\mathbf{a}^{(l)}$ ($[H, W, C]$) into $\mathbf{z}^{(l+1)}$ ($[H', W', C']$) using a convolution

our filter $\mathbf{W}^{(l+1)}$ has shape $[K, K, C', C]$

$$z^{l+1}[i,j,k] = \sum_{a=0}^{K-1} \sum_{b=0}^{K-1} \sum_{c=0}^{C-1} W^{l+1}[a,b,k,c]\, a^{l}[i+a, j+b, c]$$

matrix vector notation:

$$z^{l+1}[i,j] = \sum_{a=0}^{K-1} \sum_{b=0}^{K-1} W^{l+1}[a,b]\, a^{l}[i+a, j+b]$$

C'-dim vector

C'×C matrix

C-dim vector

Convolutions, the “backward” direction

for simplicity, let’s assume that $C' = C = 1$

so $\mathbf{z}^{(l+1)}[i,j] = \sum_{a=0}^{K-1} \sum_{b=0}^{K-1} \mathbf{W}^{(l+1)}[a,b] \times \mathbf{a}^{(l)}[i+a, j+b]$

$$\frac{\partial z^{l+1}[i,j]}{\partial a^{l}[i+a, j+b]} = W^{l+1}[a,b] \text{ for } a=0 \text{ to } K-1, \; b=0 \text{ to } K-1$$

$$\frac{\partial \ell}{\partial a^{l}[i,j]} = \sum_{a=0}^{K-1} \sum_{b=0}^{K-1} W^{l+1}[a,b] \frac{\partial \ell}{\partial z^{l+1}[i-a, j-b]}$$

convolution!

Convolutions, the “backward” direction

$$\mathbf{z}^{(l+1)}[i,j] = \sum_{a=0}^{K-1} \sum_{b=0}^{K-1} \mathbf{W}^{(l+1)}[a,b] \times \mathbf{a}^{(l)}[i+a, j+b]$$

$$\frac{\partial z^{l+1}[i,j]}{\partial W^{l+1}[a,b]} = a^{l}[i+a, j+b]$$

$$\frac{\partial \ell}{\partial W^{l+1}[a,b]} = \sum_{i=0}^{H'-1} \sum_{j=0}^{W'-1} a^{l}[i+a, j+b] \frac{\partial \ell}{\partial z^{l+1}[i,j]}$$

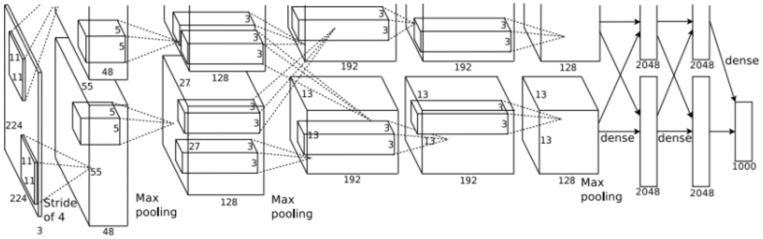
convolution!

The last decade in conv nets (sort of)

AlexNet

Krizhevsky et al, 2012

- The model that started the past decade of deep learning hype
  - Demonstrated the power of combining expressive models with lots of compute
- Widely known for being the first neural network to attain state-of-the-art results on the ImageNet large scale visual recognition challenge (ILSVRC)



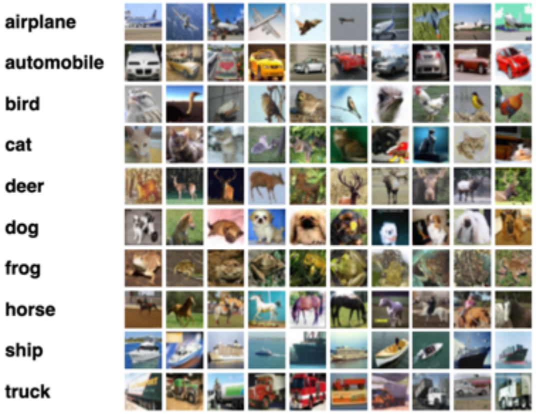
ImageNet image classification

- ImageNet consists of $224 \times 224 \times 3$ images evenly covering 1000 classes
  - There are 1.2M training images and 50000 evaluation images
- ImageNet-22K is a larger version of ImageNet (roughly $10\times$ larger) with 22000 classes, increasingly used these days due to expanding compute budgets
- It is common for computer vision applications to start from a network **pretrained** on ImageNet

Smaller image classification datasets

MNIST, CIFAR-10, and CIFAR-100

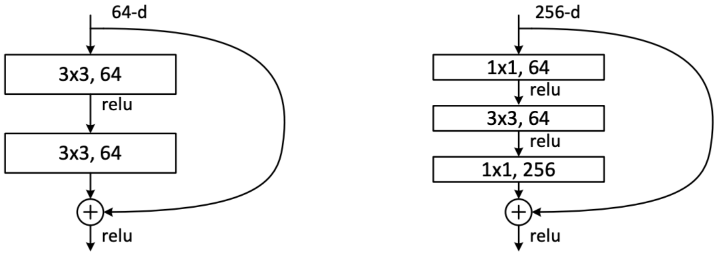
- Working with ImageNet is not a good way to prototype
- MNIST, CIFAR-10, and CIFAR-100 are much smaller datasets that increase in difficulty in that order
    - But they're all much easier than ImageNet
- MNIST: $60000/10000$ train/test, 10 classes, $28 \times 28 \times 1$ grayscale images
- CIFAR-*: $50000/10000$, * classes, $32 \times 32 \times 3$ color (RGB) images



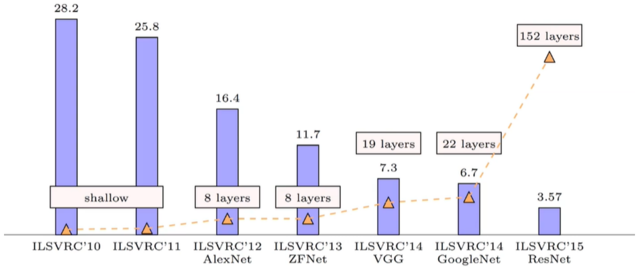
Skip connections in convolutional networks

He et al, 2015

- Recall the general idea behind skip connections: $\mathbf{a}^{(l)} = \sigma(\mathbf{z}^{(l)}) + \mathbf{a}^{(l-1)}$
- This idea was popularized by **residual networks (ResNets)**, a convolutional architecture that implemented the idea slightly differently (and in two ways)
- This allowed for better training of deeper networks, which are more performant



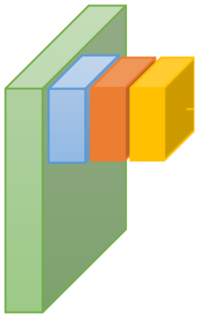
He et al, 2015

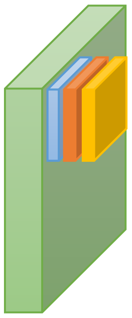


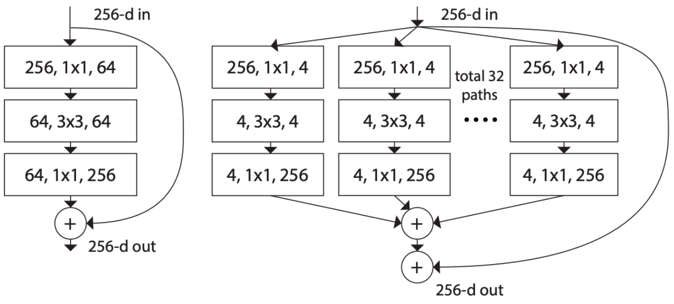
Depth wise (or grouped) convolutions

E.g., Xie et al, 2016

- In **depth wise** (resp. **grouped**) **convolutions**, the filter and input are split by channels (resp. groups of channels), convolved separately, then concatenated
- We can increase the number of channels and maintain roughly the same computational complexity with this technique, and performance often improves



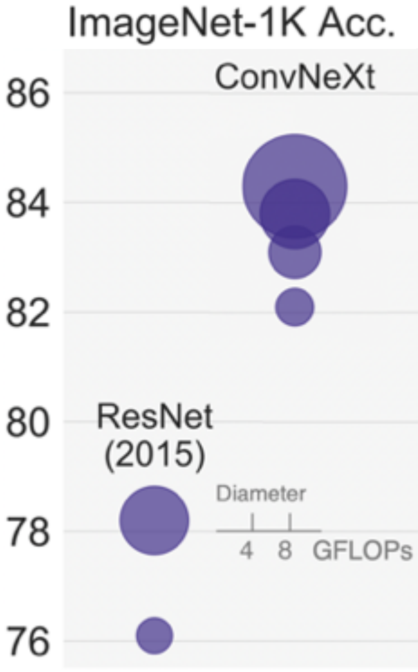




A recent state-of-the-art example

Liu et al, 2022

- ConvNeXt is a recent state-of-the-art conv net that aggregates several methods to achieve improved performance
- Improved training techniques (cosine learning rate schedule, AdamW, lots of data augmentation) turn out to help significantly
- Using depth wise convolutions (and proportionally increasing the number of channels) also significantly improves accuracy
- Some other changes, such as swapping BN for LN and swapping ReLU for GELU, provide smaller gains but appear to not be as important



Beyond image classification: MS COCO

- MS COCO is a large scale dataset that defines other computer vision tasks
- Includes labels for object detection (localization), segmentation, key point detection, and captioning
- 118000 training images, 5000 validation images, 41000 test images, 123000 unlabeled images for object detection
- Prof. Malik will tell you more about this on Wed.