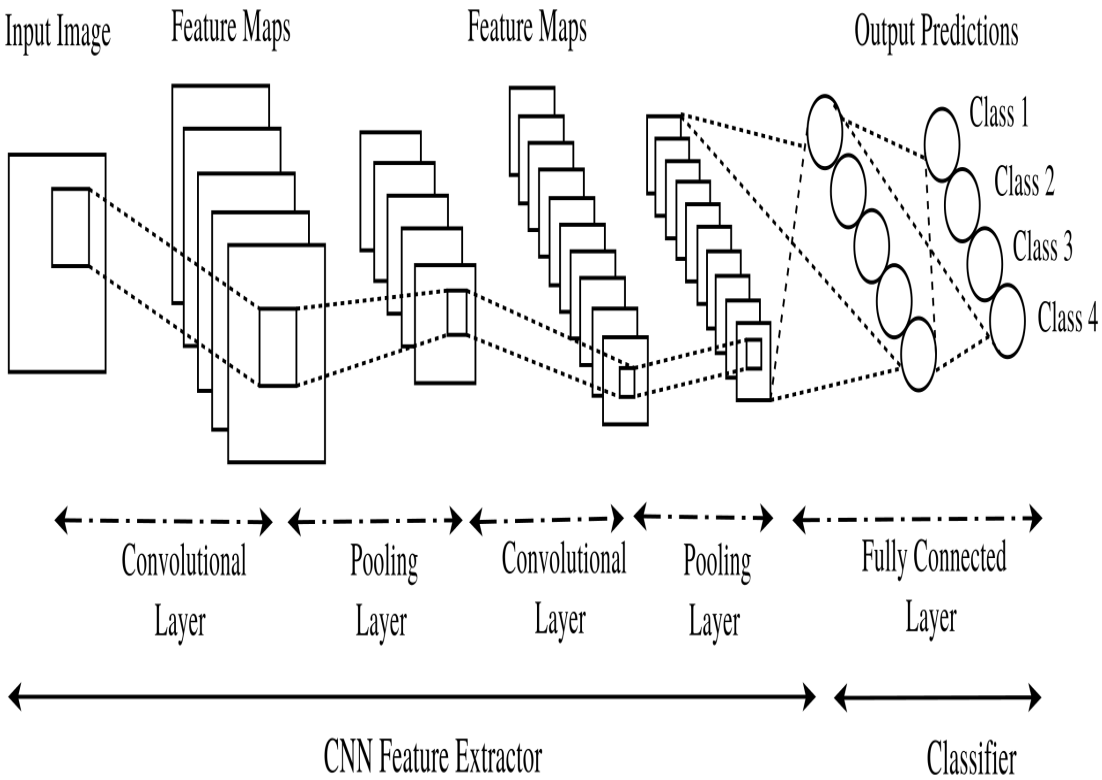
Recap

- We have been considering CNNs
- The special features of CNNs include: local connectivity, weight sharing, multiple features/filters
- Typical components of a CNN: convolutional layers (including nonlinear layer), pooling layers, fully connected layers.
- They may also include normalization layers.
- We considered CNNs mainly for object recognition. They are used in many other image processing problems too (e.g., detection, localization, segmentation).

Recap - A Typical CNN Architecture



Recap

- CNNs can be learnt using stochastic gradient descent, usually, with a cross entropy loss.
- The basic idea of backpropagation is used for gradient computation.
- CNNs have large number of parameters and generally use large training sets.
- One uses regularization such as weight decay, dropout.
- One also uses tricks such as data augmentation to get better generalization performance.
- There are also many hyperparameters to fix.

- Fixing a CNN architecture involves many issues.
- Each convolutional layer is characterized by number of filters, size of filters and stride.
- Each such layer has one weight tensor.
- The spacial extents of different convolutional layers is now determined by whether or not to we use zero-padding.
- We also need to fix details of fully connected layers.
- There are many standard architectures (e.g., Alexnet, VGGNet, Resnet etc).

Some Example CNNs

- Lenet or Lenet5 – tested on MNIST data.
  - 2 convolutional layers, each followed by a pooling layer and three fully connected (FC) layers
  - 1st Conv layer: 6 filters with 5×5 kernel; followed by 2×2 with stride 2 average pooling.
  - Image size: 28×28. Padding by 2 so that size of 1st layer is same. After pooling it becomes 14×14.
  - 2nd Conv layer: 16 filters with 5×5 kernel; followed by 2×2 with stride 2 average pooling
  - No padding. Size of 2nd layer is 10×10. After pooling, it becomes 5×5.
  - So, number of input nodes for the first FC layer would be 400 $(= 25 * 16)$
  - There are 3 fully connected layers with sizes 120, 84, 10 (This is a 10-class problem)
  - Lenet used sigmoidal activations and weight decay regularization

Alexnet

- Alexnet has 5 convolutional layers and three fully connected layers.
- Conv-1: 96 filters, 11×11 kernel, stride 4
- Conv-2: 256 filters, 5×5 kernel
- Conv-3,4,5: 384 filters, 3×3 kernel
- The three FC layers have 4096, 4096, 1000 units
- Max pool layers with 3×3 size and stride-2 are used after the first, second and fifth convolutional layers
- Alexnet uses ReLU activations, Weight decay and dropout regularization
- The input image: $3 \times 224 \times 224$
- Achieved a very high improvement in accuracy on ImageNet data.

VGG Net

- Uses so called VGG blocks: a series of convolutional layers (same number of channels and same size) followed by a max pool with size 2×2 and stride 2 (so that size becomes half)
- VGG net has five blocks:
  first two – 1 conv; last three – 2 conv
- first block has 64 channels and each subsequent one doubles number of channels
- Fully connected layers same as Alexnet
- So, it has 8 convolutional layers and 3 FC layers
- VGG net started the idea of using some 'building blocks' to build complex networks

- The Lenet took the input as 2D but convolutional layers are 3D.
- Alexnet made the structure uniform by thinking of channels in input layer too.
- This uniformity is extended by the VGG blocks.
- Many other network structures are designed by configuring different 'building blocks'. (e.g., googlenet)
- In both Alexnet and VGG net we have convolutional layers followed fully connected layers.
- One interesting question is can we have structures where 'fully connected' layers can be in between convolutional layers too.
- An interesting idea here is a convolutional layer with 1×1 kernel.

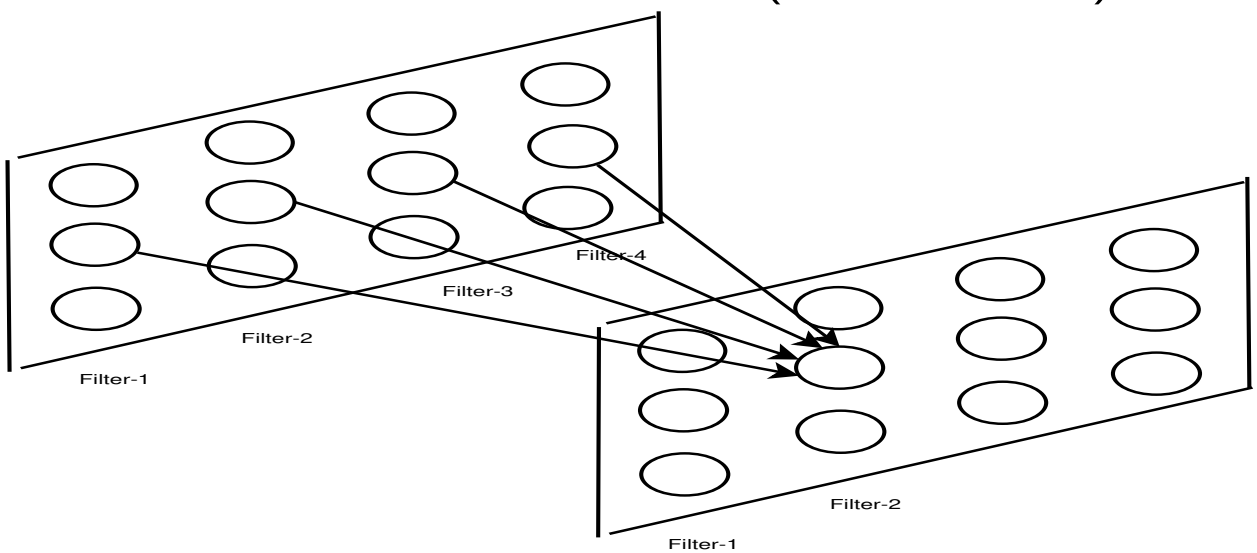
1×1 Convolutional Layer

- Consider output of convolutional layer. If kernel is 1×1

$$y_r^\ell(i,j) = \sum_c \sum_{a=-q}^{q} \sum_{b=-q}^{q} \bar{y}_c^{\ell-1}(i+a, j+b) W_r^\ell(a,b;c)$$

$$y_r^\ell(i,j) = \sum_c \bar{y}_c^{\ell-1}(i,j) W_r^\ell(c)$$

This is like a regular feedforward network (in terms of channels).
We can visualize it like this (for 1D case)

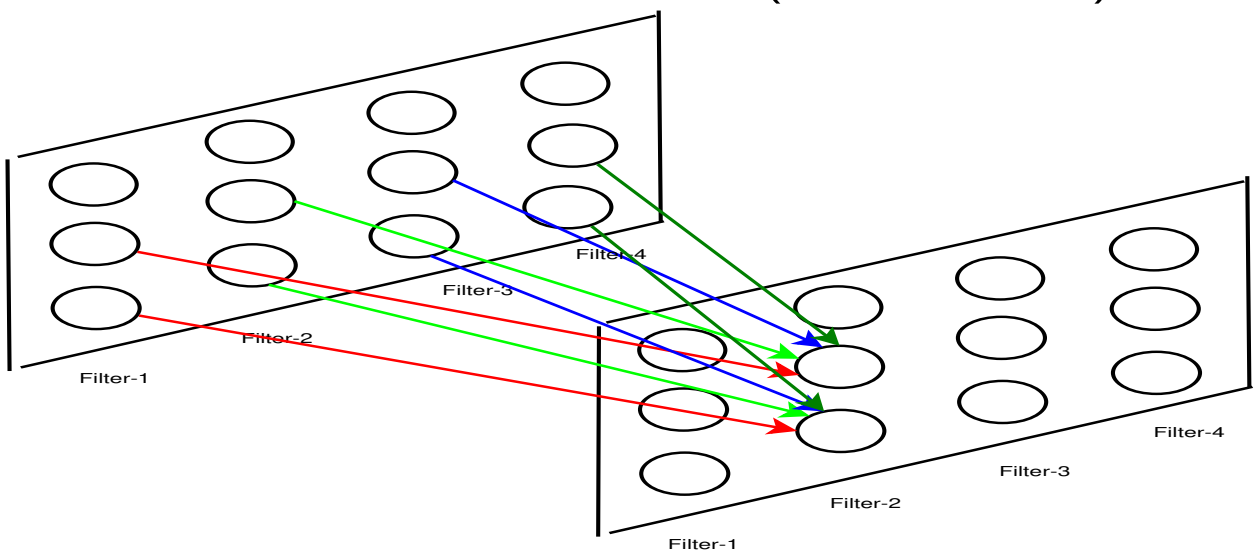


1×1 Convolutional Layer

- The output of 1×1 convolutional layer.

$$y_r^\ell(i,j) \quad = \quad \sum_c \bar{y}_c^{\ell-1}(i,j) W_r^\ell(c)$$

- This is still a convolutional layer.
- There is weight sharing. (same weight for all i, j). We can visualize it like this (for 1D case)

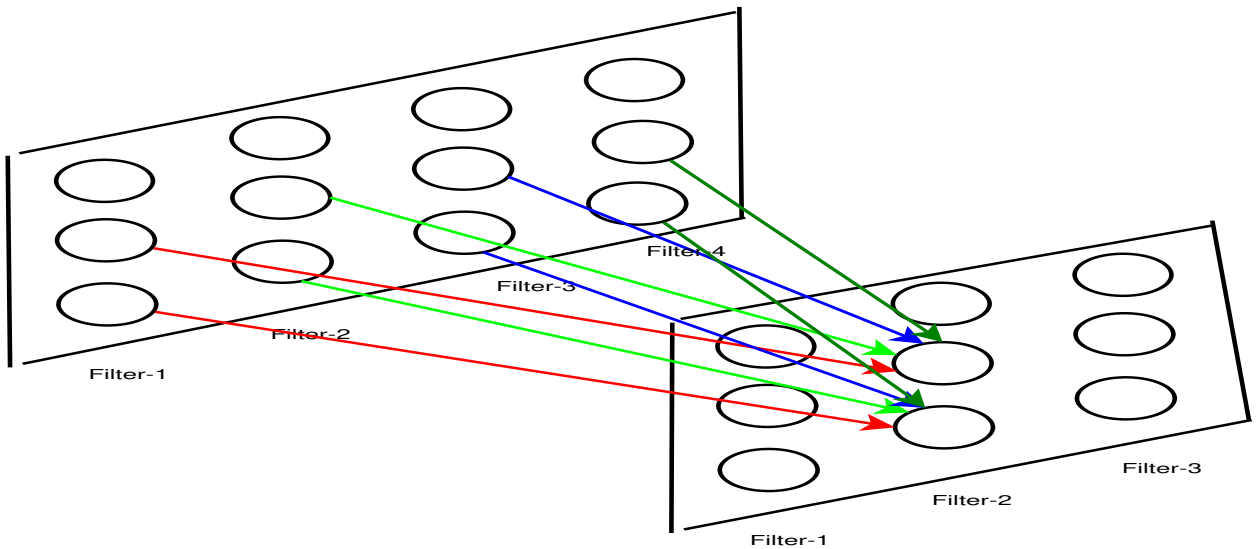


1×1 Convolutional Layer

- The output of 1×1 convolutional layer.

$$y_r^\ell(i,j) \quad = \quad \sum_c \bar{y}_c^{\ell-1}(i,j) W_r^\ell(c)$$

- This is still a convolutional layer.
- There is weight sharing. (same weight for all i, j). We can visualize it like this (for 1D case)



Different number of channels in the two layers.

- 1×1 convolutional layers can be used to, e.g., add outputs of two convolutional layers with same image size but different number of channels.
- They are used for such reshaping purposes in design of deep convolutional networks.
- For example, GoogleNet uses them in the so called inception blocks.
- Another interesting use of this is in so called residual networks.

Resnets

- The residual networks propose to have feed-forward paths to skip a layer in the middle.
- This can allow, for example, to ensure that when we add layers we do not 'lose representational abilities'.
- At the end of a block of convolutional layers, we take the input to this block and add it to the output.
- Now adding this block can only add to the function class representable.
- For such addition, we need 1×1 convolutional layer to reshape as needed.

Convolutional neural Networks

- CNNs are seen to achieve very high accuracies in a large number of applications involving classification of images, speech, text etc.
- The convolutional layer structure is very effective in extracting good feature representations using training data.
- CNNs can take a large part of the credit for the unprecedented interest in deep learning now.

Intriguing properties of CNNs

- Deep networks seem to exhibit some strange behaviour and we do not have enough understanding of deep learning yet.
- The computation by the net is Opaque.
- It is difficult to automatically (extract the) reason about why a classification decision is made.
- This needs attention because these methods are also seen to be brittle.
- We look at one important such characteristic.

Adversarial noise

- CNNs are generally robust to small random additive noise.
- But small amount of specially crafted noise can floor them.
- This so called Adversarial noise, r, can be a solution of

$$\min ||r||^2 \text{ subject to } h(x+r) \neq h(x)$$

  where h represents the CNN classifier function
- Or we can find r as

$$\min ||r||^2 \text{ subject to } P[h(x+r) \neq h(x)] \geq 1-\epsilon$$

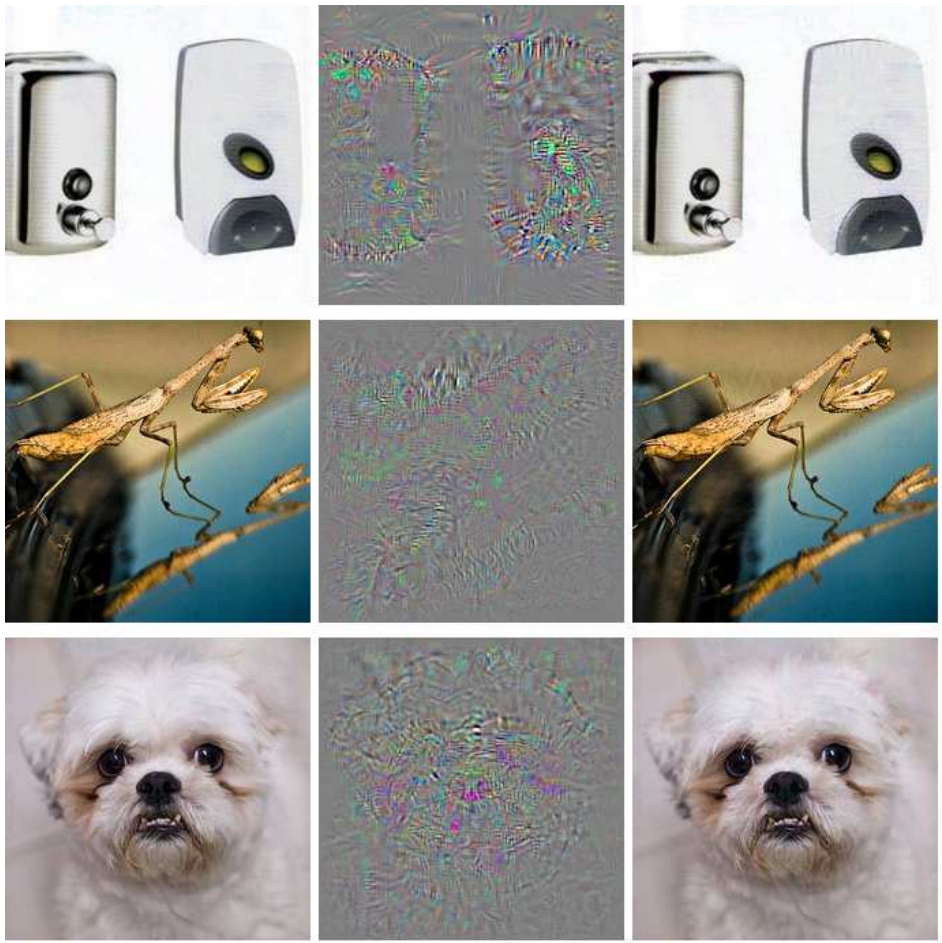
  where the probability is with respect to data distribution.
- There seem to be remarkably universal such perturbations.

An example of Brittleness of deep nets

"panda"
57.7% confidence

"gibbon"
99.3% confidence

Another example



A Caution

- Convolutional networks have seen Spectacular successes in a wide variety of applications.
- However, most of our current models and techniques are are understood only at an empirical level.
- While many ideas are well-motivated, these are all data-driven advances.
- Our explanations of why they work are, mostly, speculative.
- It is not clear, what level of rigorous theoretical understanding is possible.
- In addition, deep learning models are opaque and often brittle.
- Need to exercise sufficient caution and circumspection.