Applied Deep Learning

Chapter 4: Convolutional Neural Networks

Ali Bereyhi

ali.bereyhi@utoronto.ca

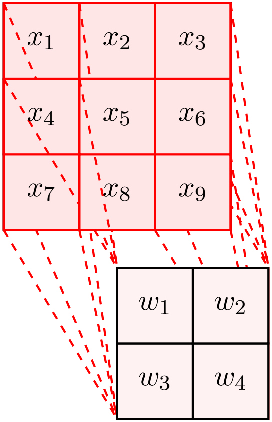
Department of Electrical and Computer Engineering

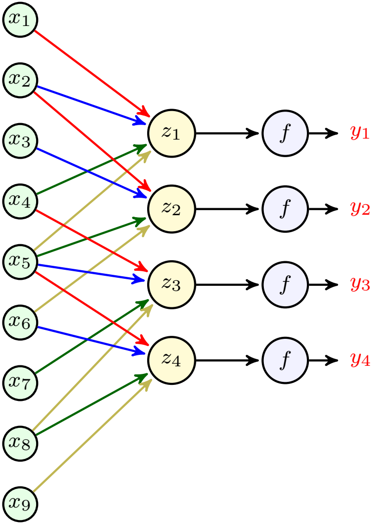
University of Toronto

Fall 2025

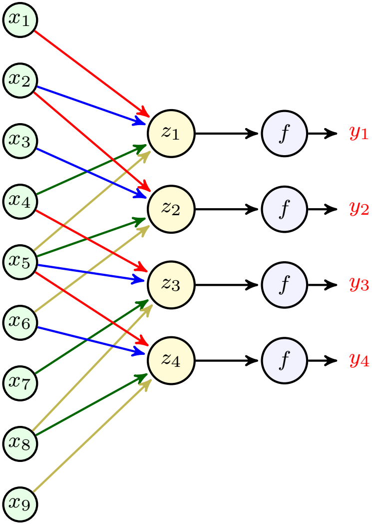
Convolutional Layer as *Set of Neurons*

We can look at a convolutional layer as a layer of neurons





Convolutional Layer as Set of Neurons



In this viewpoint

↳ *each feature is output of a neuron*

↳ *neurons share same weights and bias*

↳ *neurons are activated by $f(\cdot)$*

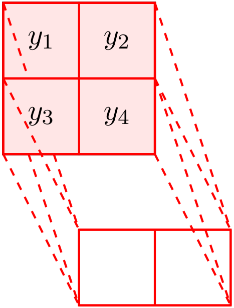
This is however not a fully-connected layer

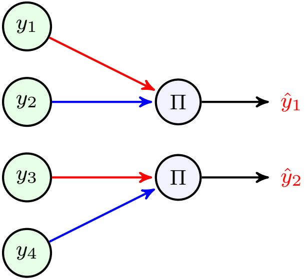
↳ *it is locally-connected*

↳ *neurons have shared parameters*

Pooling Layer as *Set of Neurons*

We can extend our viewpoint to pooling layers





Similar to convolutional layer, pooling layer is a feedforward layer with

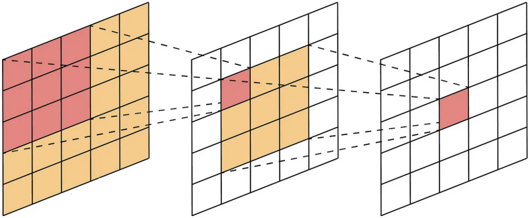
local connectivity and shared parameters

We usually have predefined parameters, i.e., no weights to be learned

Going Deep: *Receptive Field*

As we go deep in CNN, we mix features: *features in deeper layers depend on more pixels of input tensor. We often say*

deeper layers have larger receptive field



Receptive Field

The input region that each neuron in a given convolutional layer responds to

Receptive Field

Few points we may note regarding the receptive field

↳ *For first layer the receptive field is simply where the filter screen*

↳ *For deeper layers, actual receptive field is not immediately obvious and must be calculated*

This is however obvious that

↳ *Receptive field increases as we go deeper*

Receptive field also depends on the filter sizes

↳ *Large filters increase receptive field per layer*

↳ *Small filters reduce receptive field per layer*

+ *Why do we define receptive field? Does it have any particular meaning?!*
– It helps very much building intuition, *especially as we go deep*

Intuition on *Features* via *Receptive Field*

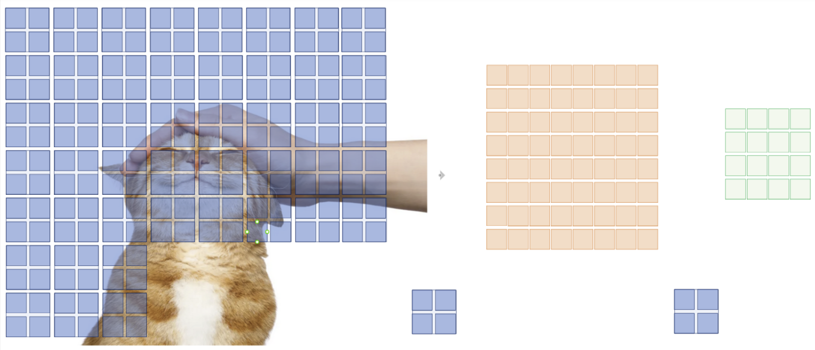
To design a deep CNN, we need to first *build a bit of more intuition*

Let's make a closer look: *say we want to classify the following image*

We use two convolutional + pooling layers cascaded both with 2×2 filters

Intuition on *Features* via *Receptive Field*

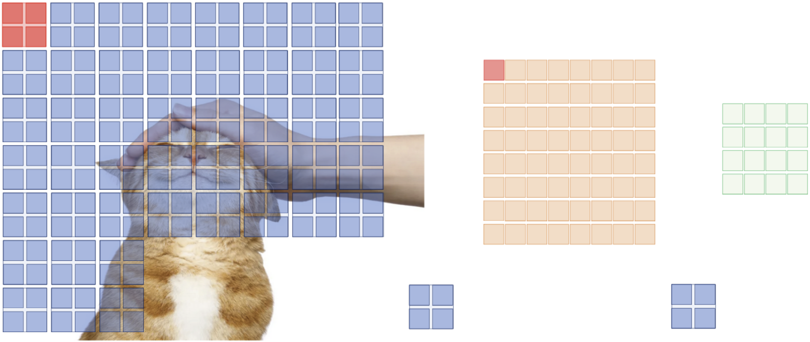
The first layer extracts features of small region of input image



Say for instance, we look at the very first feature in the orange feature map

Intuition on *Features* via *Receptive Field*

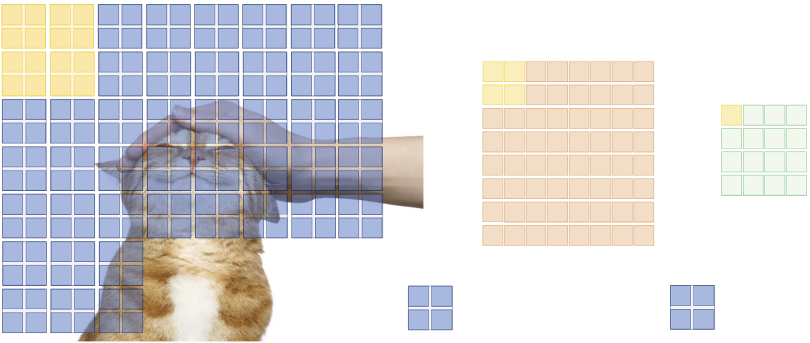
This red feature is calculated from input pixels at upper left corner



We could intuitively say that the neuron corresponding to this feature is looking locally at the upper left corner to extract the features of this region

Intuition on *Features* via *Receptive Field*

Now, let's look at the very first feature in the green feature map



This neuron is looking at a larger region of input image, but it extracts the features of this region through the features extracted via the orange feature map

General Perspective on Deep CNNs

The given example illustrates *an intuitive interpretation of deep* CNN*s*

↳ *Deep CNNs extract gradually the features of input*

↳ *In first layer they extract features of smaller regions of the input*

↳ *They gradually expand this region as they go deeper*

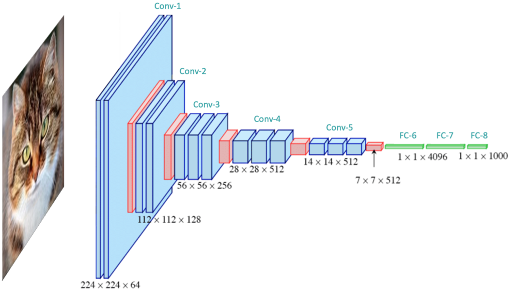
  ↳ *deeper layers look at high-level features*

This gives us a good idea on *how a good deep* CNN *looks like*

- *First layer has small filters*
  - ↳ *small filters lead to large feature maps*
    - ↳ *the number of output channels* is *small*
- *As we go deeper in the CNN filters get a bit larger*
  - ↳ *larger filters return smaller feature maps*
    - ↳ *the number of output channels* is *increased*

Deep CNN: *Example*

Recall the example of VGG-16 architecture we had a look on

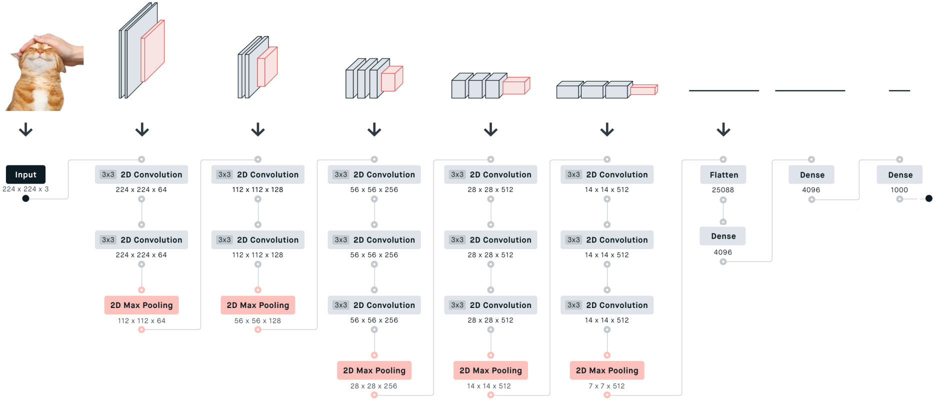


In the original proposal of this architecture, it is said

- *convolutional filters are 3×3 and similar padding is used in all layers*
- *convolution performed with unit stride and max-pooling with stride 2*

Deep CNN: *Example*

We can now break it down, as we know all the components



Deep CNN: *Example*

Let's count the number of layers with learnable parameters

$$
\begin{aligned}
\#\text{ layers with weights} = \quad & 2 && \text{first two convolutions}\\
& + 2 && \text{second two convolutions}\\
& + 3 && \text{third round of convolutions}\\
& + 3 && \text{fourth round of convolutions}\\
& + 3 && \text{fifth round of convolutions}\\
& + 3 && \text{fully-connected network}\\
& = 16
\end{aligned}
$$

This is actually why it's called VGG-16