Beyond MLPs: Convolutional Neural Networks

CS771: Introduction to Machine Learning

Piyush Rai

Recap

- Multi-layer Perceptrons (MLP)

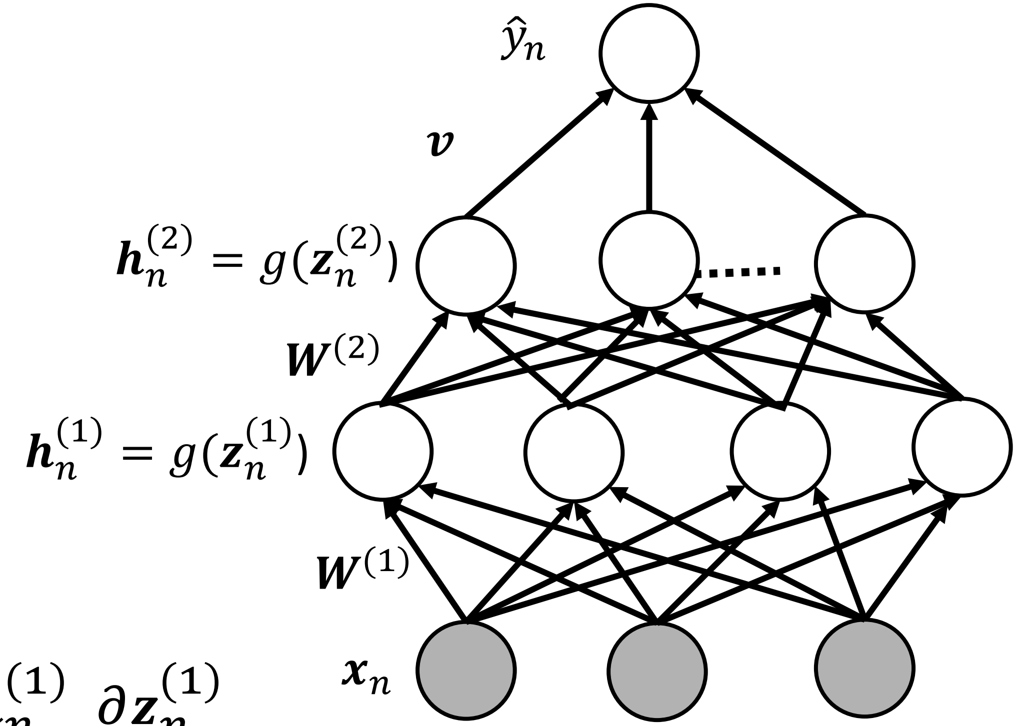
- Backpropagation algorithm to learn the weights

All these gradients are basically product of Jacobians

$$\frac{\partial \ell_n}{\partial \boldsymbol{v}} = \ell'(y_n, \hat{y}_n)\boldsymbol{h}_n^{(2)}$$

$$\frac{\partial \ell_n}{\partial \boldsymbol{W}^{(2)}} = \ell'(y_n, \hat{y}_n)\frac{\partial \hat{y}_n}{\partial \boldsymbol{h}_n^{(2)}}\frac{\partial \boldsymbol{h}_n^{(2)}}{\partial \boldsymbol{z}_n^{(2)}}\frac{\partial \boldsymbol{z}_n^{(2)}}{\partial \boldsymbol{W}^{(2)}}$$

$$\frac{\partial \ell_n}{\partial \boldsymbol{W}^{(1)}} = \ell'(y_n, \hat{y}_n)\frac{\partial \hat{y}_n}{\partial \boldsymbol{h}_n^{(2)}}\frac{\partial \boldsymbol{h}_n^{(2)}}{\partial \boldsymbol{z}_n^{(2)}}\frac{\partial \boldsymbol{z}_n^{(2)}}{\partial \boldsymbol{h}_n^{(1)}}\frac{\partial \boldsymbol{h}_n^{(1)}}{\partial \boldsymbol{z}_n^{(1)}}\frac{\partial \boldsymbol{z}_n^{(1)}}{\partial \boldsymbol{W}^{(1)}}$$



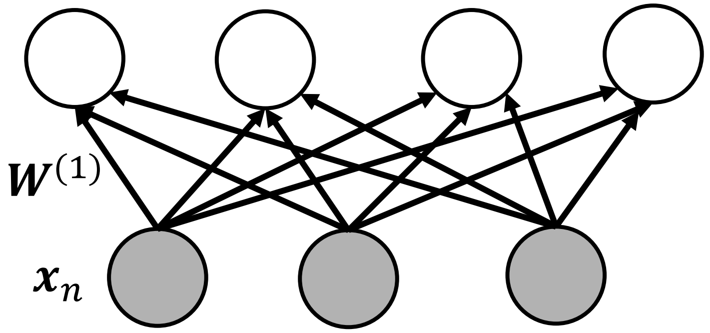
- Backprop is based on reuse of previous computations to efficiently compute the gradients required for updating the network weights using (stochastic) GD

Limitations/Shortcomings of MLP

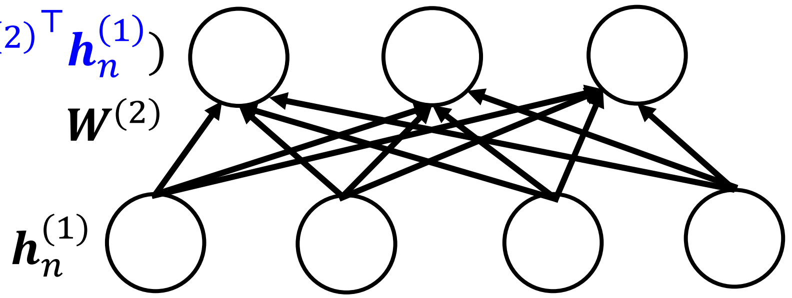
Projection using a "linear layer" + element-wise nonlinearity applied on these linear projections

- MLP uses fully connected layers defined by matrix multiplications + nonlinearity

$\boldsymbol{h}_n^{(1)} = g(\boldsymbol{W}^{(1)^\top} \boldsymbol{x}_n)$



$\boldsymbol{h}_n^{(2)} = g(\boldsymbol{W}^{(2)^\top} \boldsymbol{h}_n^{(1)})$

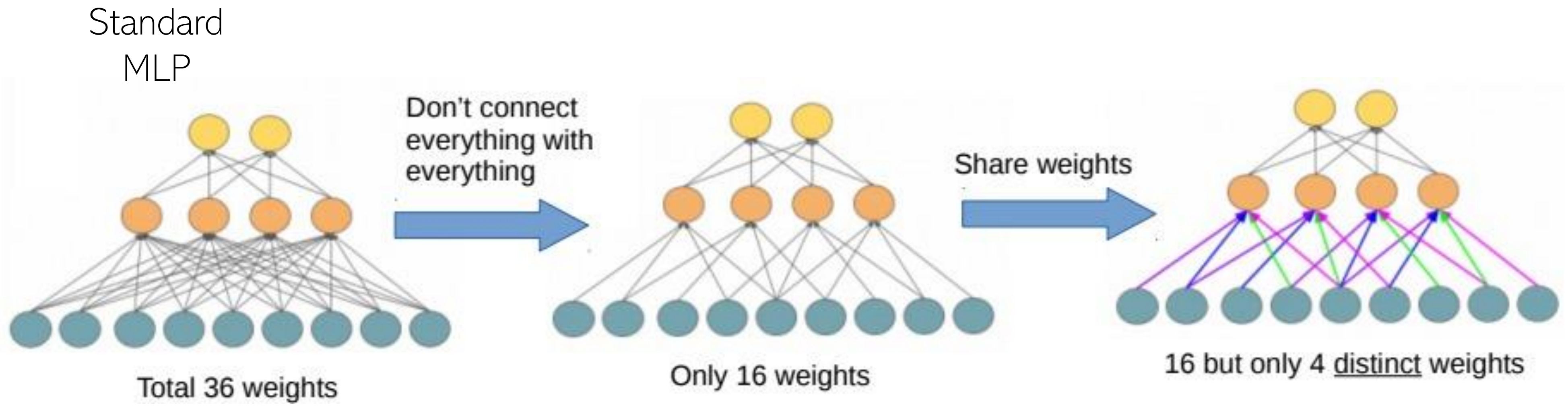


- MLP ignores structure (e.g., spatial/sequential) in the inputs
  - Not ideal for data such as images, text, etc. which are flattened as vectors when used with MLP

- Fully connected nature of MLP requires massive number of weights
  - Even a "smallish" 200x200x3 (3 channels – R,G,B) image will need 120,000 weights for each neuron in the first hidden layer (for K neurons, we will need 120,000 x K weights).
  - Recall that each layer is fully connected so each layer needs a massive number of weights!

Convolutional Neural Networks (CNN)

- CNNs use connections between layers that are different from MLPs in two key ways



- Change 1: Each hidden layer node is connected only to a local patch in previous layer
- Change 2: Same set of weights used for each local patch (purple, blue, green, pink is one set of weights, and this same set of used for all patches)
- These changes help in
  - Substantial reduction on the number of weights to be learned
  - Learning the local structures within the inputs
  - Capturing local and global structure in the inputs by repeating the same across layers

Convolutional Neural Networks (CNN)

- CNN consists of a sequence of operations to transform an input to output
  - Convolution (a linear transformation but more “local” than the one in MLP)
  - Nonlinearity (e.g., sigmoid, ReLU, etc) after the convolution operation
  - Pooling (aggregates local features into global features and reduce representation size)

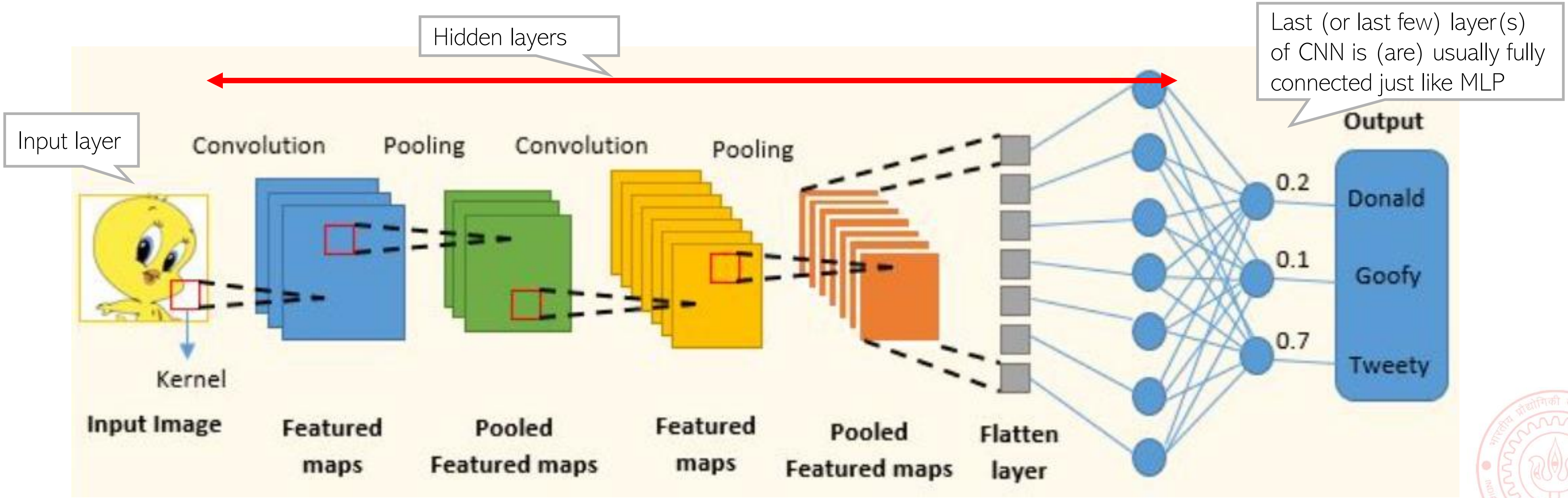
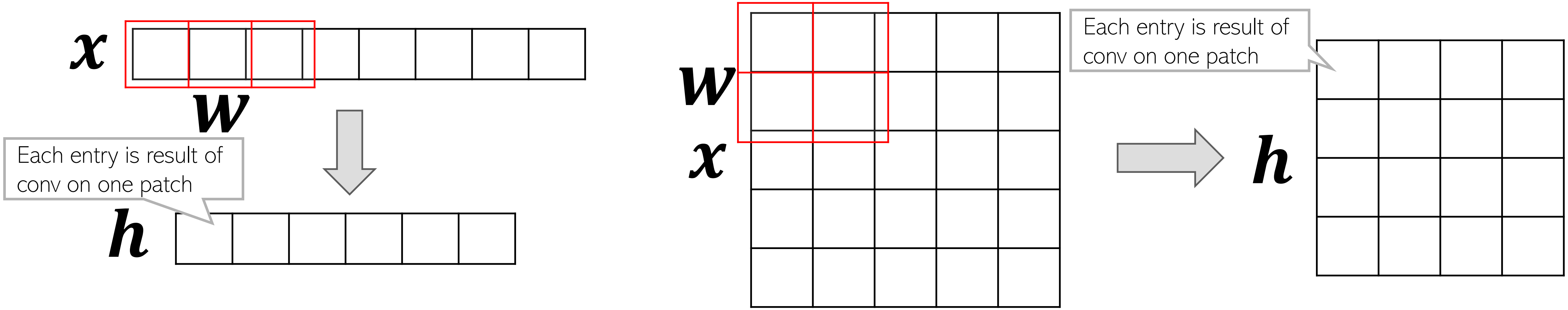


Figure credit: https://www.analyticsvidhya.com/blog/2022/01/convolutional-neural-network-an-overview/

Convolution

Sometimes also called a "kernel", though not the kernel we have seen in kernel methods ☺

- Convolution moves the same "filter"/"template" $\boldsymbol{w}$ over different patches of input $\boldsymbol{x}$
  - Filter is like a set of weights (like in MLP) but only operate on local regions of $\boldsymbol{x}$
- Convolution = dot product of $\boldsymbol{w}$ with different patches of the input $\boldsymbol{x}$



- Output $\boldsymbol{h}$ of the convolution operation is also called a "feature map"
- If $\boldsymbol{x}$ is $n_H \times n_W$, $\boldsymbol{w}$ is $k_H \times k_W$ then $\boldsymbol{h}$ is $(n_H - k_H + 1) \times (n_W - k_W + 1)$
- If we want $\boldsymbol{h}$ to have larger size than then we do zero-padding at boundaries of $\boldsymbol{x}$

Convolution

- High "match" of a filter/kernel with a patch gives high values in the feature map
- In CNN, these weights/filters are learnable. Also, usually multiple filters are used
  - Each filter gives us a different feature map (K filters will give K feature maps)
  - Each map can be seen as representing a different type of feature in the inputs

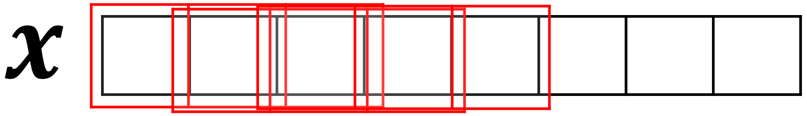
"north-west" filter

$$\begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

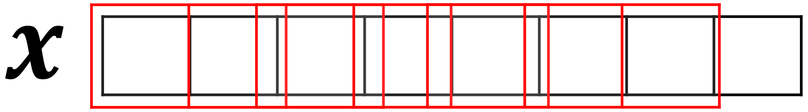
"north-east" filter

$$\begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix}$$

- When "moving" the filter across the input, the stride size can be one or more than one
  - Stride means how much the filter moves between successive convolutions



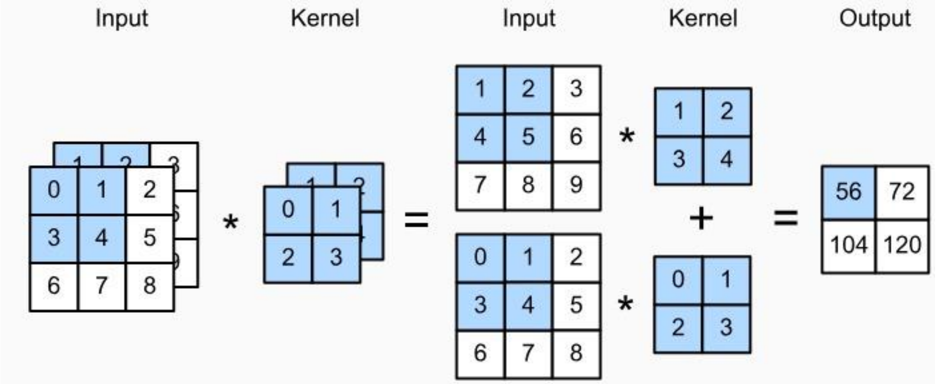
Stride of 1

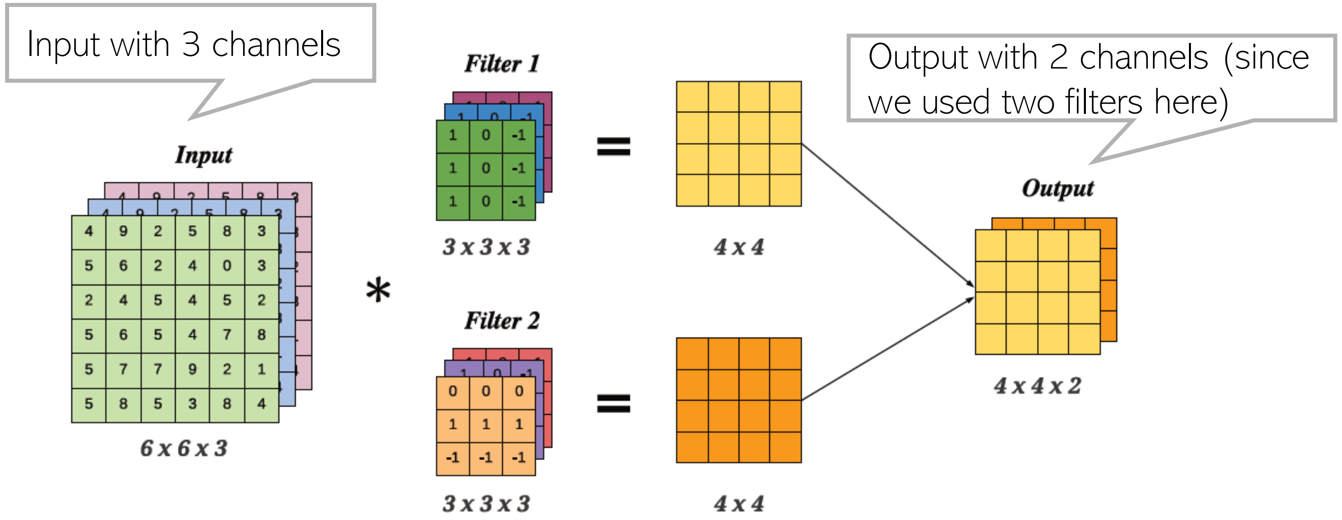


Stride of 2

Multiple Input Channels

- If the input has multiple channels (e.g., images with R,G,B channels), then each filter/kernel also needs to have multiple channels, as shown below (left figure)
- We perform per-channel convolution followed by an aggregation (sum across channels)





- Note that (right figure above) we typically also have multiple such filters (each with multiple channels) which will give us multiple such feature maps

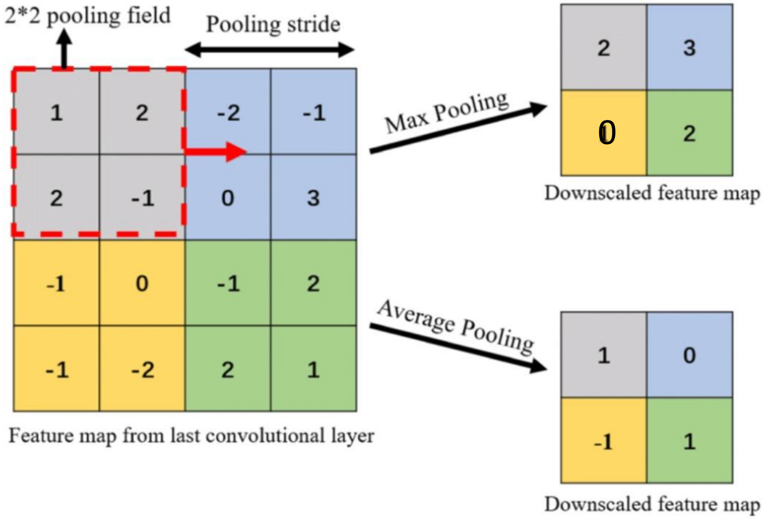
Figure credit: PML-1 (Murphy, 2022),

Pooling

- CNNs also consist of a pooling operation after each conv layer

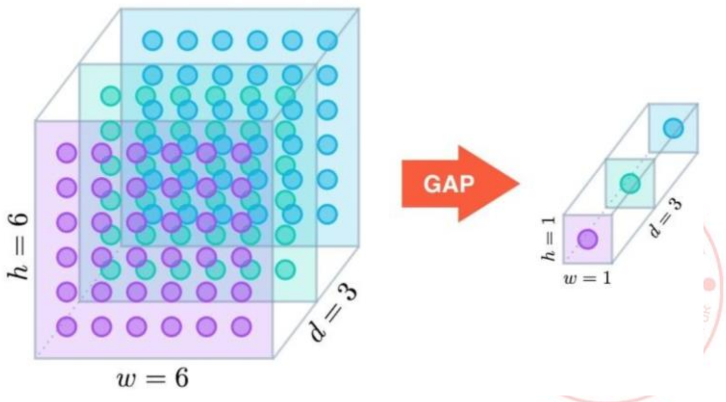
- Pooling plays two important roles
  - Reducing the size of the feature maps
  - Combining local features to make global features

- Need to specify the size of group to pool, and pooling stride



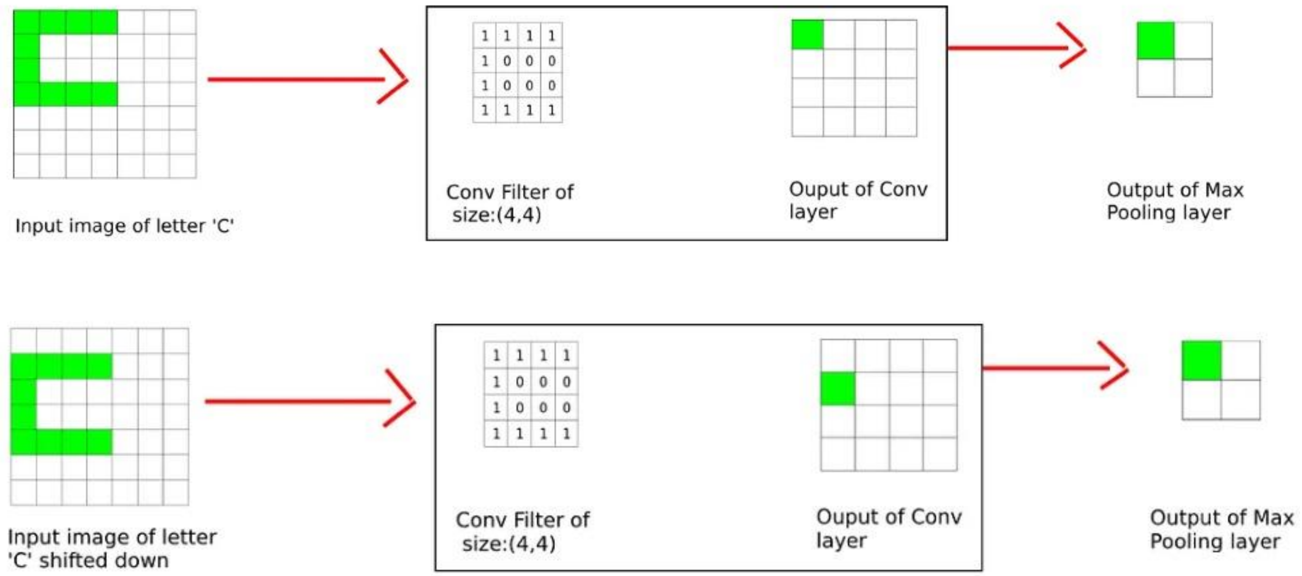
- Max pooling and average pooling are popular pooling methods

- "Global average pooling" (GAP) is another option
  - Given feature map of size $h \times w \times d$ (e.g, if there are d channels), it averages all $h \times w$ locations to give a $1 \times d$ feature map
  - Reduces the number of features significantly and also allows handling feature maps of different heights and widths



CNNs have Translation Invariance!

- Even if the object of interest has shifted/translated, CNN don't face a problem (it will be detected regardless of its location in the image)
- The simple example below shows how (max) pooling helps with this



- CNNs use a combination of conv + pooling operations in several hidden layers so CNNs remain invariant to even more significant translations

Pic credit: https://divsoni2012.medium.com/translation-invariance-in-convolutional-neural-networks-61d9b6fa03df

CNN: Summary of the overall architecture

- The overall structure of a CNN looks something like this

Thickness of this box denotes how many filters we used (each filter itself may consist of multiple channels if the input has multiple channels)

Pooling only downscales the height and width; The size of the other dimension remains the same as in the previous conv layer

Towards the end, we usually flatten the feature map (or use GAP to get a flattened input) and use one or more fully connected layers like in MLP

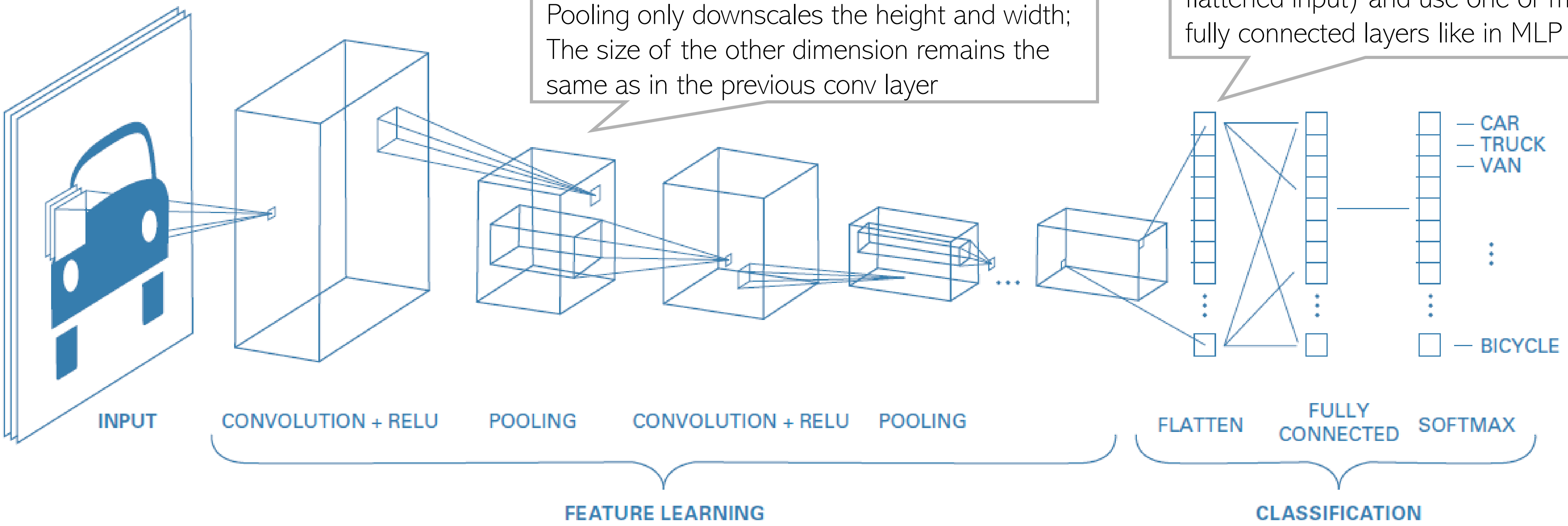


Figure credit: PML-1 (Murphy, 2022),