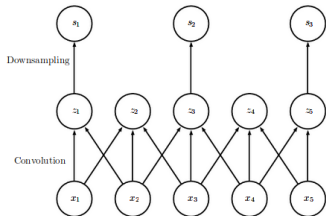
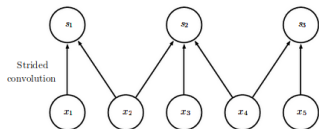


# Deep Learning

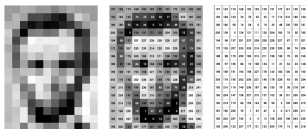
## CNN Components



### Learning goals

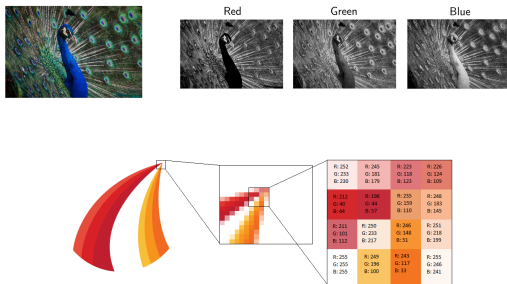
- Input Channel
- Padding
- Stride
- Pooling

# INPUT CHANNEL



**Figure:** Left: image of Lincoln; Center: pixels labeled with grayscale values; Right: grayscale values by themselves (Levin & Dorsey)

- An image consists of the smallest indivisible segments called pixels with a strength often known as the pixel intensity.
- A grayscale image has a single input channel with the value of each pixel representing the amount of light/brightness in the pixel.
- A grayscale value can lie between 0 and 255 (0 = black, 255 = white).

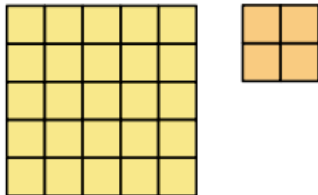


**Figure:** How AI sees an image (Maladkar, 2020)

- A colored digital image usually comes with three color channels, i.e. the Red-Green-Blue channels, popularly known as the RGB values.
- Each pixel can be represented by a vector of three numbers (each ranging from 0 to 255) for the three primary color channels.

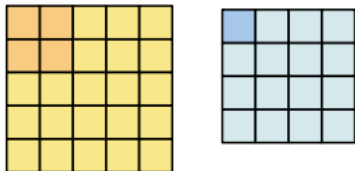
# VALID PADDING

Suppose we have an input of size  $5 \times 5$  and a filter of size  $2 \times 2$ .



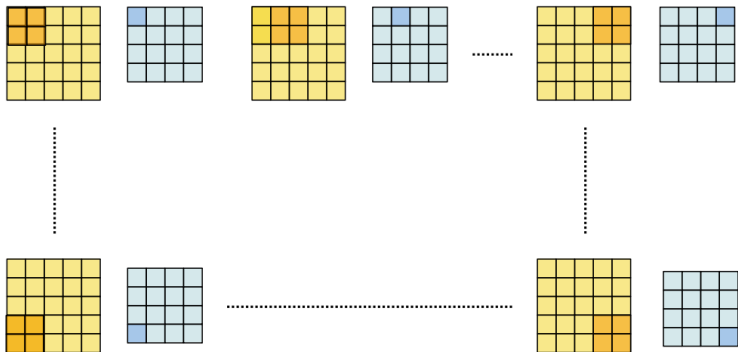
# VALID PADDING

The filter is only allowed to move inside of the input space.



# VALID PADDING

That will inevitably reduce the output dimensions.

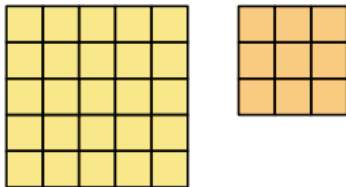


In general, for an input of size  $i \times i$  and filter size  $k \times k$ , the size of the output feature map  $o \times o$  calculated by:

$$o = i - k + 1$$

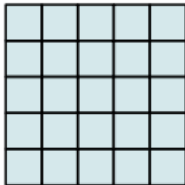
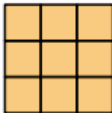
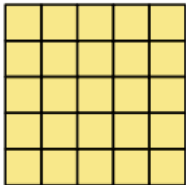
# SAME PADDING

Suppose the following situation: an input with dimensions  $5 \times 5$  and a filter with size  $3 \times 3$ .



# SAME PADDING

We would like to obtain an output with the same dimensions as the input.

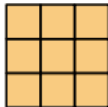




# SAME PADDING

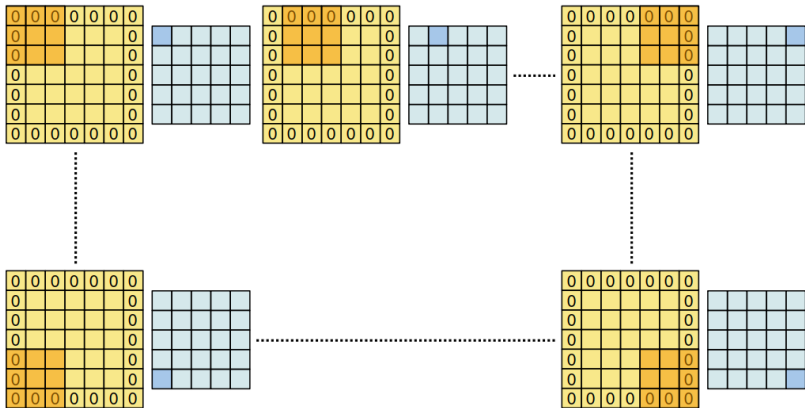
Hence, we apply a technique called zero padding. That is to say “pad” zeros around the input:

0	0	0	0	0	0	0
0						0
0						0
0						0
0						0
0						0
0	0	0	0	0	0	0

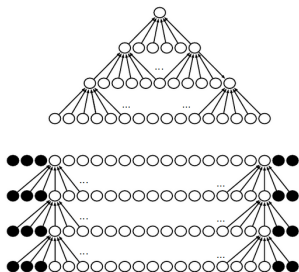


# SAME PADDING

That always works! We just have to adjust the zeros according to the input dimensions and filter size (ie. one, two or more rows).



# PADDING AND NETWORK DEPTH



**Figure:** “Valid” versus “same” convolution.

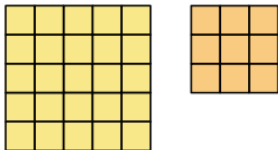
*Top* : Without padding, the width of the feature map shrinks rapidly to 1 after just three convolutional layers (filter width of 6 shown in each layer). This limits how deep the network can be made.

*Bottom* : With zero padding (shown as solid circles), the feature map can remain the same size after each convolution which means the network can be made arbitrarily deep (Goodfellow et al., 2016).

# Strides

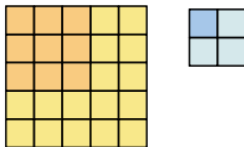
# STRIDES

- Stepsize “strides” of our filter (stride = 2 shown below).



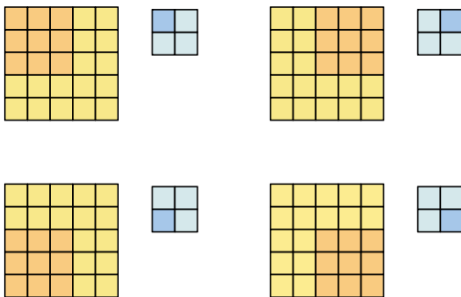
# STRIDES

- Stepsize “strides” of our filter (stride = 2 shown below).



# STRIDES

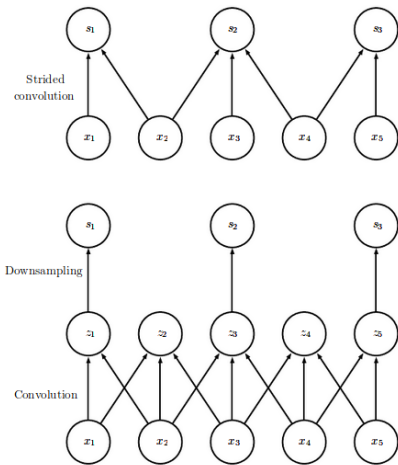
- Stepsize “strides” of our filter (stride = 2 shown below).



In general, when there is no padding, for an input of size  $i$ , filter size  $k$  and stride  $stride$ , the size  $o$  of the output feature map is:

$$o = \left\lfloor \frac{i - k}{stride} \right\rfloor + 1$$

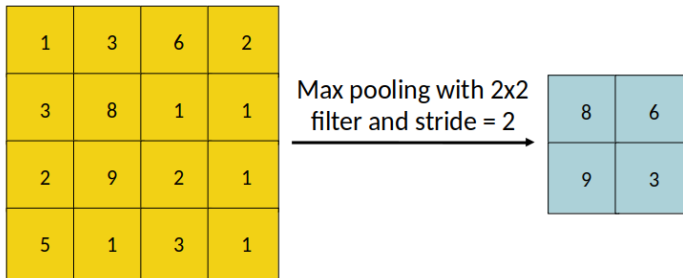
# STRIDES AND DOWNSAMPLING



**Figure:** A strided convolution is equivalent to a convolution without strides followed by downsampling (Goodfellow et al., 2016).

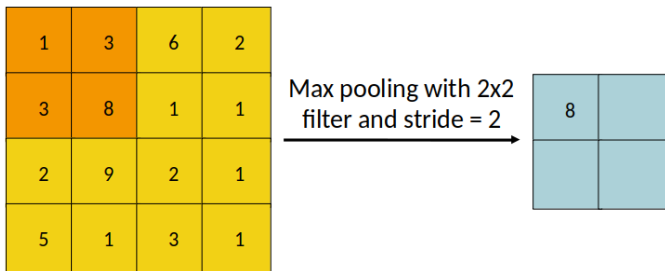


# MAX POOLING



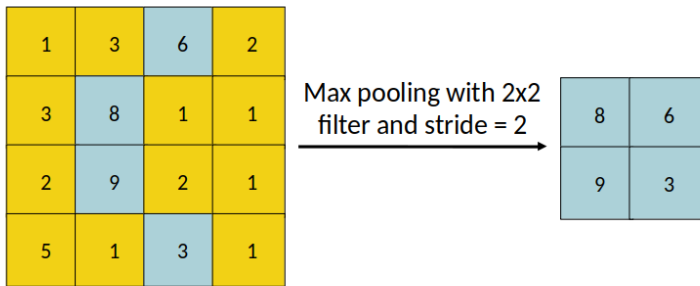
- We've seen how convolutions work, but there is one other operation we need to understand.
- We want to downsample the feature map but optimally lose no information.

# MAX POOLING



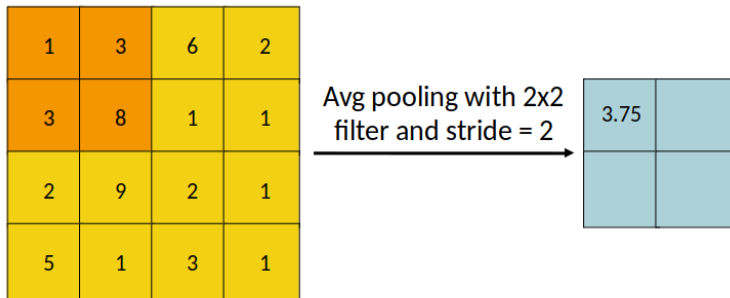
- Applying the max pooling operation, we simply look for the maximum value at each spatial location.
- That is 8 for the first location.
- Due to the filter of size 2 we have the dimensions of the original feature map and obtain downsampling.

# MAX POOLING



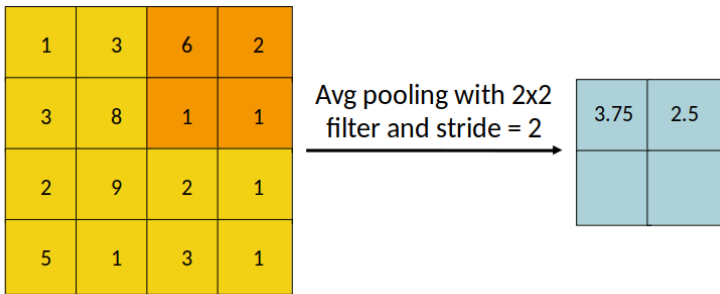
- The final pooled feature map has entries 8, 6, 9 and 3.
- Max pooling brings us 2 properties: 1) dimension reduction and 2) spatial invariance.
- Popular pooling functions: max and (weighted) average.

# AVERAGE POOLING



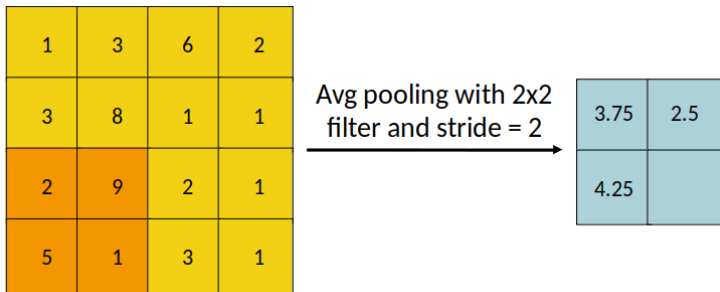
- We've seen how max pooling worked, there are exists other pooling operation such as avg pooling, fractional pooling, LP pooling, softmax pooling, stochastic pooling, blur pooling, global average pooling, and etc.
- Similar to max pooling, we downsample the feature map but optimally lose no information.

# AVERAGE POOLING



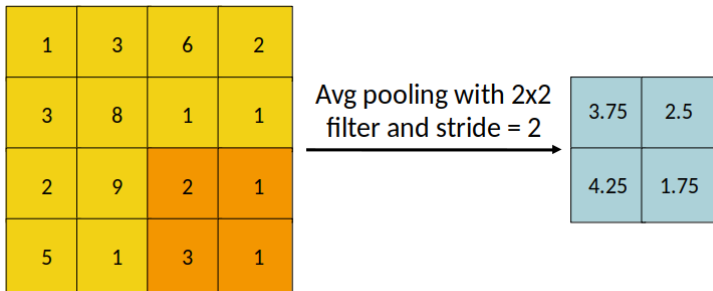
- Applying the average pooling operation, we simply look for the mean/average value at each spatial location.

# AVERAGE POOLING



- We use all information by sum and backpropagated to all responses.
- It is not robust to noise.

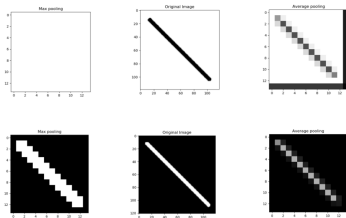
# AVERAGE POOLING



- The final pooled feature map has entries 3.75, 2.5, 4.25 and 1.75.

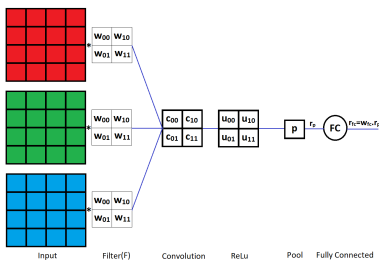
# COMPARISON OF MAX AND AVERAGE POOLING

- Avg pooling use all information by sum but max pooling use only highest value.
- In max pooling operation details are removed therefore it is suitable for sparse information (Image Classification) and avg pooling is suitable for dense information (NLP).



**Figure:** Shortcomings of max and average pooling using toy image (Sharma, 2020)





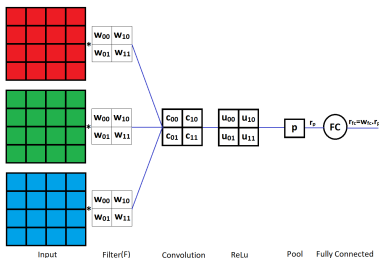
**Figure:** For colored images each of the Red, Green and Blue (RGB) color spectrums serve as separate input channel (Belwal, 2018)

In this CNN:

- there are 3 input channel, with the size of 4x4 as an input matrices,
- one 2x2 filter (also known as kernel),
- a single ReLu layer,
- a single pooling layer (which applies the MaxPool function),

- and a single fully connected (FC) layer.
- The elements of the filter matrix are equivalent to the unit weights in a standard NN and will be updated during the backpropagation phase.
- Assuming a stride of 2 with no padding, the output size of the convolution layer is determined by the following equation:
- $o = \frac{i-k+2p}{stride} + 1$  where:
  - o: is the dimension (rows and columns) of the output square matrix,
  - i: is the dimension (rows and columns) of the input square matrix,
  - k: is the dimension (rows and columns) of the filter (kernel) square matrix,
  - p: is the number of pixels(cells) of padding added to each side of the input,

- stride: is the stride, or the number of cells skipped each time the kernel is slid.








Inserting the values shown in the figure into the equation,

$$o = \frac{i - k + 2p}{stride} + 1 = \frac{(4 - 2 + 2 \cdot 0)}{2} + 1 \quad (1)$$

$$= 2 \quad (2)$$

# REFERENCES

-  Levin, G. (n.d.). *Image Processing and Computer Vision*. ofBook - Image Processing and Computer Vision. [https://openframeworks.cc/ofBook/chapters/image\\_processing\\_computer\\_vision.html](https://openframeworks.cc/ofBook/chapters/image_processing_computer_vision.html)
-  Ian Goodfellow, Yoshua Bengio, & Aaron Courville (2016). *Deep Learning*. MIT Press.
-  Sharma, P. (2020, April 21). *MaxPool vs avgpool*. OpenGenus IQ: Computing Expertise % Legacy. <https://iq.opengenus.org/maxpool-vs-avgpool/> .
-  Belwal, C. (2018, June 12). *Part III: Backpropagation mechanics for a convolutional neural network*. Part III: Backpropagation mechanics for a Convolutional Neural Network. <http://cbelwal.blogspot.com/2018/05/part-iii-backpropagation-mechanics-for.html>
-  Maladkar, K. (2020, October 16). *Computer vision primer: How ai sees an image*. Analytics India Magazine. <https://analyticsindiamag.com/computer-vision-primer-how-ai-sees-an-image/>