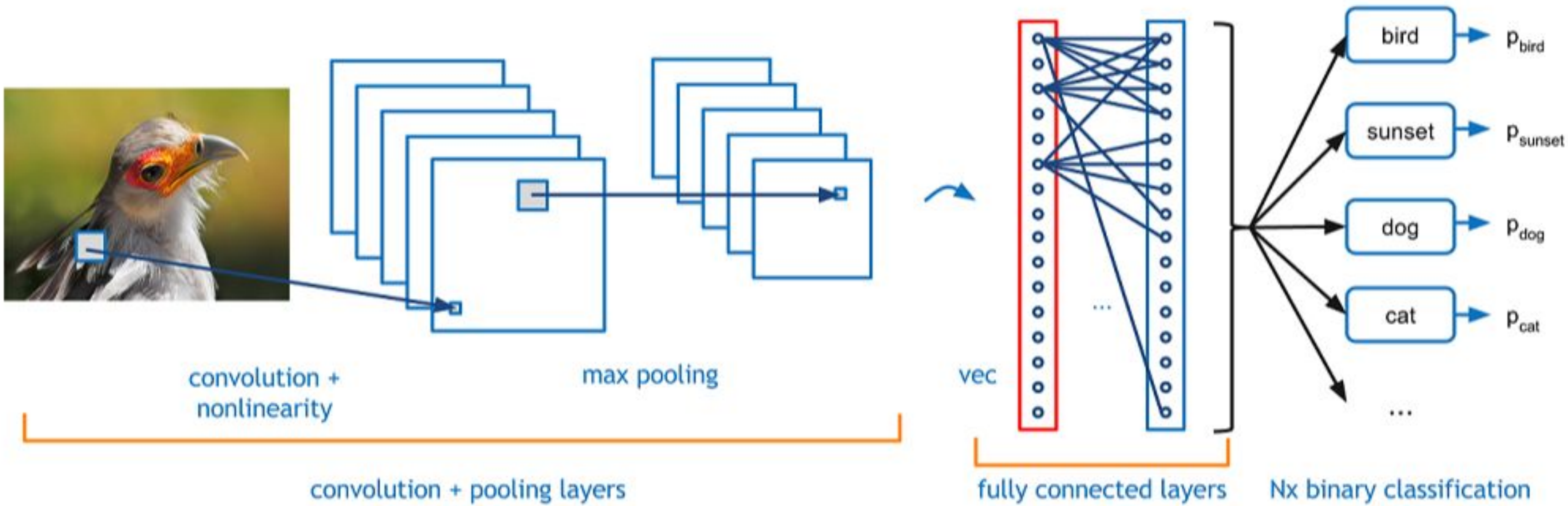
Convolutional Neural Network for Image and Text Data

CNN for Image Data

Convolutional Neural Network



Extracting useful features of data

Perform a ML task (like classification based on the vectorized data)

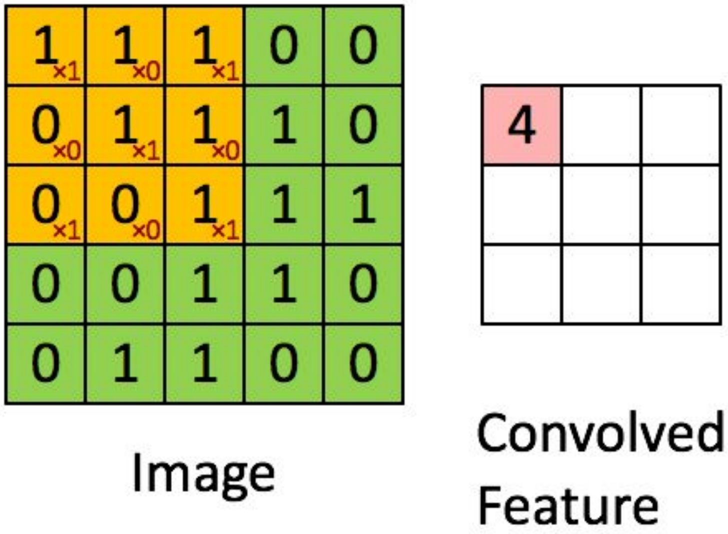
Convolutional Operation

- Filter Size: K
- Stride Size: S
- Padding Size: P

Input size

$$o = \frac{W - K + 2P}{S} + 1$$

Output size



Stanford UFLDL

Multi-Channel CNN

- A color image is a 3-D tensor
- 400 (height) 630 (width) 3 (R,G,B channels)

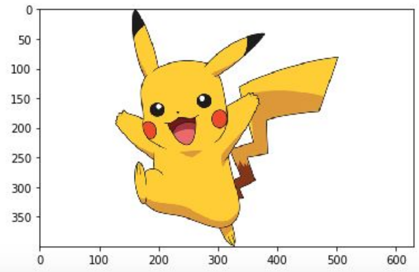
```
from matplotlib.image import imread
import numpy as np
img = imread('pikka_3.jpg')
```

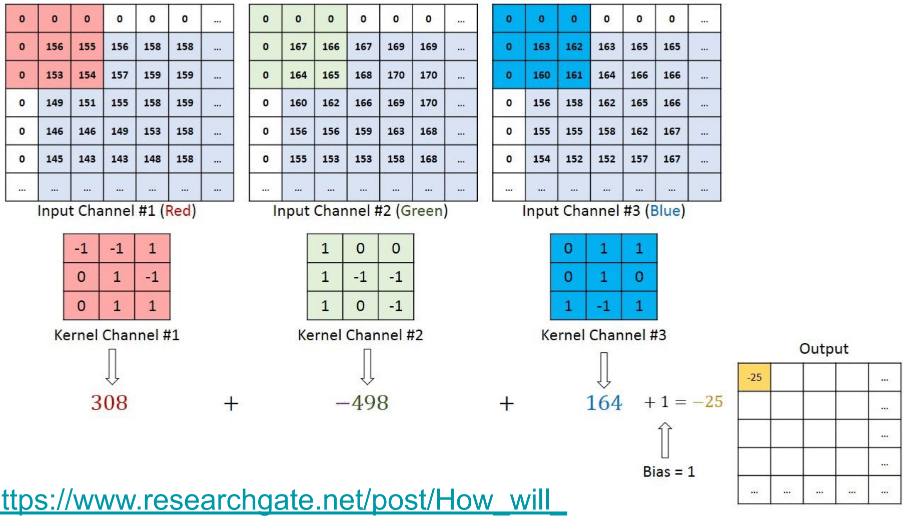
```
print(img.shape)
```

```
(400, 630, 3)
```

```
plt.imshow(img, interpolation='nearest')
```

```
<matplotlib.image.AxesImage at 0x11b404278>
```





https://www.researchgate.net/post/How_will_channels_RGB_effect_convolutional_neural_network

From Keras Layers Conv2D

Input shape

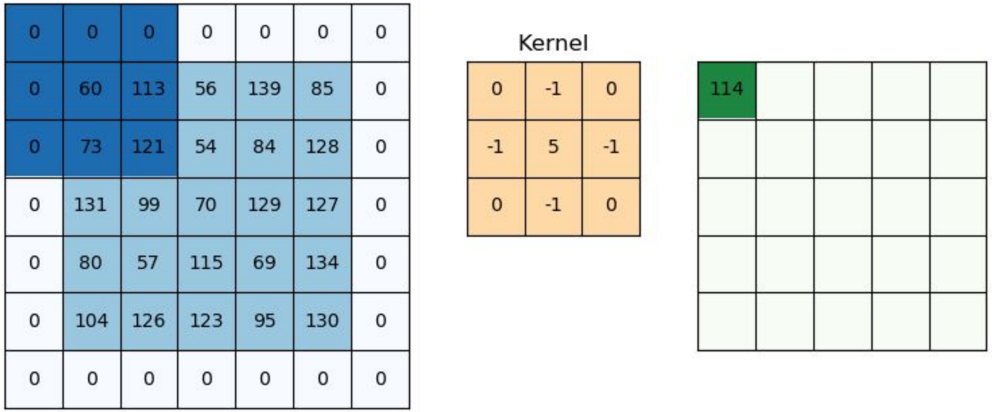
4D tensor with shape: `(batch, channels, rows, cols)` if `data_format` is `"channels_first"` or 4D tensor with shape: `(batch, rows, cols, channels)` if `data_format` is `"channels_last"`.

Output shape

4D tensor with shape: `(batch, filters, new_rows, new_cols)` if `data_format` is `"channels_first"` or 4D tensor with shape: `(batch, new_rows, new_cols, filters)` if `data_format` is `"channels_last"`. `rows` and `cols` values might have changed due to padding.

Zero Padding

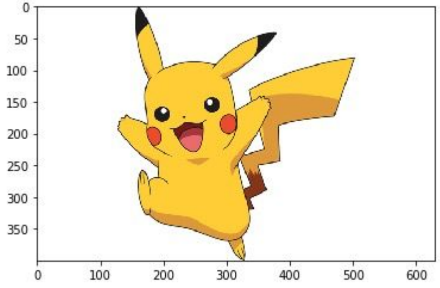
- Pads the image with zeros around the **border**
- Make the input image and feature map have the same spatial dimensions



Stride: 1
Size of zero padding:
(k-1)/2

https://stackoverflow.com/questions/52067833/how-to-plot-an-animated-matrix-in-matplotlib

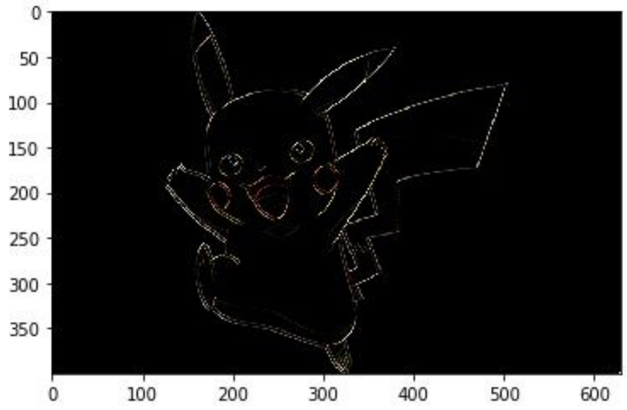
Filter comes from “Image Processing”



Image

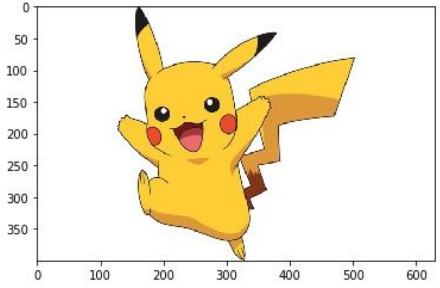
```
print(kernel)

[[ 1  0 -1]
 [ 0  0  0]
 [-1  0  1]]
```

Edge Detection



Convolved Features

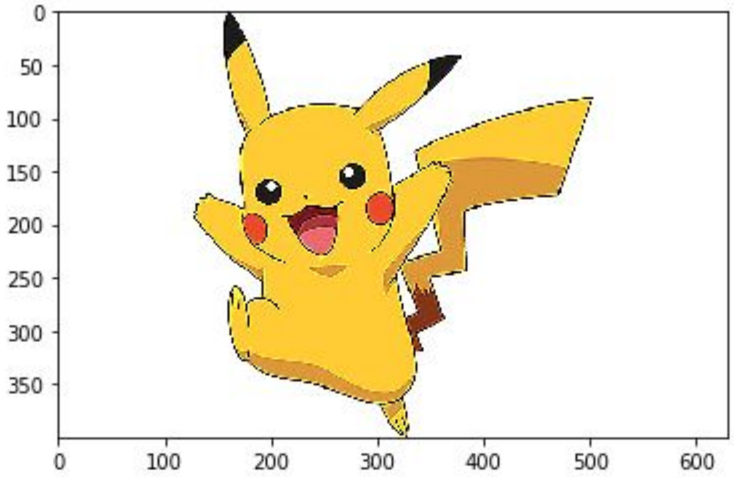
Filter comes from “Image Processing”



Image

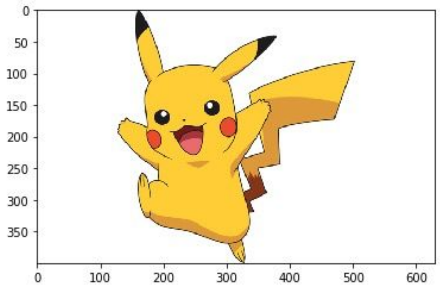
```
print(kernel)
[[ 0 -1  0]
 [-1  5 -1]
 [ 0 -1  0]]
```

Sharpen



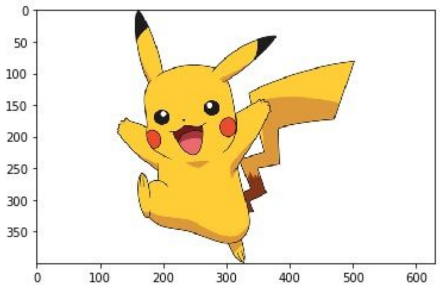
Convolved Features

Filter comes from “Image Processing”



Image

Identity

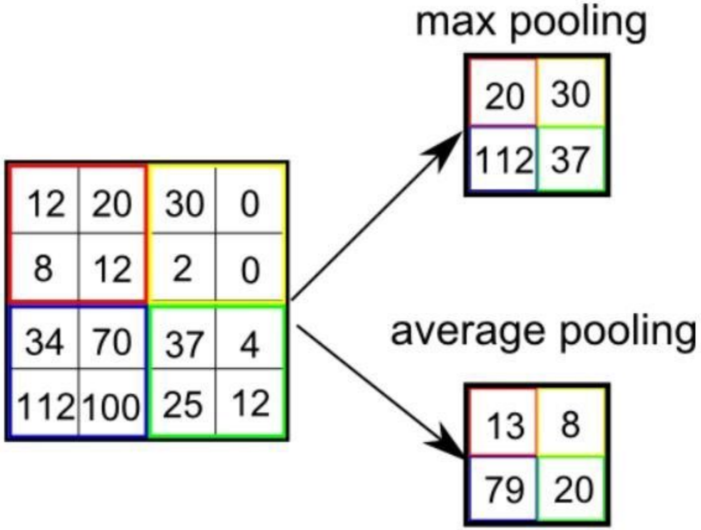


Convolved Features

Pooling Operation

- Pooling Size: the box size. Here is 2 * 2
- Stride Size: how much pixel the window move
- Reduce the dimensionality

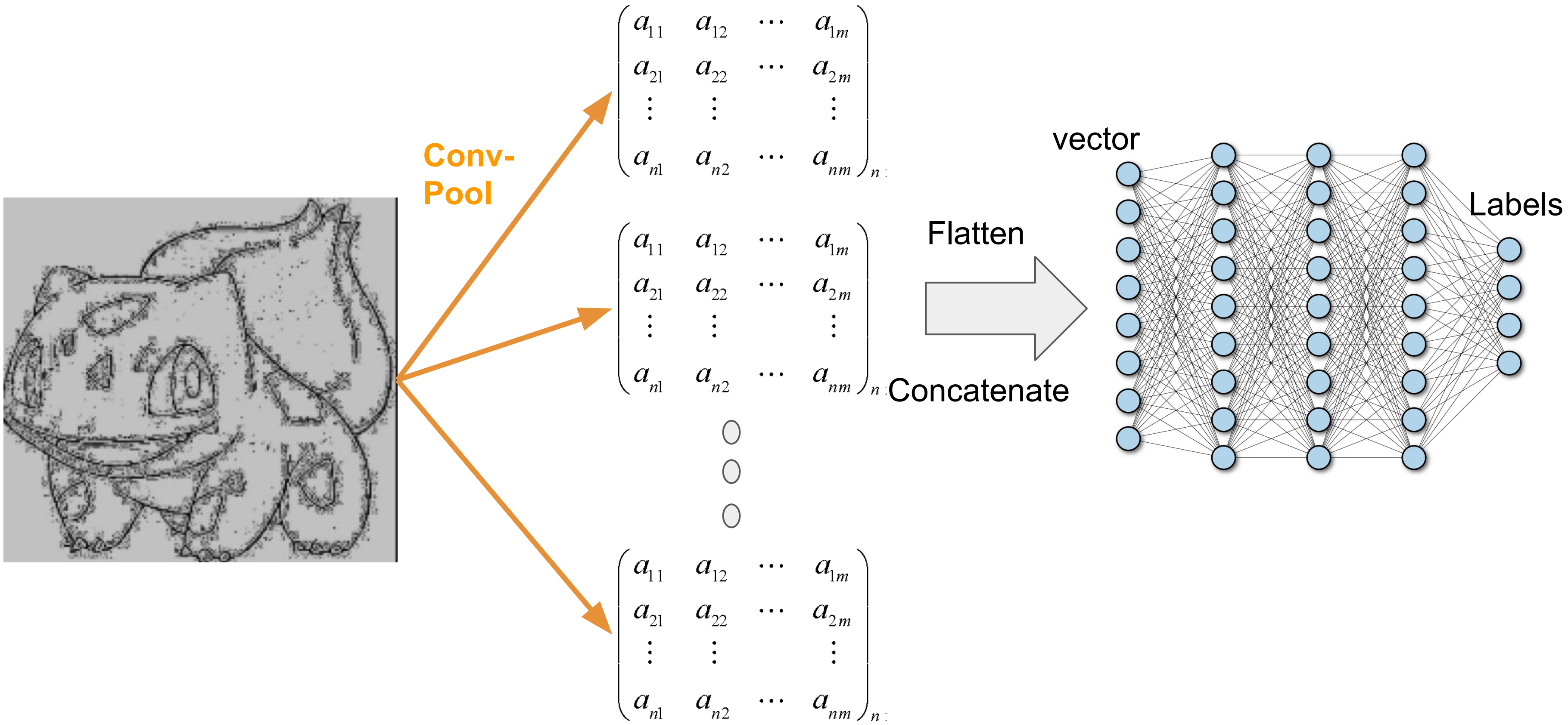
What is stride size here？



Filter then Pool

1. The size is **one quarter** the original size
2. The **vertical line** features are **enhanced**.

Conv-Pool



Where are these filters from?

- Filters, in nature, are model parameters, which can be **learned** by backpropagation.

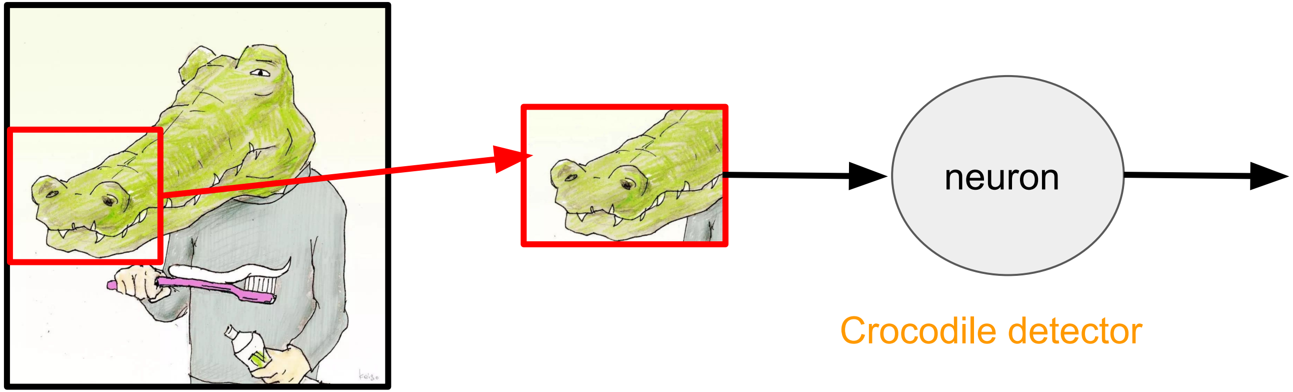
- These filters weights are firstly randomly initialized, and then updated during training process.

- End-to-End optimization: Backpropagation.

- More details:
  https://towardsdatascience.com/training-a-convolutional-neural-network-from-scratch-2235c2a25754

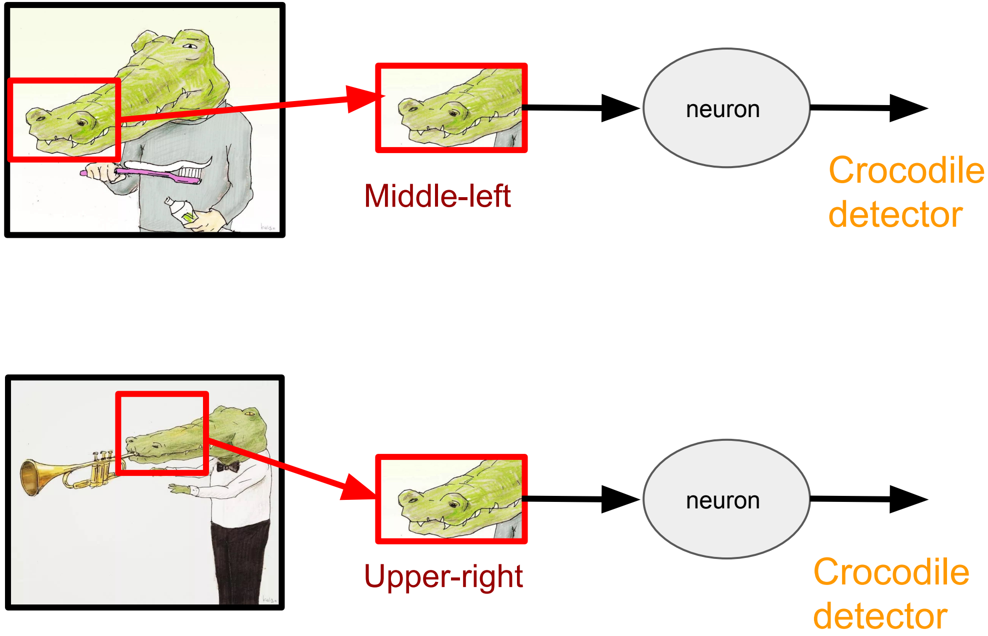
Local Features Matter

- Discriminative patterns are much smaller than the whole image
- A neuron does not have to see the whole image
- Less parameters required



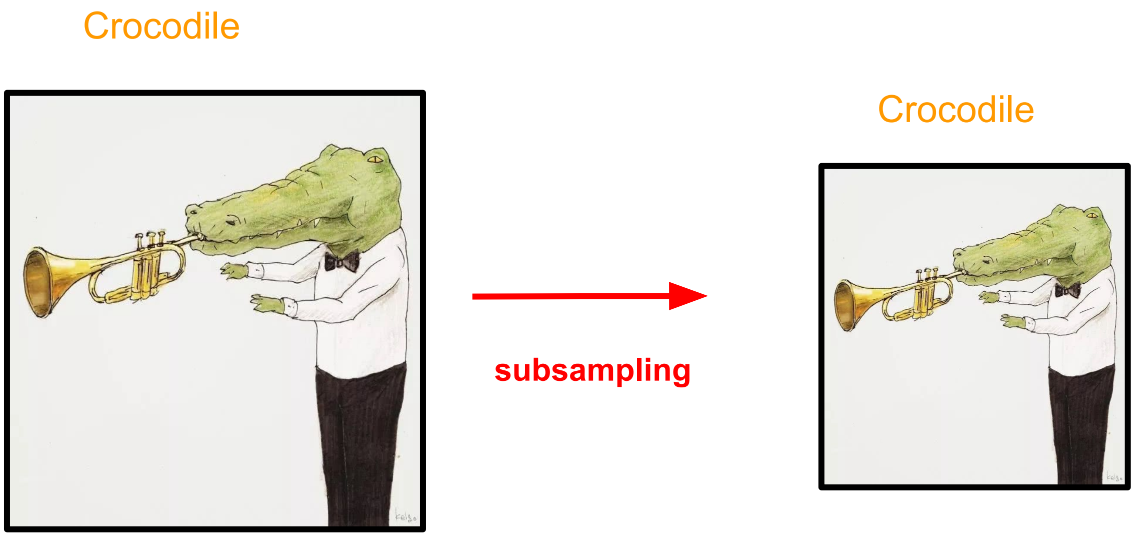
Location Insensitive

- The same patterns appear in different regions
- A neuron should be location insensitive.



Subsampling Works

- Subsampling the pixels will not change the object
- We can subsample the pixels to make images smaller -> less parameters required



CNN for Images

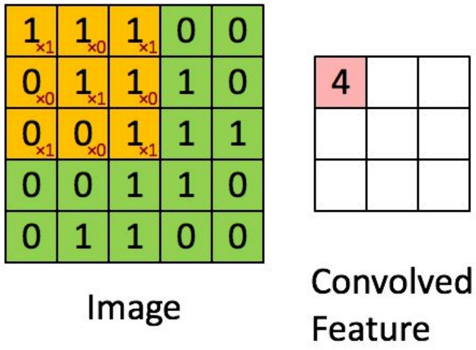
CNN:

1. **Convolutional Layer:** from local regions in images to feature map

2. **Pooling Layer:** reduce the dimensionality of feature maps

3. 2d example ->

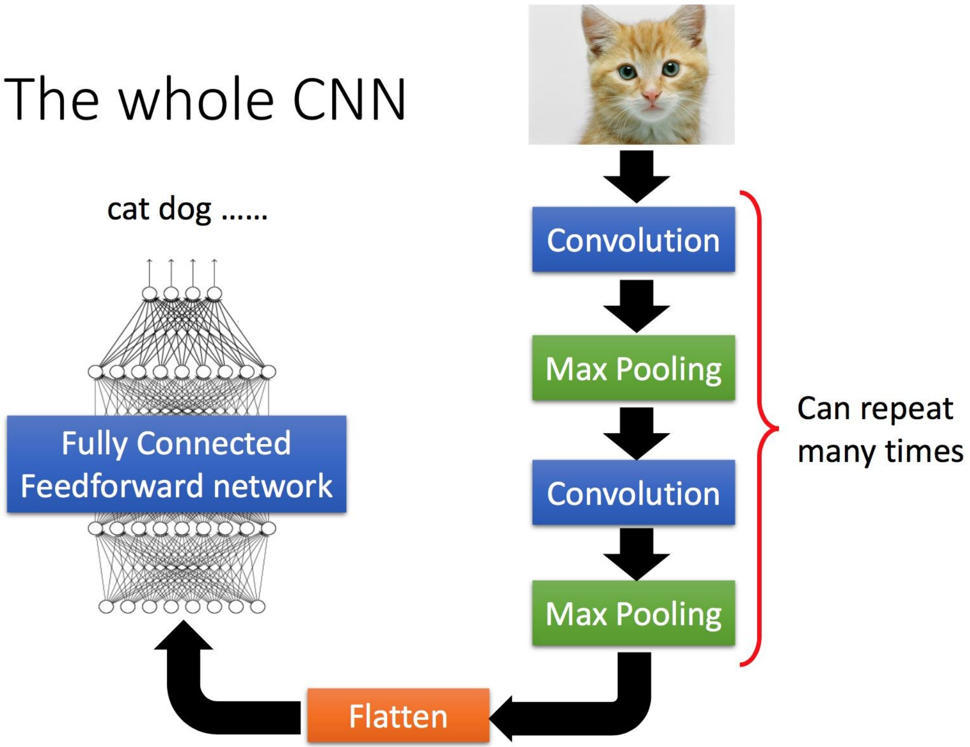
Yellow shows filter weights
Green shows input



Stanford UFLDL

CNN for Images

- Convolution Layer：
  - Local features matters
  - Location Insensitive
- Max Pooling Layer:
  - Subsampling works



CNN for Text

CNN works for Text

Images

- Local Features Matter
- Locations Insensitive
- Subsampling Works

Texts

- Key n-grams define semantics

  *Pulp fiction's director is Quentin. I **am obsessed of** it.*

- Locations of key n-grams Insensitive?

  *I **am obsessed of** Pulp fiction, whose director is Quentin.*

  *Pulp fiction's director is Quentin. I **am obsessed of** it.*

  I owe you ten dollars
  You owe me ten dollars.

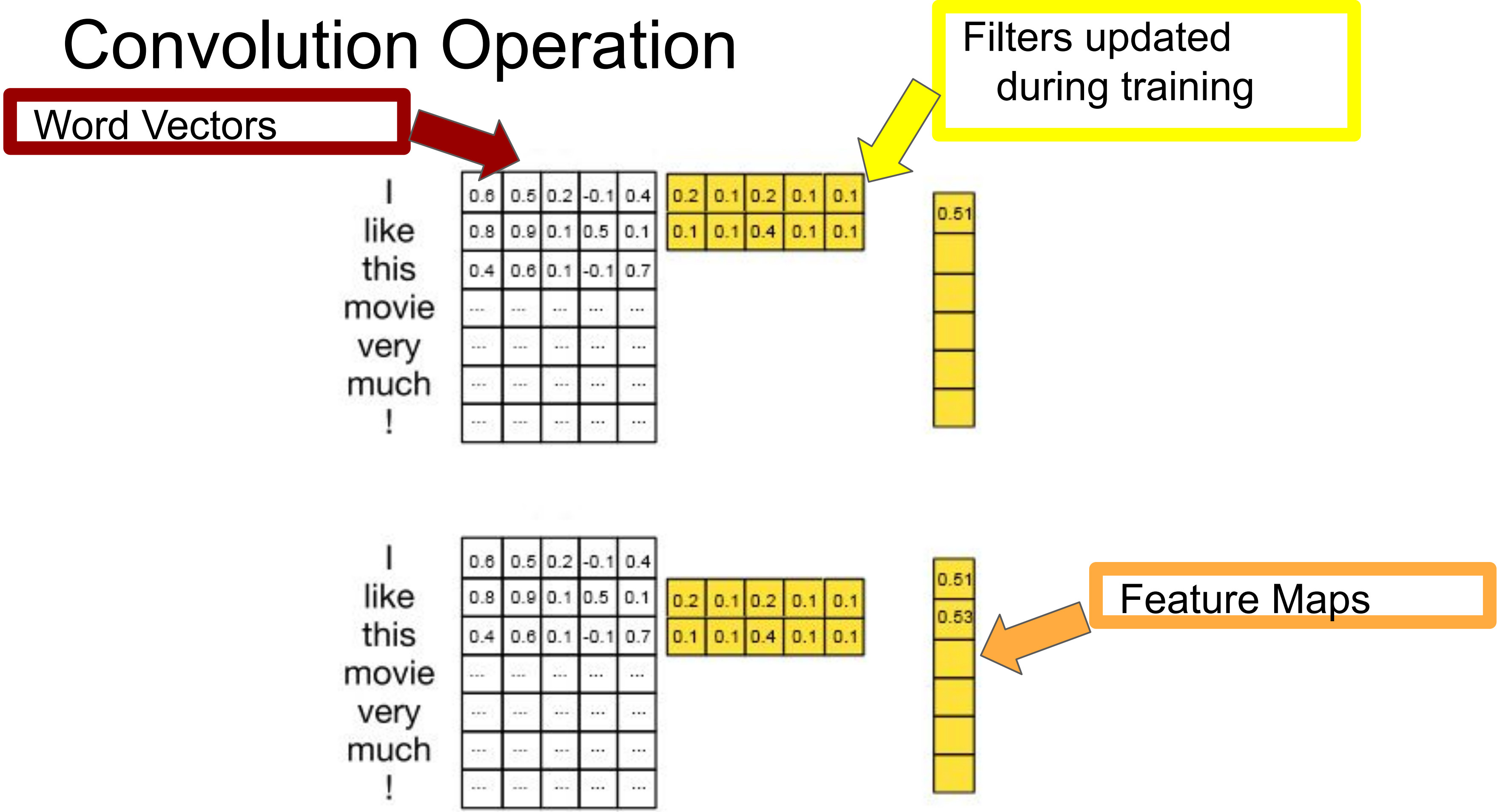
- Doc. Summarization

Combinations

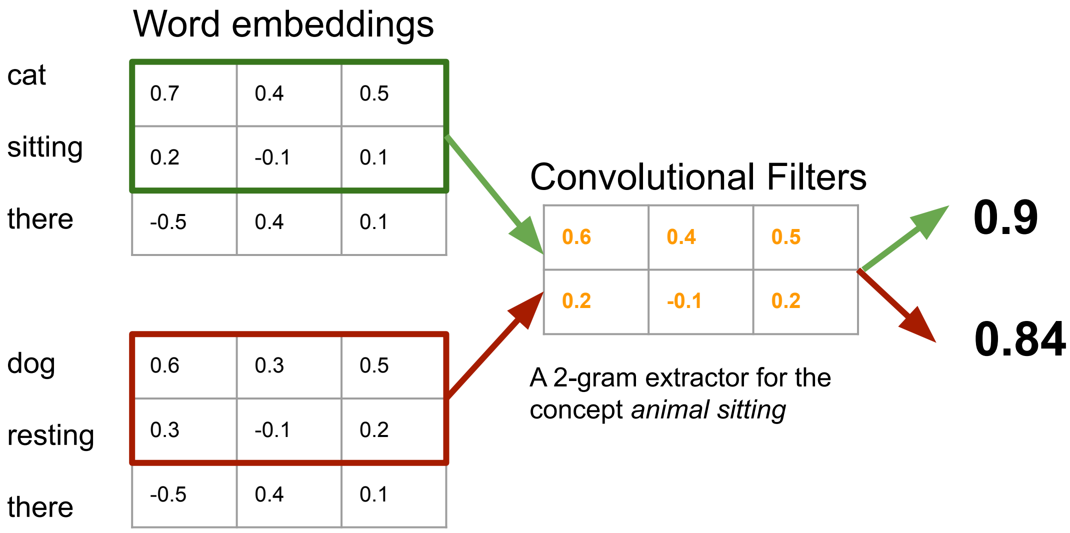
E.g., *I hate this movie*

- Compute vectors for every possible phrase
    - *I hate this movie* ----> I hate; hate this; this movie
- Compute these vectors for these phrases

Convolution Operation



Toy Example



A 2-gram extractor for the concept *animal sitting*

- This convolution provides high activations for 2-grams with certain meaning
- Can be extended to 3-grams, 4-grams, etc.
- Can have various filters, need to track many n-grams.
- They are called 1D since we only slice the windows only in one direction

Why is it better than BoW?

Convolution Operation

$$p = \tanh\left(W \left[\begin{array}{c} c_1 \\ c_2 \end{array} \right] + b\right)$$

Word Vector: $c \in R^k$

[ ]: concatenation operation

W: linear matrix $W \in R^{d \times nk}$

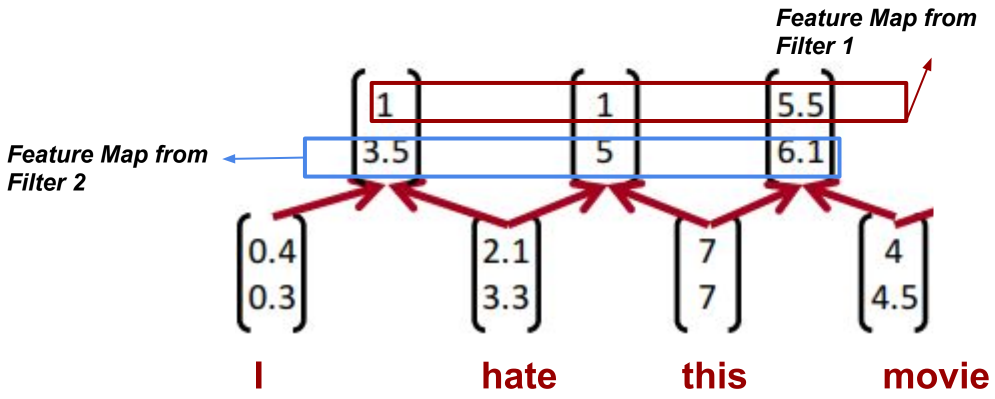
b: bias vector $b \in R^d$

W and b are the network parameters to be learned.

Convolution Layer Weights

W: linear matrix $W \in R^{d \times nk}$

- Each **row vector** in linear matrix can be regarded as one **n-gram extractor**, i.e., filter.
- Number of row d can be regarded as number of extractors.
- The output is the feature value for the input n-words.
- They are called 1D because we slide the window only in one direction.

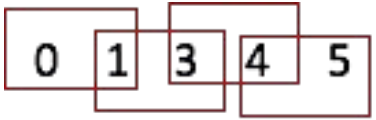


Padding

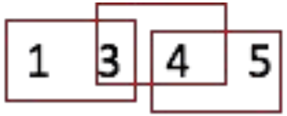
Padding: After convolution, the lengths of feature maps depends on padding

Toy Sequence: 1, 3, 4, 5

a. To be the same as the input length **(*same*)**

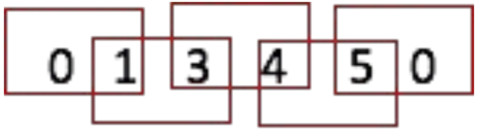


b. Input length - window size + 1 **(*valid or narrow*)**



Variable Feature Dimension after Convolution

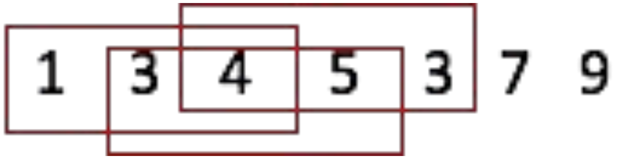
c. Input length + window size - 1 **(*wide*)**



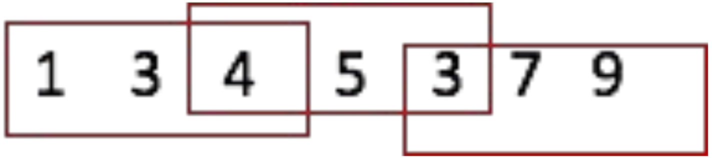
Stride

- Control how the filters move along the input sequence

- **N-stride** means that the filter convolve around the input sequence by **shifting n units every time**.



Stride = 1



Stride = 2

Pooling Operation

1. Calculates some reduction function feature-wise

2. Pooling is conducted over the sequence direction

    a. **K-range Max pooling:** *Did u see this feature anywhere in the k window*

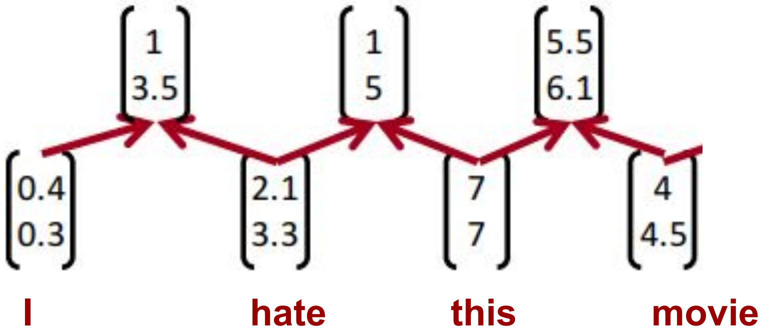
    $$[5.5 \ ; \ 6.1]^T$$

    b. **Average pooling:** *How prevalent is this feature over the entire range?*

    $$[2.9 \ ; \ 4.5]^T$$

Lose the order information.

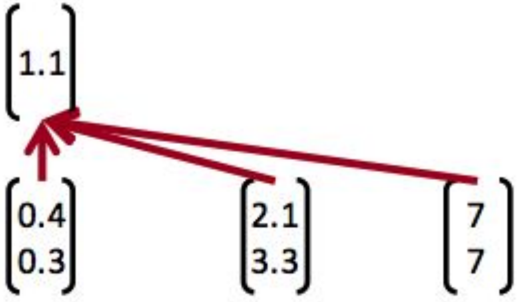
We do not care the position of the key n-grams in the sentence.
Whether the key n-grams is in the sentence or not is important.



Convolution Operation Hyper-parameters

Windows size n: how many words are considered together?

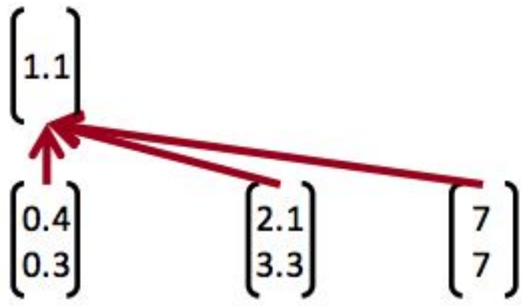
Output dimension d: how many filters are used for word windows?



Windows Size

Windows size n: capture the n-gram features in convolutional layer

- CNN **automatically** learns the values of its filters
- Compared to n-grams model, CNN is **efficient** and **compact** in representation without representing all vocabulary

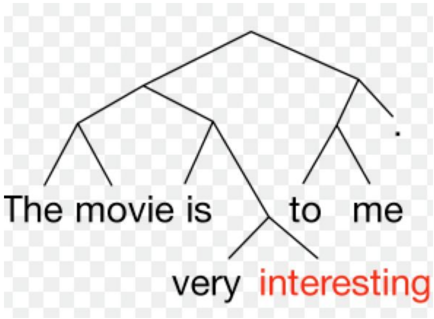


Stride Size

Stride size: how much you want to shift your filter at each step

- It is usually set to be one
- If large stride size, build a model that behaves somewhat similarly to a tree

stride size=2



Multiple Filters

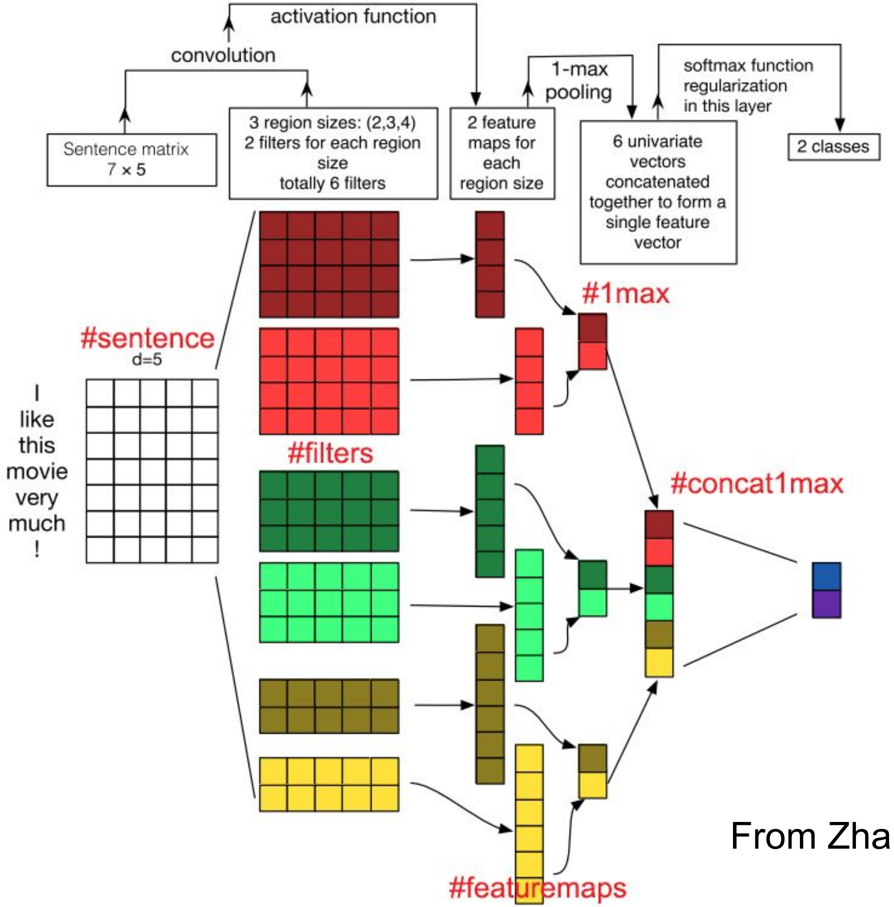
- Use multiple filter weights w (initialize randomly)
- Use different window sizes n
- Then, we can have features for bigrams, tri-grams, 4-grams

Classification after one CNN layer

1. First one convolution, followed by one max-pooling
2. Obtain final features vectors: z = [c_1,....,c_m] where m is the number of filters
3. Apply softmax layers on final vector

$$y = softmax\left(W^{(S)}z + b\right)$$

CNN Framework



From Zhang 2015

Multiple Channels

- Like image, CNN is applied on R-G-B channels
- For NLP, different word embeddings can be regarded as different channels

CNN for NLP

1. n-grams features are important (window size)

2. Location of key n-grams are trivial (pooling)

3. Stack of Convolutional layer or large window size can also capture long-range information