Deep Learning

Classification & Convolutional Neural Networks

Dr. Mohammed Salah Al-Radhi

(slides by: Dr. Bálint Gyires-Tóth)

Copyright

Lecture Overview

1. Data Visualization
2. Classification
    a. Cost function
    b. Confusion matrix
    c. ROC & AUC curves
3. Convolution Neural Network
    a. Stride
    b. Zero padding
    c. Pooling
    d. Dilation
    e. Layers
    f. General architecture

Data Visualization

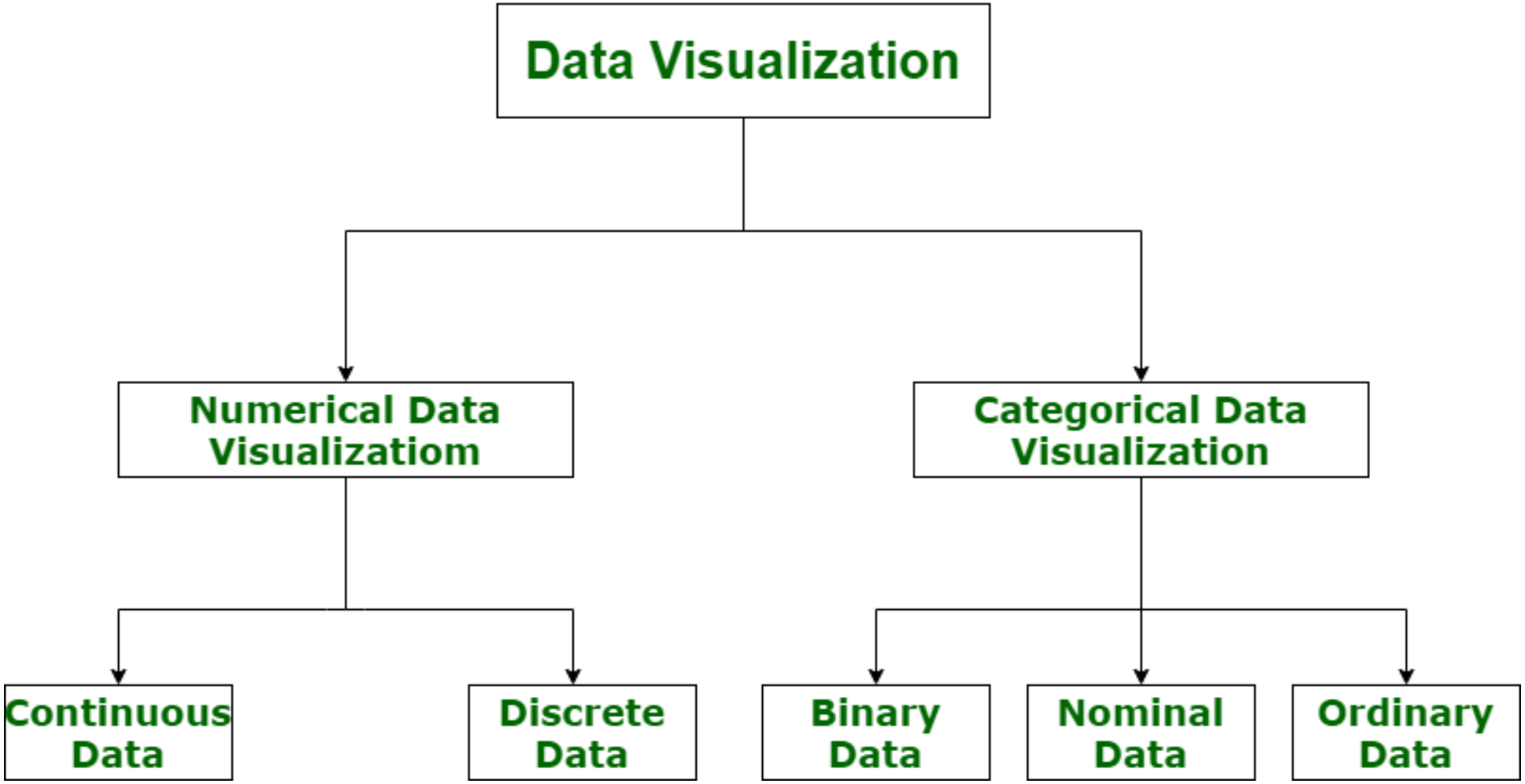
Data Visualization

images by Mark P. Witton

What is Data Visualization?

- presentation of data in a graphic format.
- helps people see patterns and trends.
- used from simple graphs to complex network diagrams.
- puts the Data into the correct context
- Saves Time
- Tells a Data Story and making it more accessible, understandable, and usable.

Data Visualization



Data Visualization

Top Data Visualization Tools:

- Tableau
- Looker
- Zoho Analytics
- Sisense
- IBM Cognos Analytics
- Qlik Sense
- Microsoft Power BI
- SAP Analytics Cloud

Top Data Visualization Libraries Available in Python Libraries:

- Matplotlib
- Plotly
- Seaborn
- Altair
- Bokeh
- Scikit Learn
- Pandas
- YellowBrick

Classification

Is this Classification?

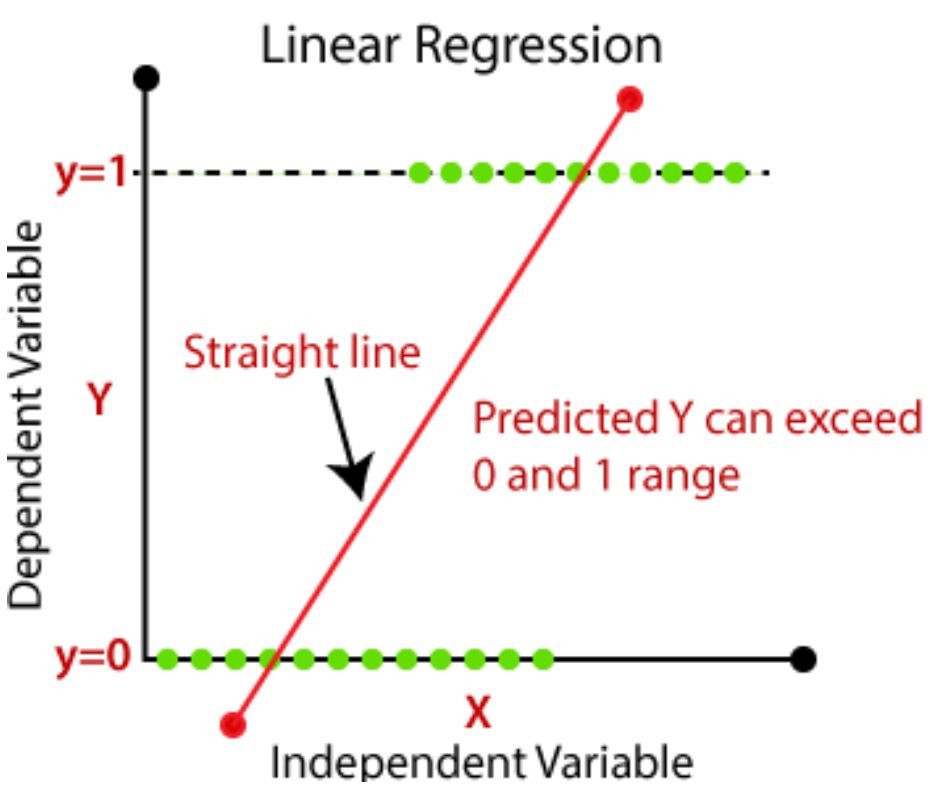
- Is this bank transfer fraudulent? 

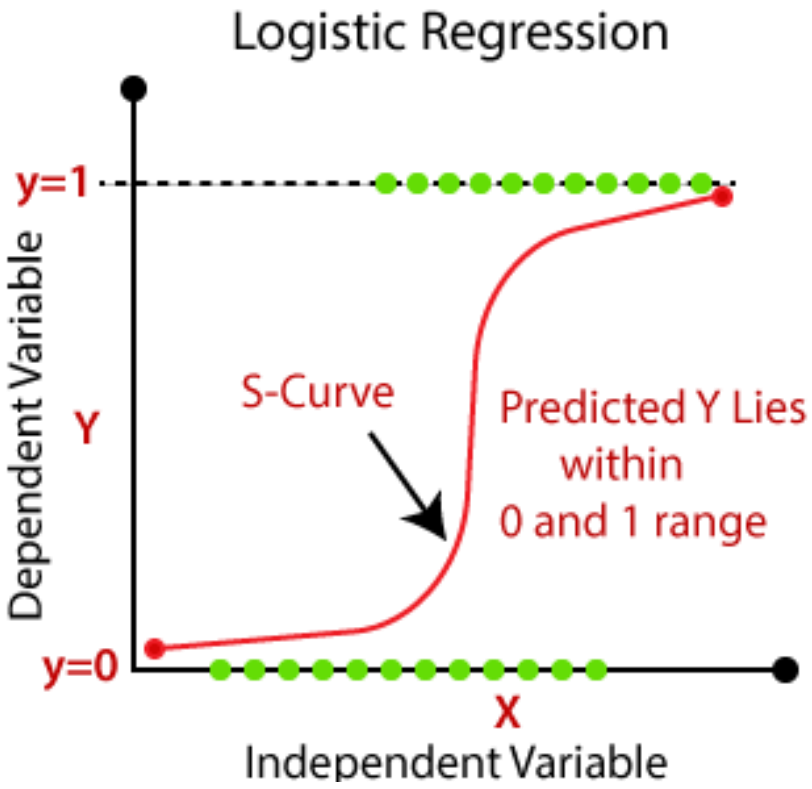
- Is this patient healthy? 
- Will you vote for me, for X, or for Y? 

- Will these two people fit to each other? 
- Is this an apple? 

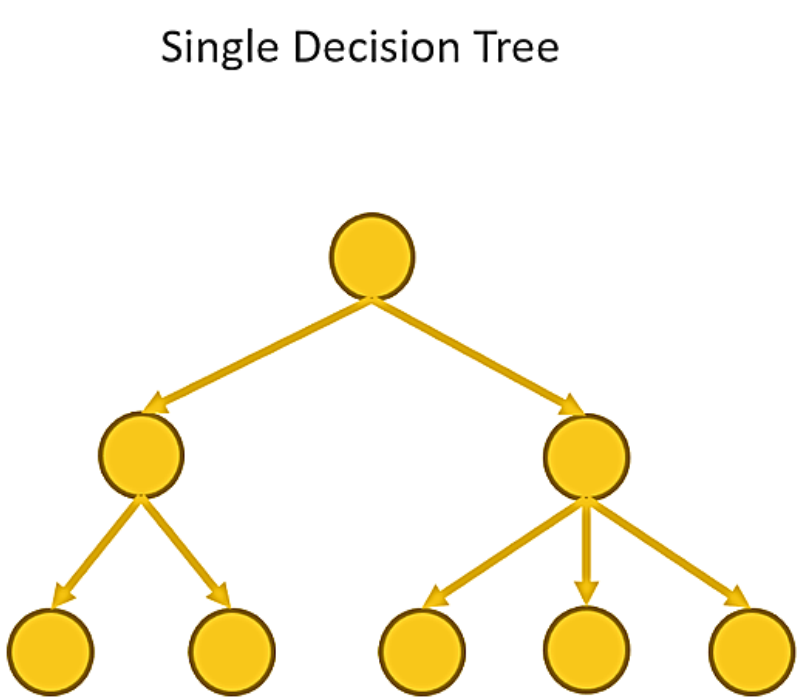
Given a number of examples, identify to which class a given observation belongs to.

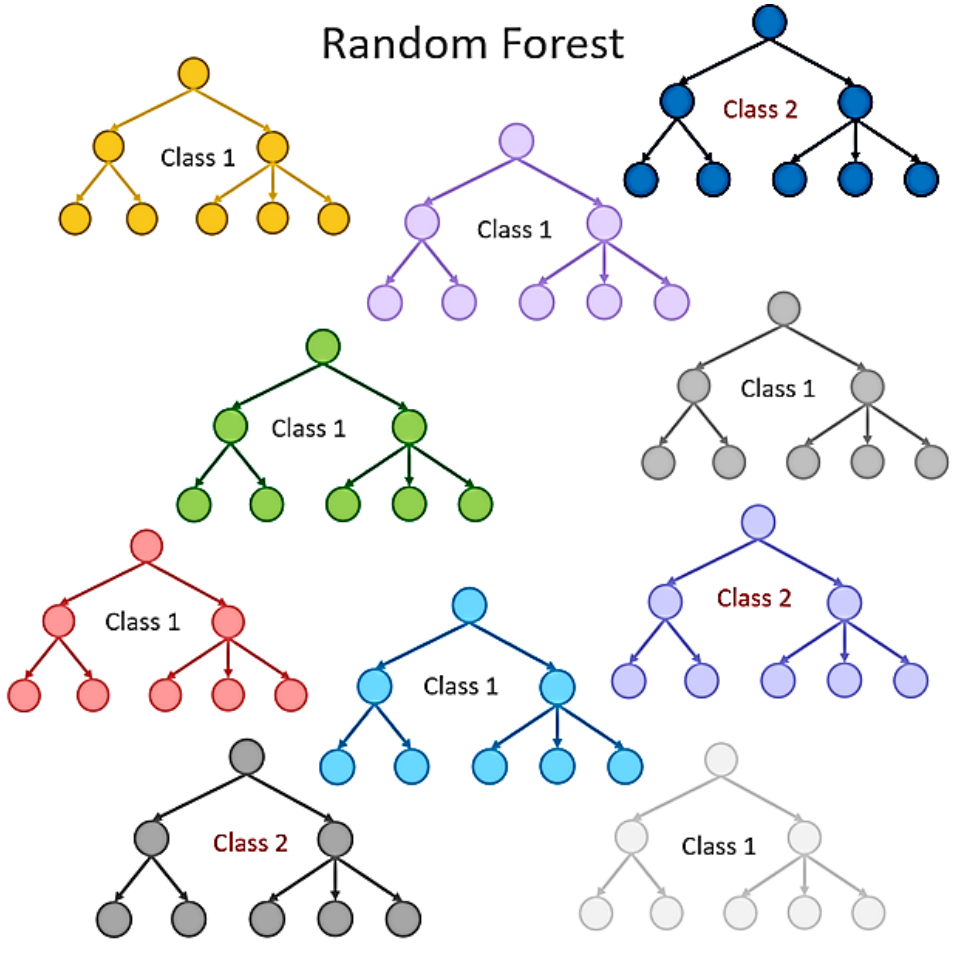
Regression


Linear Regression
y=1
y=0
Y
Dependent Variable
Straight line
Predicted Y can exceed
0 and 1 range
X
Independent Variable


Logistic Regression
y=1
y=0
Y
Dependent Variable
S-Curve
Predicted Y Lies
within
0 and 1 range
X
Independent Variable

Tree


Single Decision Tree


Random Forest
Class 1
Class 2
Class 1
Class 1
Class 1
Class 1
Class 2
Class 1
Class 2
Class 1

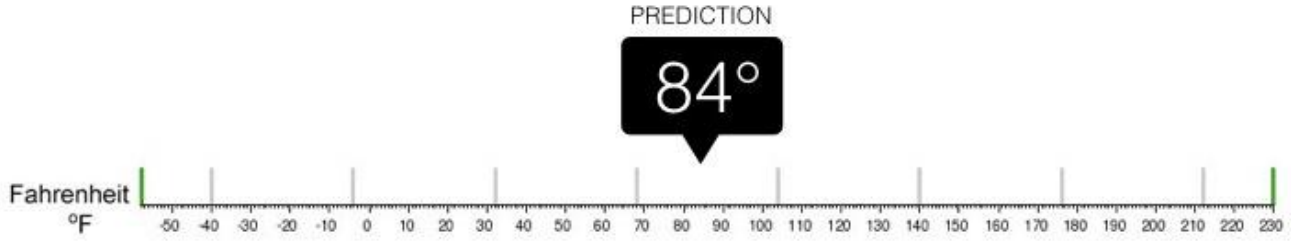
Regression vs. Classification

- **Regression**: predict a continuous output value
  - e.g., temperature of a room

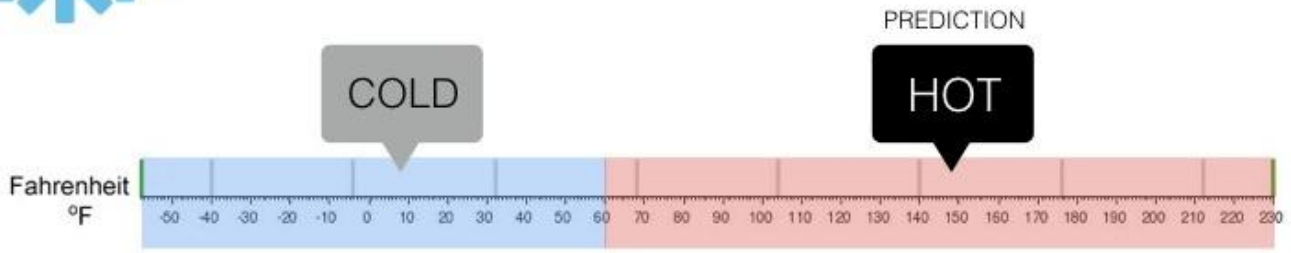
- **Classification**: predicts a discrete value
  - Binary classification: output is either 0 or 1
  - Multi-class classification: set of N classes

Regression vs. Classification

What is the temperature going to be tomorrow?



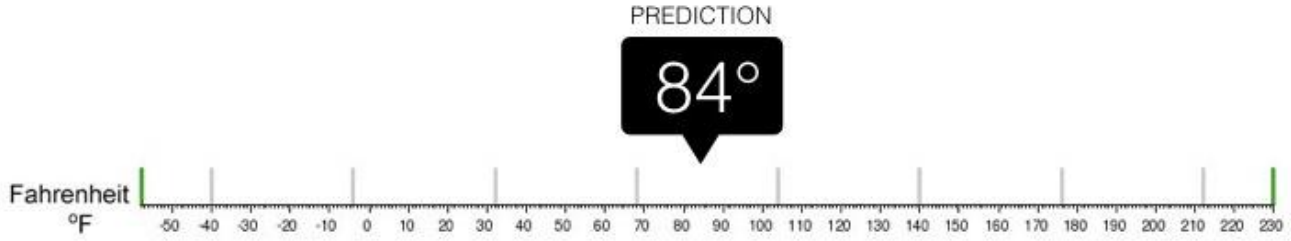
Will it be Cold or Hot tomorrow?



Regression vs. Classification

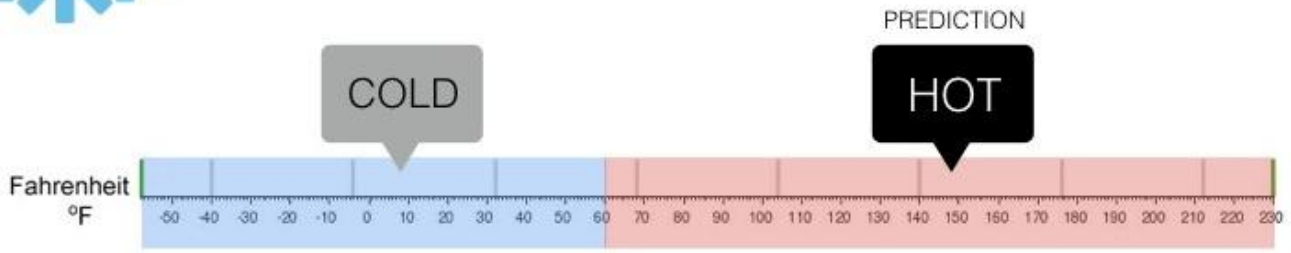
Regression

What is the temperature going to be tomorrow?

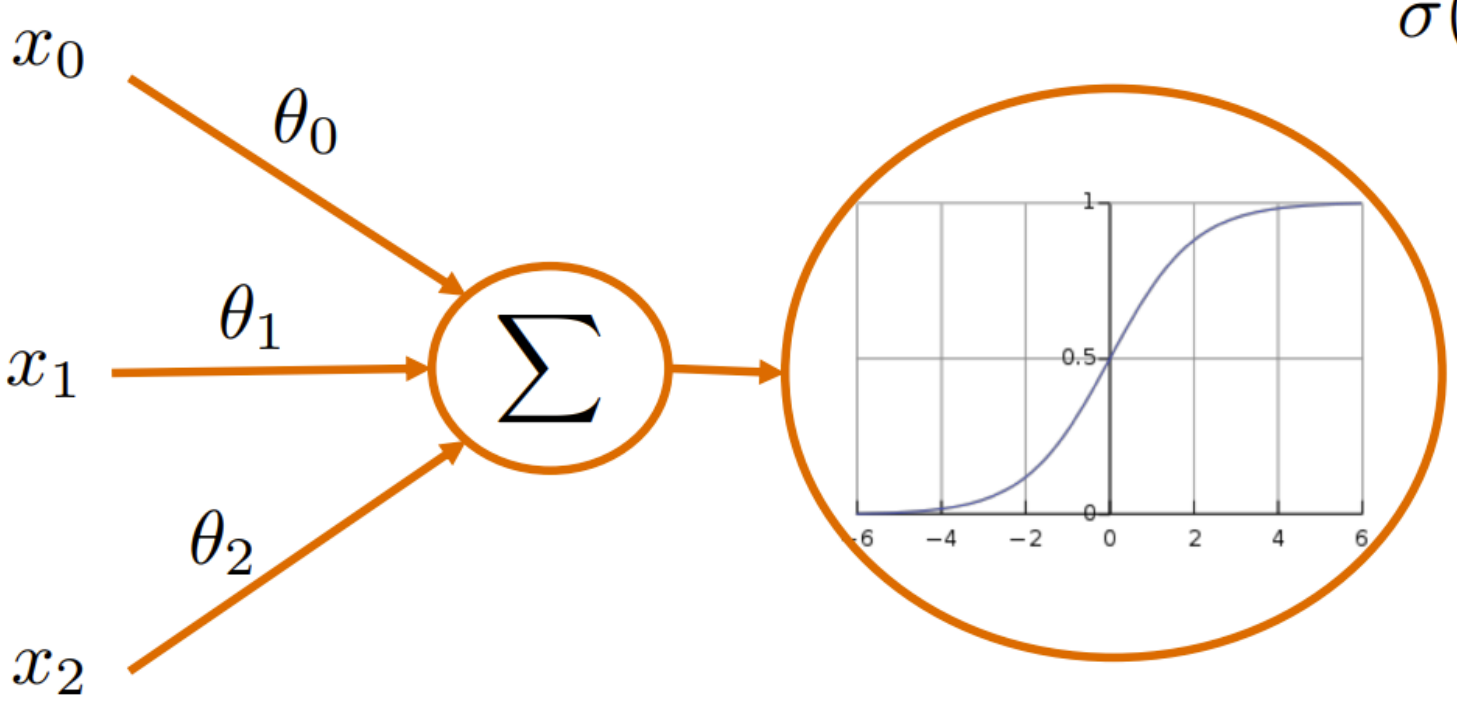


Classification

Will it be Cold or Hot tomorrow?



Sigmoid for binary predictions



$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

1

Can be interpreted as a probability

0

$$p(y_i = 1 | \mathbf{x}_i, \boldsymbol{\theta})$$

Cost functions, output layer types

- Regression
  - Sigmoid / tanh
  - Data scaled to: between 0.01..0.99 or -0.99 .. 0.99
  - Loss function: mean_squared_error

- Classification
  - Two classes: sigmoid (one output) + binary_crossentropy
  - Multiple (k) labels: sigmoid (k outputs, multihot encoding) + binary_crossentropy
  - Multiple (n) classes: softmax (n outputs, onehot encoding) + categorical_crossentropy

Classification vs regression

- Regression: sigmoid / tanh / linear + MSE
- Classification: softmax + cross-entropy
- Cost function: cross-entropy

$$C = -\sum_{n=1}^{M}\sum_{i=1}^{K} y_i^{(n)} \cdot \ln\left(\hat{y}_i^{(n)}\right)$$

M: number of samples
K: number of classes

- SoftMax: converts numbers to „probabilities” (with sum of 1)

$$y_i(x) = \frac{e^{x_i}}{\sum_{k=1}^{K} e^{x_k}} \qquad i = 1 \dots K$$

Cross-entropy for classification 1

Y_target (OneHot)			Y_predicted (probability)			Class	Correct?
1	0	0	0.1	0.1	0.8	car	no
0	1	0	0.3	0.4	0.3	rocket	yes
0	0	1	0.3	0.3	0.4	airplane	yes

Classification accuracy: $\frac{2}{3} = 0.6$

Cross-entropy:

$$-\{(\ln(0.1)\cdot 1)+(\ln(0.1)\cdot 0)+(\ln(0.8)\cdot 0)\} = -\ln(0.1)$$

$$-\{(\ln(0.3)\cdot 0)+(\ln(0.4)\cdot 1)+(\ln(0.3)\cdot 0)\} = -\ln(0.4)$$

$$-\{(\ln(0.3)\cdot 0)+(\ln(0.3)\cdot 0)+(\ln(0.4)\cdot 1)\} = -\ln(0.4)$$

Average cross-entropy: $\frac{-(\ln(0.1)+\ln(0.4)+\ln(0.4))}{3} \approx 1.38$

MSE:

$$(0.1-1)^2+(0.1-0)^2+(0.8-0)^2 = 1.46$$
$$(0.3-0)^2+(0.4-1)^2+(0.3-0)^2 = 0.54$$
$$(0.3-0)^2+(0.3-0)^2+(0.4-1)^2 = 0.54$$

$$\frac{1.46+0.54+0.54}{3} \approx 0.847$$

Cross-entropy for classification 2

Y_target (OneHot)			Y_predicted (probability)			Class	Correct?
1	0	0	0.3	0.4	0.3	Car	no
0	1	0	0.1	0.8	0.1	Rocket	yes
0	0	1	0.1	0.1	0.8	Airplane	yes

Classification accuracy: $\frac{2}{3} = 0.6$

Cross-entropy:

$$-\{(\ln(0.3)\cdot 1)+(\ln(0.4)\cdot 0)+(\ln(0.3)\cdot 0)\} = -\ln(0.3)$$

$$-\{(\ln(0.1)\cdot 0)+(\ln(0.8)\cdot 1)+(\ln(0.1)\cdot 0)\} = -\ln(0.8)$$

$$-\{(\ln(0.1)\cdot 0)+(\ln(0.1)\cdot 0)+(\ln(0.8)\cdot 1)\} = -\ln(0.8)$$

Average cross-entropy: $\frac{-(\ln(0.1)+\ln(0.8)+\ln(0.8))}{3} \approx 0.91$

MSE:

$$(0.3-1)^2+(0.4-0)^2+(0.3-0)^2 = 0.74$$
$$(0.1-0)^2+(0.8-1)^2+(0.1-0)^2 = 0.06$$
$$(0.1-0)^2+(0.1-0)^2+(0.8-1)^2 = 0.06$$

$$\frac{0.74+0.06+0.06}{3} = 0.286$$

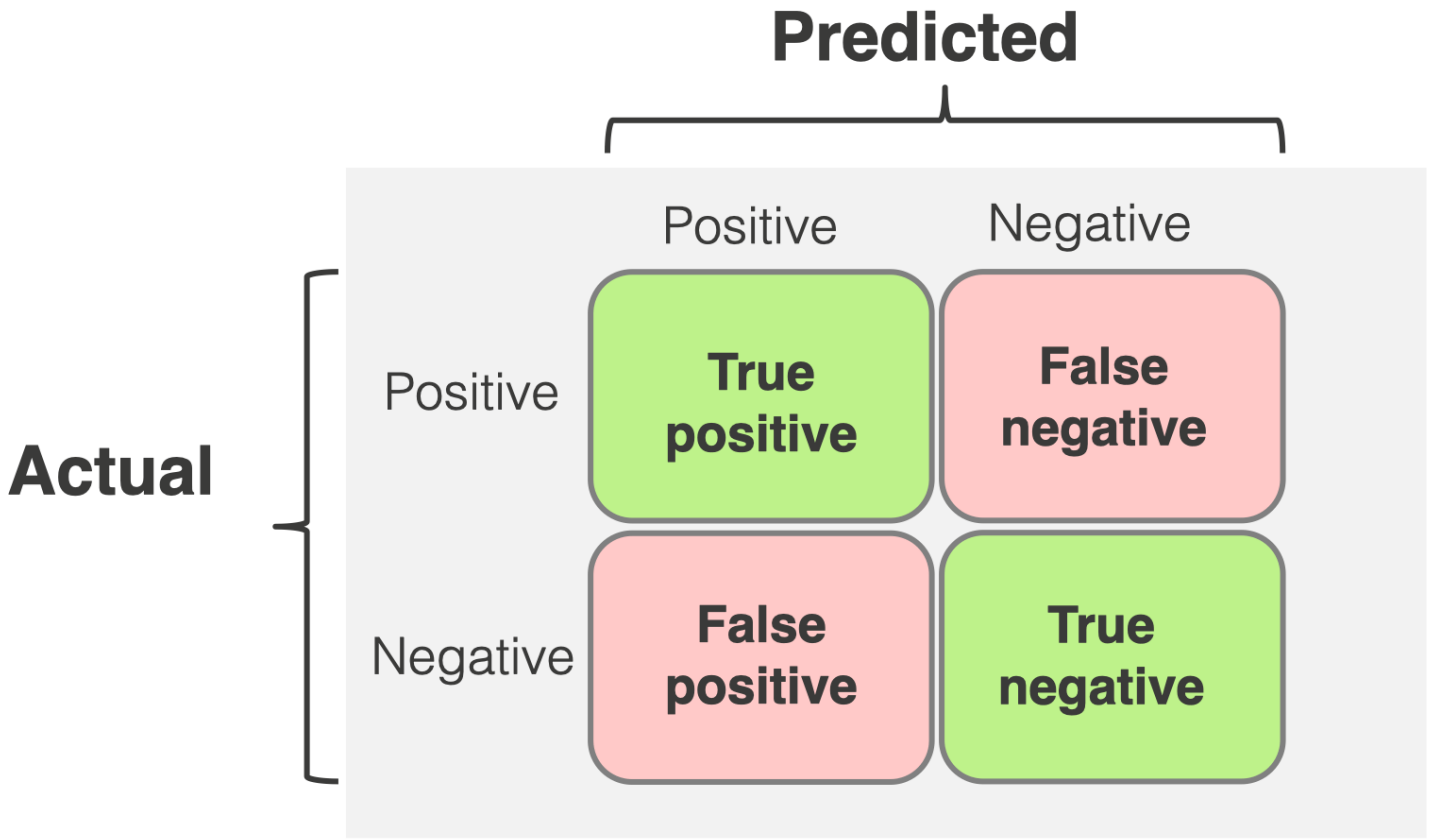
Evaluation methods

Classification: evaluation methods

- Confusion matrix (TP, TN, FP, FN)
- Classification metrics
  - Recall
  - Precision
  - Accuracy
  - F1
- ROC
- AUC

Confusion matrix

- Easy to read, includes all results.



Confusion matrix

- E.g. 3 classes:

		Y_predicted		
	Class	Rocket	Car	Airplane
Y_target	Rocket	**15**	**3**	**8**
	Car	**4**	**23**	**1**
	Airplane	**9**	**2**	**19**

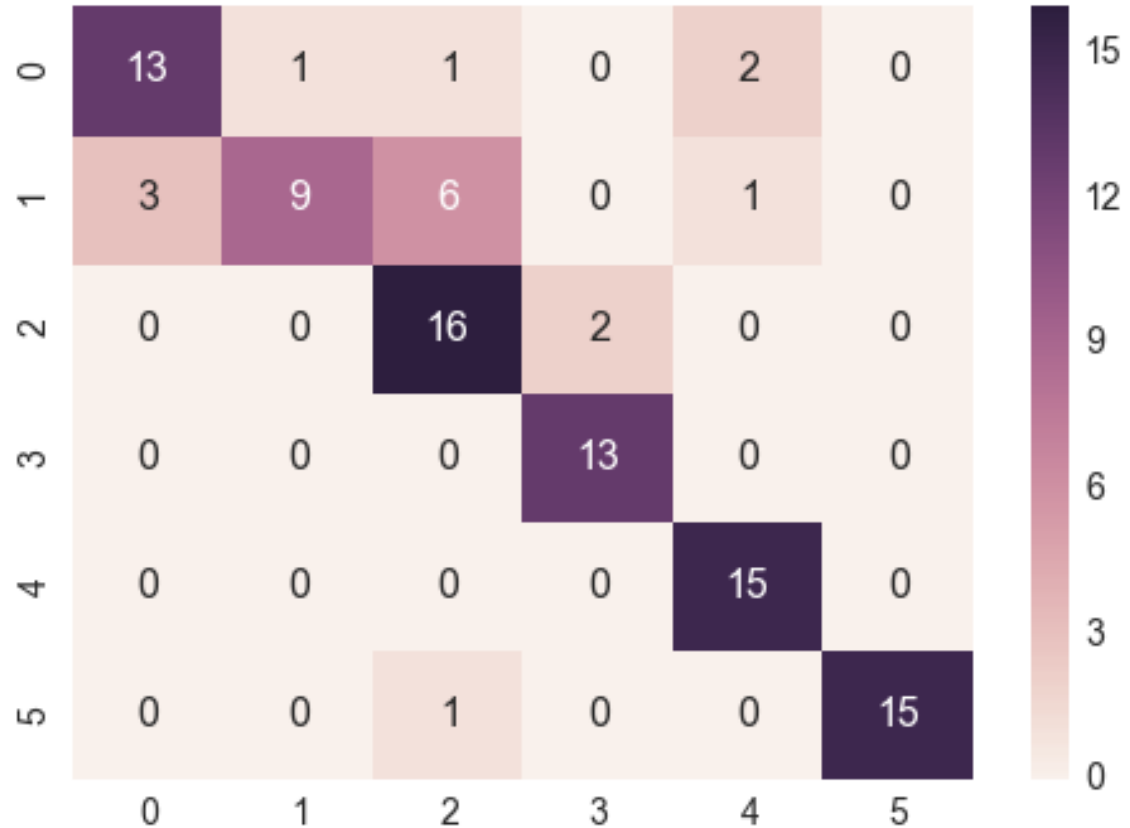
Confusion matrix: binary classification

Rocket

		Y_predicted	
	Condition	Positive	Negative
Y_target	Positive	15 True Positive (TP)	3+8=11 False Negative (FN)
	Negative	4+9=13 False Positive (FP)	23+19+1+2=42 True Negative (TN)

Pl. shoot, if rocket! (to protect the city):

- **True Positive (TP)**: We correctly identified it as a rocket (✓).
- **True Negative (TN)**: We correctly identified it as not a rocket (✓).
- **False Positive (FP)**: We incorrectly identified it as a rocket (✗).
- **False Negative (FN)**: We incorrectly identified it as not a rocket (✗).




```
sns.heatmap(confusion_matrix, annot=True)
```

Classification metrics

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision or } = \frac{TP}{TP + FP}$$

$$\text{Recall or TPR } = \frac{TP}{TP + FN}$$

$$\text{Specificity or TNR } = \frac{TN}{TN + FP}$$

$$\text{F1 Score } = 2 * \frac{\text{Recall } \times \text{Precision}}{\text{Recall} + \text{Precision}}$$

$$\text{False Positive Ratio(FPR)} = \frac{FP}{FP+FN}$$

More: MAP, ROC, AUROC

SkLearn Precision/Recall/F1 means

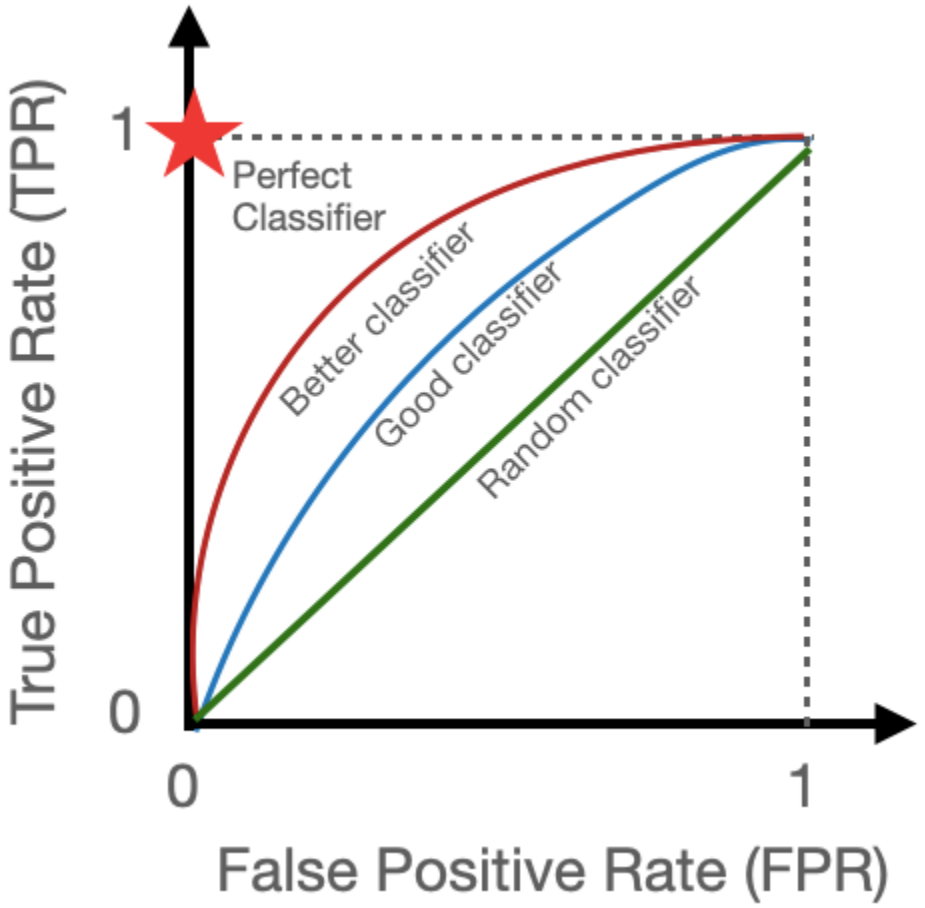
- **Macro:** calculate the value for all classes separately, and take the average.
- **Micro:** calculate the values for all classes, and calculate precision/recall/F1 from that.
- **Weighted:** weights the classes based as a function of the imbalances. Those classes which have more values have higher weight.

ROC Curve, AUC

Receiver Operating Characteristic

TP and FP rates under various Decision threshoulds

AUC: Area Under Curve



Convolution Neural Network

“CNN” vs “ConvNet”

- There are many papers that use either phrases, but:
- “ConvNet” is the preferred term, since “CNN” clashes with other things called CNN ☺

Deep learning news station

- Video generation based on text: https://makeavideo.studio/

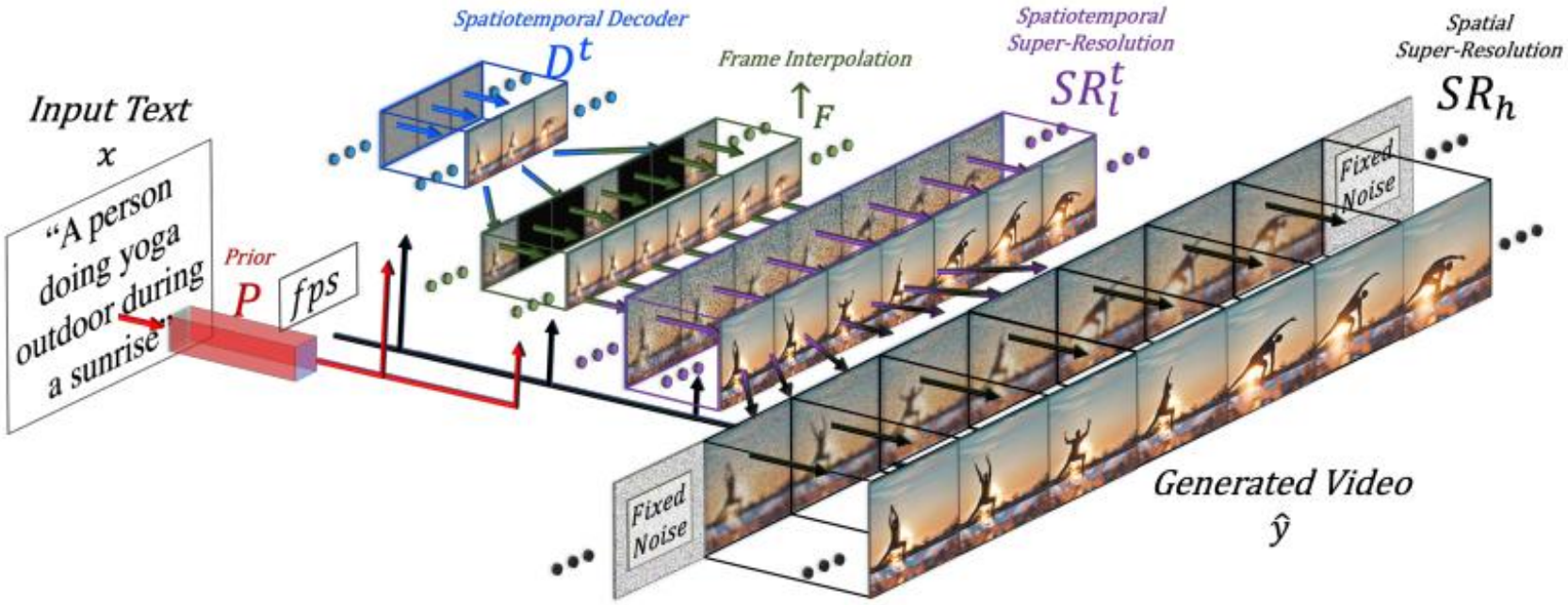
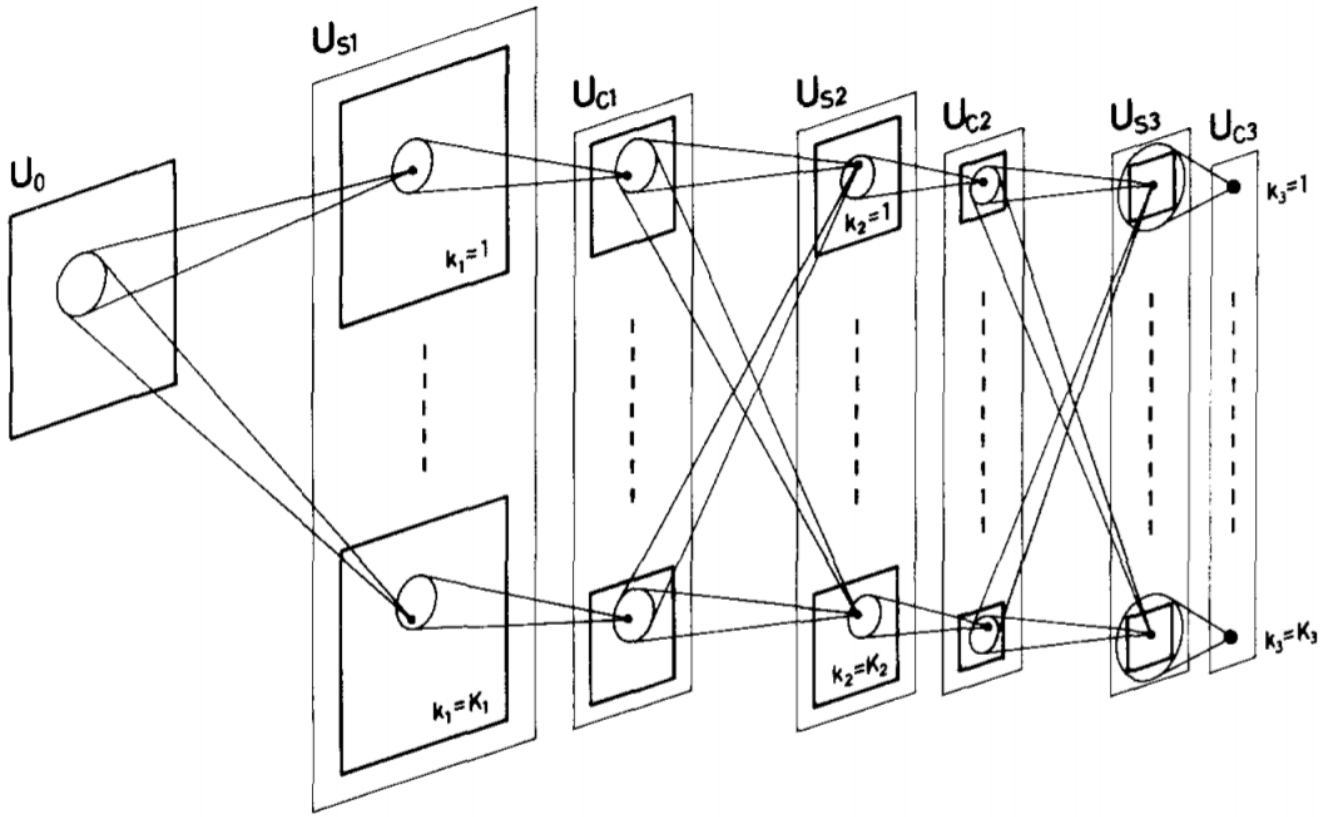


Figure 2: **Make-A-Video high-level architecture.** Given input text x translated by the prior P into an image embedding, and a desired frame rate fps, the decoder D^{t} generates 16 64×64 frames, which are then interpolated to a higher frame rate by $\uparrow_F$, and increased in resolution to 256×256 by SR_l^t and 768×768 by SR_h, resulting in a high-spatiotemporal-resolution generated video $\hat{y}$.

Major milestones

- **Neocognitron:** Fukushima, Kunihiko, and Sei Miyake. "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition." In Competition and cooperation in neural nets, pp. 267-285. Springer, Berlin, Heidelberg, 1982.



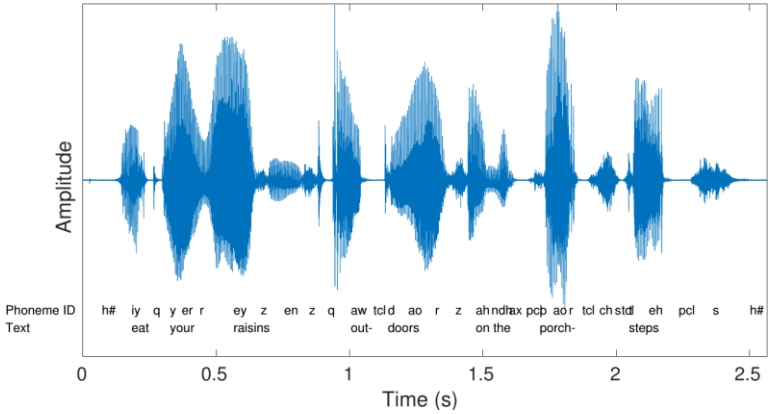
Major milestones

- **LeNet:** LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, *3361*(10), 1995.
- **LeNet5:** LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.
- **AlexNet:** Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

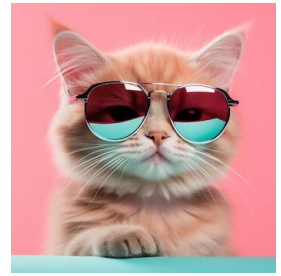
But what is CNN?

Version of deep neural networks designed for signals:

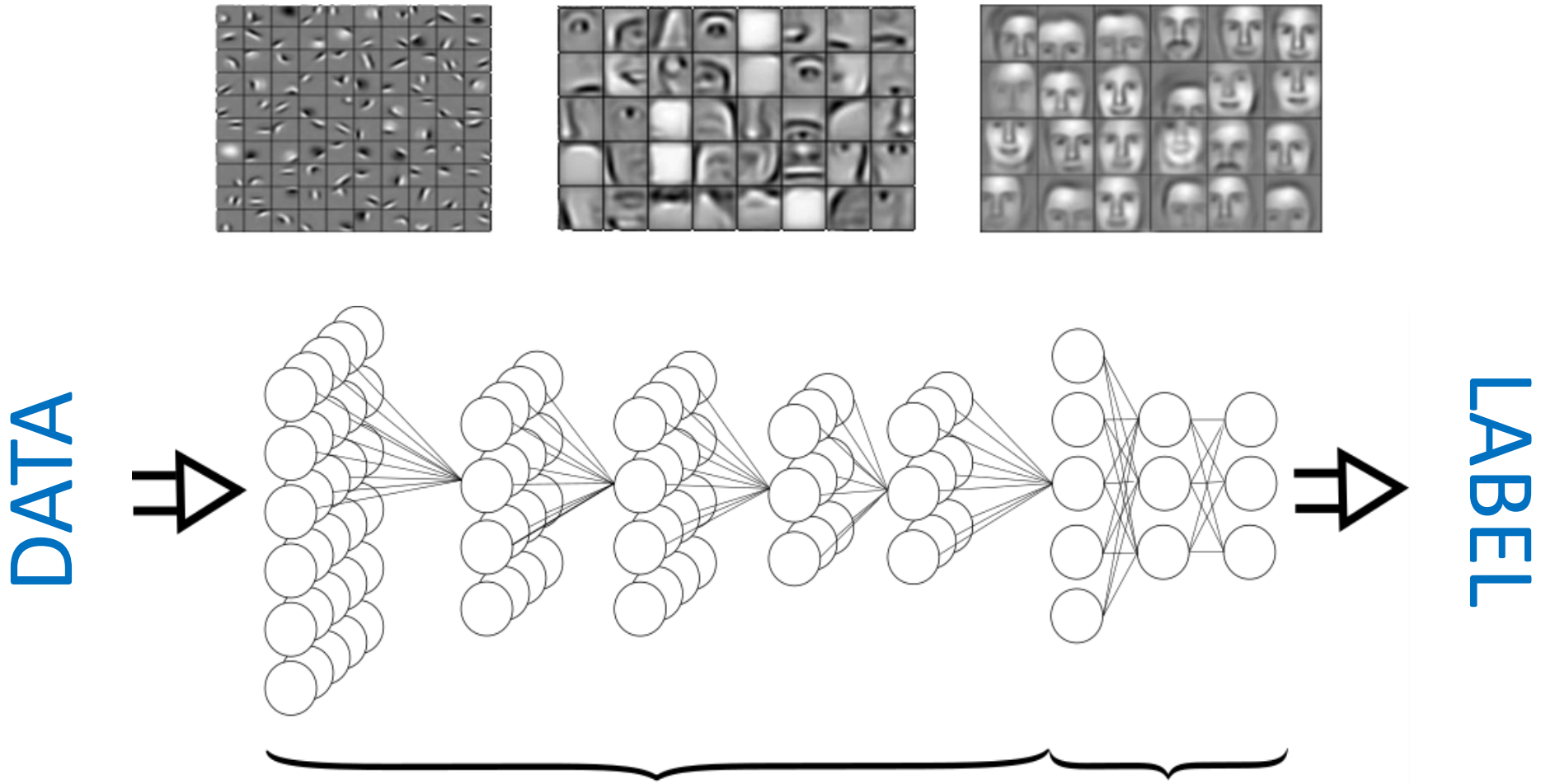
- 1D signals (e.g., speech waveforms)



- 2D signals (e.g., images)



General structure of a CNN network

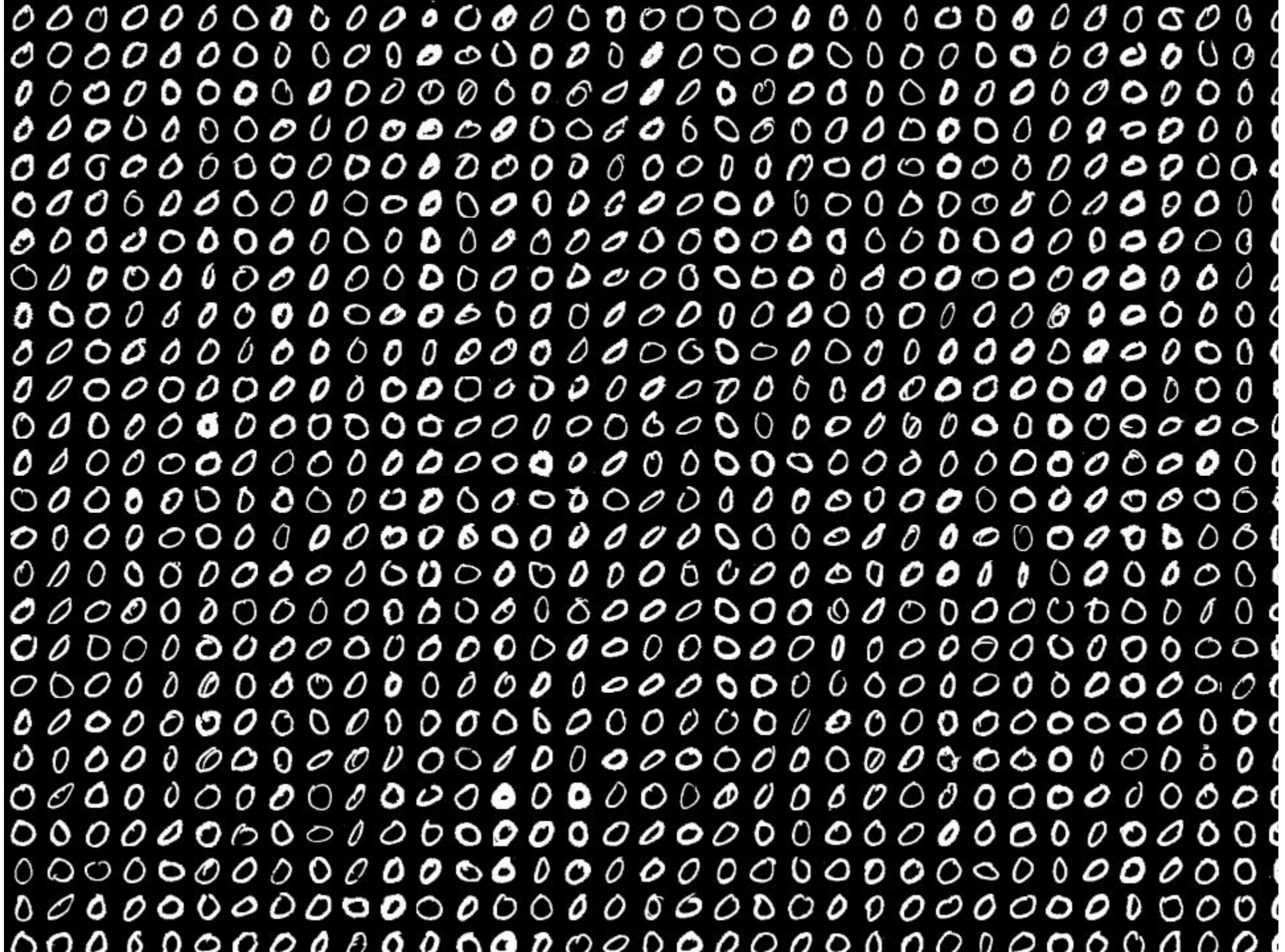


MNIST: Yann LeCun handwriting database

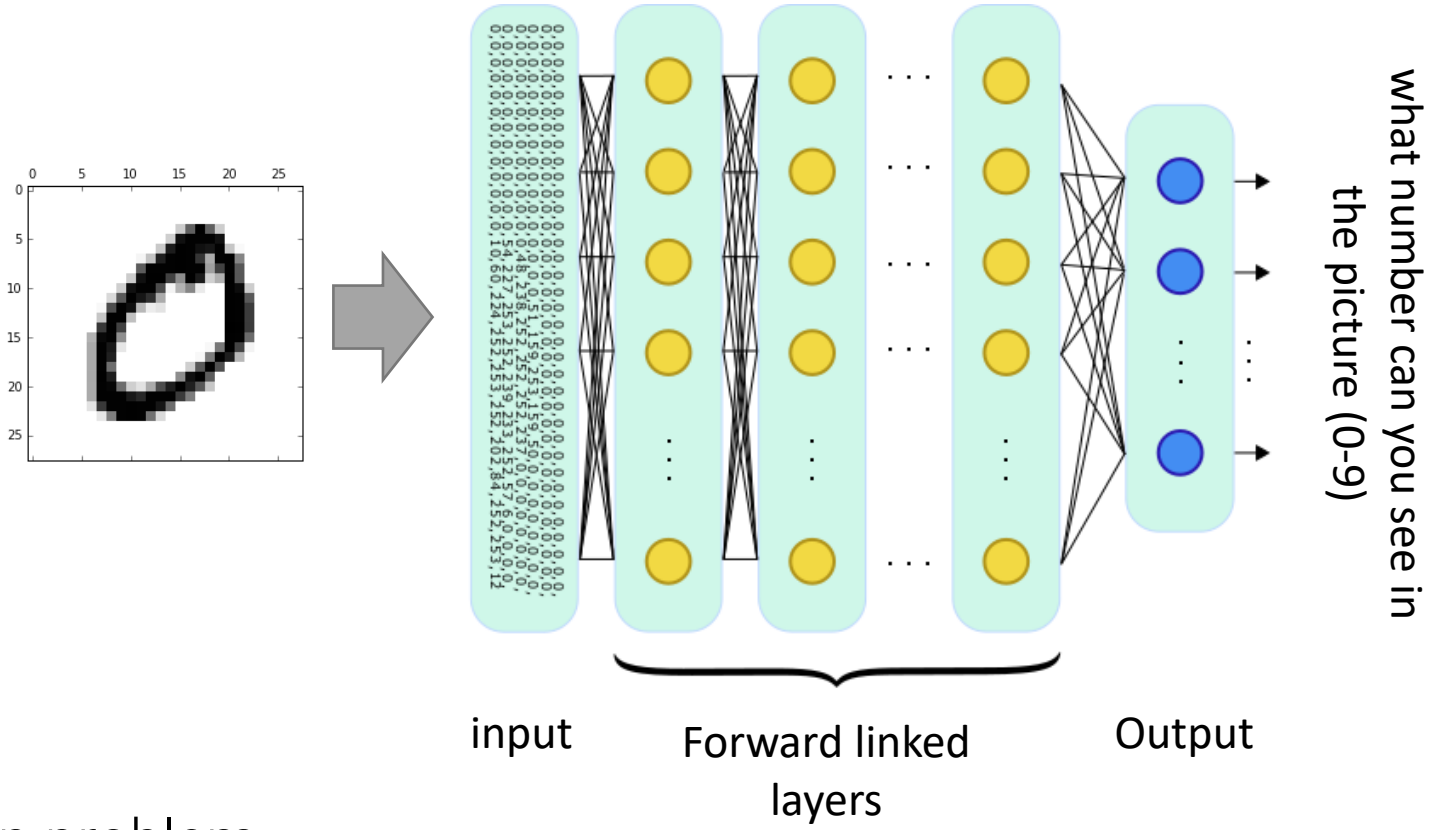
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,51,159,253,159,50,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,48,238,2
52,252,252,237,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,54,227,253,252,239,233,252,57,6,0,0,0,0,0,0,0,0,0,0
,0,0,0,0,0,0,0,10,60,224,252,253,252,202,84,252,253,122,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,163,252,252,252,25
3,252,252,96,189,253,167,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,51,238,253,253,190,114,253,228,47,79,255,168,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,48,238,252,252,179,12,75,121,21,0,0,253,243,50,0,0,0,0,0,0,0,0,0,0,0,0,0,38,165,253,
233,208,84,0,0,0,0,0,0,253,252,165,0,0,0,0,0,0,0,0,0,0,0,0,7,178,252,240,71,19,28,0,0,0,0,0,0,253,252,195,0,
0,0,0,0,0,0,0,0,0,0,0,57,252,252,63,0,0,0,0,0,0,0,0,0,253,252,195,0,0,0,0,0,0,0,0,0,0,0,0,198,253,190,0,0,0,
0,0,0,0,0,0,0,255,253,196,0,0,0,0,0,0,0,0,0,0,0,76,246,252,112,0,0,0,0,0,0,0,0,0,0,253,252,148,0,0,0,0,0,0,0
,0,0,0,0,85,252,230,25,0,0,0,0,0,0,0,0,7,135,253,186,12,0,0,0,0,0,0,0,0,0,0,0,85,252,223,0,0,0,0,0,0,0,0,7,1
31,252,225,71,0,0,0,0,0,0,0,0,0,0,0,0,85,252,145,0,0,0,0,0,0,0,48,165,252,173,0,0,0,0,0,0,0,0,0,0,0,0,0,0,86
,253,225,0,0,0,0,0,0,114,238,253,162,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,85,252,249,146,48,29,85,178,225,253,223,1
67,56,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,85,252,252,252,229,215,252,252,252,196,130,0,0,0,0,0,0,0,0,0,0,0,0,0,0
,0,0,28,199,252,252,253,252,252,233,145,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,25,128,252,253,252,141,37,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0

MNIST

```
[[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  51, 159, 253, 159,  50,   0,   0,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  48, 238, 252, 252, 252, 237,   0,   0,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  54, 227, 253, 252, 239, 233, 252,  57,   6,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  10,  60, 224, 252, 253, 252, 202,  84, 252, 253, 122,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 163, 252, 252, 252, 253, 252, 252,  96, 189, 253, 167,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  51, 238, 253, 253, 190, 114, 253, 228,  47,  79, 255, 168,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,  48, 238, 252, 252, 179,  12,  75, 121,  21,   0,   0, 253, 243,  50,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,  38, 165, 253, 233, 208,  84,   0,   0,   0,   0,   0,   0, 253, 252, 165,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   7, 178, 252, 240,  71,  19,  28,   0,   0,   0,   0,   0,   0, 253, 252, 195,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,  57, 252, 252,  63,   0,   0,   0,   0,   0,   0,   0,   0,   0, 253, 252, 195,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0, 198, 253, 190,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 255, 253, 196,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,  76, 246, 252, 112,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 253, 252, 148,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,  85, 252, 230,  25,   0,   0,   0,   0,   0,   0,   0,   0,   7, 135, 253, 186,  12,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,  85, 252, 223,   0,   0,   0,   0,   0,   0,   0,   0,   7, 131, 252, 225,  71,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,  85, 252, 145,   0,   0,   0,   0,   0,   0,   0,  48, 165, 252, 173,   0,   0,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,  86, 253, 225,   0,   0,   0,   0,   0,   0, 114, 238, 253, 162,   0,   0,   0,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,  85, 252, 249, 146,  48,  29,  85, 178, 225, 253, 223, 167,  56,   0,   0,   0,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,  85, 252, 252, 252, 229, 215, 252, 252, 252, 196, 130,   0,   0,   0,   0,   0,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,  28, 199, 252, 252, 253, 252, 252, 233, 145,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,  25, 128, 252, 253, 252, 141,  37,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 0,   0],
 [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 0,   0]]
```



Feedforward network approach



Main problem

- Parameter space too large
  - Pl. 200x200x3=120000 input, additional connections …)
- It recognizes data, not patterns

Convolutional neural network approach

- The convolution operation involves element-wise **multiplication** of a filter (also known as a kernel) with a corresponding region of the input, followed by the **summation** of these products to produce a single value in the output feature map.

CNN – Producing Feature Maps

Original

Sharpen

Edge Detect

"Strong" Edge Detect

CNN main ideas

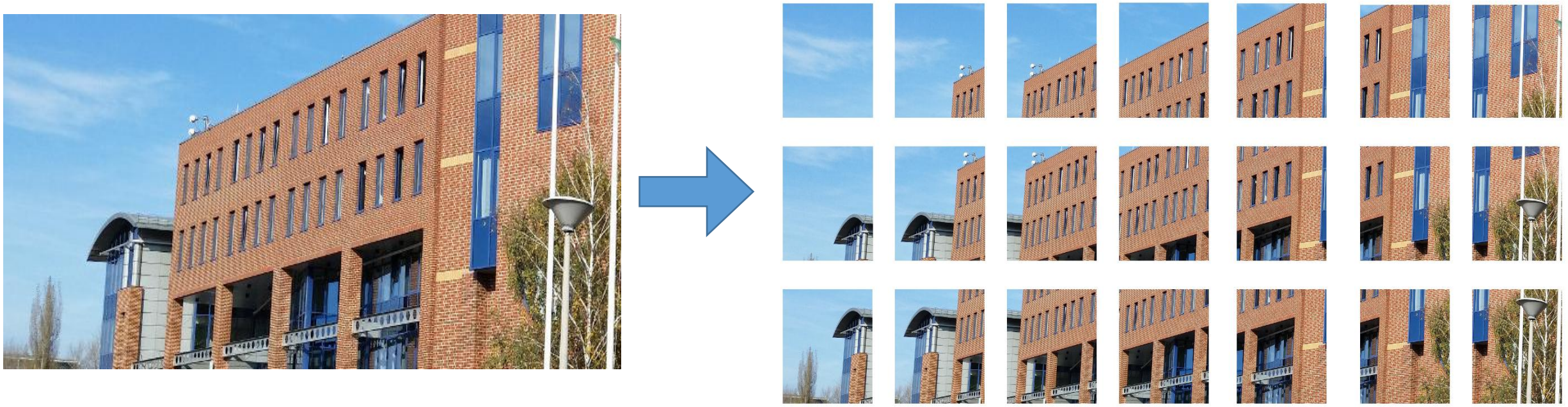
Dependence on space and/or time

Shared weights

Multiple (shared) weight matrix per layer

Spatial dependence

- By breaking down the data into smaller pieces, we train the network at different levels.

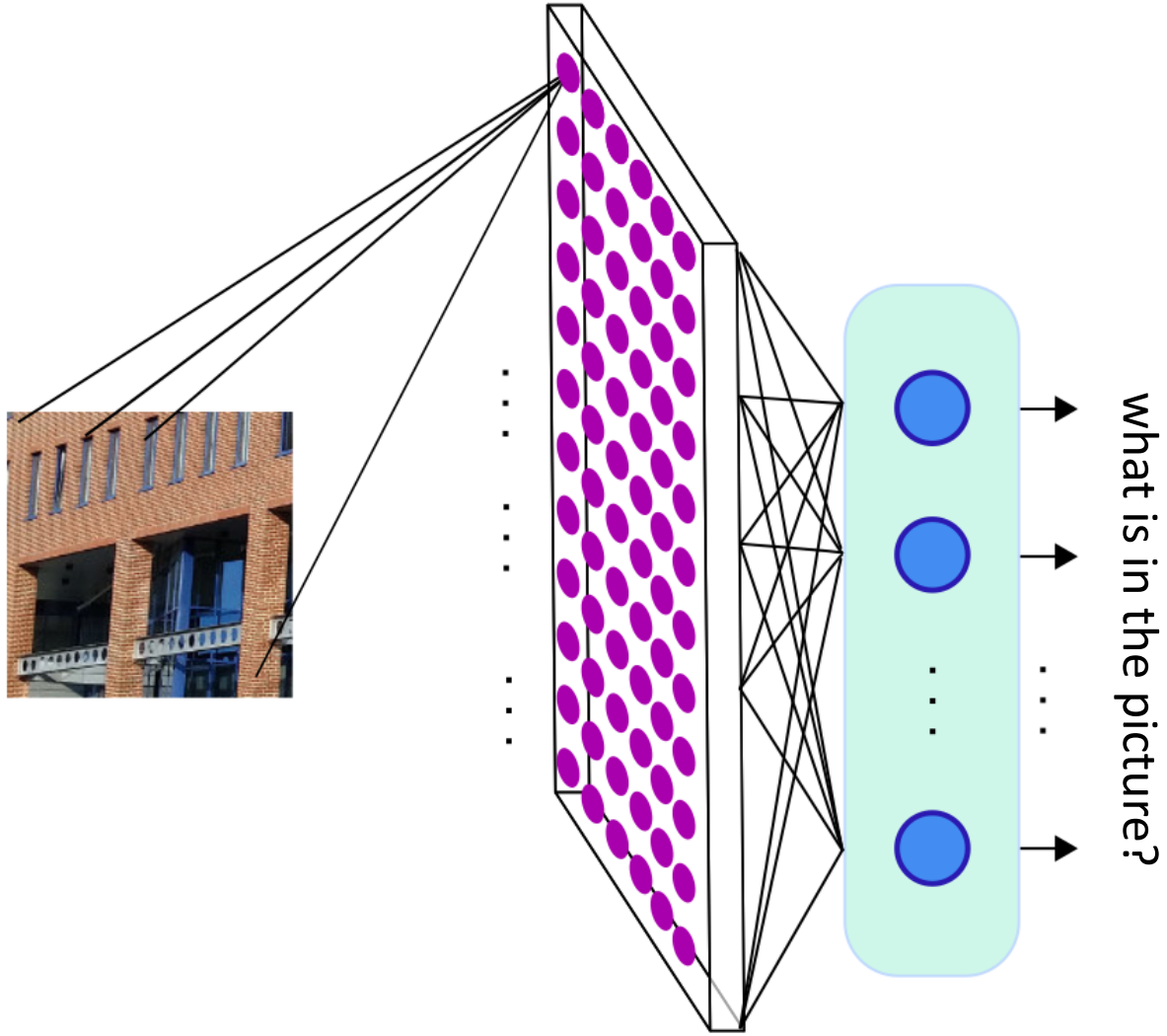


Building blocks: convolutional layer 1

Training the net with pieces of the original image.

Pl. 20x20 px piece, then 400 weight + 1 bias.

Now the neurons of the feed-forward network are arranged in 2D.



Building blocks: convolutional layer 2

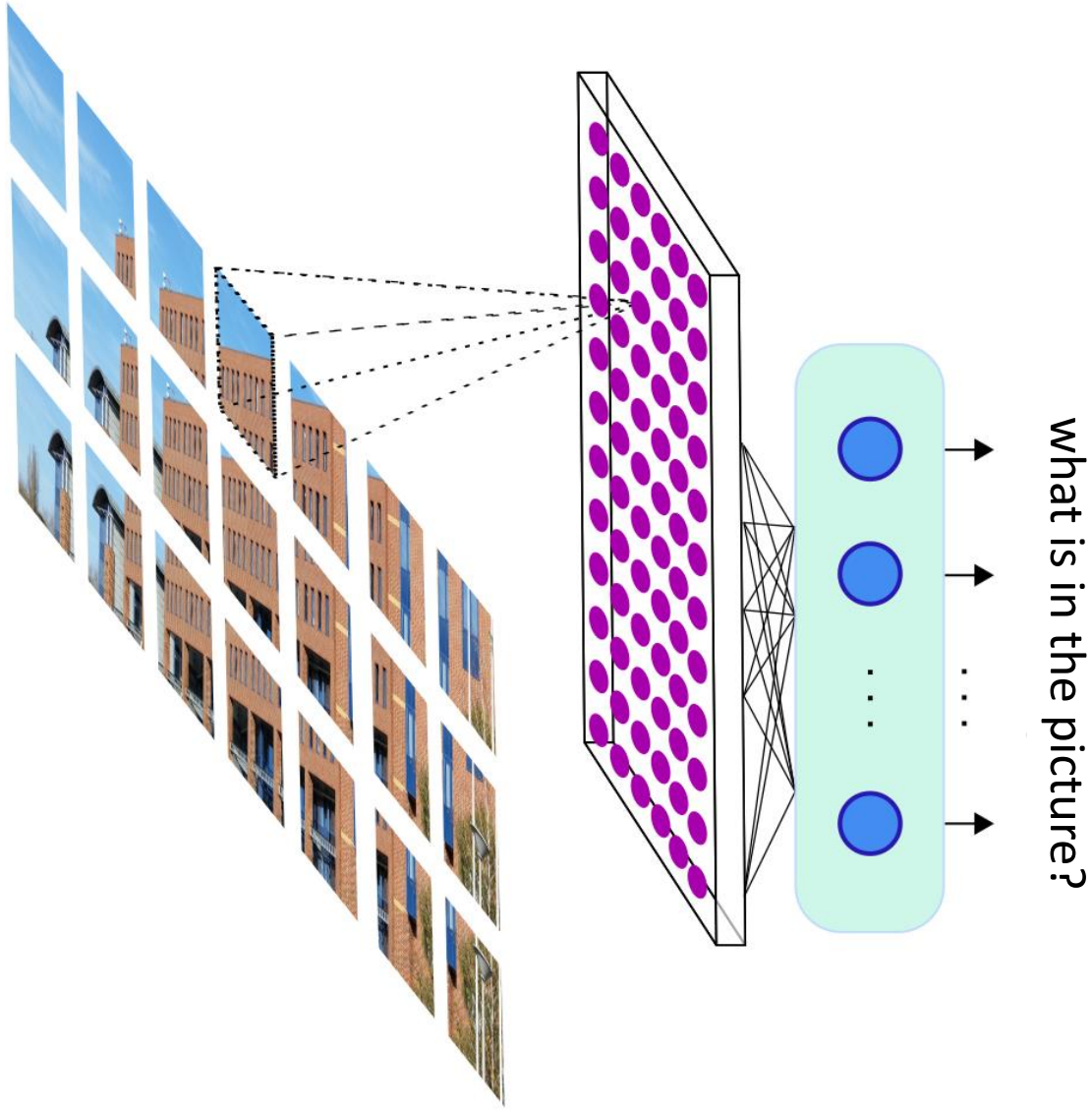
Shared weights:

We use the same weights for each piece of the original image.

If a feature is important at one point in the picture, it will be important elsewhere!

Why we call it CNN?

Convolution of input and shared weights (filters).



Hyperparameters (conv2d)

INPUT: `{depth}@{input_x}x{input_y}`

FILTER / KERNEL: `{filter_x}x{filter_y}@{filter_depth}`

STRIDE: `{stride_x}x{stride_y}`

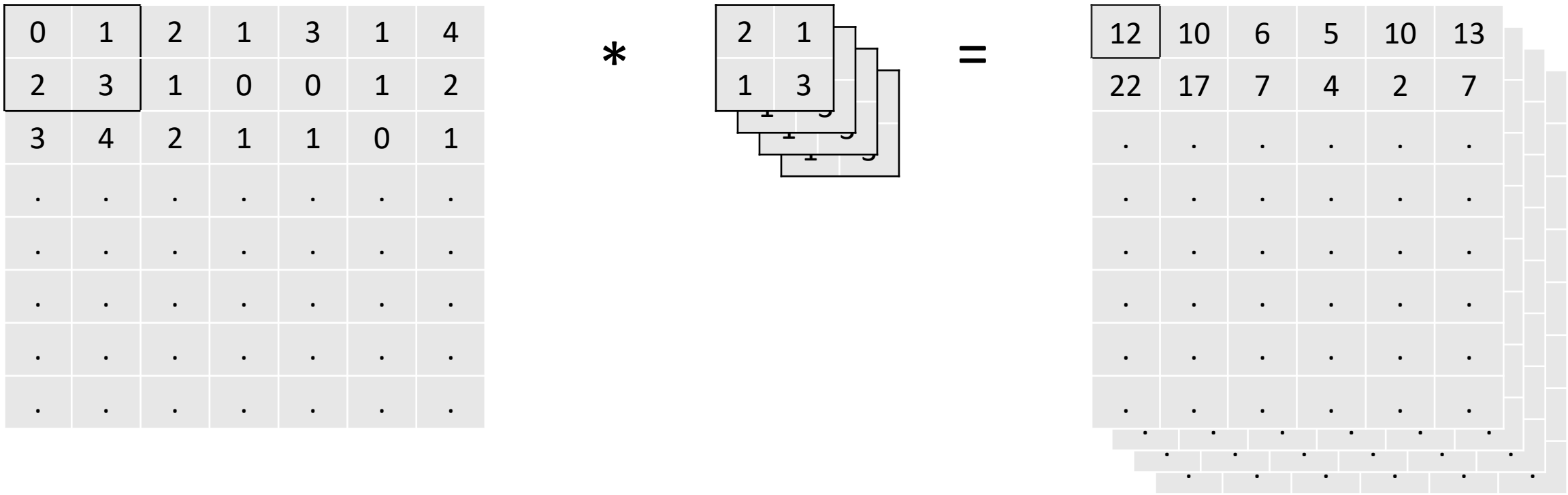
ZERO PADDING: `{zeropadding_x}x{zeropadding_y}`

DILATATION: `{dilatation_x}x{dilatation_y}`

Output:

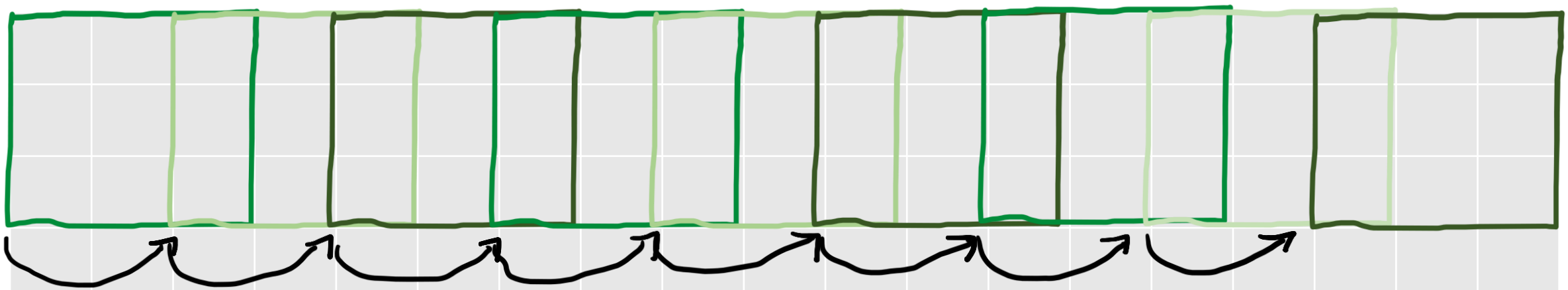
```
{filter_depth}@
{((input_x-filter_x+2*zeropadding_x)/stride_x)+1}x
{((input_y-filter_y+2*zeropadding_y)/stride_y)+1}
```

Convolution operation

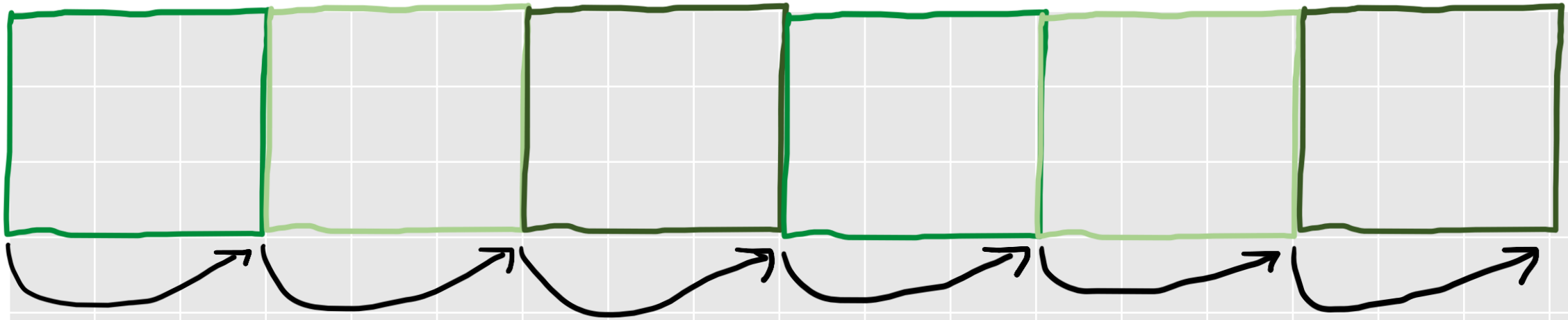


Input Kernel Output

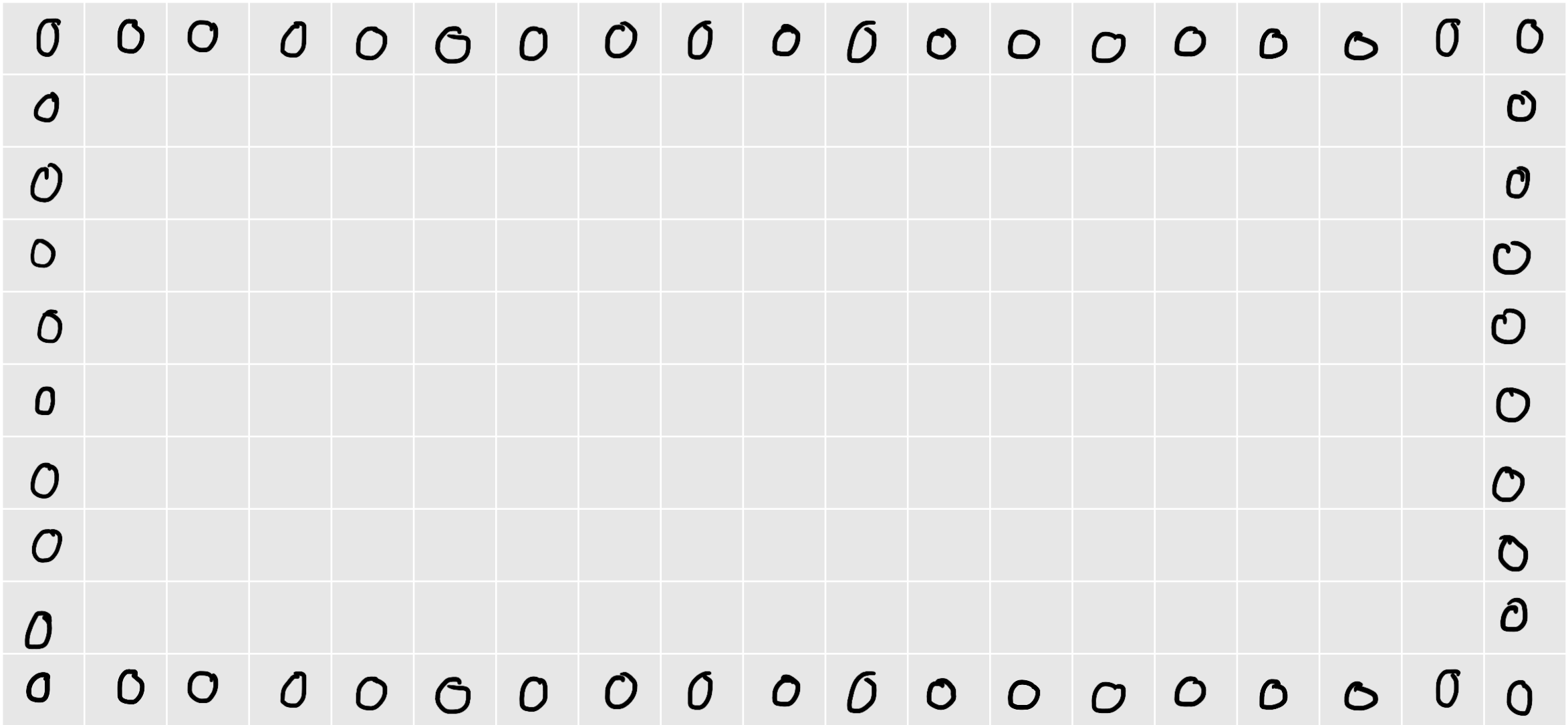
Stride = 2



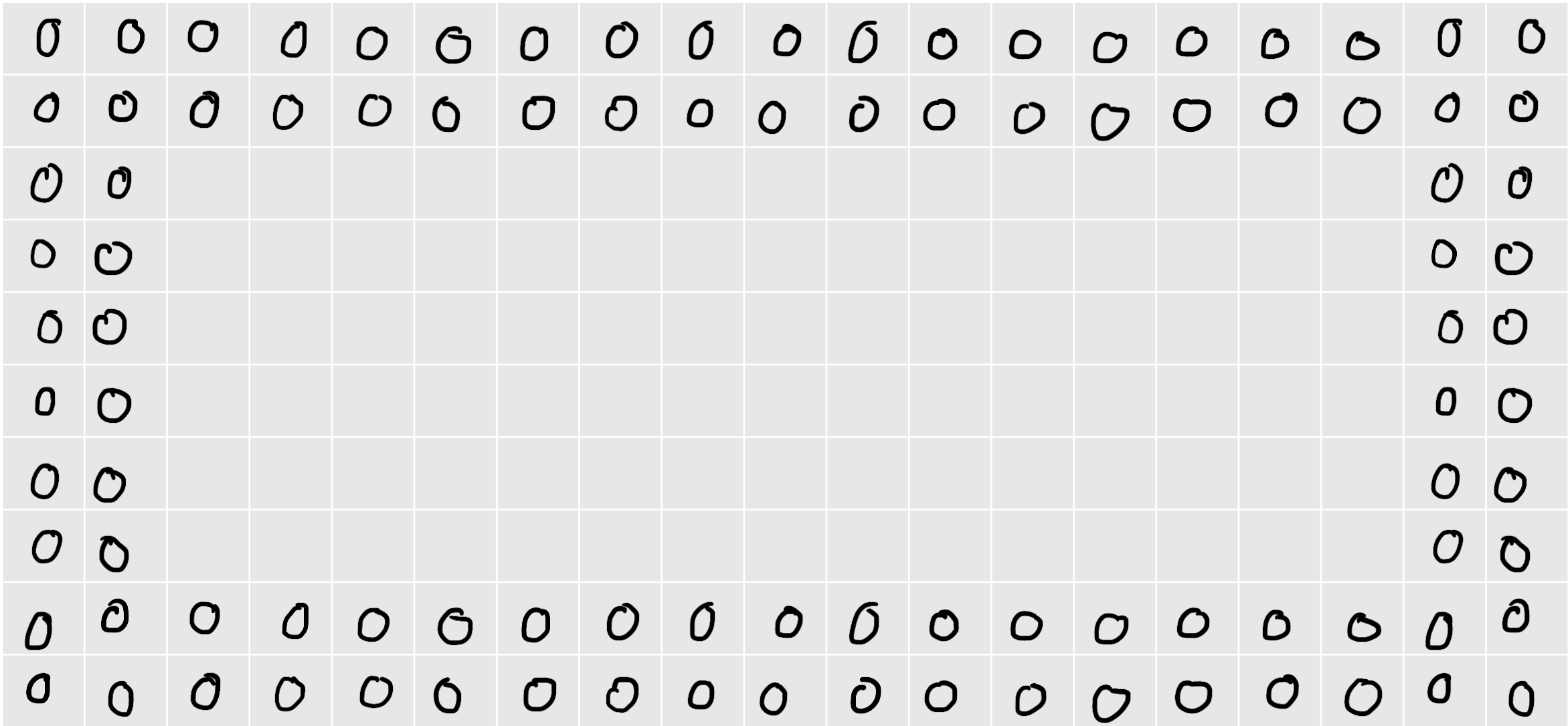
Stride = 3



Zeropadding = 1x1

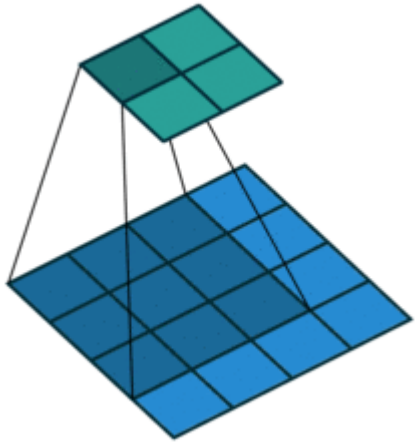


Zeropadding = 2x2

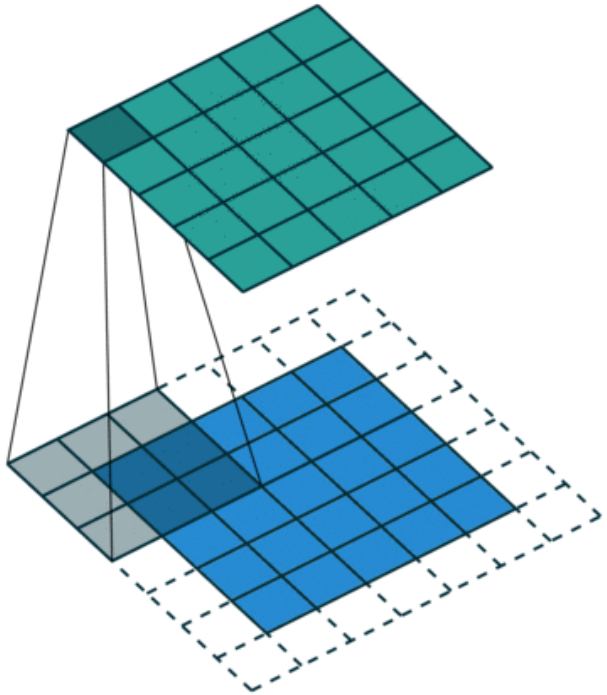


Zeropadding

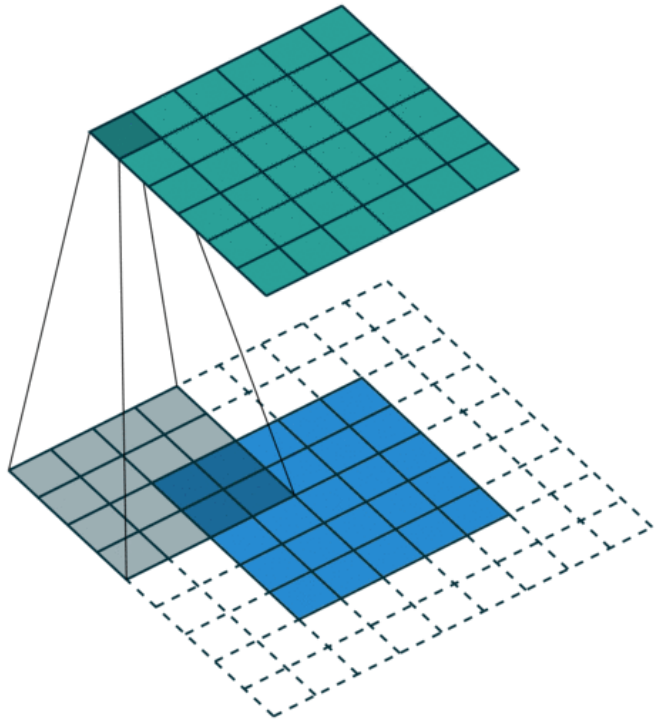
No zero padding, unit strides



Zero padding, unit strides



1 x 1



2 x 2

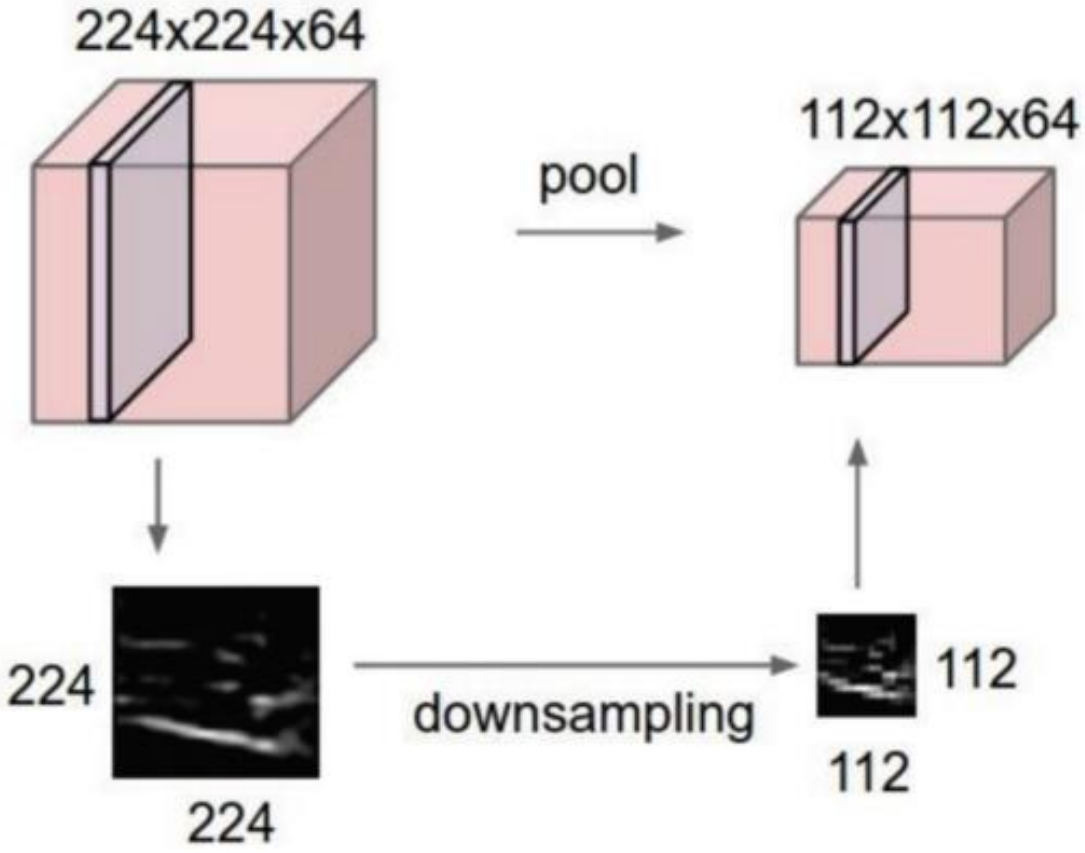
https://theano.readthedocs.io/en/master/tutorial/conv_arithmetic.html

Pooling

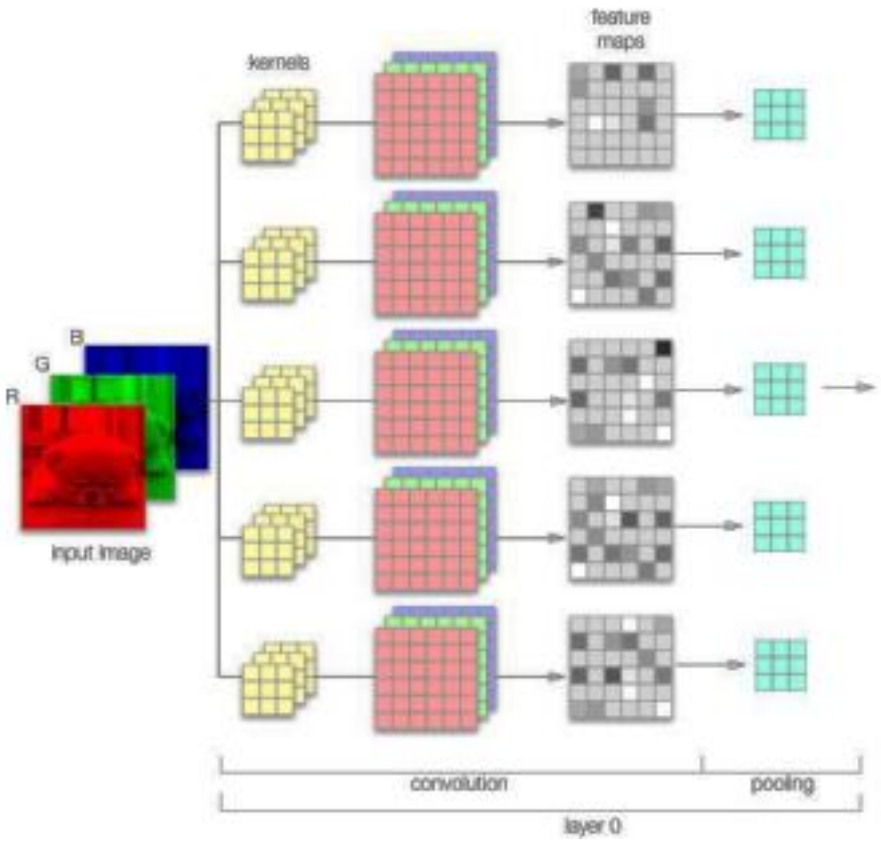
- makes the representations smaller and more manageable
- operates over each activation map independently

- **Pooling** is a form of downsizing that uses a 2D sliding filter. The filter passes over the input slice according to a configure parameter called the stride.
- **Stride** is the number pixels the filter moves across the input slice from one position to the next.
- There are two types of pooling operations: **average pooling** and **max pooling**. However, max pooling is the most common.

```
tf.keras.layers.Max
    Pool2D(
    pool_size=(2,2),
)   strides=2
```



Pooling



2	2	7	3
9	4	6	1
8	5	2	4
3	1	2	6

Max Pool

Filter - (2 x 2)
Stride - (2, 2)

9	7
8	6

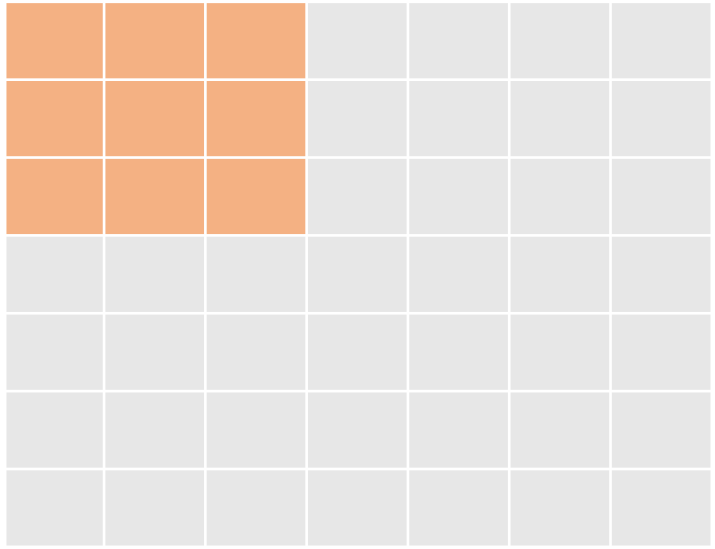
Average Pool

Filter - (2 x 2)
Stride - (2, 2)

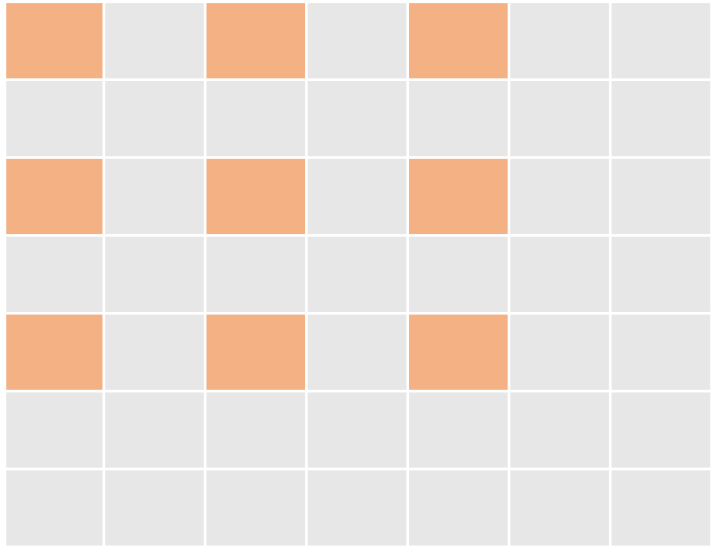
4.25	4.25
4.25	3.5

Reduce dimensionality

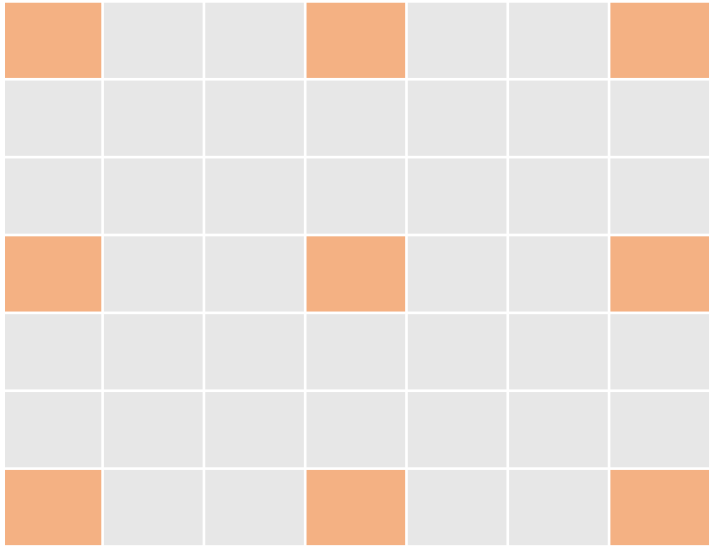
Dilation



Kernel: 3x3
Dilation: 1x1

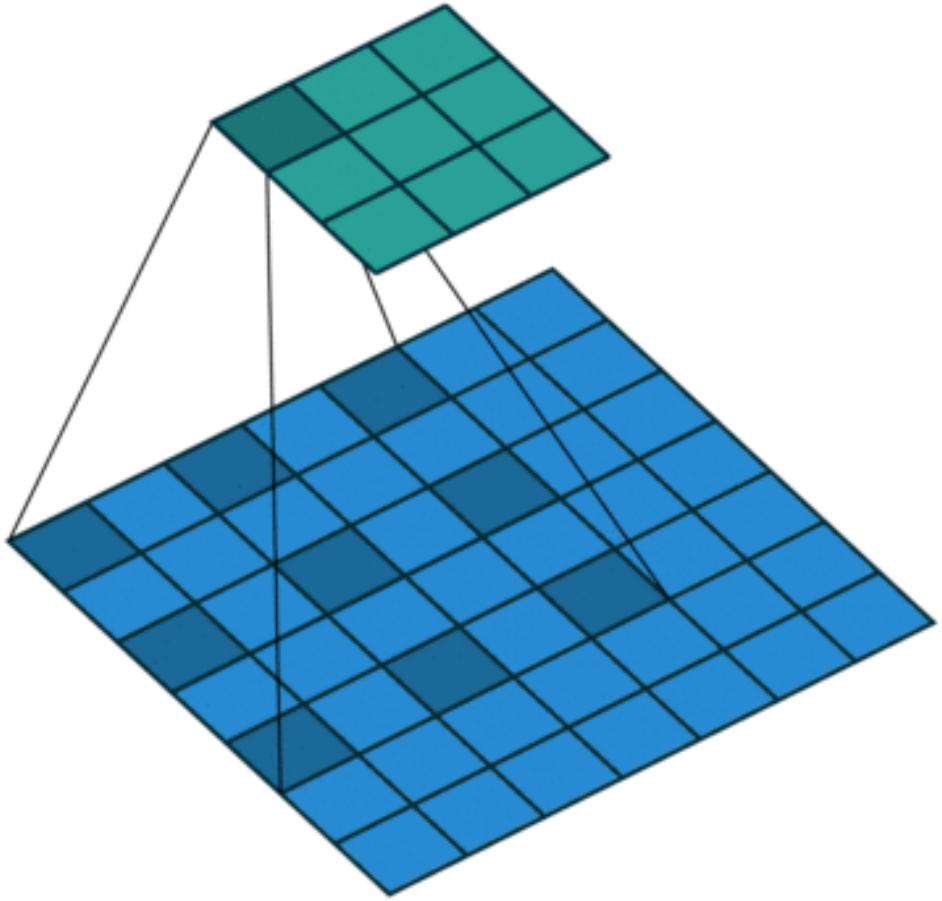


Kernel: 3x3
Dilation: 2x2



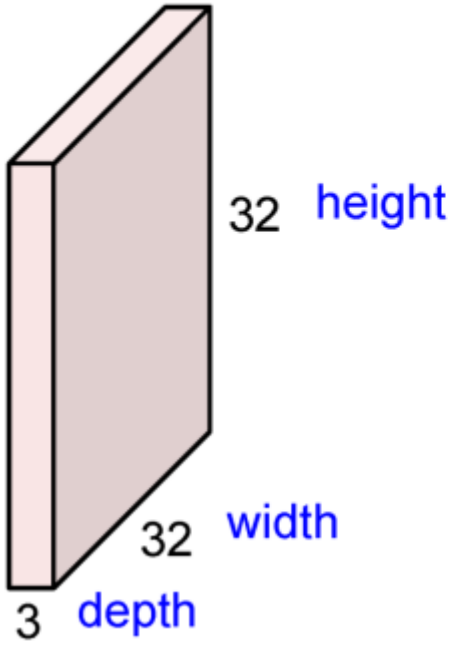
Kernel: 3x3
Dilation: 3x3

Dilation



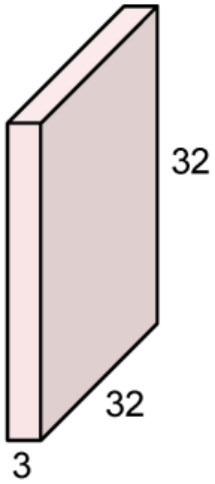
Convolution Layer

32x32x3 image -> preserve spatial structure

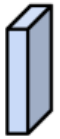


Filters always extend the full depth of the input volume

32x32x3 image



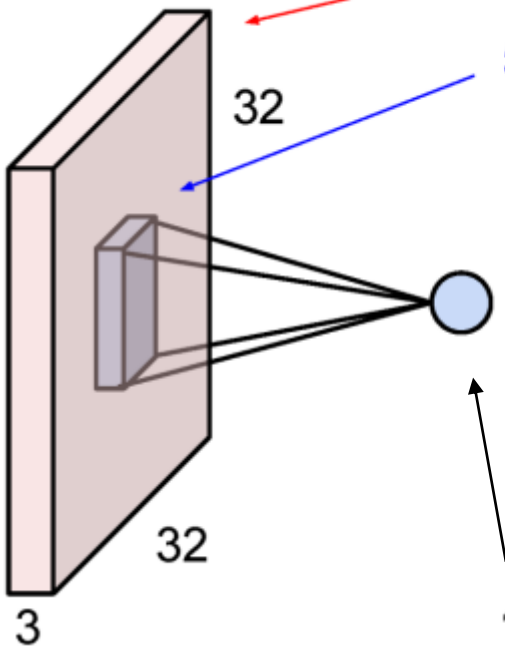
5x5x3 filter



Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

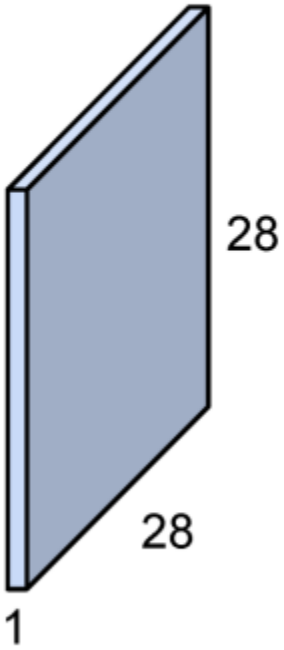
Number of weights: 5 x 5 x 3 + 1 = **76**
(vs. 3072 for a fully-connected layer)
(+1 for bias)

Convolution Layer



convolve (slide) over all spatial locations

activation map

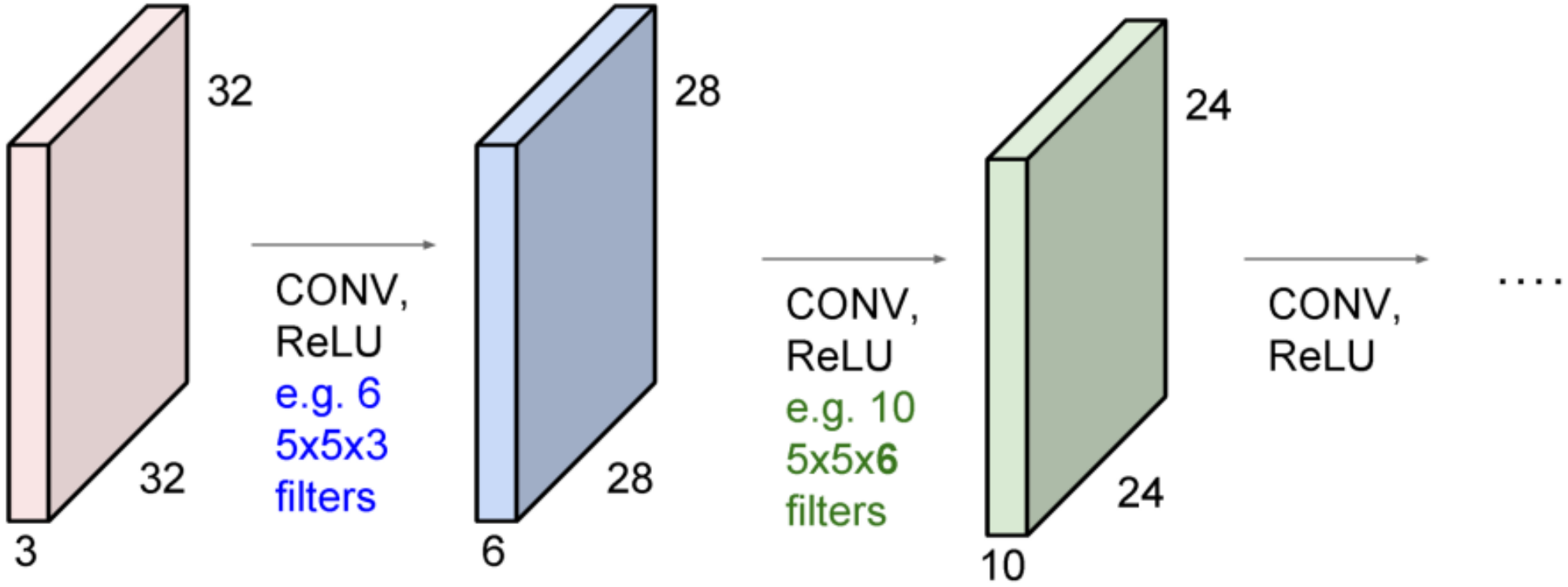


1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

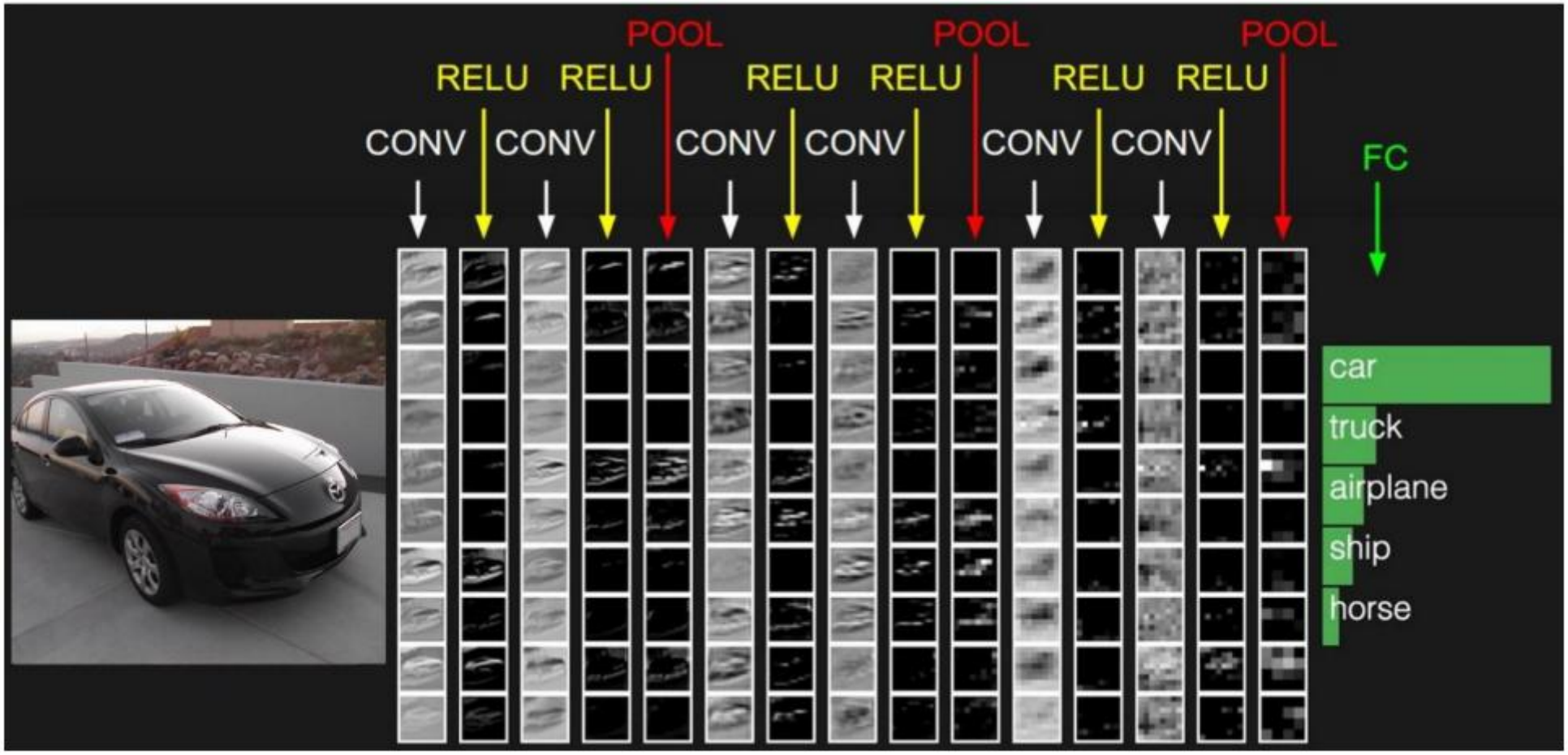
Convolution Layer

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



Fully Connected Layer

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



Example: 3@32x32,f:5x5@12,s:1x1,z:0x0

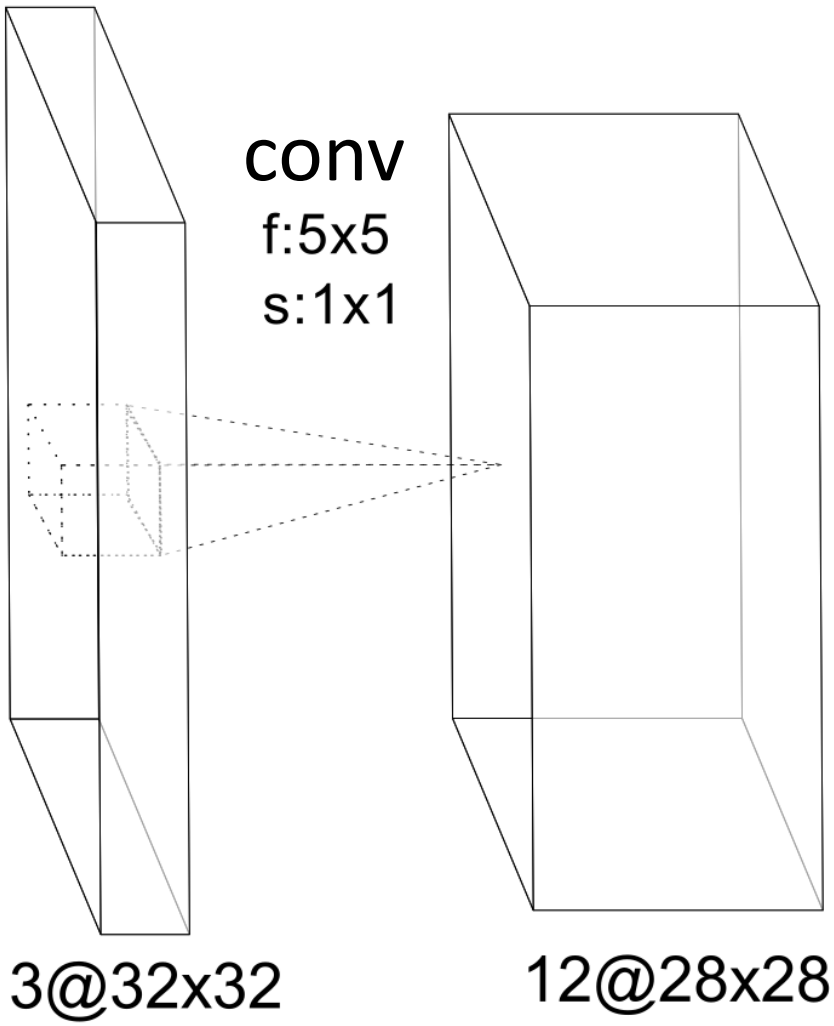
Output: 12@28x28

Number of weights between a first layer and a plane in a second layer:

5*5*3 + 1 (bias) = 76

Total number of weights:

(5*5*3 +1)*12 = 912

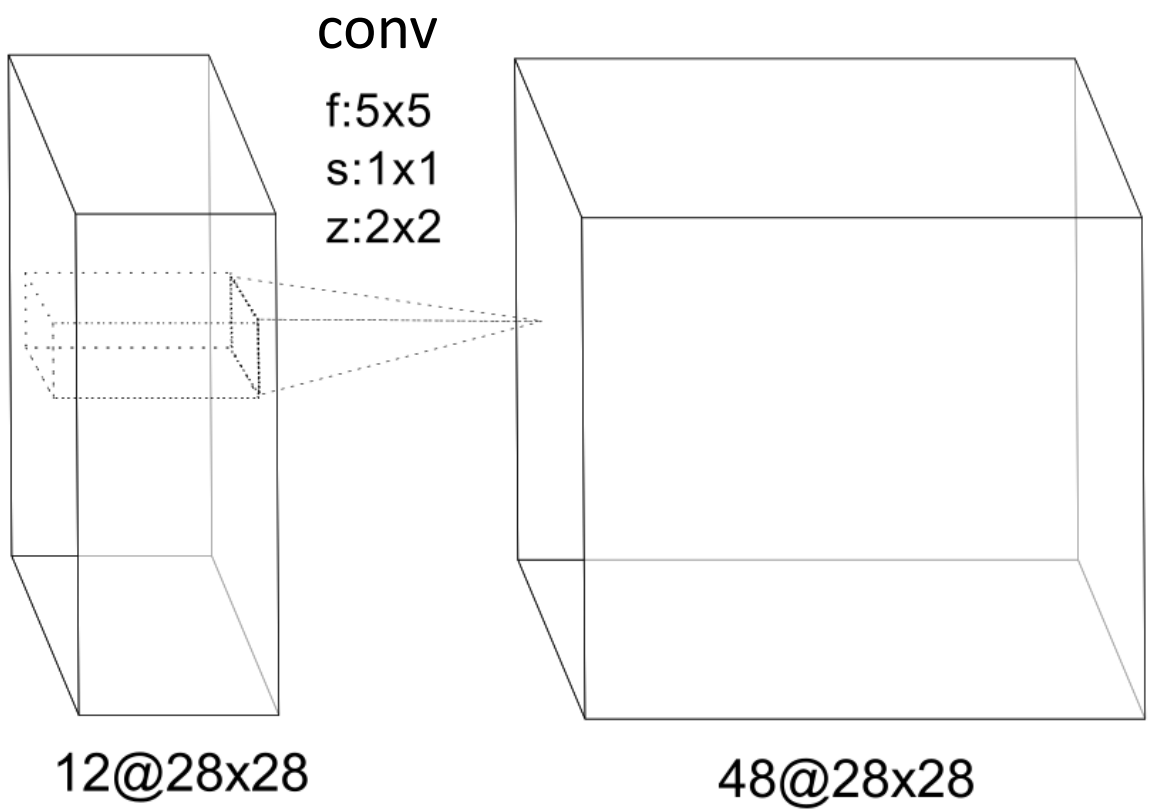


Example: 12@28x28,f:5x5@48,s:1x1,z:2x2

Output: 48@28x28

Number of weights per plane: 5*5*12 + 1 (bias) = 301

Number of weights: (5*5*12+1)*48 = 14448

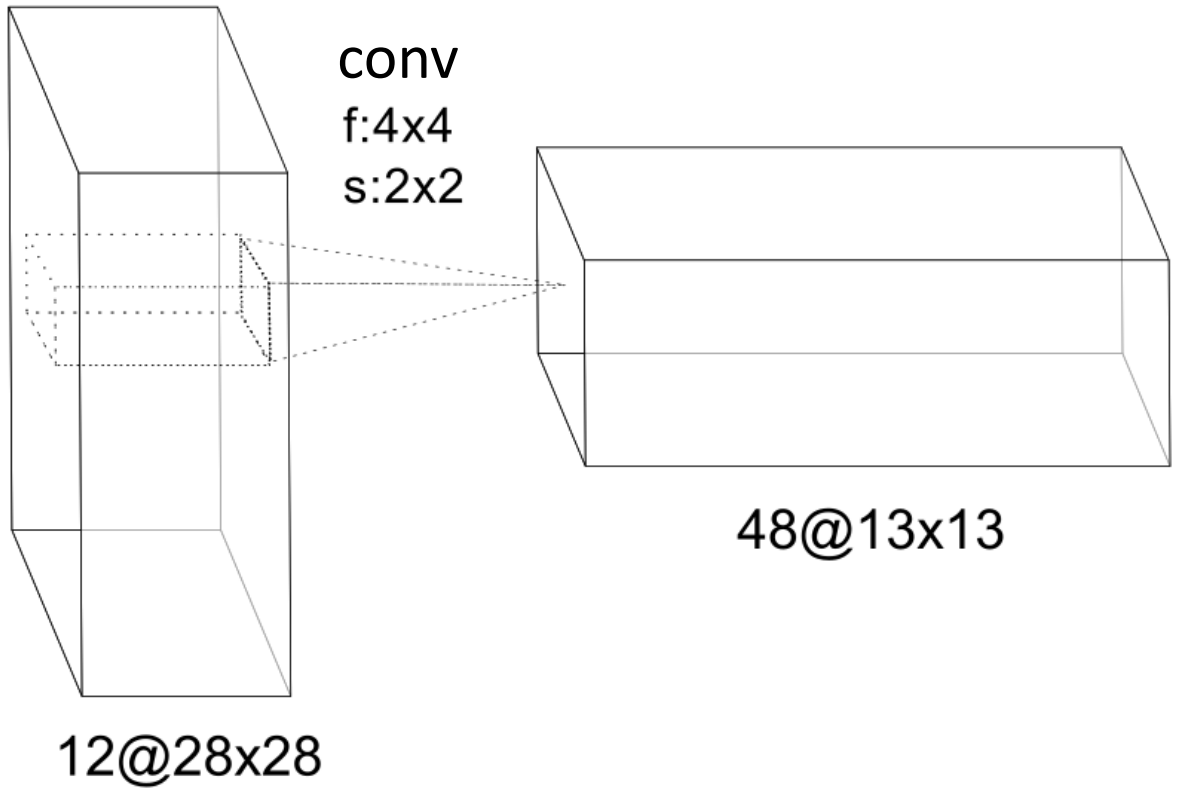


Example: 12@28x28,f:4x4@48,s:2x2,z:0x0

Output: 48@13x13

Number of weights per plane: 4*4*12+1=193

Number of weights: (4*4*12+1)*48=9264



[ConvNetJS demo: training on CIFAR-10]

ConvNetJS CIFAR-10 demo

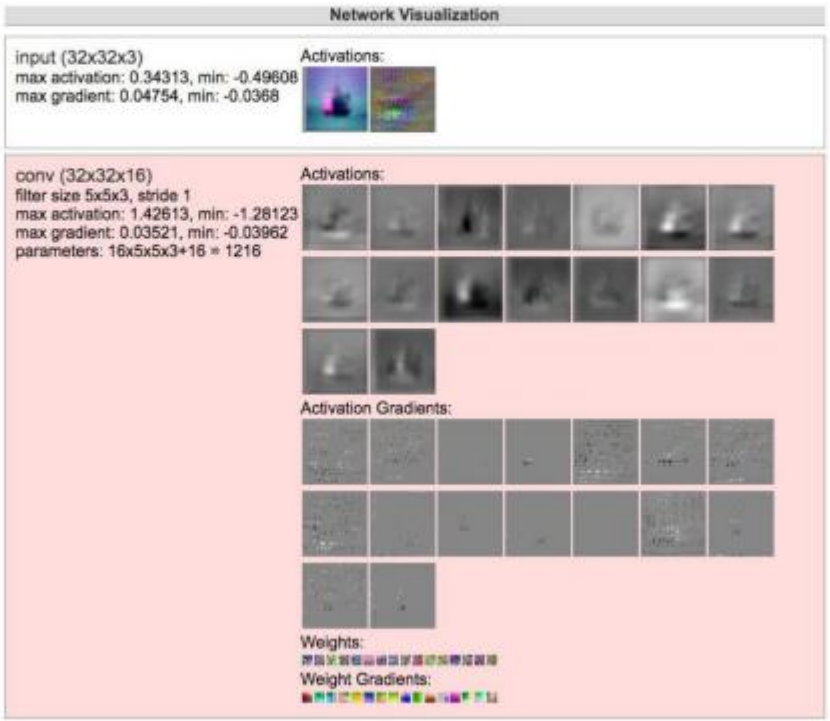
Description

This demo trains a Convolutional Neural Network on the CIFAR-10 dataset in your browser, with nothing but Javascript. The state of the art on this dataset is about 90% accuracy and human performance is at about 94% (not perfect as the dataset can be a bit ambiguous). I used this python script to parse the original files (python version) into batches of images that can be easily loaded into page DOM with img tags.

This dataset is more difficult and it takes longer to train a network. Data augmentation includes random flipping and random image shifts by up to 2px horizontally and verically.

By default, in this demo we're using Adadelta which is one of per-parameter adaptive step size methods, so we don't have to worry about changing learning rates or momentum over time. However, I still included the text fields for changing these if you'd like to play around with SGD+Momentum trainer.

Report questions/bugs/suggestions to @karpathy.



https://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

To be continued next week

Please, don’t forget to send feedback:

https://bit.ly/bme-dl

Thank you for your attention

Dr. Mohammed Salah Al-Radhi
malradhi@tmit.bme.hu

(slides by: Dr. Bálint Gyires-Tóth)

08 October 2024