

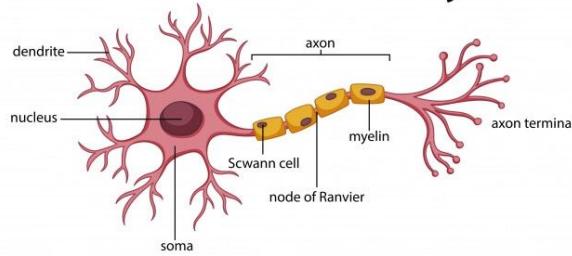
Data Science Survival Skills

Artificial Intelligence, Machine Learning, Deep Learning

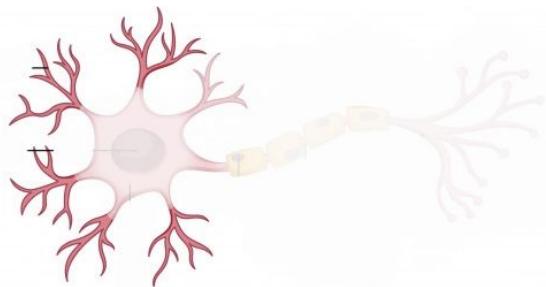
Part II:

Deep Learning

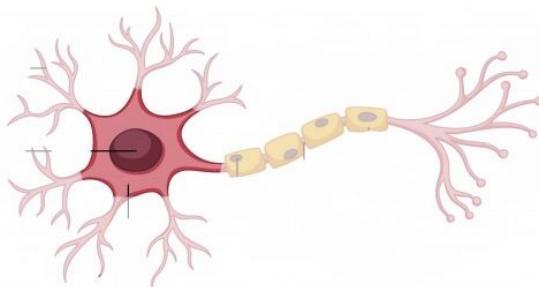
Perceptrons



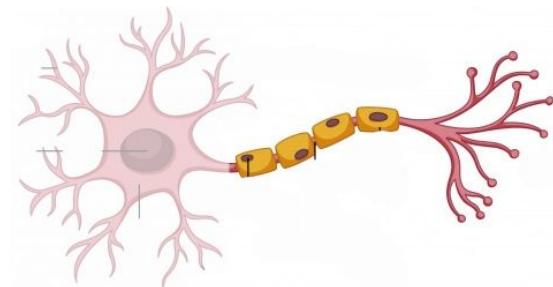
INPUT



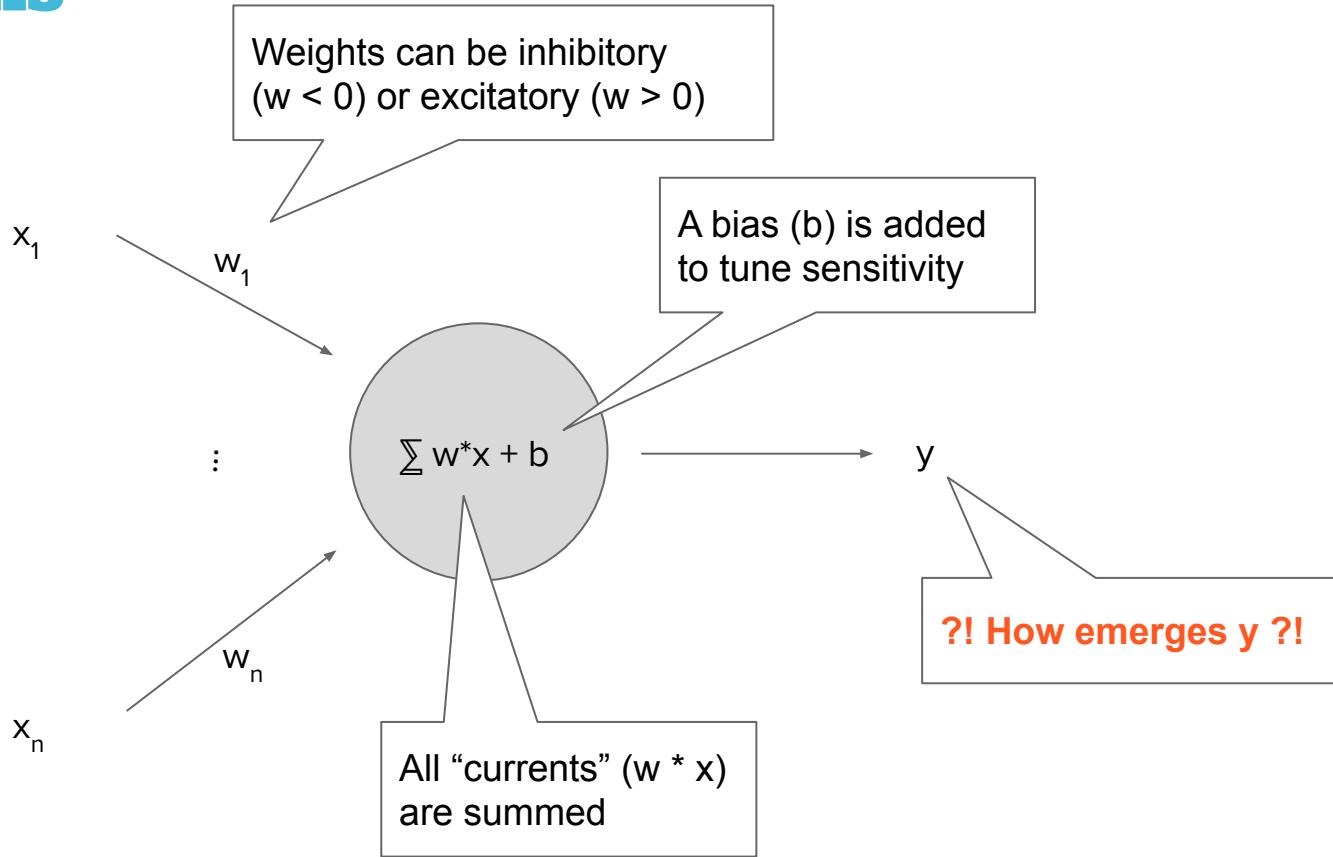
COMPUTATION



OUTPUT



Perceptrons



Perceptrons

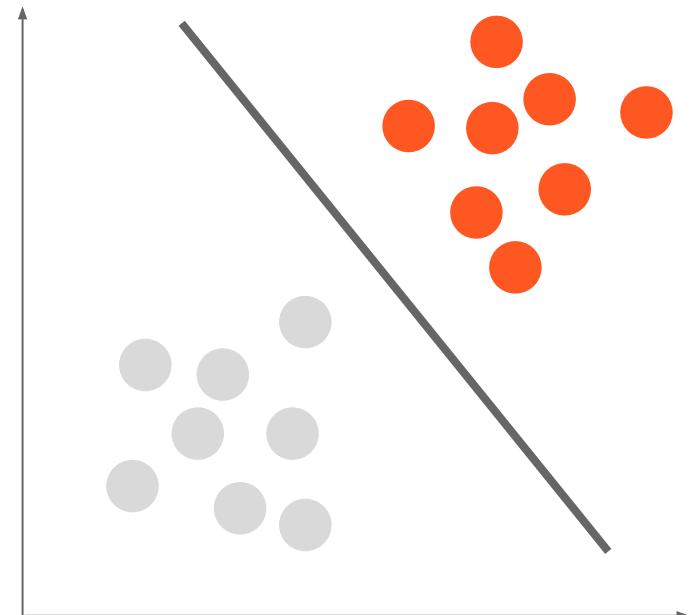
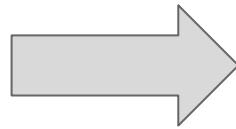
Perceptrons are **LINEAR CLASSIFIER!**

Threshold
step function:

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

A thought bubble containing the mathematical formula for a perceptron's decision function.

$$\sum_{i=1}^m w_i x_i,$$



Teaching and Learning

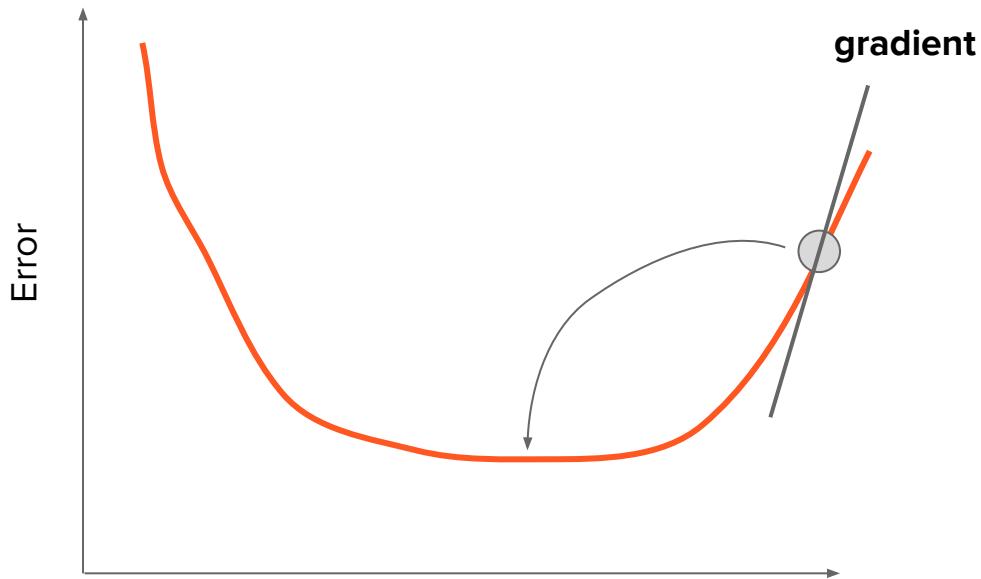
What are the parameters one could tune?

WEIGHTS

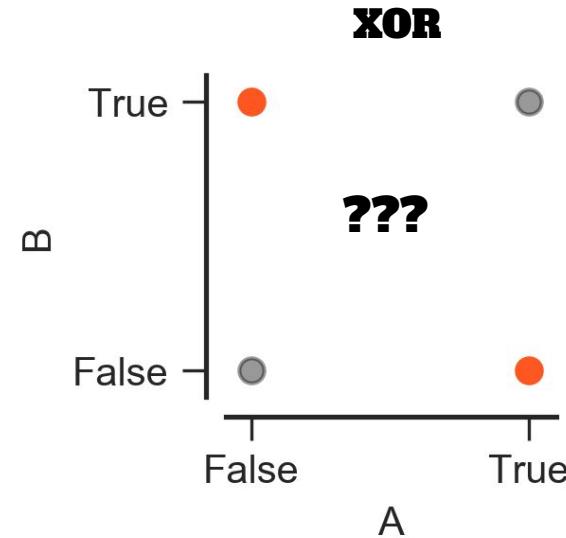
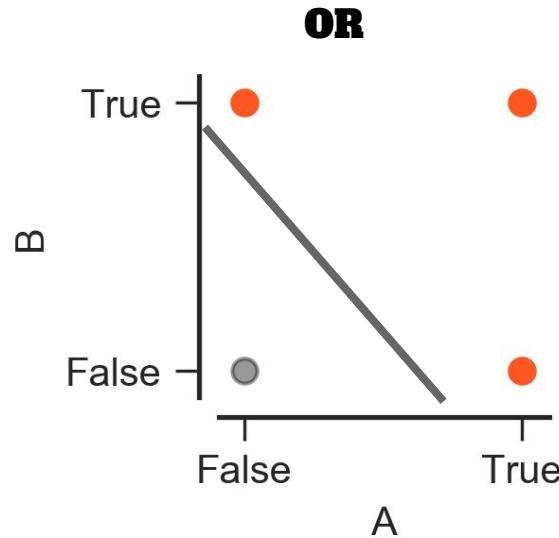
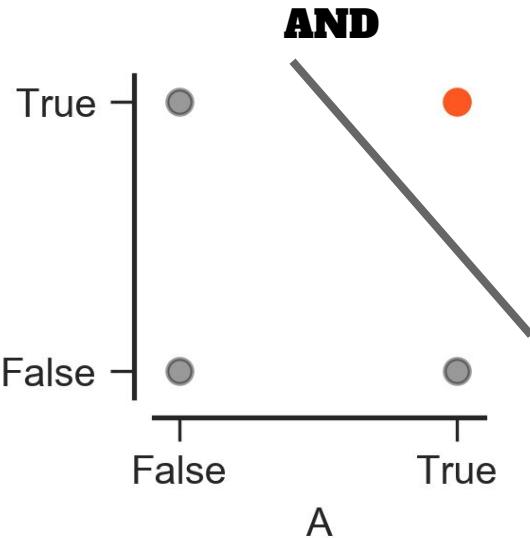
BIAS

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0, \\ 0 & \text{otherwise} \end{cases}$$

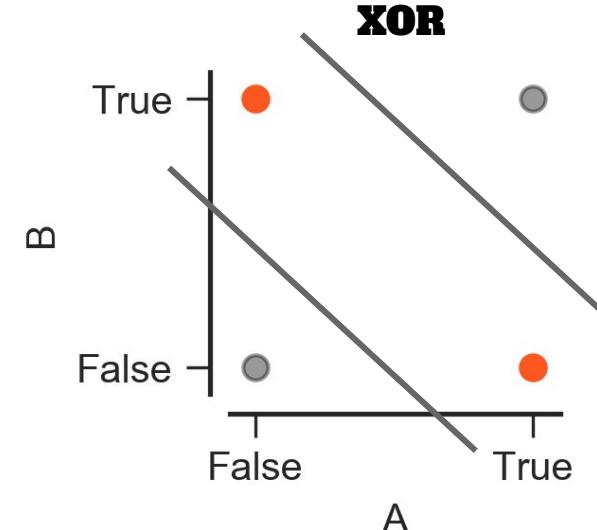
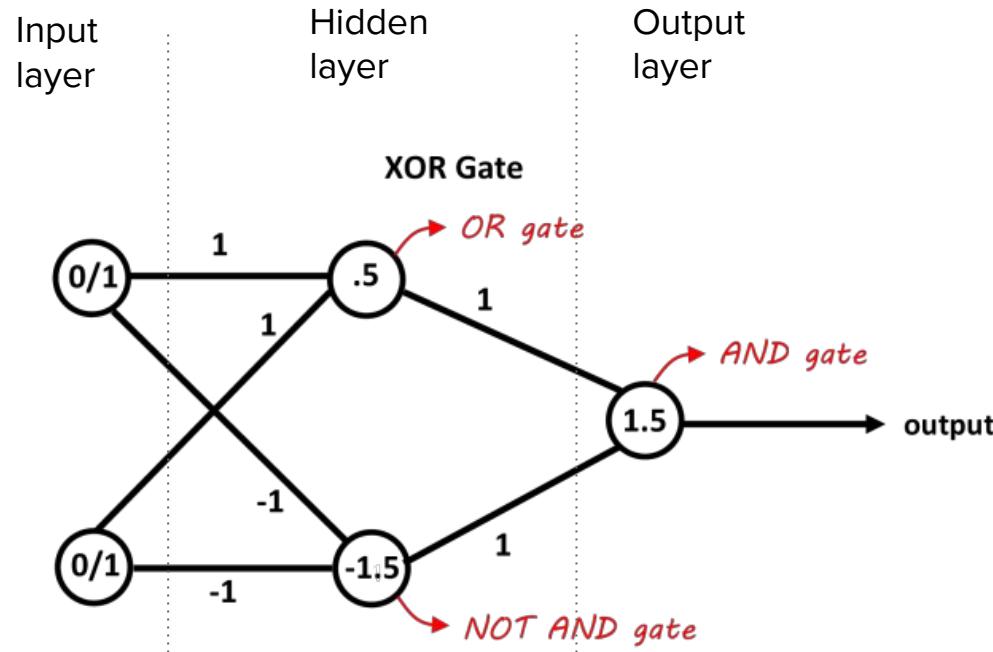
Learning weights with gradient descent



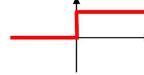
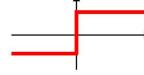
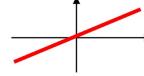
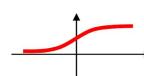
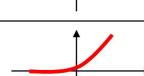
Why perceptrons were abandoned...



But there is a solution: The multilayer perceptron!

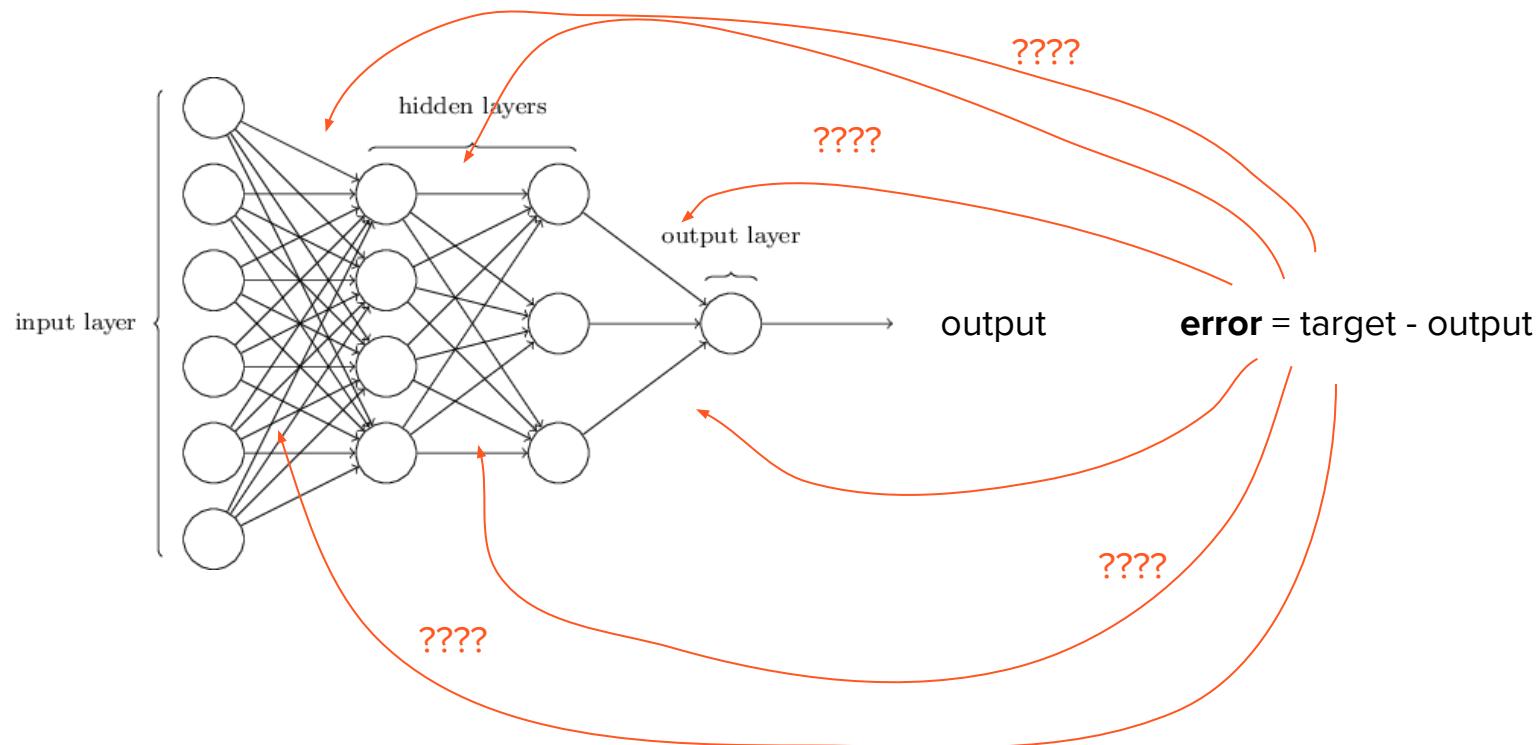


Activation functions

Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = \max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^z)$	Multi-layer Neural Networks	

Assigning the error

→ Something called backpropagation.
It is done automatically, but you should
KNOW WHAT IT IS DOING!



Yes you should understand backprop



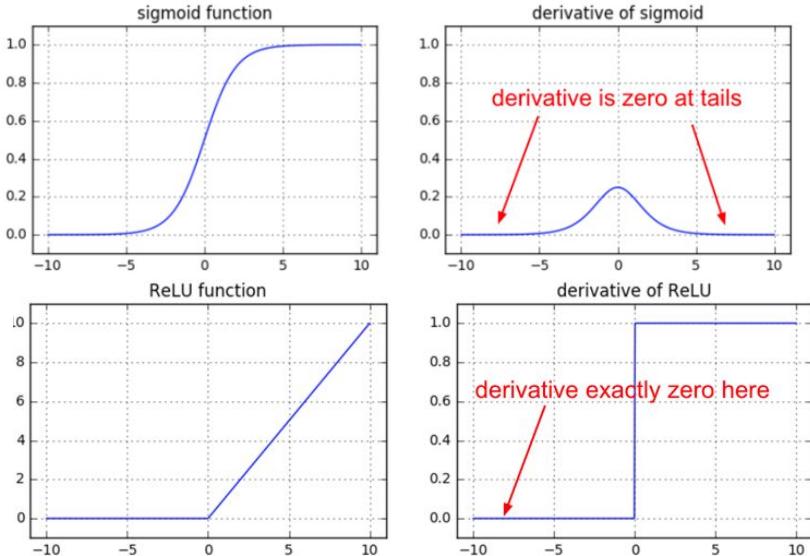
Andrey Karpathy Dec 19, 2016 · 7 min read



17K



46



In conclusion

Backpropagation is a leaky abstraction; it is a credit assignment scheme with non-trivial consequences. If you try to ignore how it works under the hood because “TensorFlow automagically makes my networks learn”, you will not be ready to wrestle with the dangers it presents, and you will be much less effective at building and debugging neural networks.

TensorFlow/Keras



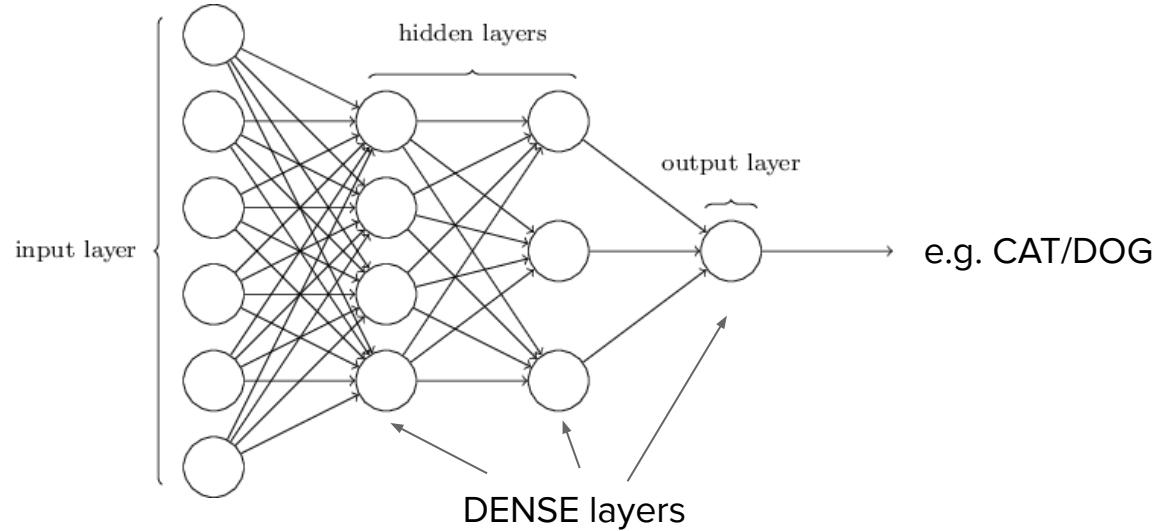
1	1	0
4	2	1
0	2	1



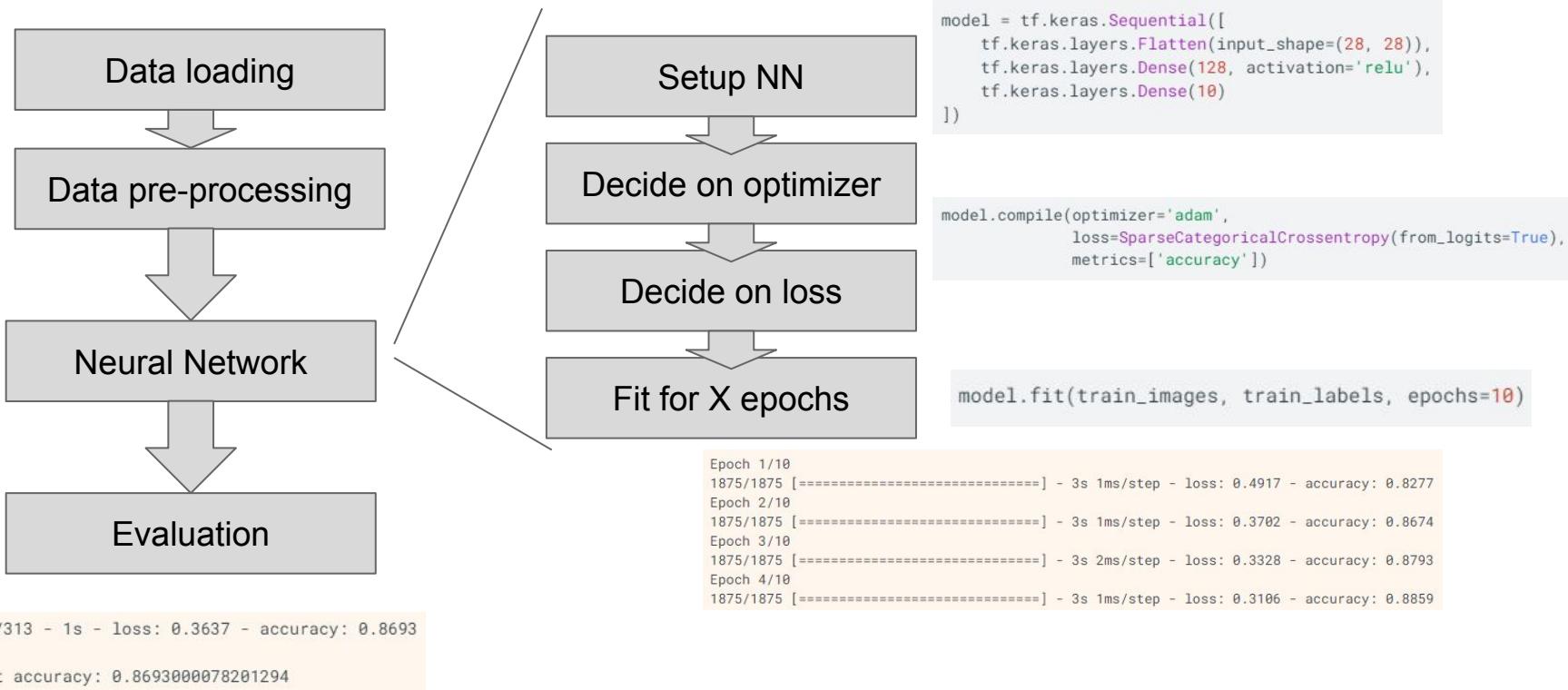
1
1
0
4
2
1
0
2
1

INPUT layer

FLATTEN layer

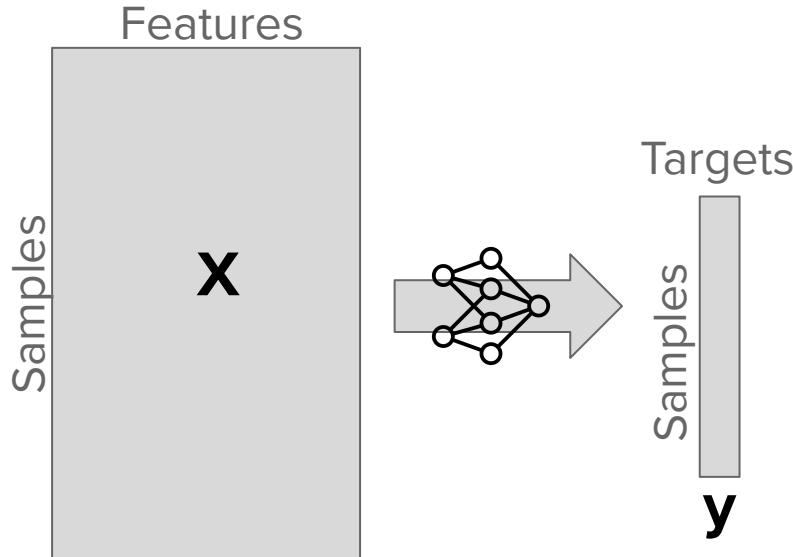


How to create a NN (in Keras)



Loading data

What is your X, what is your y/Y?



Steps

1. Does my data fit completely in memory? Yes: load it
No: use lazy loading

2. Does my data has always the same shape? Yes: Good! No:
Reshape/Resize/Crop

3. Normalize your data.
Min-max: $[0...1]$ or $[-1...1]$
Z-scoring: $x' = (x - \text{x_mean}) / \text{std}$

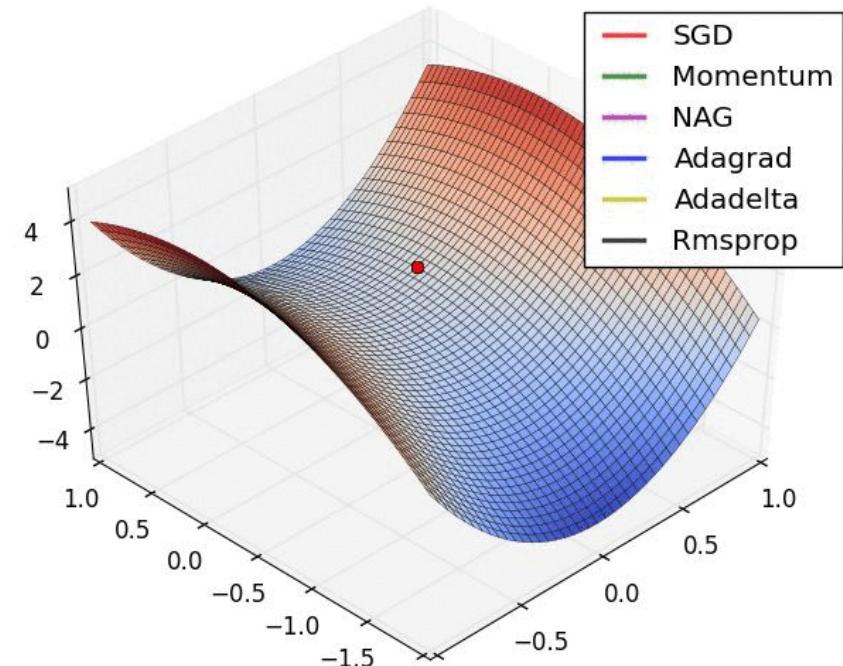
4. How large should the batch be?

Optimizers

Strategies to optimize the network and perform gradient descent:

- Stochastic gradient descent (SGD)
- SGD + momentum
- RMSprop
- Adadelta
- Adam
- ...

Try SGD(momentum=.9) and Adam



The loss function

Important:

your objective function should represent what you would like to optimize!!

Regression Losses

Mean Square Error/Quadratic Loss/L2 Loss

Mathematical formulation :-

$$MSE = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}$$

Mean Squared Error

Cross Entropy Loss/Negative Log Likelihood

This is the most common setting for classification problems. Cross-entropy loss increases as the predicted probability diverges from the actual label.

Mathematical formulation :-

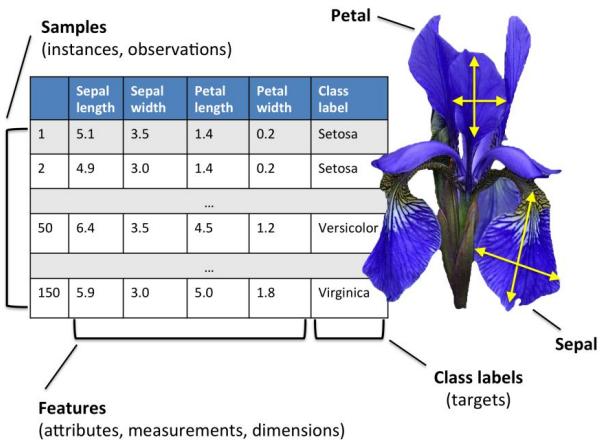
$$CrossEntropyLoss = -(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

Cross entropy loss

Code for an MLP

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
]) # Multilayer perceptron
```

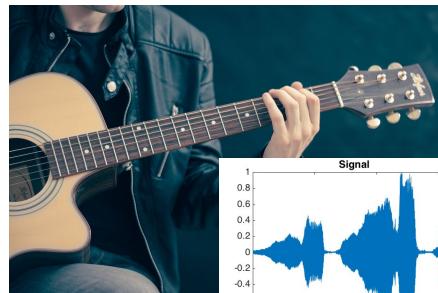
What kind of data do you have...?



Structured data!



images



audio

text



Johann Wolfgang Goethe: *Erlkönig* -

Wer ruelt so spät durch Nacht und Wind?
Es ist der Vater mit seinem Kind.
Er hat den Knaben wohl in dem Arm,
Er fasst ihn sicher, er hält ihn warm.

„Mein Sohn, wie bringst du so bang dein Gesicht?“ –
„Siehst, Vater, du den Erlkönig nicht?
Den Erlkönig mit Kron’ und Schweif?“ –
„Mein Sohn, es ist ein Nebelstreif.“ –

„Du, liebes Kind, komm, geh mit mir!
Geschnüre Spiele spiell’ ich mit dir:
Manch bunte Blumen sind an dem Strand,
Meine Mutter hat manch gülden Gewand.“

„Mein Vater, mein Vater, und hörest du nicht,
Was Erlkönig mir leise verspricht?“ –
„Sei ruhig, bleibe ruhig, mein Kind;
In dünnen Blättern säuselt der Wind.“ –

„Willst, feiner Knabe, du mit mir gehen?
Meine Töchter sollen dich warten schön;
Meine Töchter führen den nächtlichen Reih,
Und wiegen und tanzen und singen dich ein.“

„Mein Vater, mein Vater, und siehst du nicht dort
Erlköings Töchter am düstern Ort?“ –
„Mein Sohn, mein Sohn, ich seh es genau:
Es scheinen die alten Weiden so grau.“ –

„Ich liebe dich, mich reizt deine schöne Gestalt;
und bist du willig, so brauch ich Gewalt.“ –
„Mein Vater, mein Vater, jetzt fasst er mich an!
Erlkönig hat mir ein Leids gelan!“ –

Dem Vater grauset’s, er reitet geschwind,
Er hält in den Armen das schützende Kind,
Erreicht den Hof mit Muhe und Not;
In seinen Armen das Kind war tot.

Images

Imagine a noisy pixel



12	11	11
13	255	10
14	15	10

v
u

$$\frac{12 + 11 + 11 + 13 + 255 + 10 + 14 + 15 + 10}{9} = 39$$

12	11	11
13	39	10
14	15	10

$$I'(u, v) = \frac{\sum_{i=-1}^1 \sum_{j=-1}^1 I(u+i, v+j)}{9}$$

1	1	1
1	1	1
1	1	1

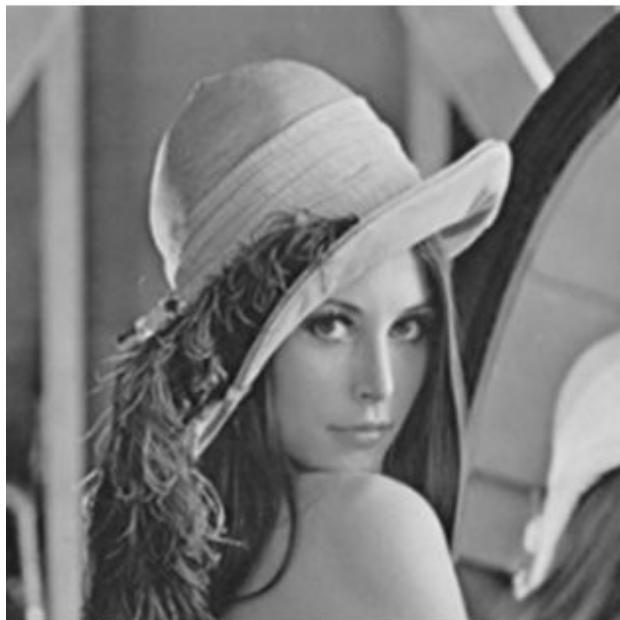
Applying a linear filter across a whole image



$$\begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$



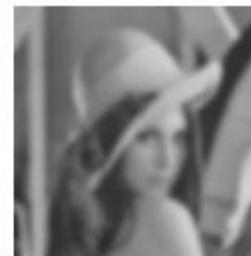
Kernel size effect



$$\otimes \quad \square =$$



$$\otimes \quad \boxed{\square} =$$



$$\otimes \quad \boxed{\boxed{\square}} =$$



Increase of
receptive field

Edge detection linear filters

Prewitt-Filter

$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

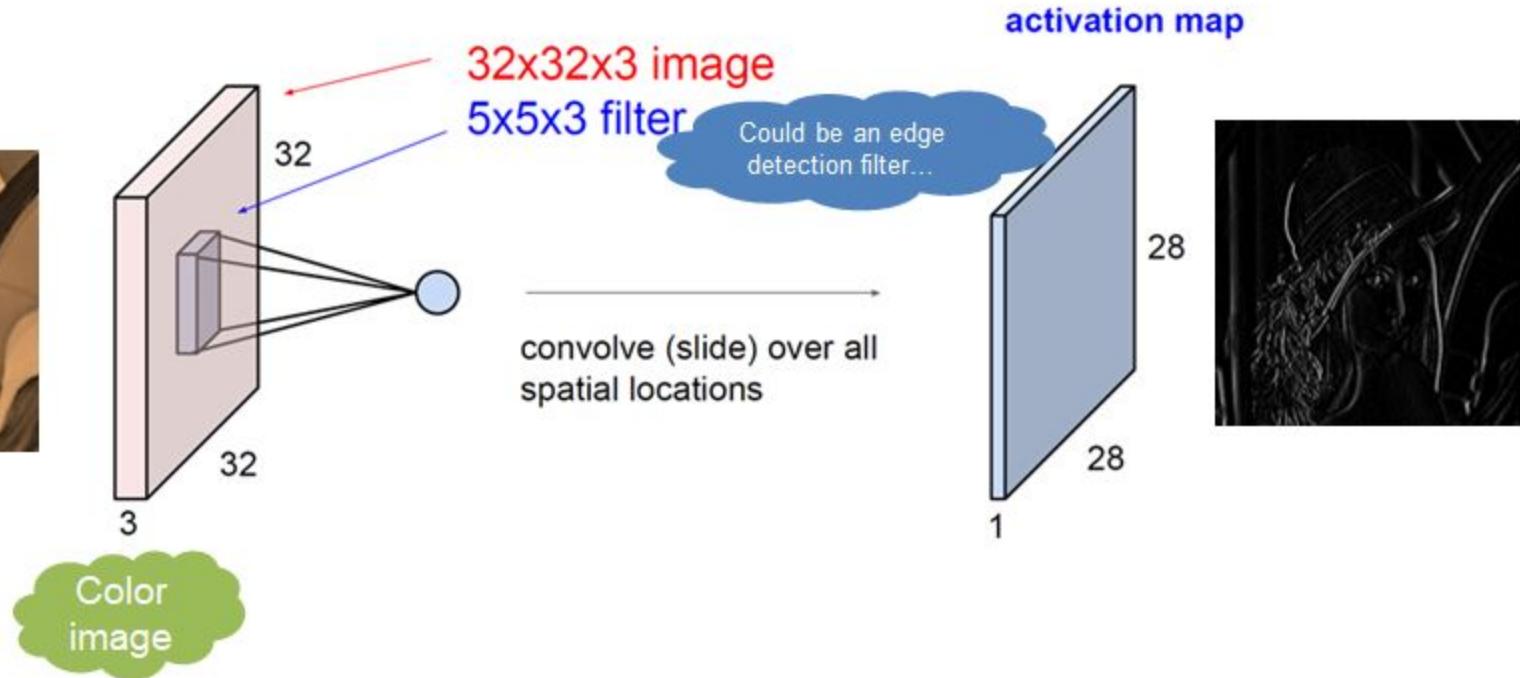


Sobel-Filter

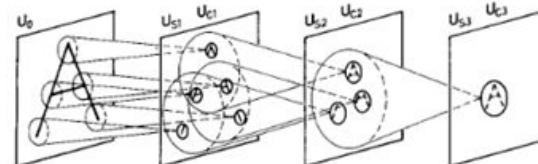
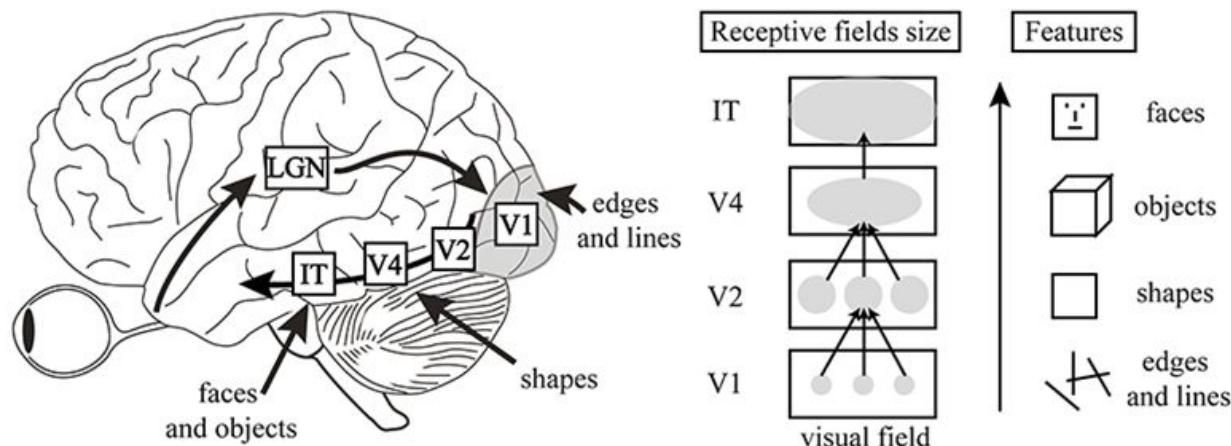
$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$



Basis of convolutional neural networks



How does our visual cortex work?



Convolutional NNs (1998)

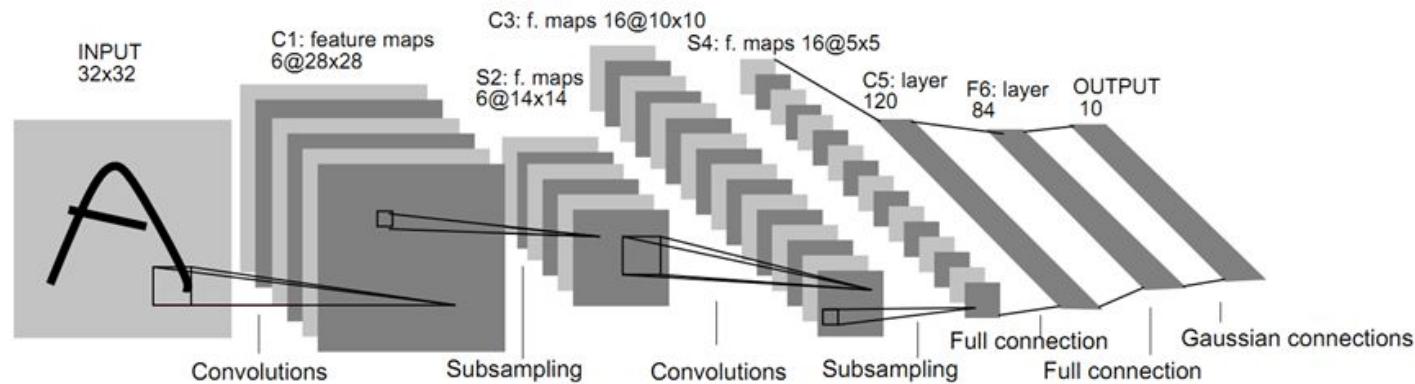


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

3 elements:

- Convolutions
- Subsamplings
- Fully Connected/Dense/Perceptron layers

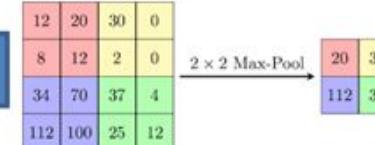
Missing the last bits....

Filters/
Activation maps

$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ and } H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

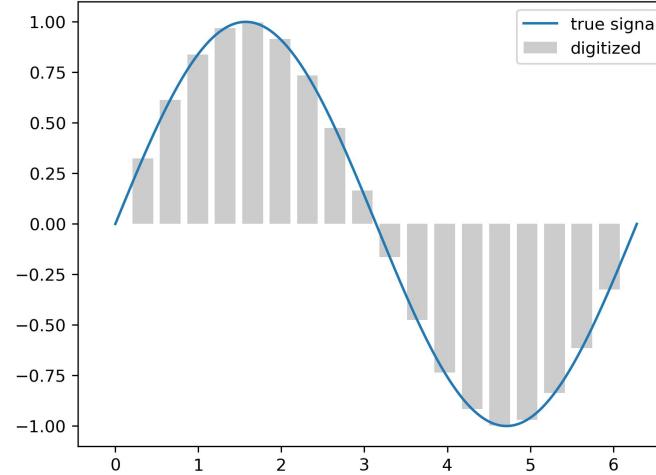
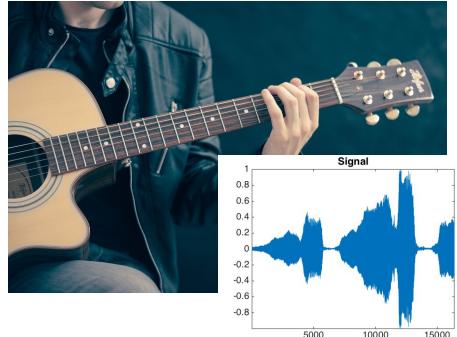
3x3			5x5		
0.86	0.92	0.97	0.27	0.34	0.44
0.70	0.26	0.95	0.28	0.39	0.99
0.99	0.98	0.24	0.94	0.67	0.98
0.94	0.23	0.24	0.97	0.98	0.98
0.87	0.86	0.87	0.75	0.96	0.98

```
In [46]: 1 cnn = tf.keras.models.Sequential([
2     tf.keras.layers.Conv2D(8, (3,3), input_shape=(28,28,1), padding='same', activation='relu'),
3     tf.keras.layers.MaxPool2D(),
4     tf.keras.layers.Conv2D(16, (3,3))
5     tf.keras.layers.MaxPool2D(),
6     tf.keras.layers.Flatten(),
7     tf.keras.layers.Dense(128, activation='relu'),
8     tf.keras.layers.Dropout(0.2),
9     tf.keras.layers.Dense(10, activation='softmax')
10    ])
11 ])
```



Same as in MLPs...

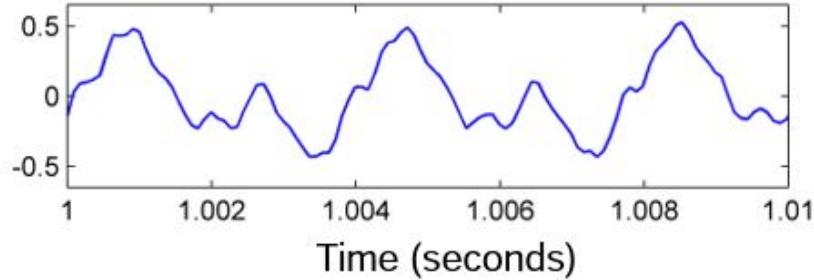
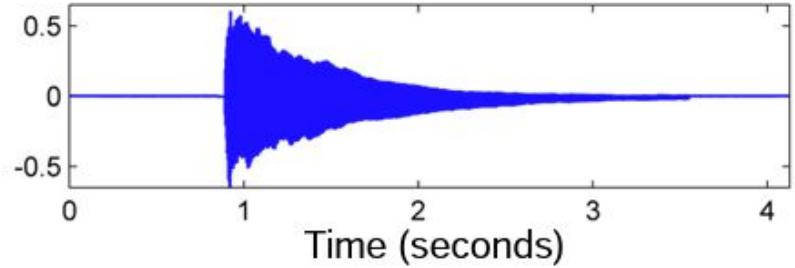
Audio data



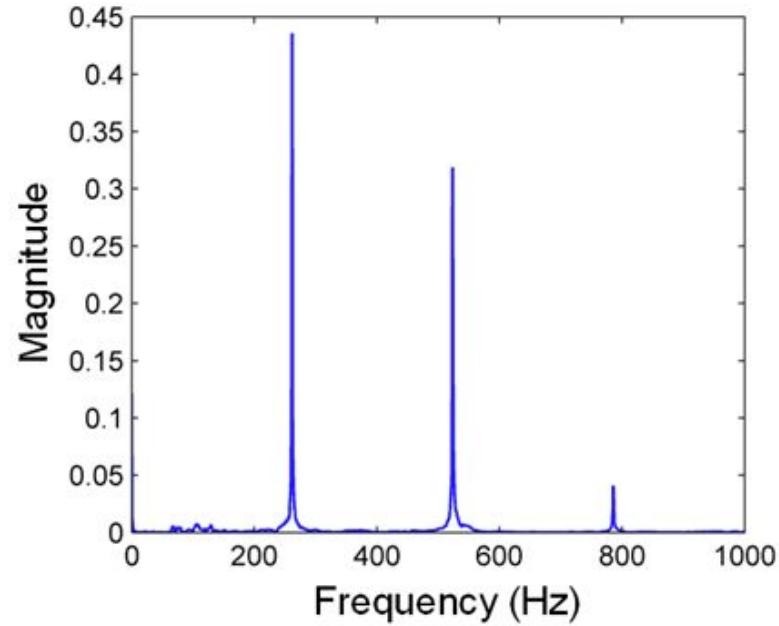
Audio signals are **raw 1D signal** with one (mono) or two channels (stereo), typically.

1D Convolutions for raw audio processing!

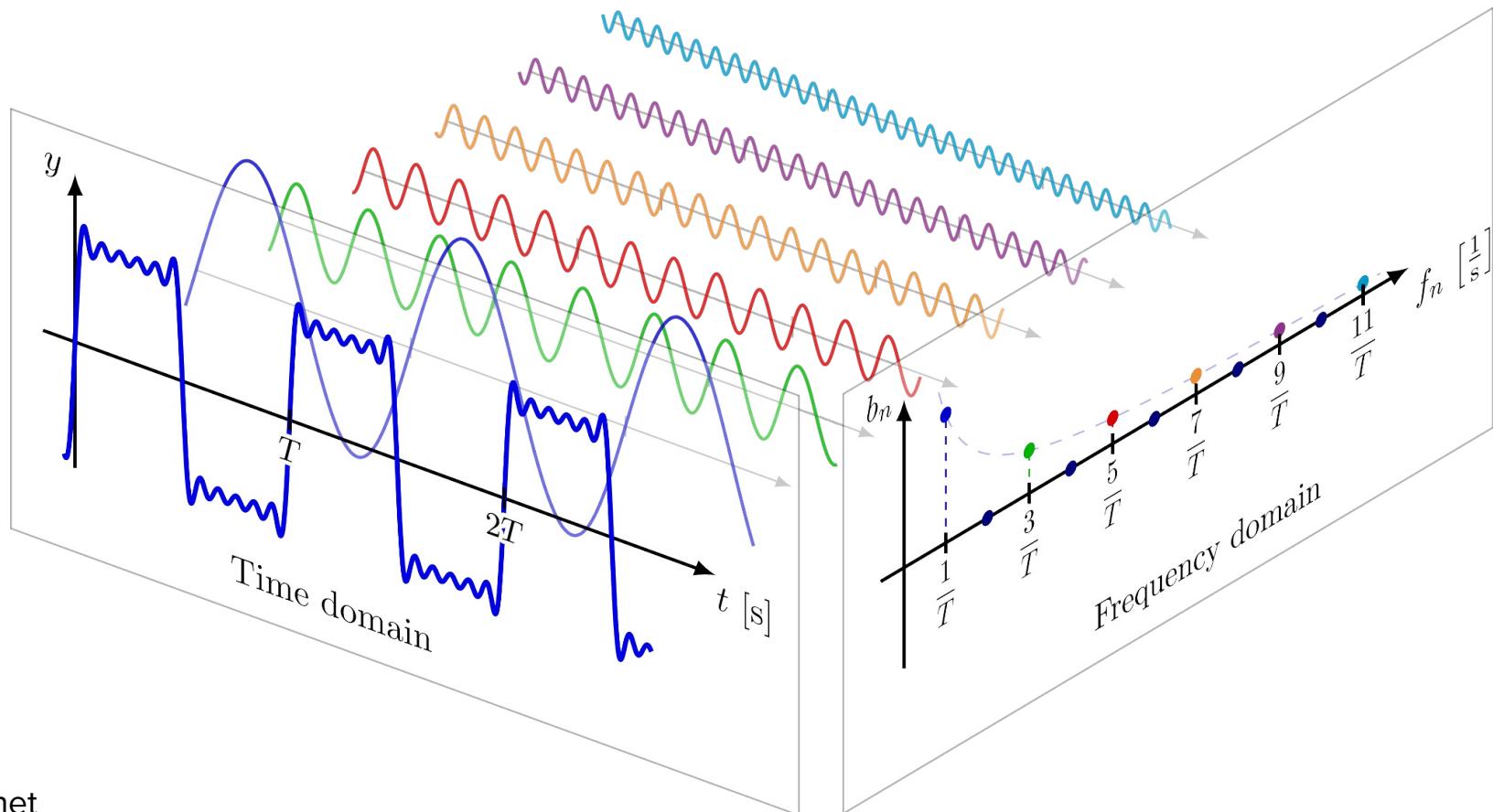
How do we get more information across time?



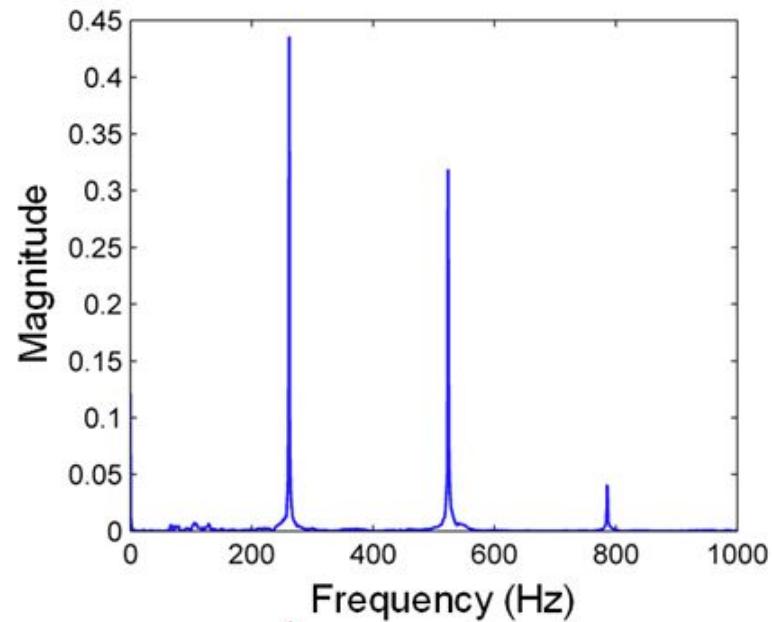
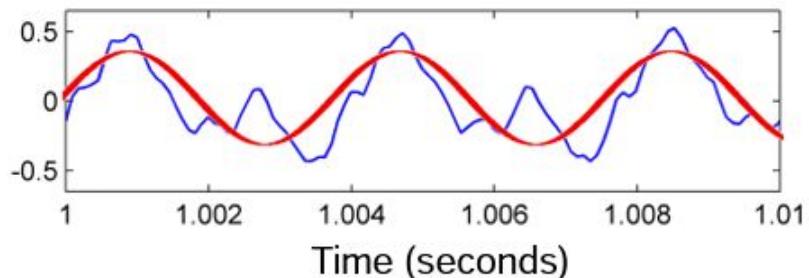
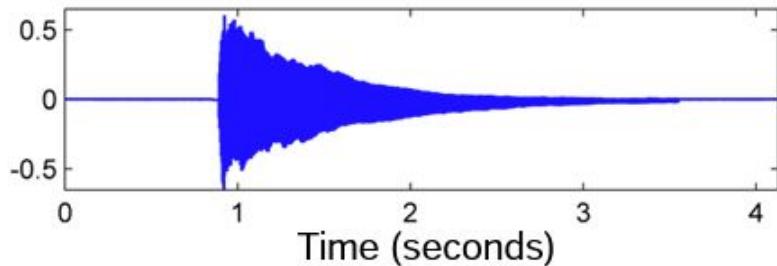
FOURIER SPACE!



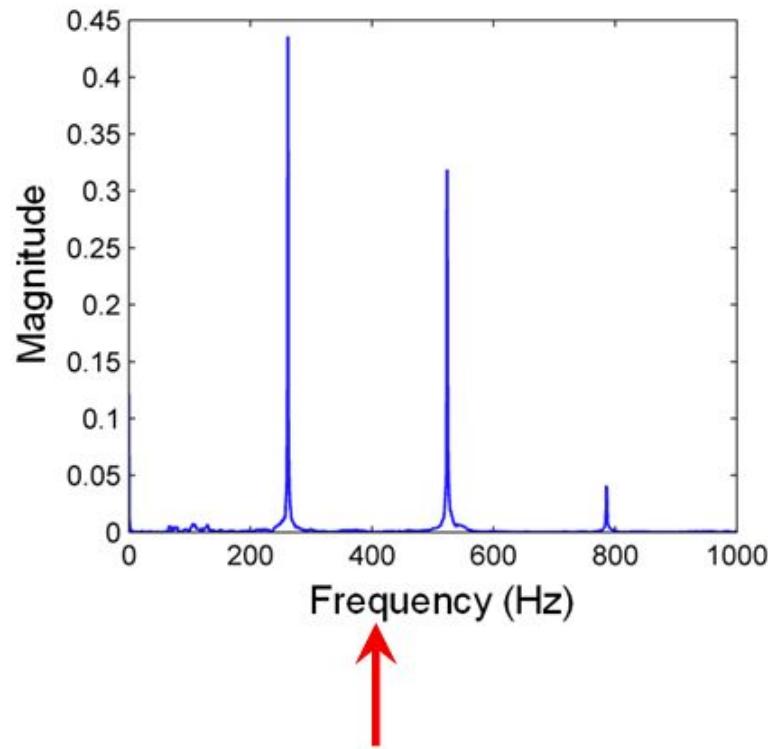
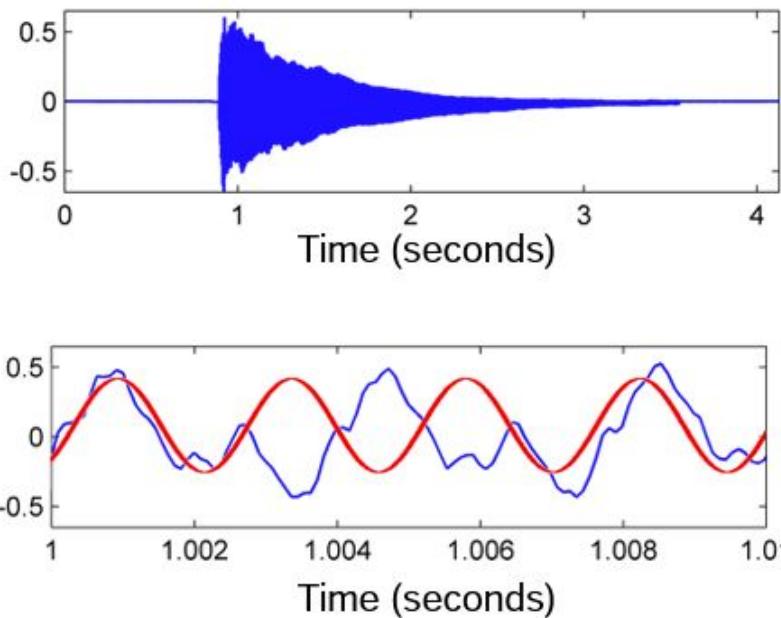
Fourier transformation ctn.



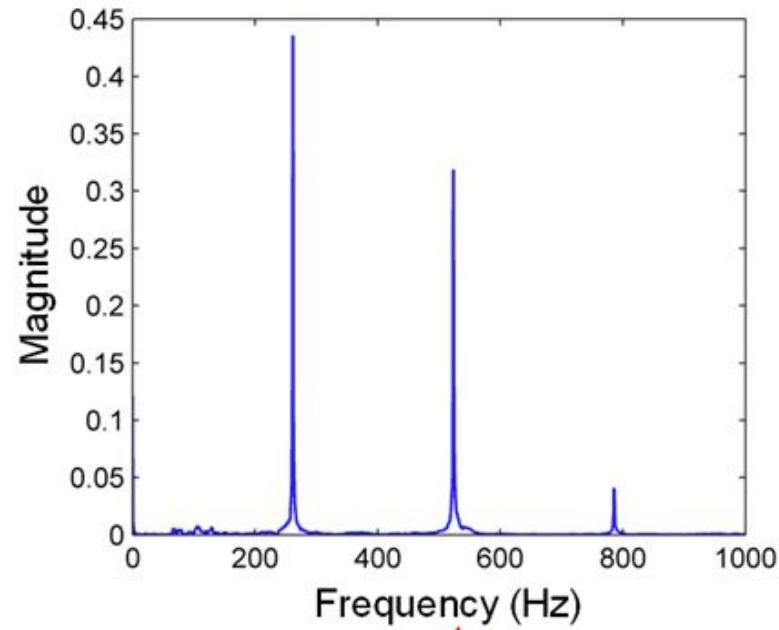
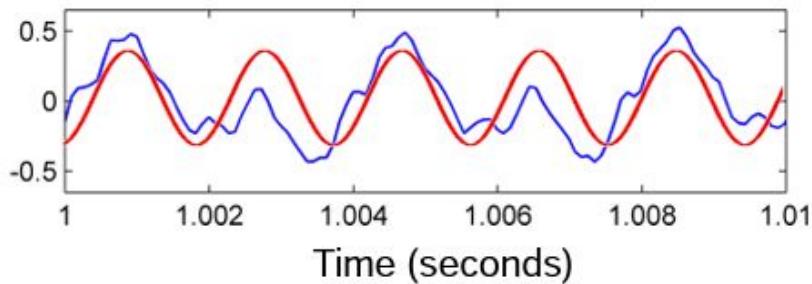
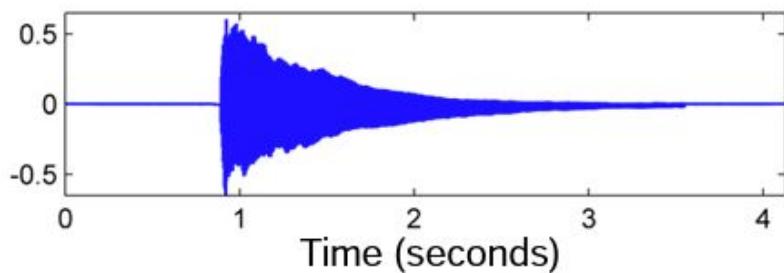
Fourier ctn.



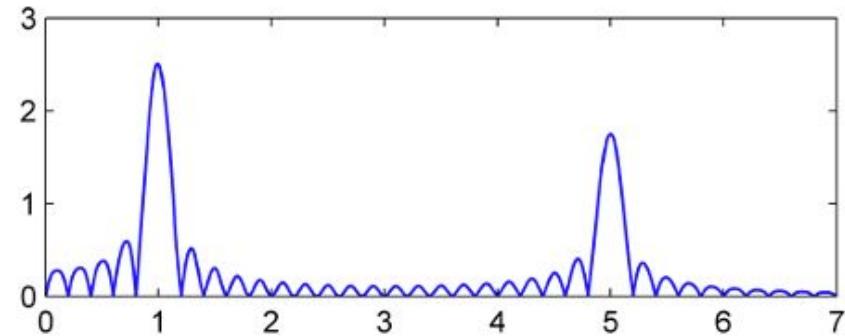
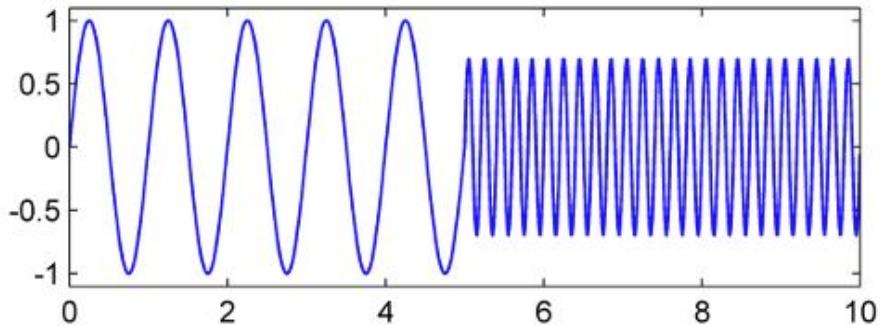
Fourier ctn.



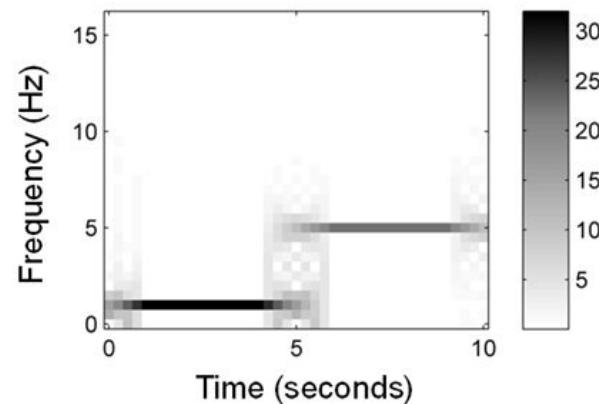
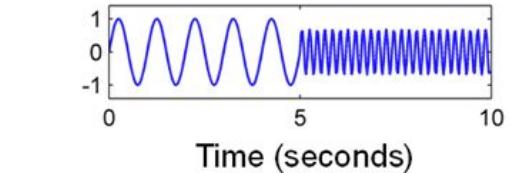
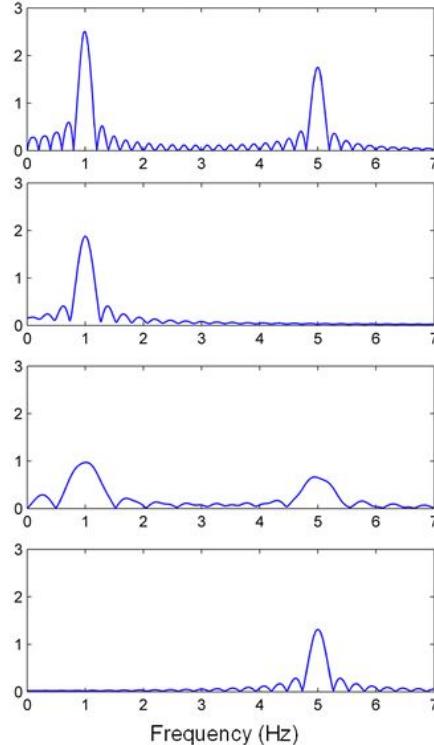
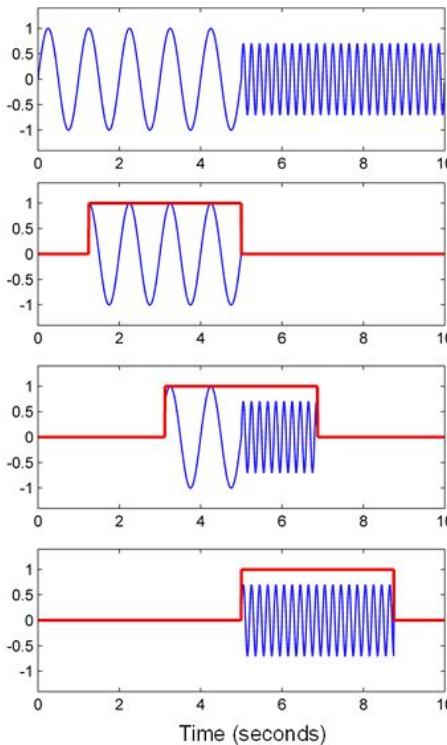
Fourier ctn.



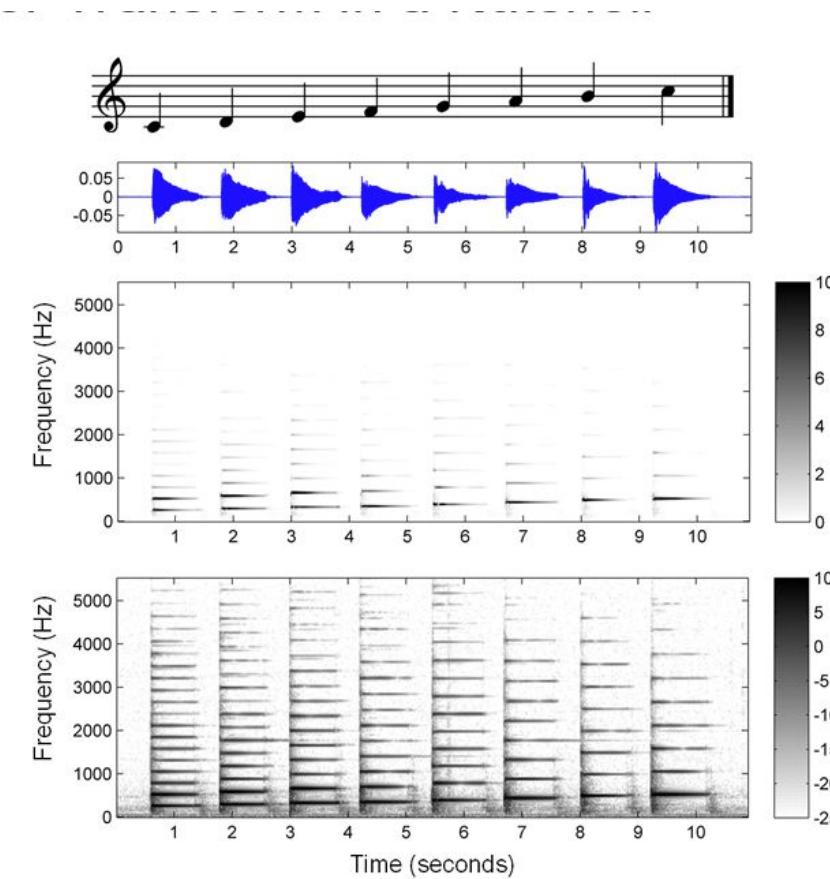
Pitch change...



FFT across time: short-term fourier transformation (STFT)

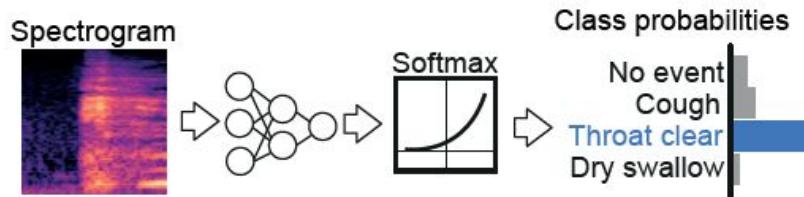


More complex input

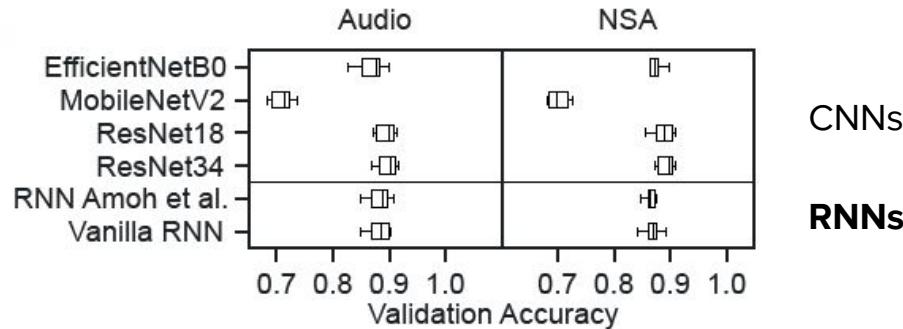


Audio input as spectrogram to DNNs

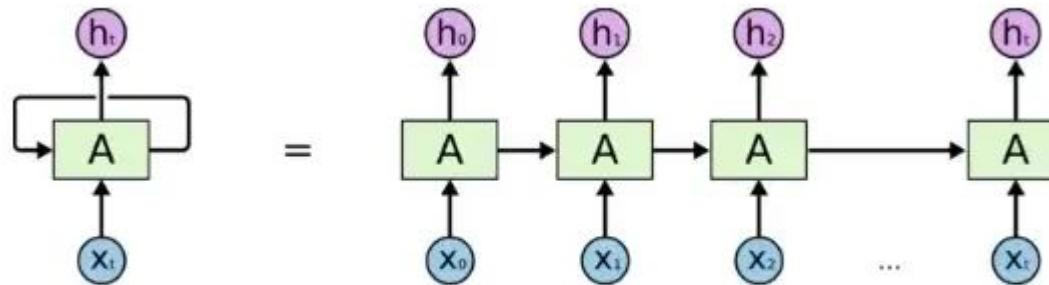
A



B



RNNs and LSTMs



An unrolled recurrent neural network.

$$h_t = \tanh (W_{hh}h_{t-1} + W_{xh}x_t) \quad \longrightarrow \quad y_t = W_{hy}h_t$$

RNNs pros and cons

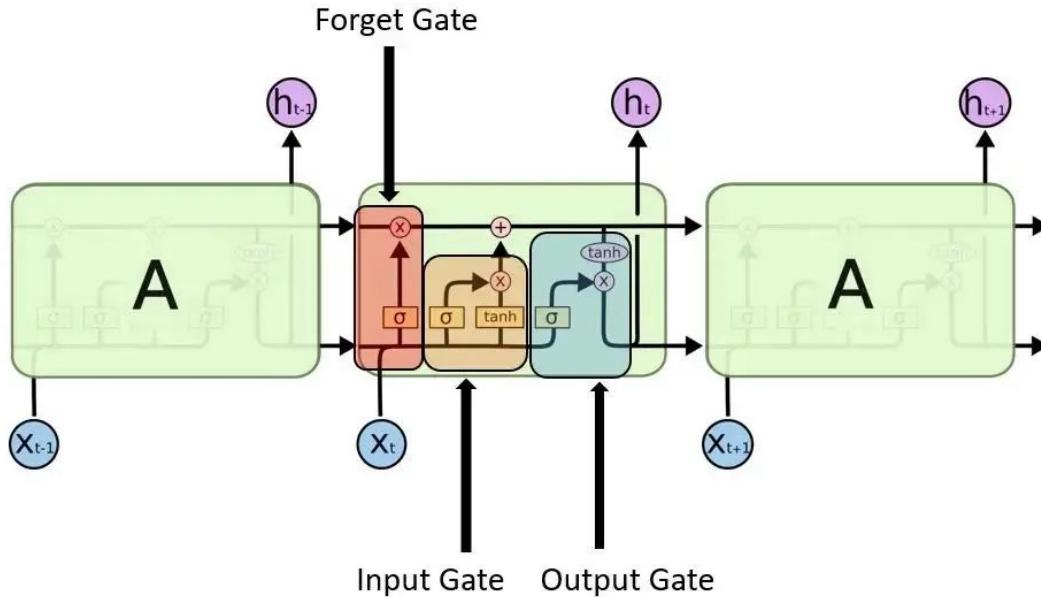
Advantages of Recurrent Neural Network

1. RNN can model sequence of data so that each sample can be assumed to be **dependent on previous ones**
2. Recurrent neural network are even used with convolutional layers to extend the effective pixel neighbourhood.

Disadvantages of Recurrent Neural Network

1. Gradient vanishing and exploding problems.
2. Training an RNN is a very difficult task.
3. It cannot process very long sequences if using *tanh* or *relu* as an activation function.

Long-short term memory cells



Forget gate: decides what to keep and what to forget

$$f_t = \sigma (W_f \cdot [h_{t-1}, x_t] + b_f)$$

Input gate: how to shape new memory

Let through stage
 $i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i)$

Importance stage
 $\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$

Output gate: How to combine input and memory to shape output

$$o_t = \sigma (W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

More time variant, and time dependent data

Fold napkins. Polish silverware. Wash dishes.

French napkins. Polish silverware. German dishes.

⇒ Context is important!

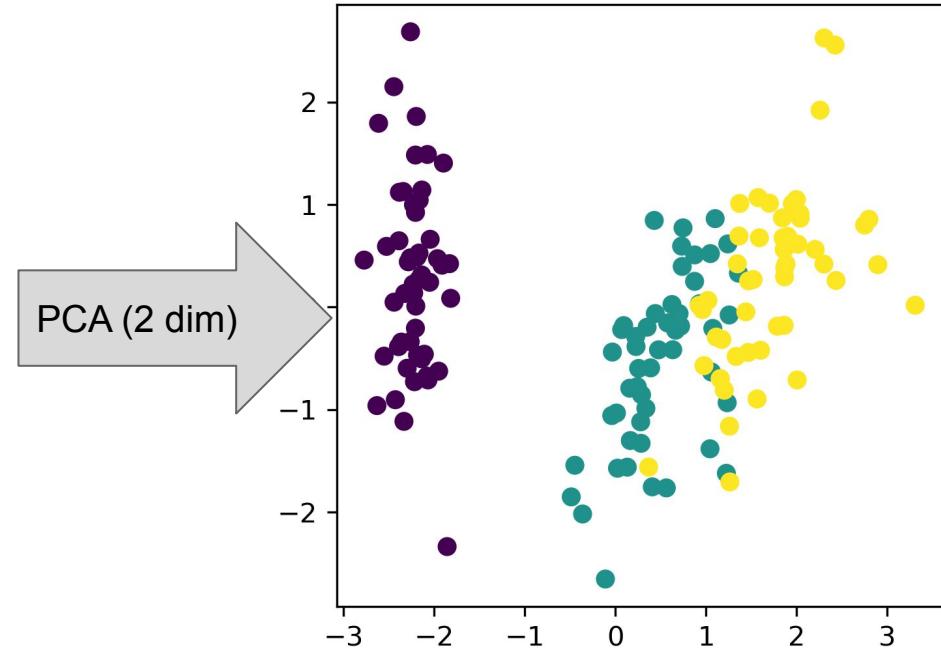
However, how to feed text towards DNNs?

Embeddings

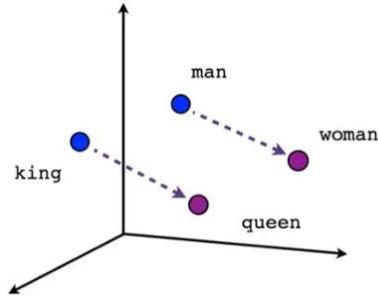
An embedding can always be created if you can compute a distance between values.

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

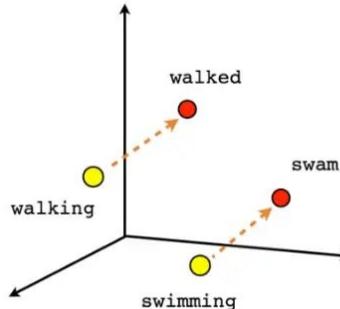
150 rows × 4 columns



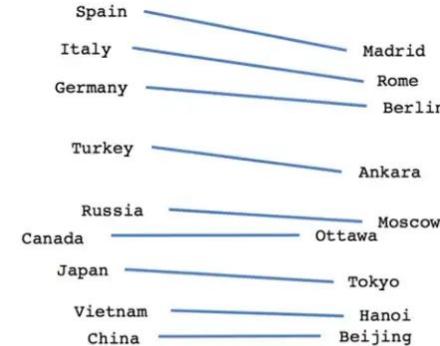
Word embeddings



Male-Female



Verb tense



Country-Capital

“**Word embedding** is the collective name for a set of language modeling and feature learning techniques in natural language processing (NLP) where **words or phrases from the vocabulary are mapped to vectors of real numbers**. The term word2vec literally translates to **word to vector**.” (Wikipedia, good enough)

“dad” = [0.1548, 0.4848, ..., 1.864]

“mom” = [0.8785, 0.8974, ..., 2.794]

Context!

The future king is the prince
Daughter is the princess
Son is the prince
Only a man can be a king
Only a woman can be a queen
The princess will be a queen
Queen and king rule the realm
The prince is a strong man
The princess is a beautiful woman
The royal family is the king and queen and their children
Prince is only a boy now
A boy will be a man

Window context (w=2)

The *future king* **is** the **prince**

All pairs

(**The**, *future*), (**The**, *king*),
(**future**, *the*), (**future**, *king*), (**future**, *is*)
(**king**, *the*), (**king**, *future*), (**king**, *is*), (**king**, *the*)
(**is**, *future*), (**is**, *king*), (**is**, *the*), (**is**, *prince*),
(**the**, *king*), (**the**, *is*), (**the**, *prince*)
(**prince**, *is*), (**prince**, *the*)

All pairs, cleaned

```
['future', 'king'],
['future', 'prince'],
['king', 'prince'],
['king', 'future'],
['prince', 'king'],
['prince', 'future'],
['daughter', 'princess'],
['princess', 'daughter'],
['son', 'prince']  
...  
...
```

Encoding and training

```
unique_word_dict = {  
    'beautiful': 0,  
    'boy': 1,  
    'can': 2,  
    'children': 3,  
    'daughter': 4,  
    'family': 5,  
    'future': 6,  
    'king': 7,  
    'man': 8,  
    'now': 9,  
    'only': 10,  
    'prince': 11,  
    'princess': 12,  
    'queen': 13,  
    'realm': 14,  
    'royal': 15,  
    'rule': 16,  
    'son': 17,  
    'strong': 18,  
    'their': 19,  
    'woman': 20  
}
```

One-hot encoding:

```
'blue' = [1, 0, 0]  
'car' = [0, 1, 0]  
'sky' = [0, 0, 1]
```

```
a = ['blue', 'sky', 'blue', 'car']
```



```
A =  
[  
    1, 0, 0  
    0, 0, 1  
    1, 0, 0  
    0, 1, 0  
]
```

X= focus
words

Y= context
words

```
['future', 'king'],  
 ['future', 'prince'],  
 ['king', 'prince']
```

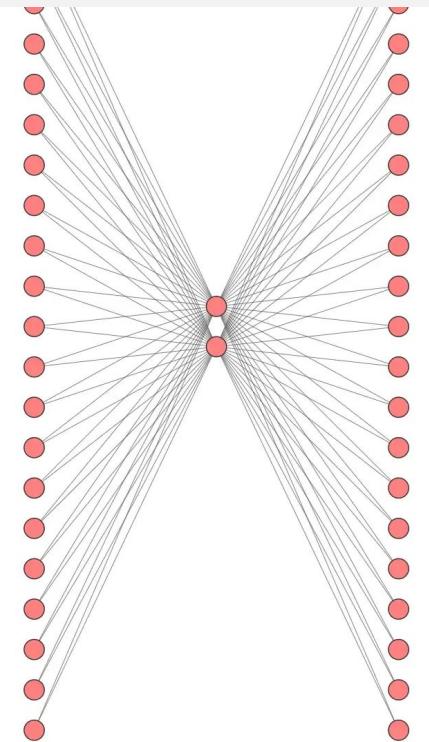
```
[array([0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,  
        0., 0.,  
        0., 0., 0., 0.]),  
 array([0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,  
        0., 0.,  
        0., 0., 0., 0.]),  
 array([0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0.,  
        0., 0.,  
        0., 0., 0., 0.])]
```

```
[array([0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.,  
        0., 0.,  
        0., 0., 0., 0.]),  
 array([0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.,  
        0., 0.,  
        0., 0., 0., 0.]),  
 array([0., 0., 0., 0., 0., 0., 1., 0., 0., 0., 0., 0., 0., 0., 0.,  
        0., 0.,  
        0., 0., 0., 0.])]
```

Training embedding

```
['future', 'king'],
['future', 'prince'],
['king', 'prince']
```

X= focus
words

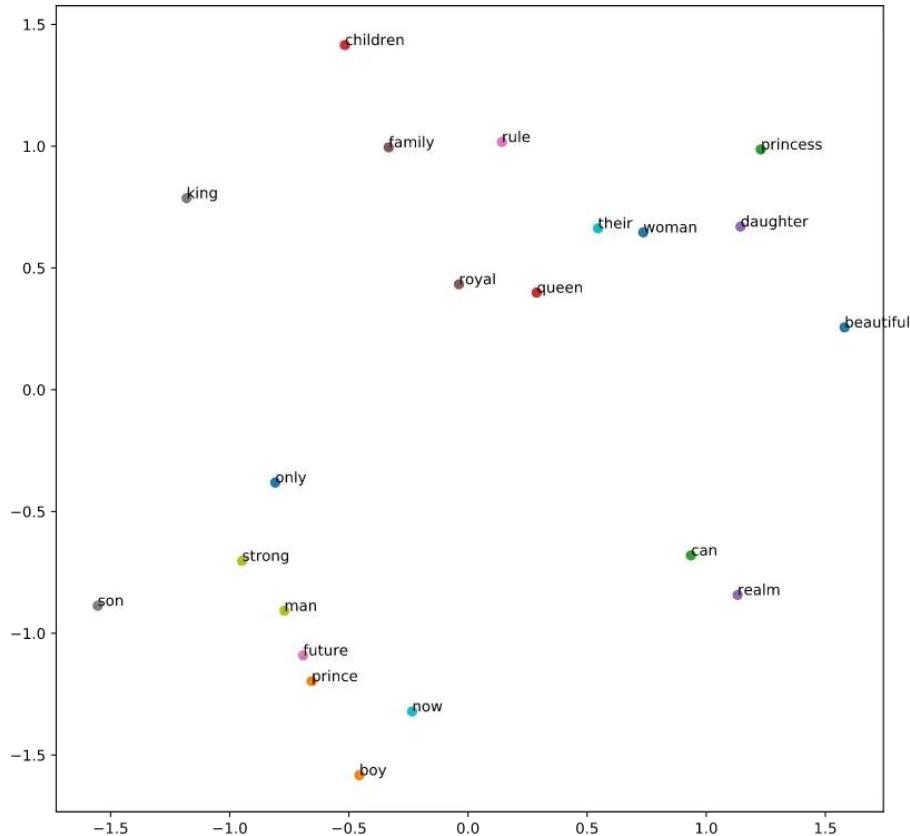


Input Layer $\in \mathbb{R}^{21}$

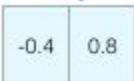
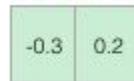
Hidden Layer $\in \mathbb{R}^2$

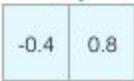
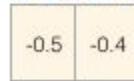
Output Layer $\in \mathbb{R}^{21}$

Y= context
words

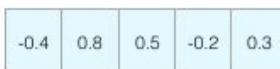


Similarity using distances

`cosine_similarity(`  `,`  `) = 0.87` ✓

`cosine_similarity(`  `,`  `) = -0.20`

1- We can represent things (and people) as vectors of numbers
(Which is great for machines!)

Jay 

2- We can easily calculate how similar vectors are to each other

The people most similar to Jay are:

`cosine_similarity` ▼

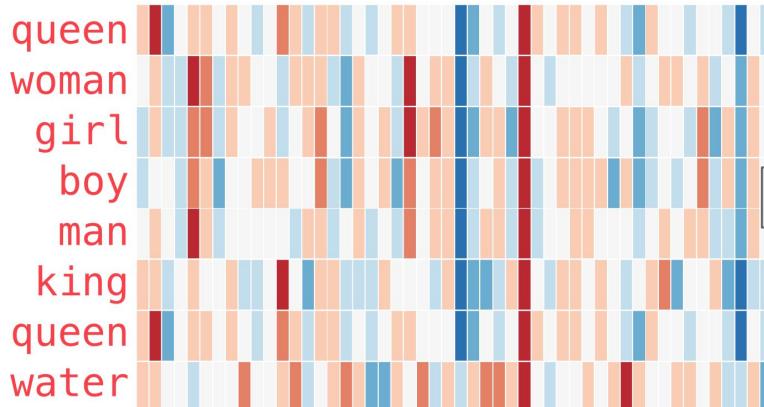
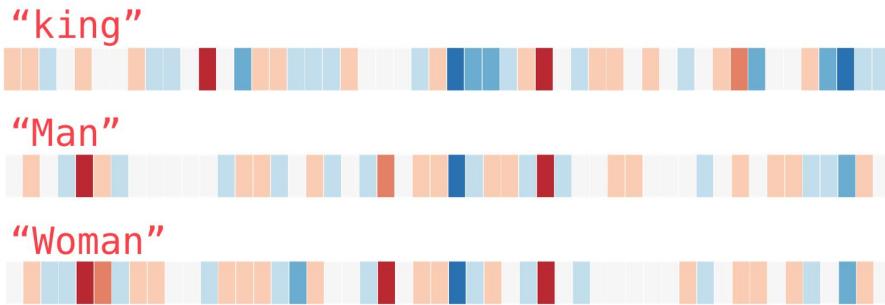
Person #1	0.86
Person #2	0.5
Person #3	-0.20

Embedding size:

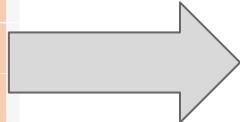
2-15 rather interchangeable (also consider antonyms - similar context such as good and bad)

15-50 similarity rather defines relatedness

Example for logic



$$\text{king} - \text{man} + \text{woman} \approx \text{queen}$$



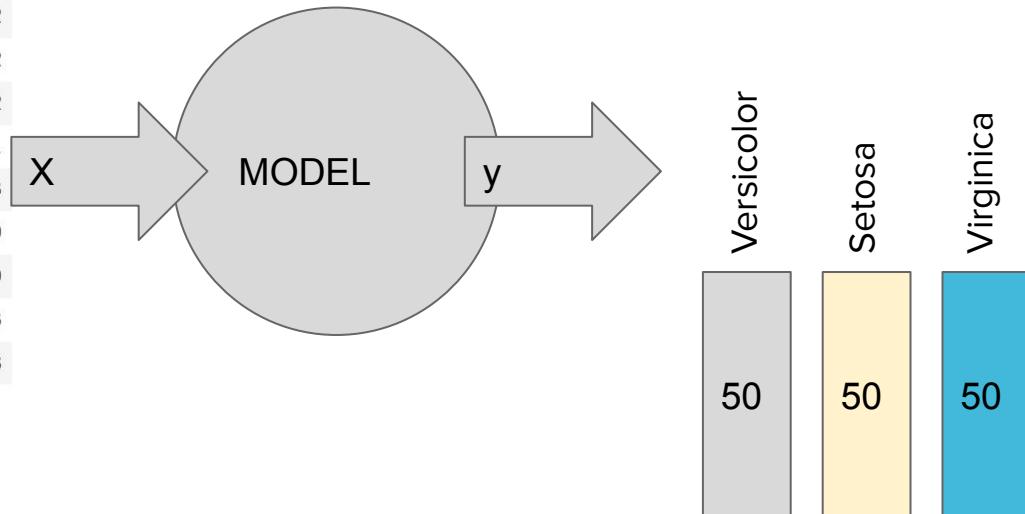
Part III: Model evaluation

Iris dataset

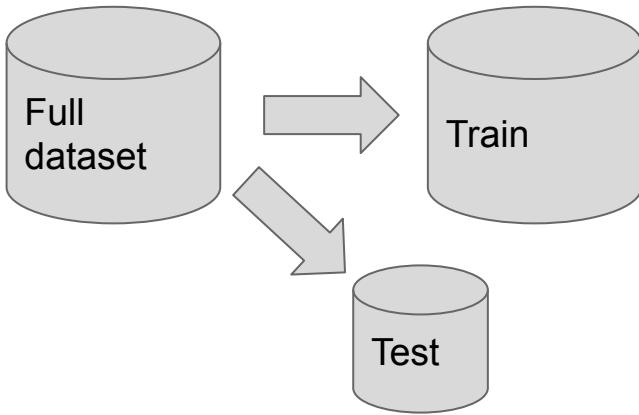
Baseline accuracy: 33%!

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns



Okay, can we determine if the model learnt?



- training set → $38 \times$ Setosa, $28 \times$ Versicolor, $34 \times$ Virginica
- test set → $12 \times$ Setosa, $22 \times$ Versicolor, $16 \times$ Virginica

i.i.d. We assume that the training examples are i.i.d (independent and identically distributed), which means that all examples have been drawn from the same probability distribution and are statistically independent from each other. A scenario where training examples are not independent would be working with temporal data or time-series data.

⇒ STRATIFICATION of your data
Allowing equal sampling of classes

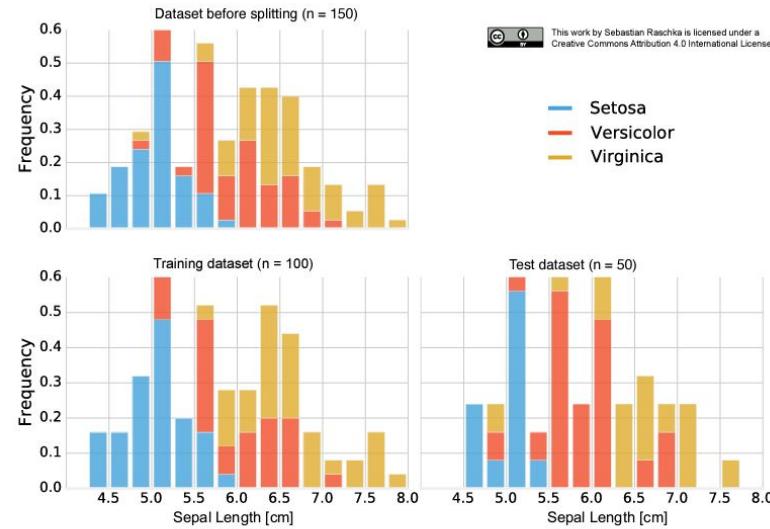


Figure 1: Distribution of *Iris* flower classes upon random subsampling into training and test sets.

Holdout validation

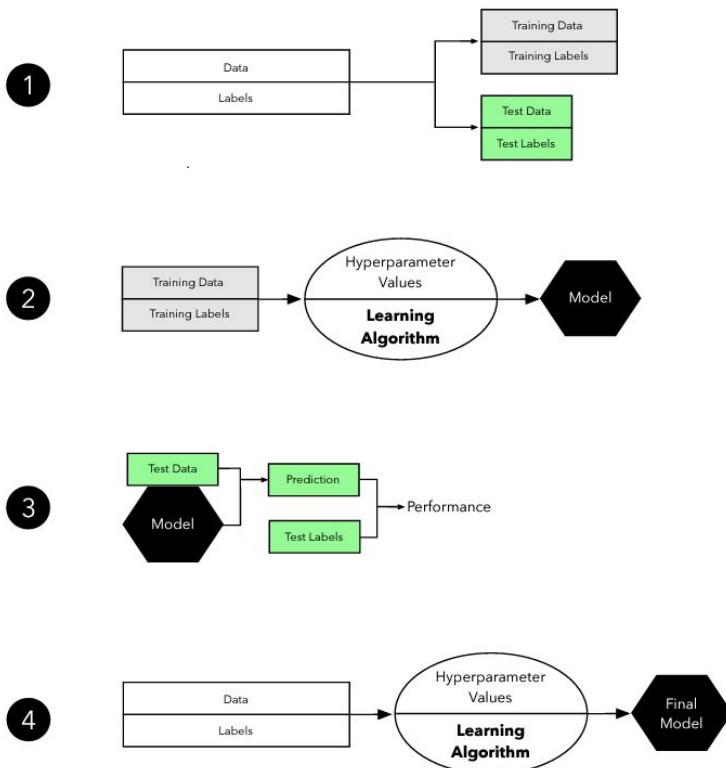
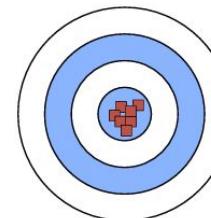


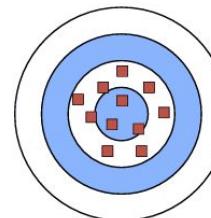
Figure 2: Visual summary of the holdout validation method.

Increasing
training/decreasing test

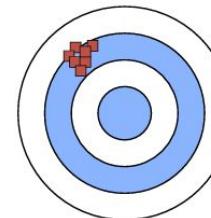
Low Variance
(Precise)



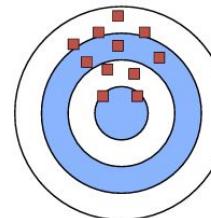
High Variance
(Not Precise)



Low Bias
(Accurate)



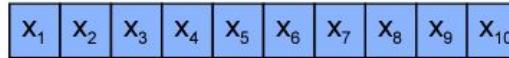
High Bias
(Not Accurate)



Training = test set \Rightarrow optimistically biased, test set too large in respect to training set \Rightarrow pessimistically biased

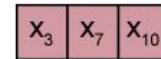
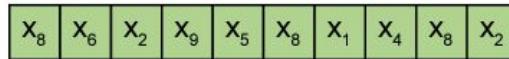
Bootstrapping

Original Dataset

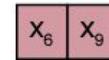
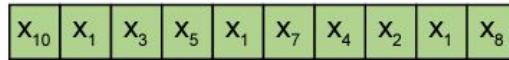


Sampling/drawing always **from the original dataset!**

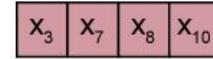
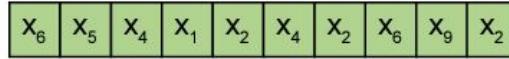
Bootstrap 1



Bootstrap 2



Bootstrap 3



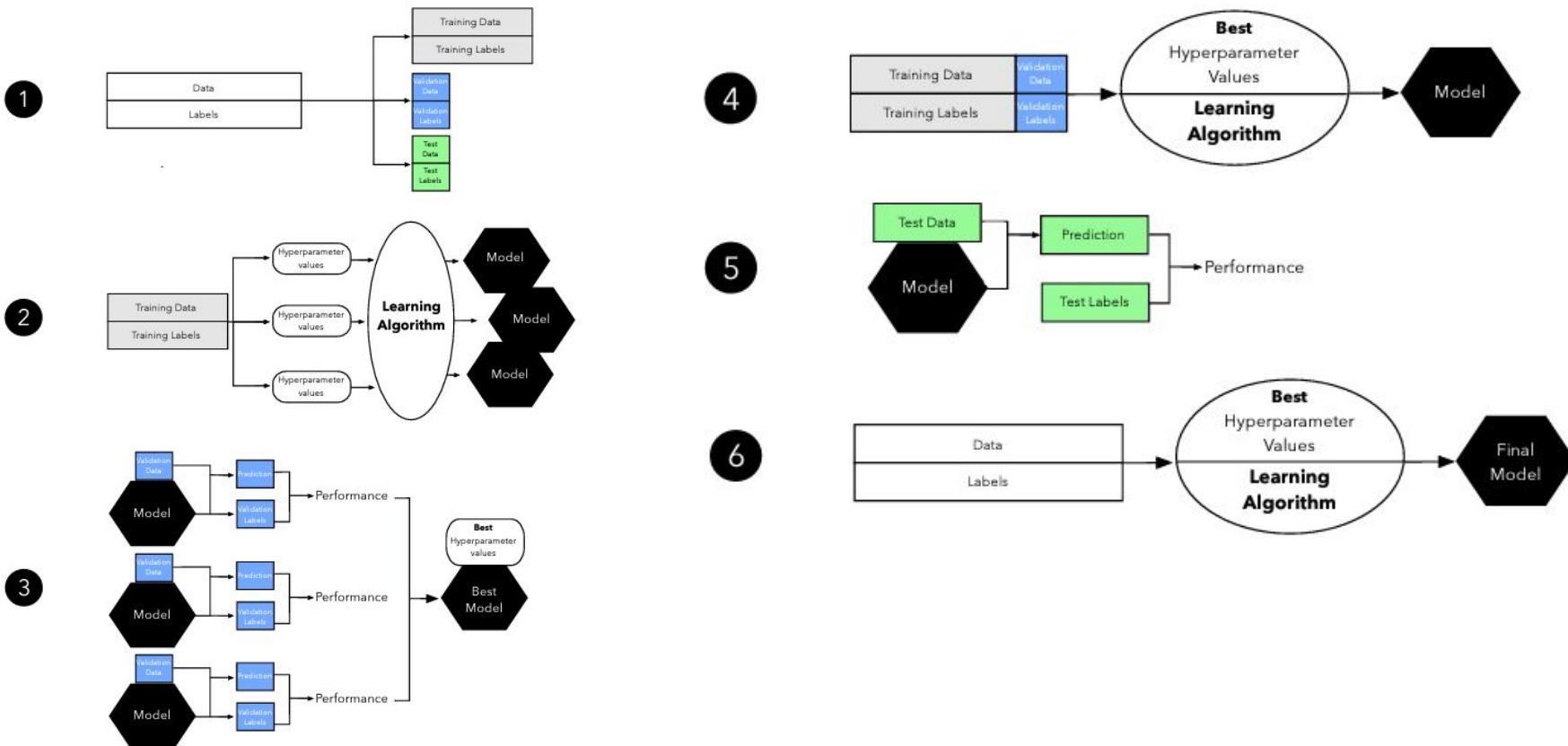
Training Sets

Test Sets

Figure 7: Illustration of training and test data splits in the Leave-One-Out Bootstrap (LOOB).

50-200 bootstrap samples should be good
[Efron and Tibshirani, 1994]

Hyperparameter tuning/model selection



K-fold crossvalidation

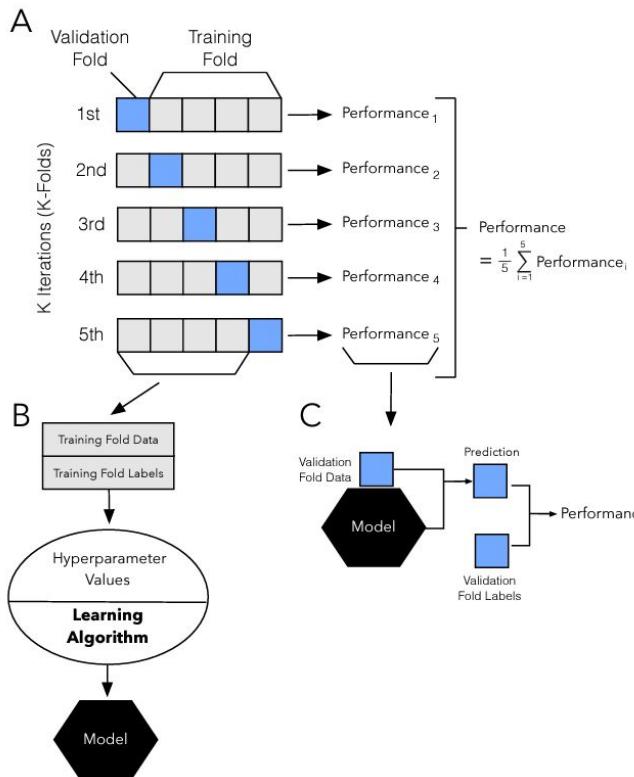


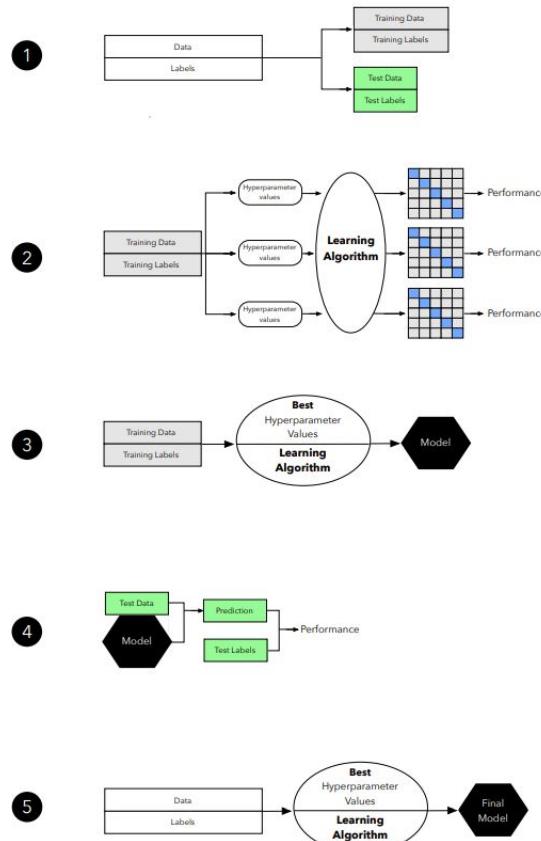
Figure 13: Illustration of the k -fold cross-validation procedure.

Leave-one-out-Val.



Figure 15: Illustration of leave-one-out cross-validation.

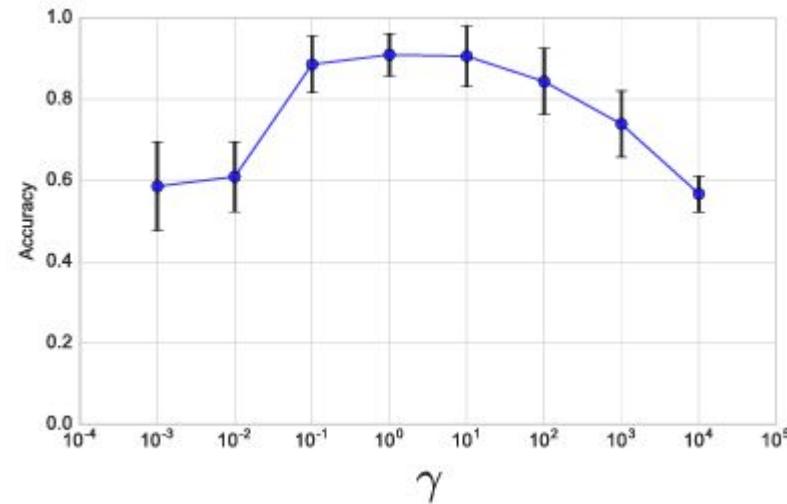
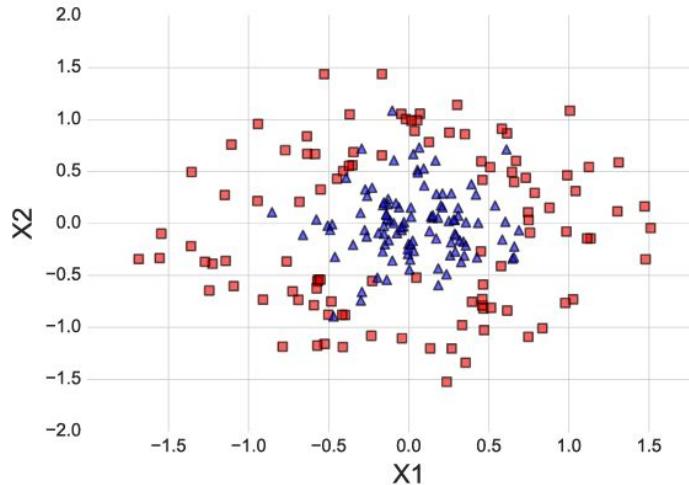
Model selection w/ k-fold crossval



Example: tuning hyperparameter Gamma in SVMs

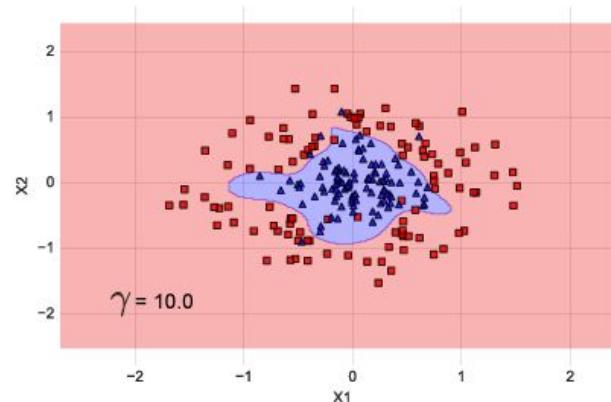
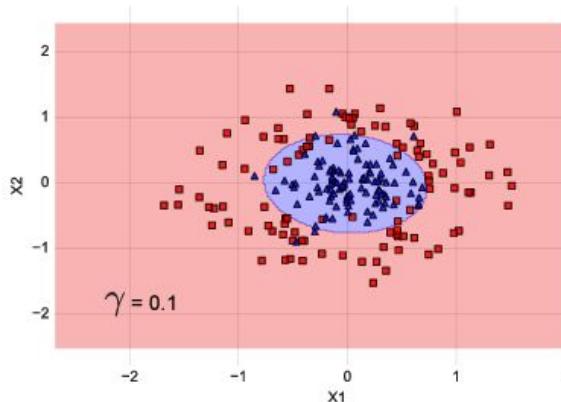
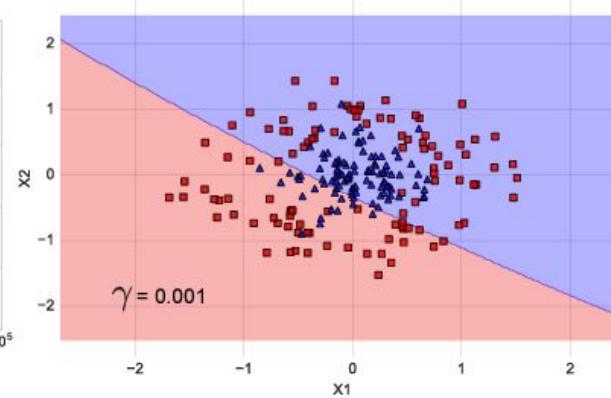
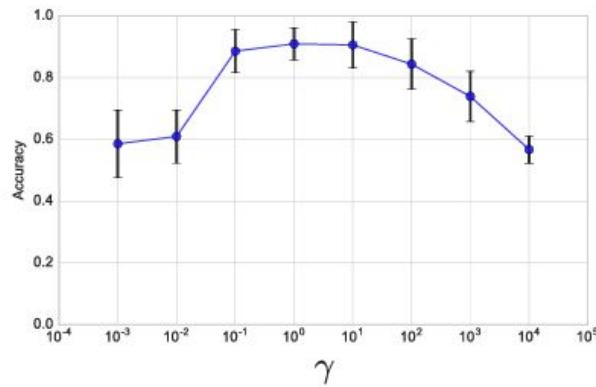
Now, let us assume that our goal is to optimize the γ (gamma) hyperparameter of a Support Vector Machine (SVM) with a non-linear Radial Basis Function-kernel (RBF-kernel), where γ is the free parameter of the Gaussian RBF:

$$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2), \gamma > 0. \quad (46)$$

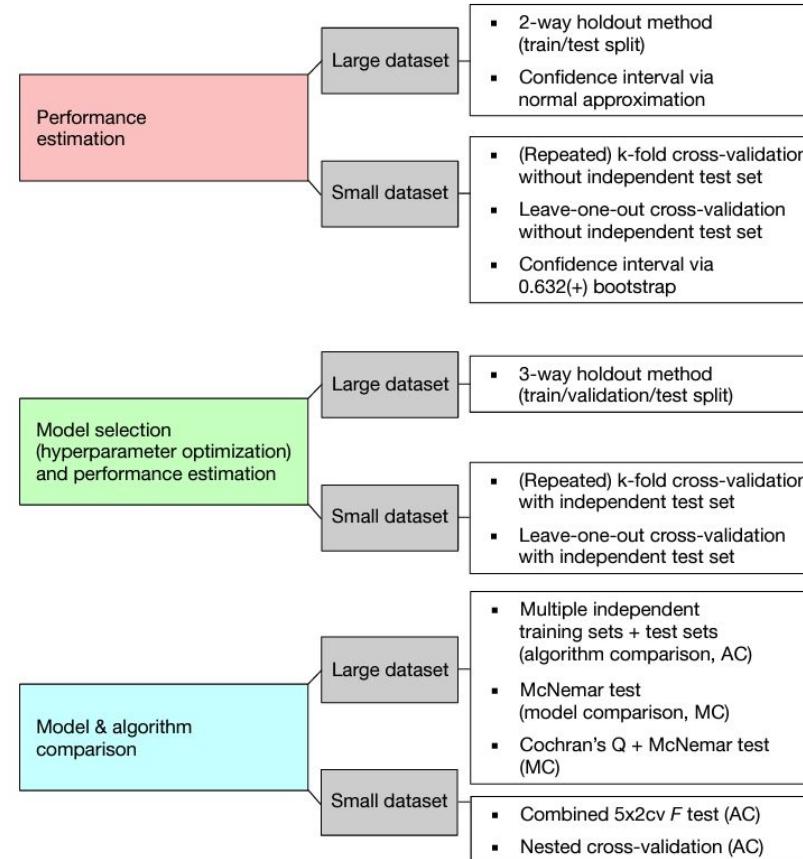


10-fold crossvalidation, mean performance, error bars indicate std.

Looking at Gamma in detail



Summary



Homework

In this homework, you will build, train and evaluate a Convolutional Neural Network using TensorFlow.

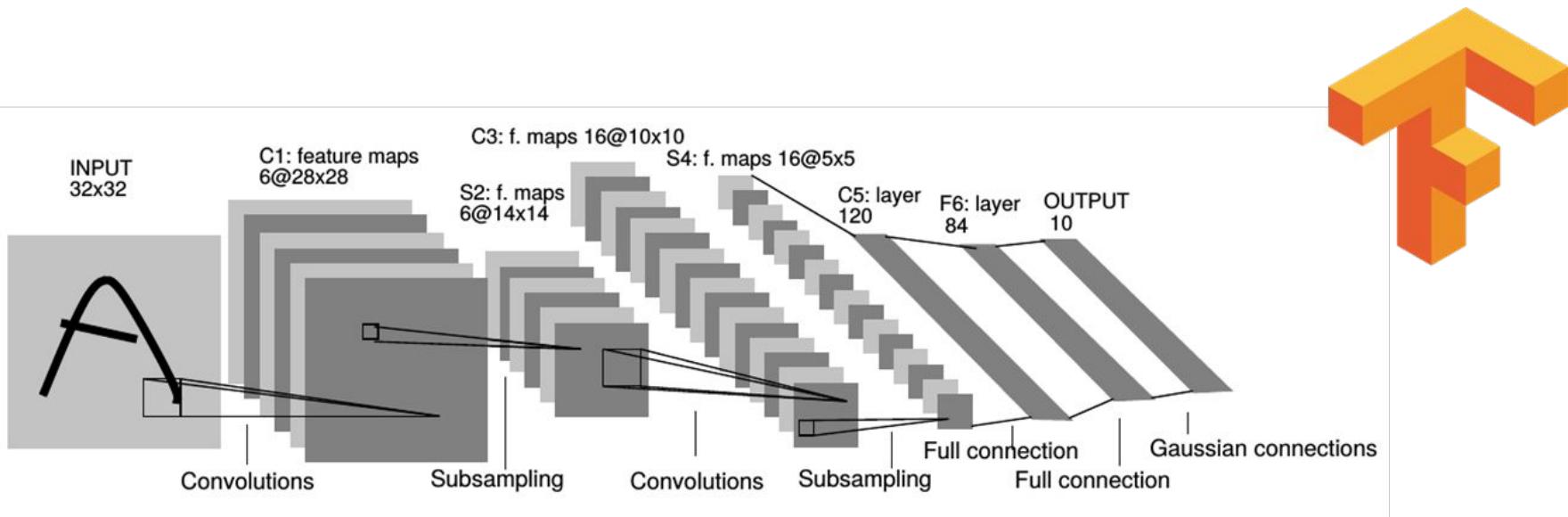


Fig. 2. Architecture of LeNet-5, a convolutional NN, here used for digits recognition. Each plane is a feature map, i.e., a set of units whose weights are constrained to be identical.