ML Fundamentals

Instituto Balseiro

Instituto Balseiro
20/09/2022

Lecture 10
Deep learning II

Announcements

- Ya (casi) subimos el test set
  - Tienen que generar predicciones con su modelo y reportar el error de test en el Classroom (mismo lugar desde donde mandaron el informe).
    El resultado no impacta en su evaluación.
- Las notas van a estar antes del viernes,
- Subimos también el demo de la guía 03,
- En breve subimos la guía 04, la última.

ML Fundamentals – Lecture 10

- **Backpropagation**
- **CNNs**
- RNNs
- Autoencoders
- GANs

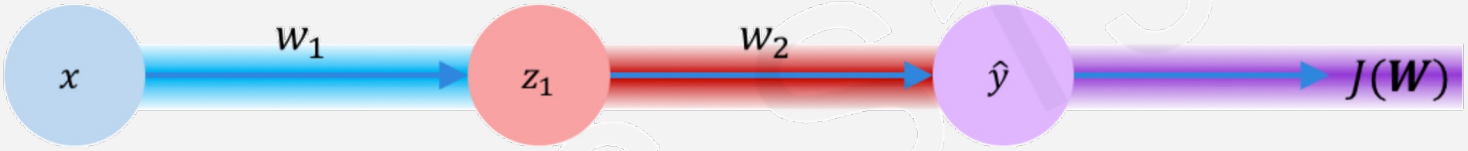
Backpropagation

aka *Reverse mode differentiation*

Learning representations by back-propagating errors

David E. Rumelhart*, Geoffrey E. Hinton† & Ronald J. Williams*

* Institute for Cognitive Science, C-015, University of California, San Diego, La Jolla, California 92093, USA
† Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Philadelphia 15213, USA



$$\frac{\partial J(\boldsymbol{W})}{\partial w_1} = \frac{\partial J(\boldsymbol{W})}{\partial \hat{y}} * \frac{\partial \hat{y}}{\partial z_1} * \frac{\partial z_1}{\partial w_1}$$

Repeat this for **every weight in the network** *using gradients from later layers*

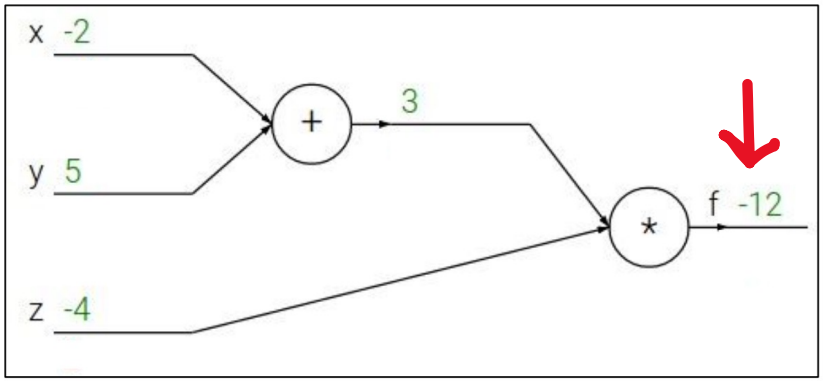
Computational graphs

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

Forward pass: f(-2, 5, -4)=-12

Graph connecting primitive operations



What's the influence of each input value on the output of the expression?

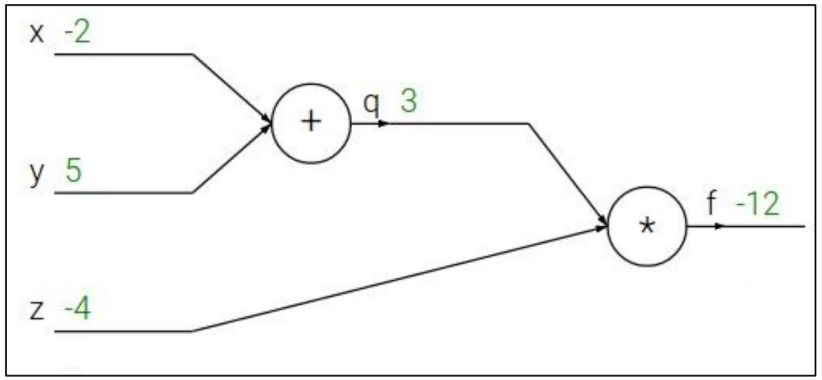
Key to computing gradient: understand derivatives in the edges

- In these graphs, edges have associated derivatives, measuring how downstream variable affects upstream variable
- Differentiable functions can be computed locally analytically
- The overall result is compounded via the *chain rule*

Computational graphs

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

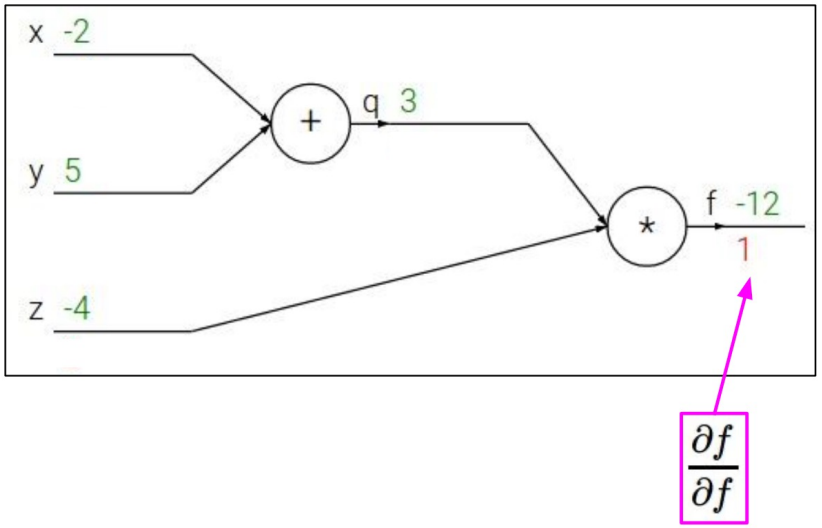
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

- In these graphs, edges have associated derivatives, measuring how downstream variable affects upstream variable
- Differentiable functions can be computed locally analytically
- The overall result is compounded via the *chain rule*

Computational graphs

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

- In these graphs, edges have associated derivatives, measuring how downstream variable affects upstream variable
- Differentiable functions can be computed locally analytically
- The overall result is compounded via the *chain rule*

Computational graphs

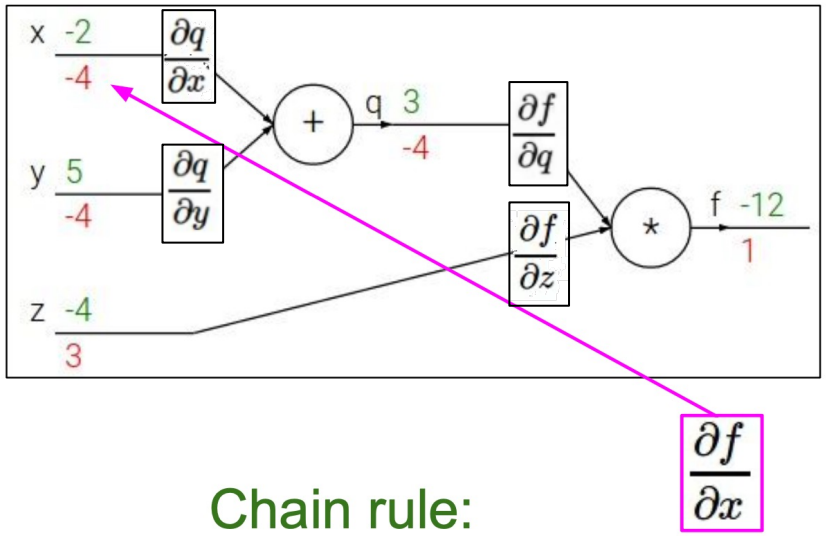
$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

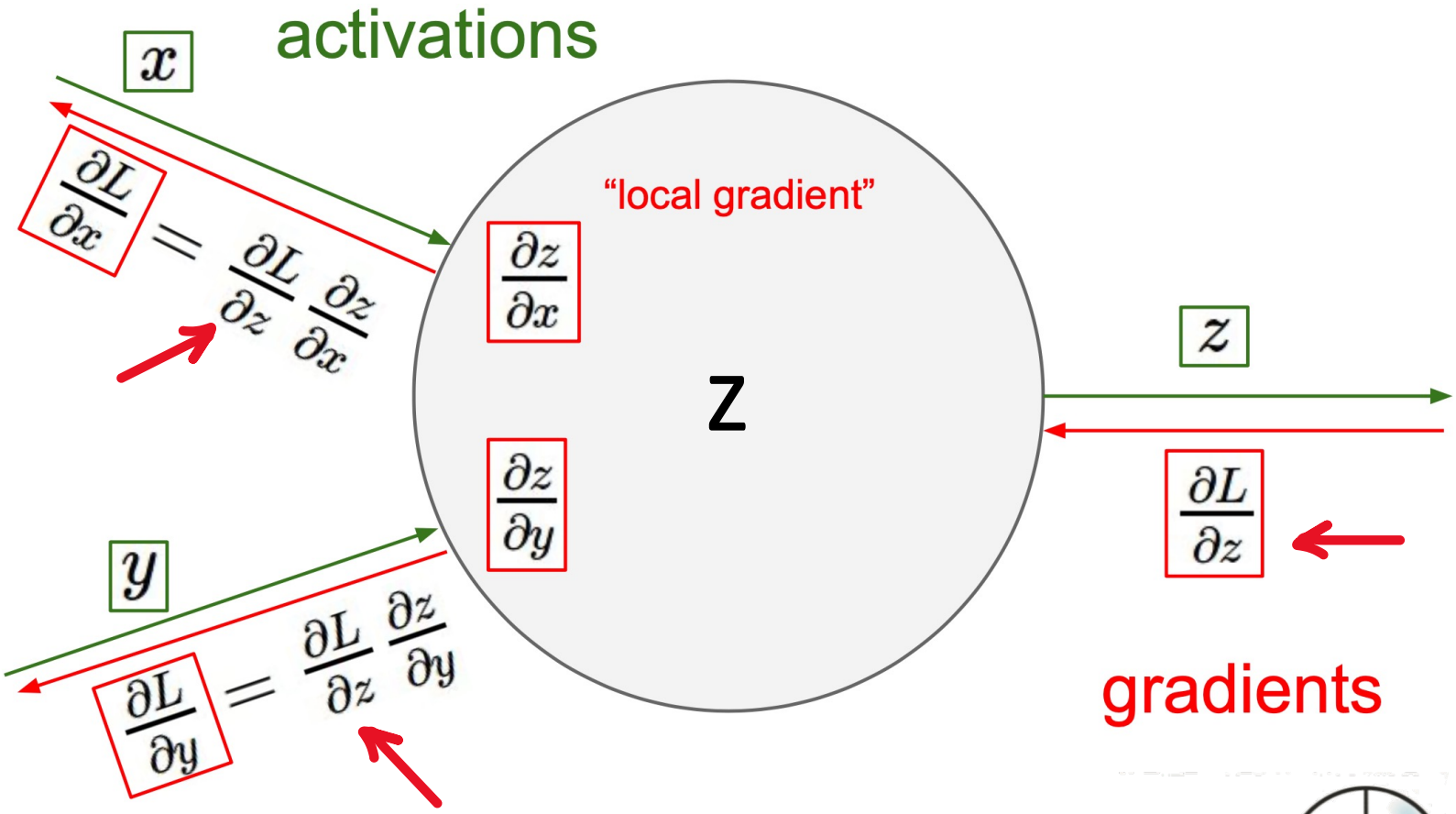


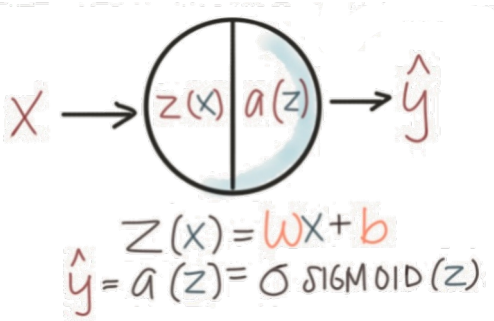
Chain rule:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x}$$

- In these graphs, edges have associated derivatives, measuring how downstream variable affects upstream variable
- Differentiable functions can be computed locally analytically
- The overall result is compounded via the *chain rule*

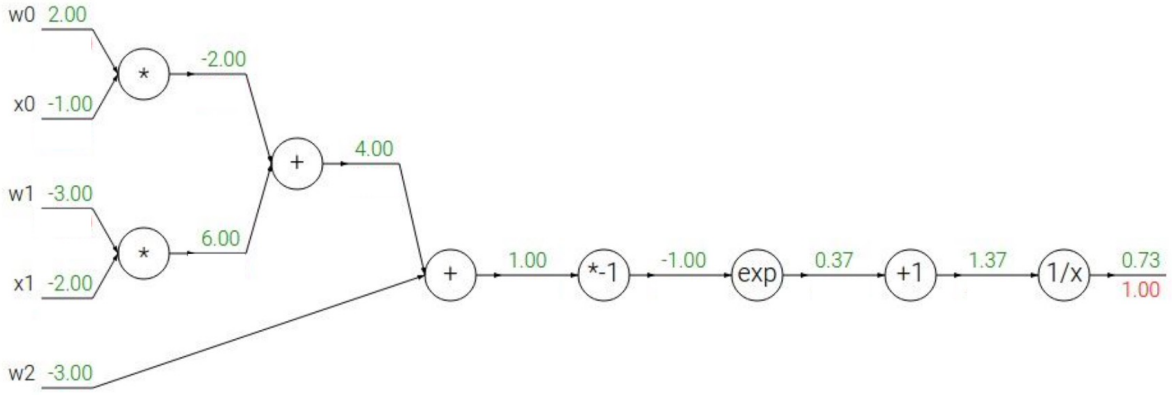
Gradients flow through the graph





Gradients flow through the graph

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$

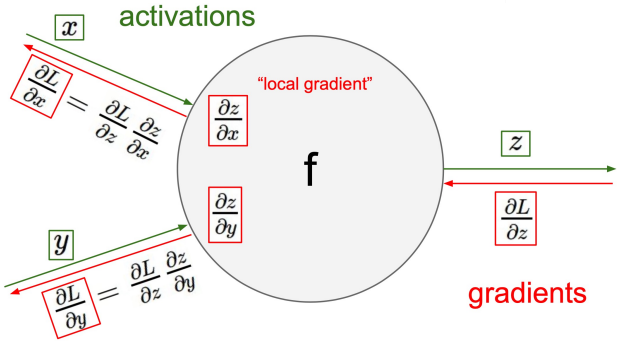


$z(x) = e^x \rightarrow \frac{\partial z}{\partial x} = e^x$

$z_a(x) = ax \rightarrow \frac{\partial z}{\partial x} = a$

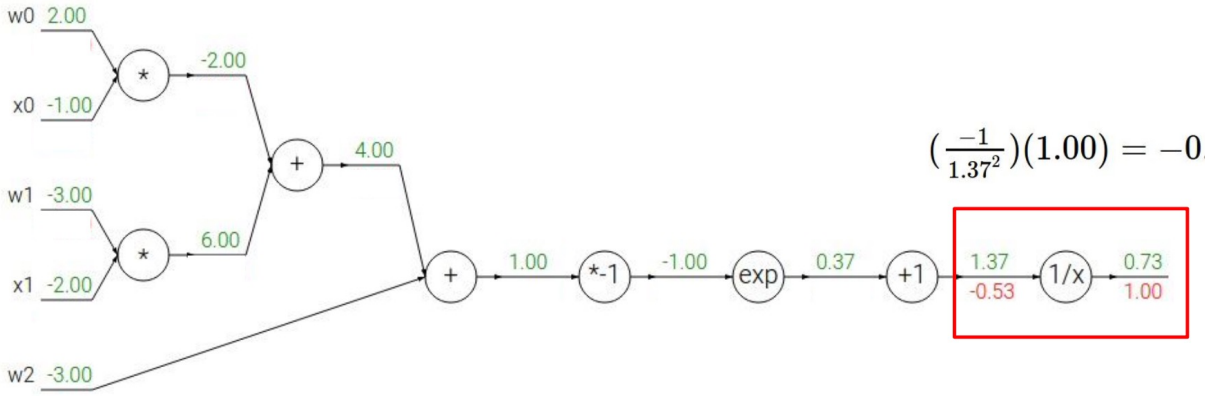
$z(x) = \frac{1}{x} \rightarrow \frac{\partial z}{\partial x} = -1/x^2$

$z_c(x) = c + x \rightarrow \frac{\partial z}{\partial x} = 1$



Gradients flow through the graph

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



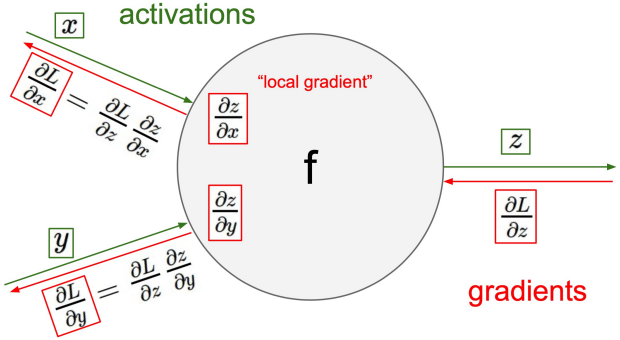
$(\frac{-1}{1.37^2})(1.00) = -0.53$

$z(x) = e^x \rightarrow \frac{\partial z}{\partial x} = e^x$

$z_a(x) = ax \rightarrow \frac{\partial z}{\partial x} = a$

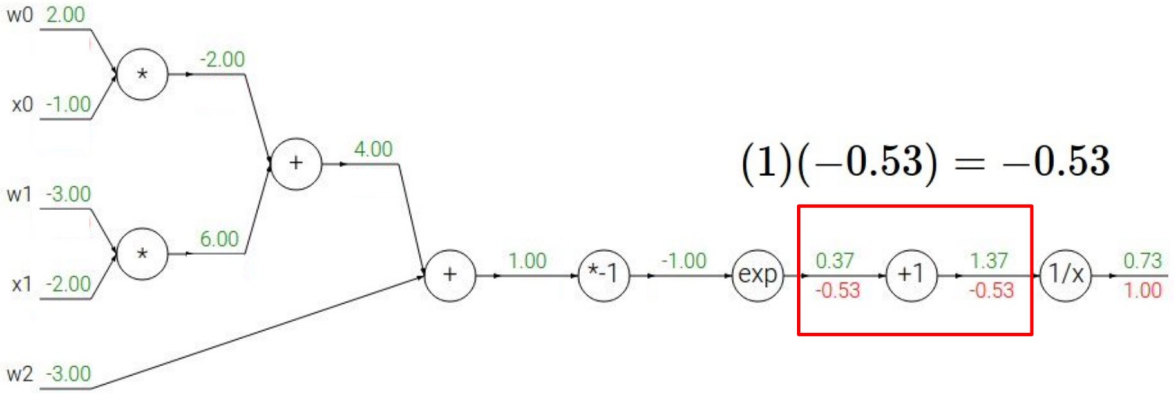
$z(x) = \frac{1}{x} \rightarrow \frac{\partial z}{\partial x} = -1/x^2$

$z_c(x) = c + x \rightarrow \frac{\partial z}{\partial x} = 1$



Gradients flow through the graph

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$

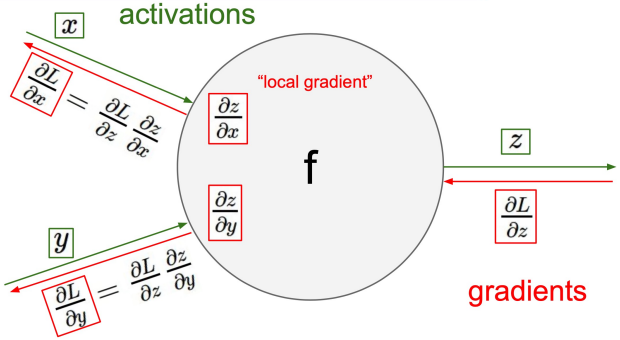


$z(x) = e^x \rightarrow \frac{\partial z}{\partial x} = e^x$

$z_a(x) = ax \rightarrow \frac{\partial z}{\partial x} = a$

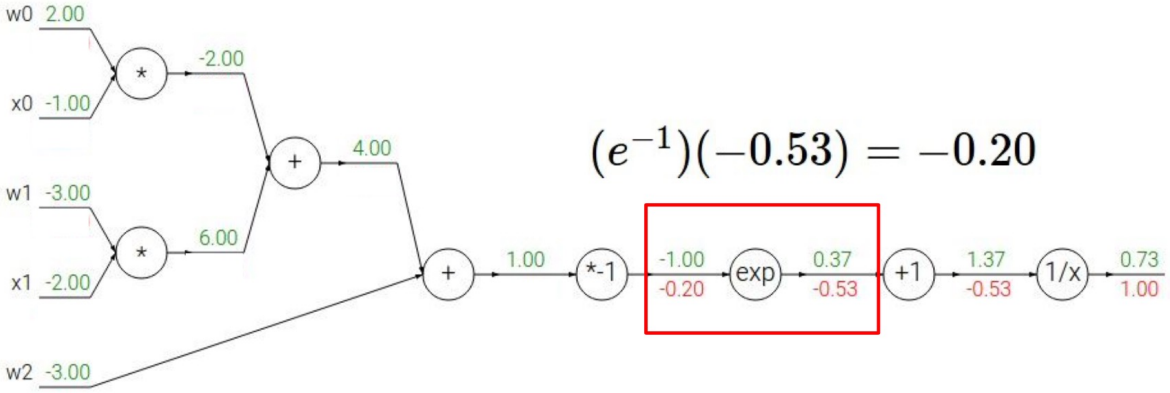
$z(x) = \frac{1}{x} \rightarrow \frac{\partial z}{\partial x} = -1/x^2$

$z_c(x) = c + x \rightarrow \frac{\partial z}{\partial x} = 1$



Gradients flow through the graph

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



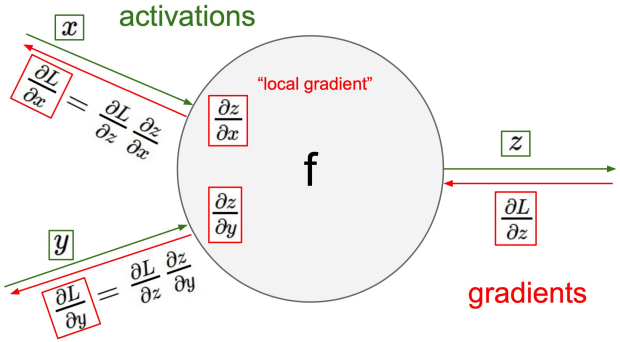
$(e^{-1})(-0.53) = -0.20$

$z(x) = e^x \rightarrow \frac{\partial z}{\partial x} = e^x$

$z_a(x) = ax \rightarrow \frac{\partial z}{\partial x} = a$

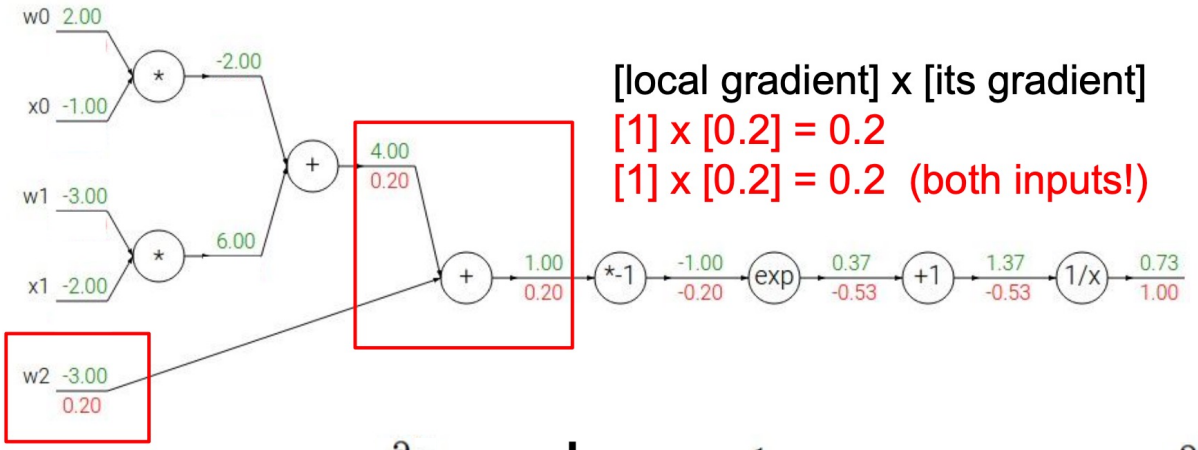
$z(x) = \frac{1}{x} \rightarrow \frac{\partial z}{\partial x} = -1/x^2$

$z_c(x) = c + x \rightarrow \frac{\partial z}{\partial x} = 1$



Gradients flow through the graph

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



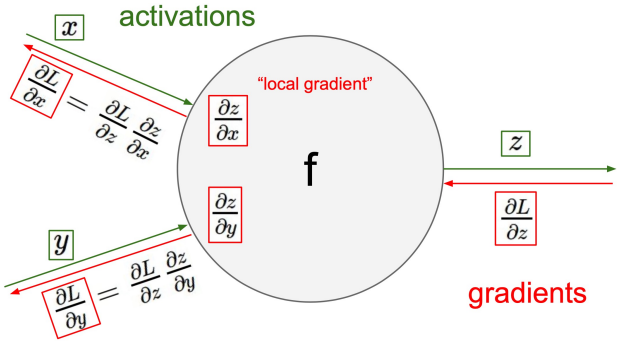
[local gradient] x [its gradient]
[1] x [0.2] = 0.2
[1] x [0.2] = 0.2 (both inputs!)

$z(x) = e^x \rightarrow \frac{\partial z}{\partial x} = e^x$

$z_a(x) = ax \rightarrow \frac{\partial z}{\partial x} = a$

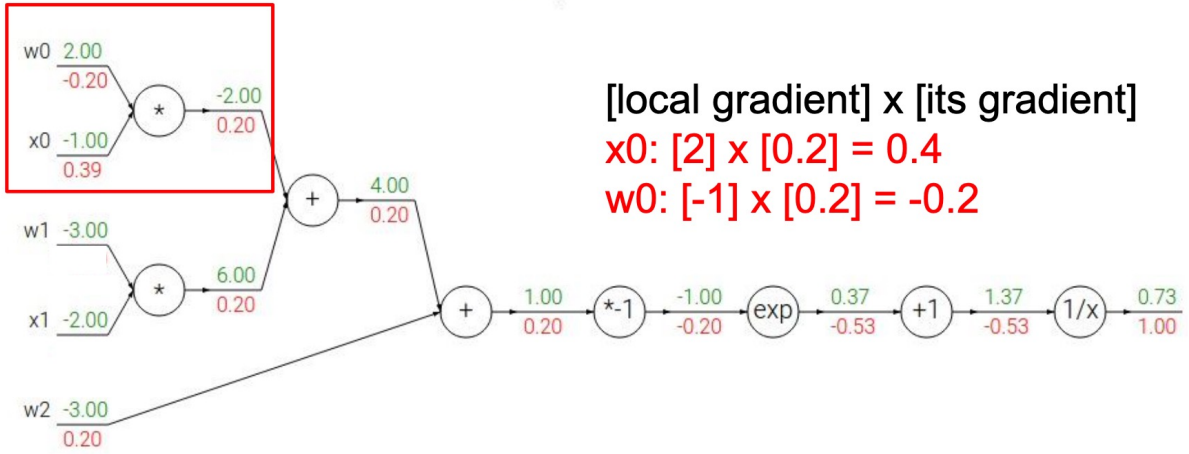
$z(x) = \frac{1}{x} \rightarrow \frac{\partial z}{\partial x} = -1/x^2$

$z_c(x) = c + x \rightarrow \frac{\partial z}{\partial x} = 1$



Gradients flow through the graph

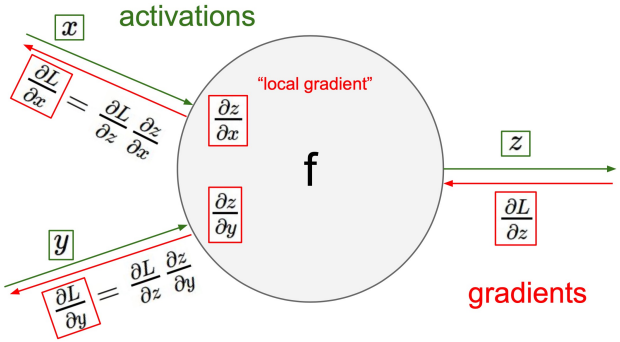
Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$



[local gradient] x [its gradient]
x0: [2] x [0.2] = 0.4
w0: [-1] x [0.2] = -0.2

$$z(x) = e^x \rightarrow \frac{\partial z}{\partial x} = e^x \qquad z(x) = \frac{1}{x} \rightarrow \frac{\partial z}{\partial x} = -1/x^2$$

$$z_a(x) = ax \rightarrow \frac{\partial z}{\partial x} = a \qquad z_c(x) = c + x \rightarrow \frac{\partial z}{\partial x} = 1$$

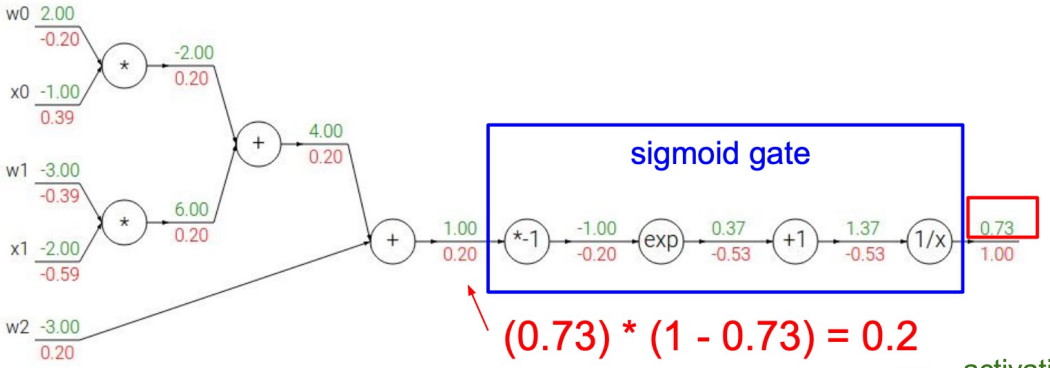


Gradients flow through the graph

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

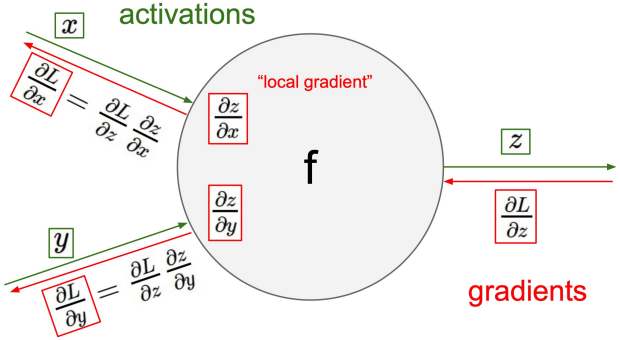
$$\sigma(x) = \frac{1}{1+e^{-x}}$$ sigmoid function

$$\frac{d\sigma(x)}{dx} = \frac{e^{-x}}{(1+e^{-x})^2} = \left(\frac{1+e^{-x}-1}{1+e^{-x}}\right)\left(\frac{1}{1+e^{-x}}\right) = (1-\sigma(x))\,\sigma(x)$$

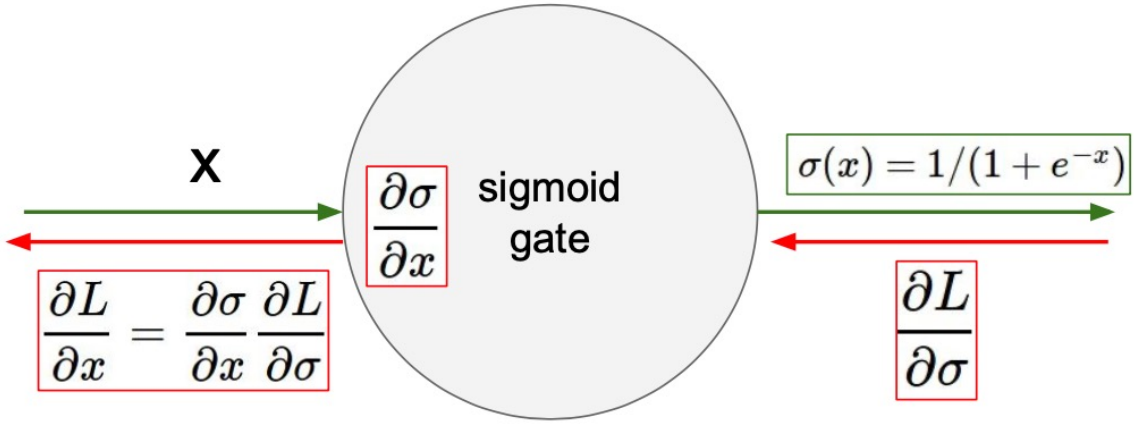


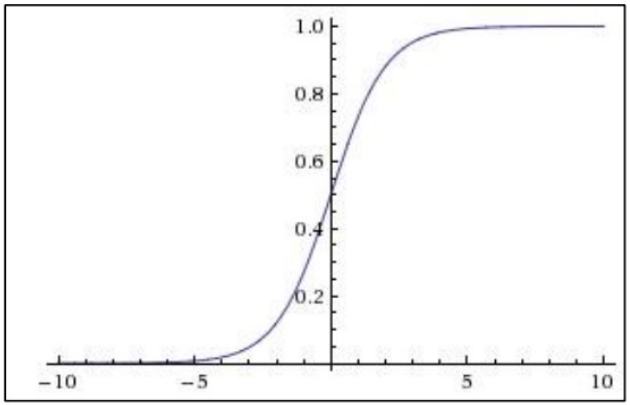
(0.73) * (1 - 0.73) = 0.2

σ'(x)= σ(x)(1- σ(x))



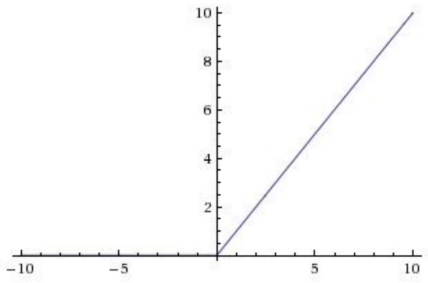
Vanishing/exploding gradients



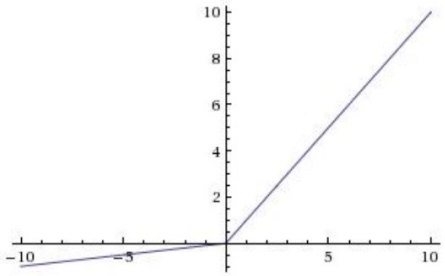


What happens when x = -10?
What happens when x = 0?
What happens when x = 10?

Activation functions



ReLU
(Rectified Linear Unit)



Leaky ReLU

$$f(x) = \max(0.01x, x)$$

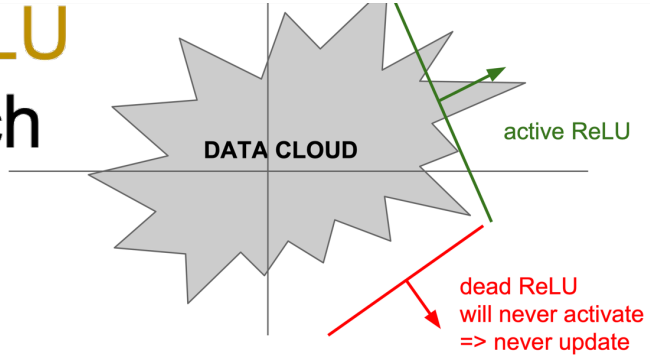
- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice! (e.g. 6x)
- **will not "die".**

Parametric Rectifier (PReLU)

$$f(x) = \max(\alpha x, x)$$

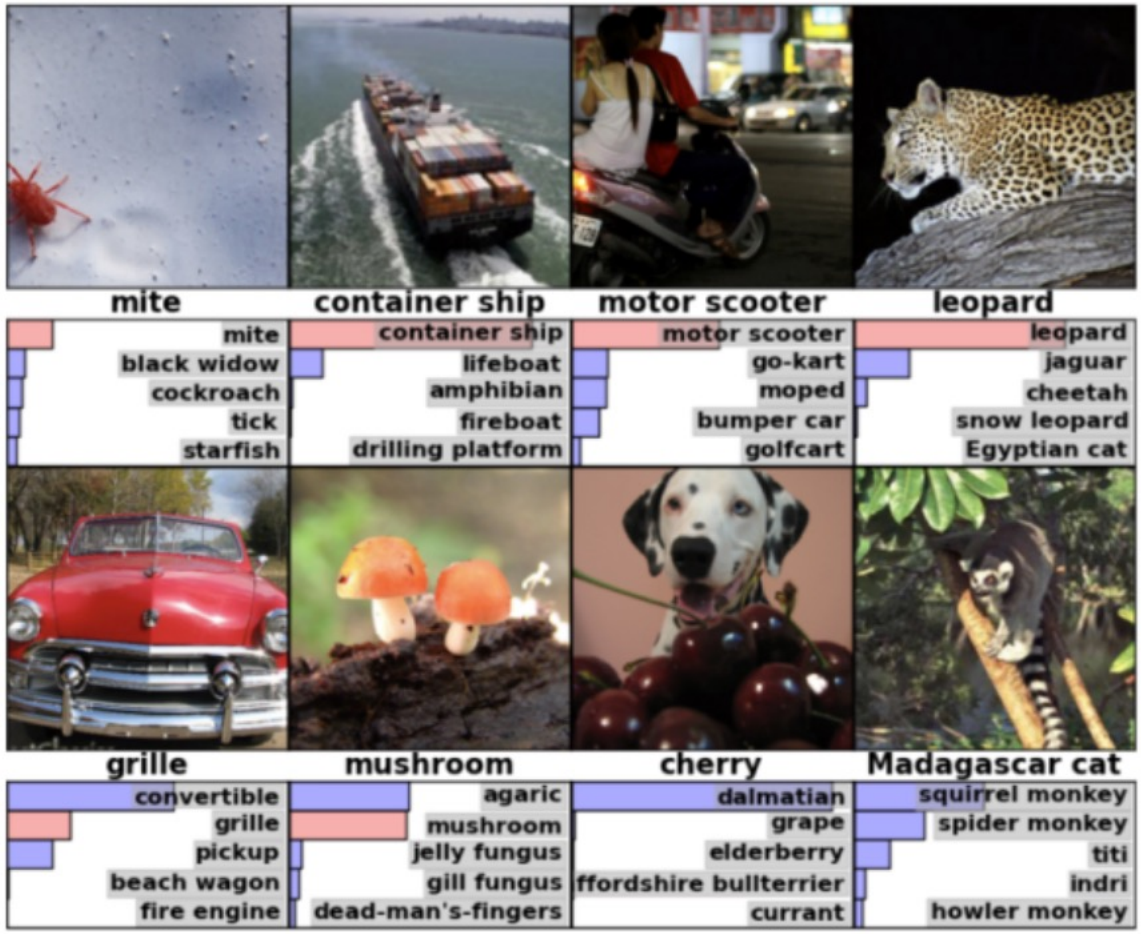
backprop into \alpha
(parameter)

- Use ReLU. Be careful with your learning rates
- Try out Leaky ReLU / Maxout / ELU
- Try out tanh but don't expect much
- Don't use sigmoid



ImageNet Challenge 2012

1000 categories - 1.4 million images

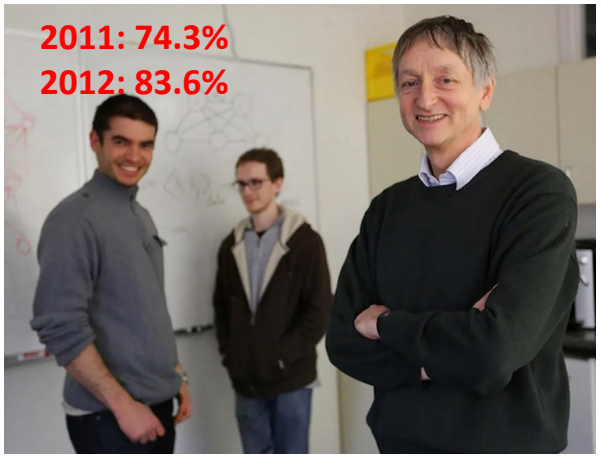


ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky University of Toronto kriz@cs.utoronto.ca
Ilya Sutskever University of Toronto ilya@cs.utoronto.ca
Geoffrey E. Hinton University of Toronto hinton@cs.utoronto.ca

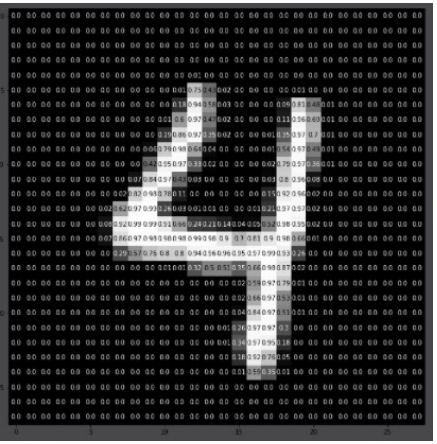
Abstract

We trained a large, deep convolutional neural network to classify the 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest into the 1000 different classes. On the test data, we achieved top-1 and top-5 error rates of 37.5% and 17.0% which is considerably better than the previous state-of-the-art. The neural network, which has 60 million parameters and 650,000 neurons, consists of five convolutional layers, some of which are followed by max-pooling layers, and three fully-connected layers with a final 1000-way softmax. To make training faster, we used non-saturating neurons and a very efficient GPU implementation of the convolution operation. To reduce overfitting in the fully-connected layers we employed a recently-developed regularization method called "dropout" that proved to be very effective. We also entered a variant of this model in the ILSVRC-2012 competition and achieved a winning top-5 test error rate of 15.3%, compared to 26.2% achieved by the second-best entry.

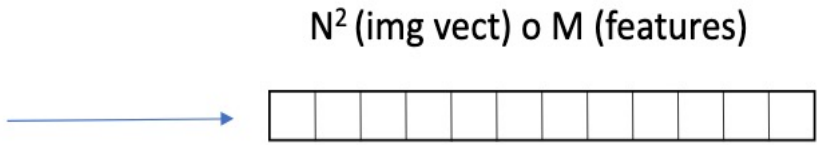


60 million parameters – 650 000 neurons - five convolutional layers

Why convnets? Why not feedforward?

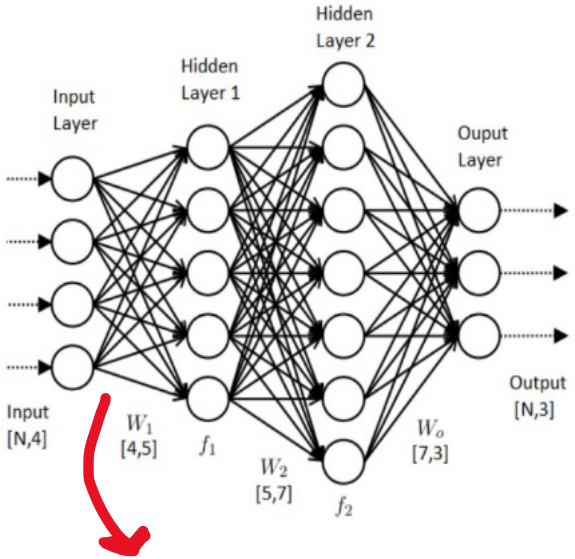


1024x1024



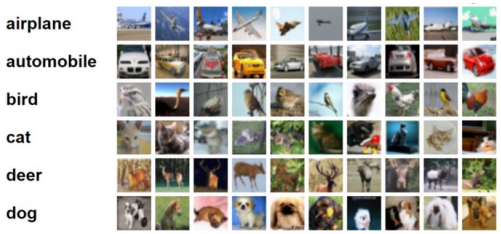
Vectorized image

o(10^6) elements 😬

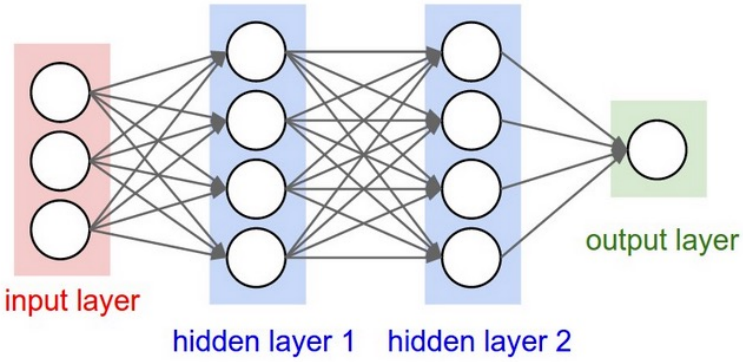


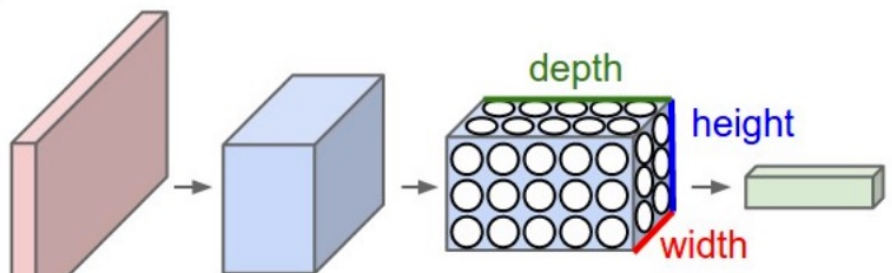
o(10^{6-8}) weights 😱

- FFN do not scale well as the image size increases.
- Poor accuracy (standard FFN on CIFAR-10 got 52% accuracy).



Convolutional Networks





Convnets building blocks

- **Convolutional (CONV)**
- **Fully connected (FC)**
- Activation (ACT or RELU)
- Pooling (POOL)
- Batch normalization (BN)
- Dropout (DO)

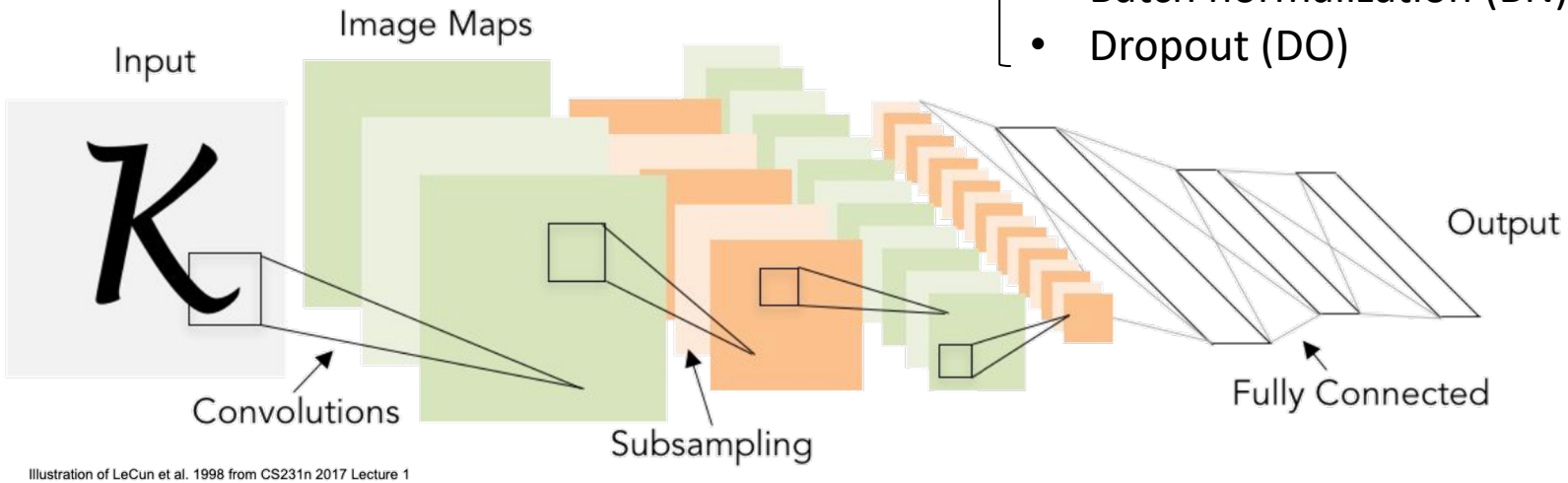


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

Luis G. Moyano - Fundamentos de ML Instituto Bakseiro

Convolutional layer

Extract information by applying a sliding filter/kernel

Image

$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$	0	0
$0_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$	1	0
$0_{\times 1}$	$0_{\times 0}$	$1_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

Convolved feature

4		

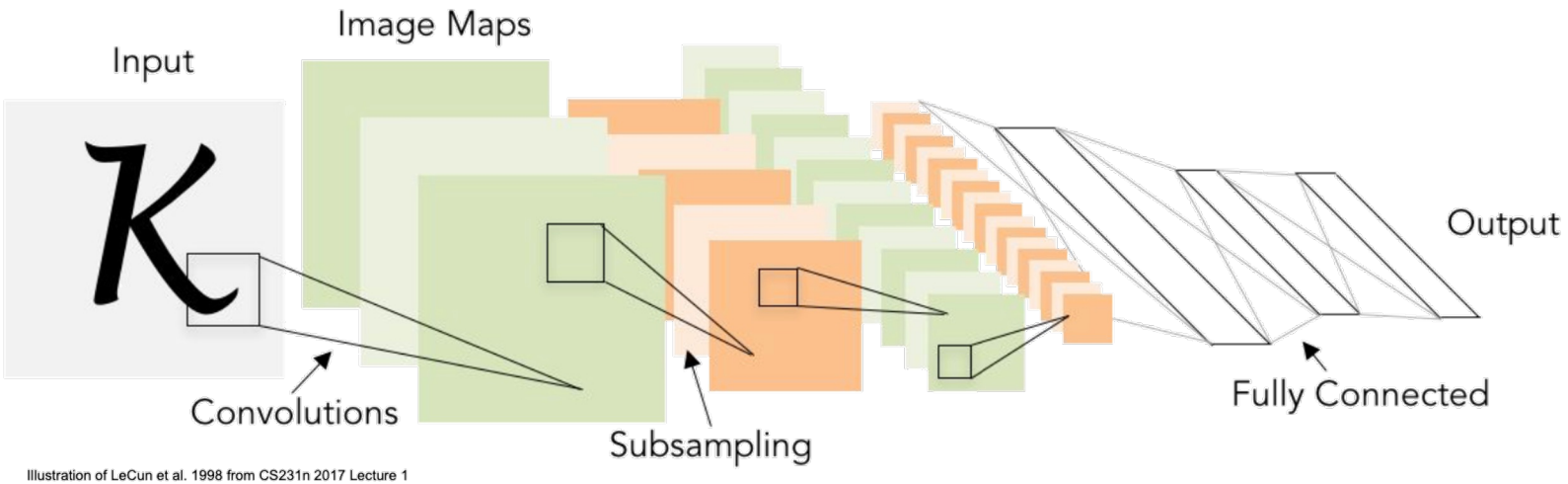
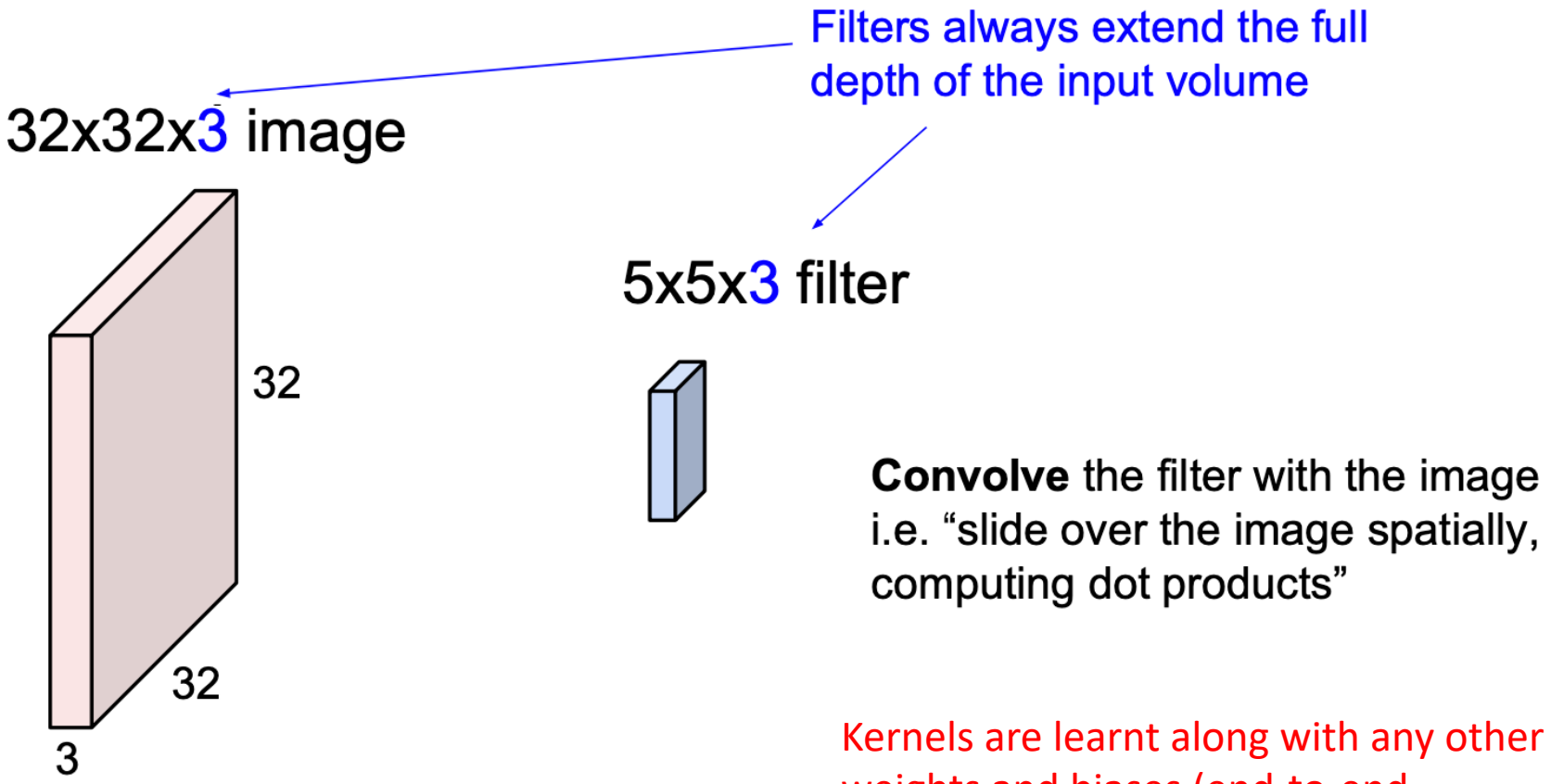
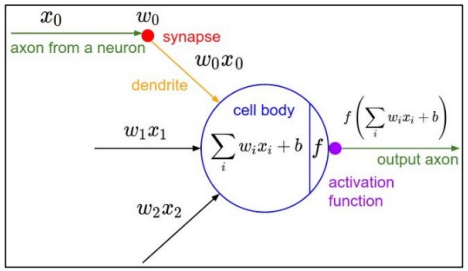


Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

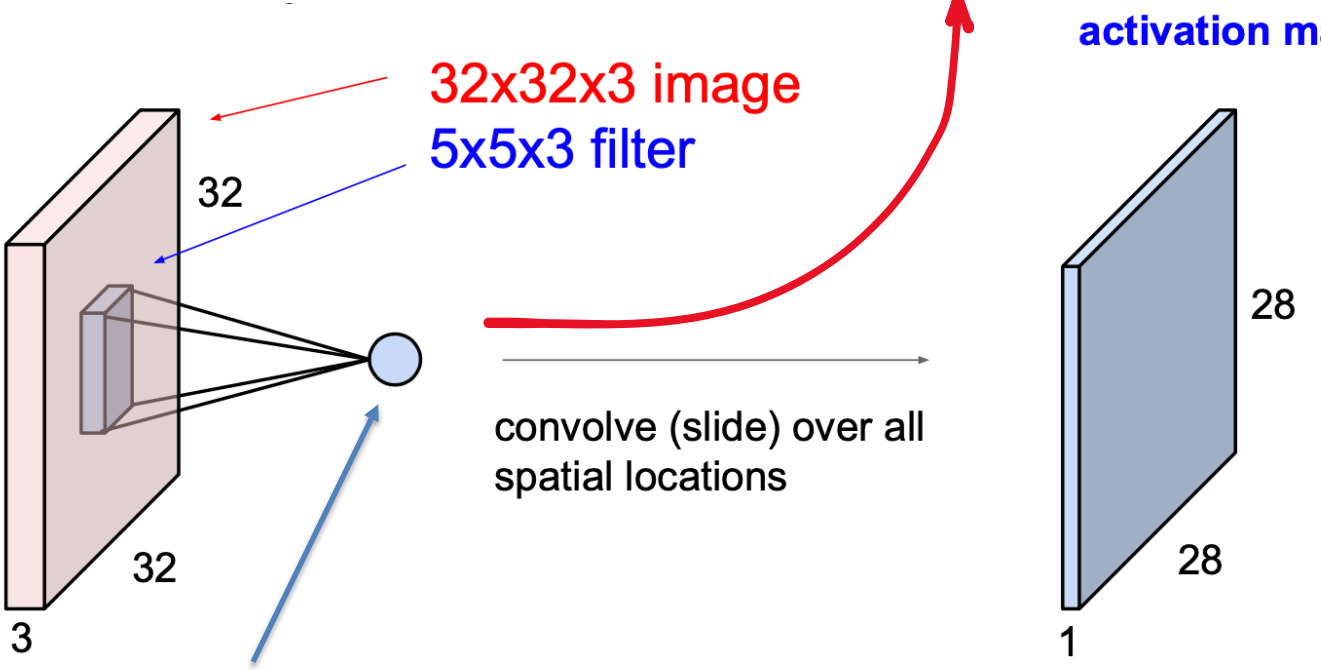
Convolution layer



Convolution layer



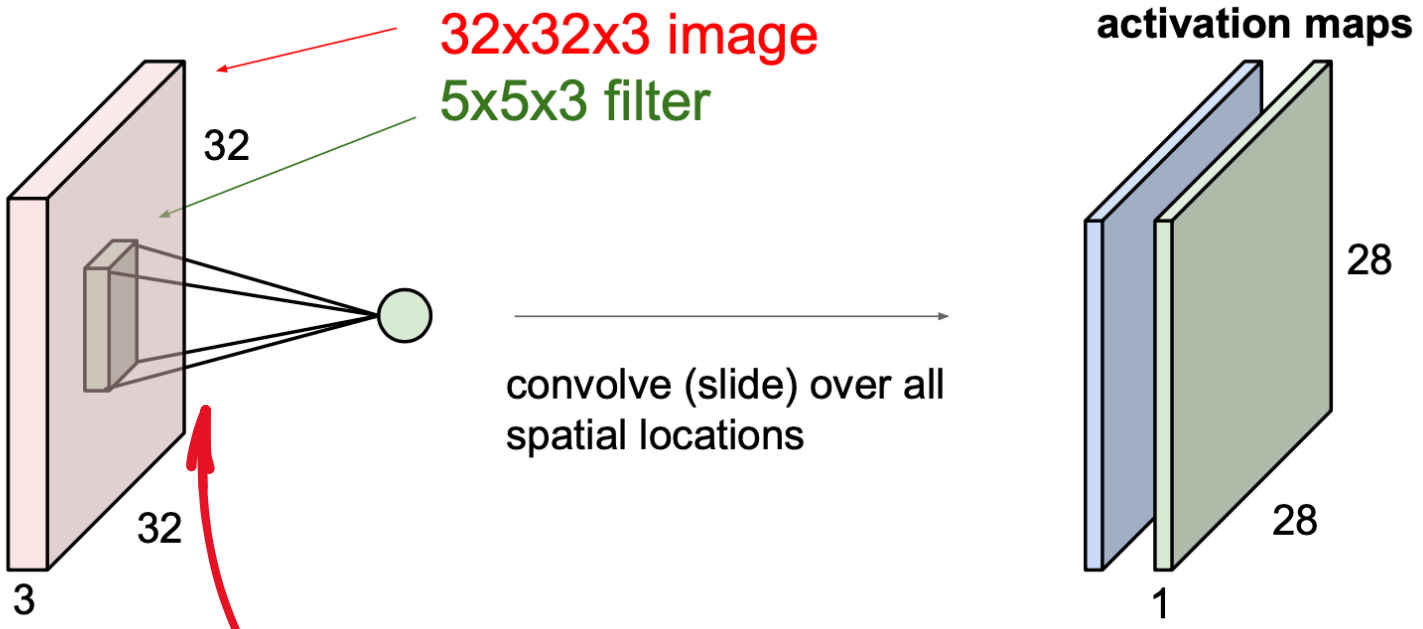
It's just a neuron with local connectivity...



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

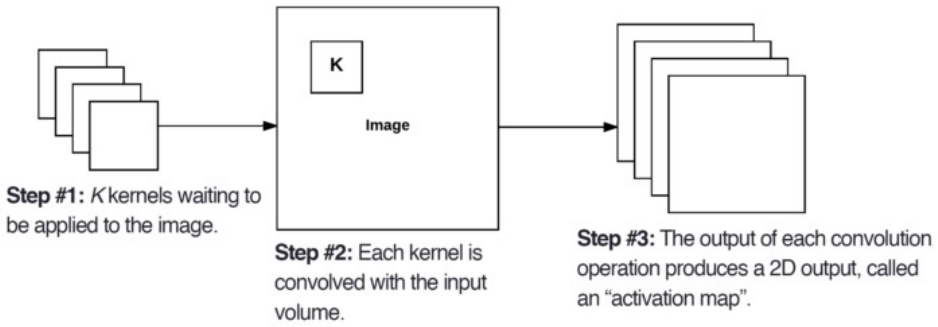


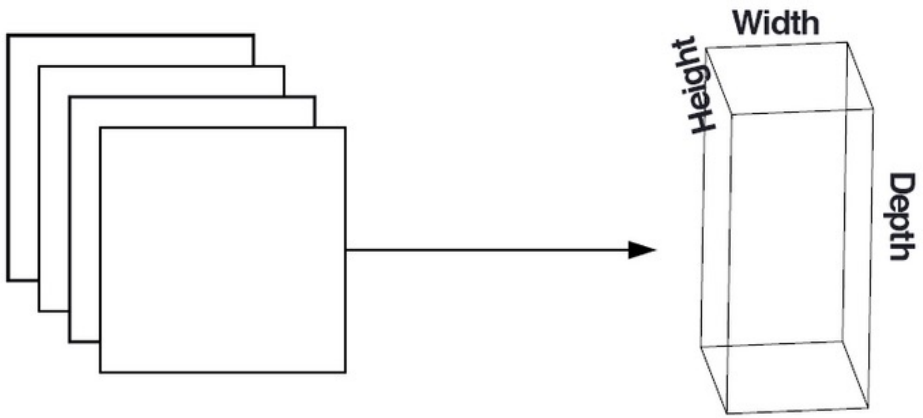
An activation map is a 28x28 sheet of neuron outputs:

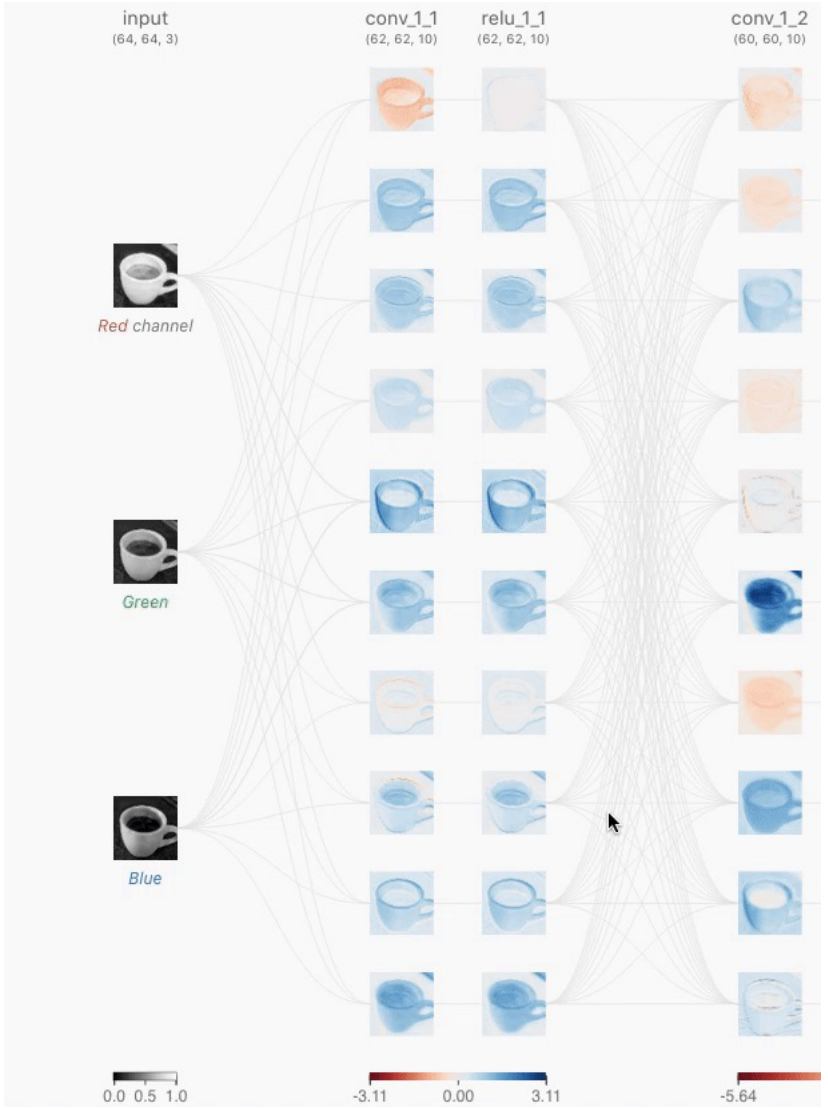
1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" -> "5x5 receptive field for each neuron"

Activation maps,
aka feature maps



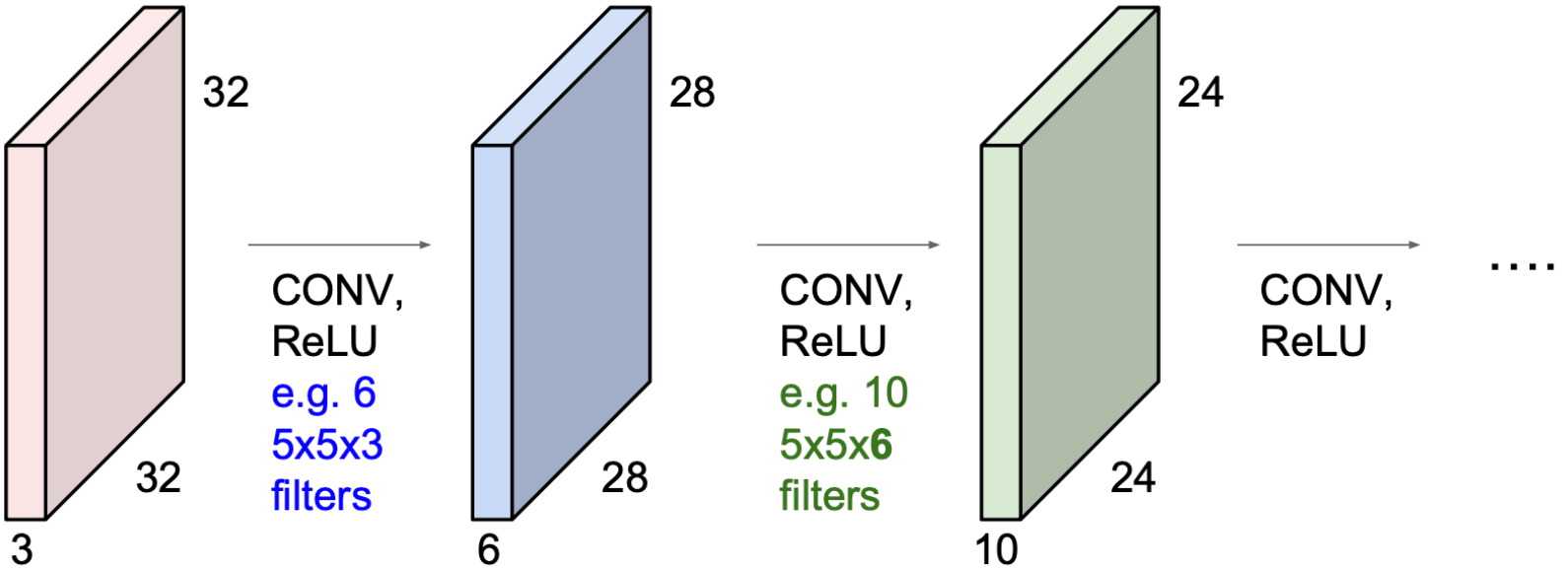


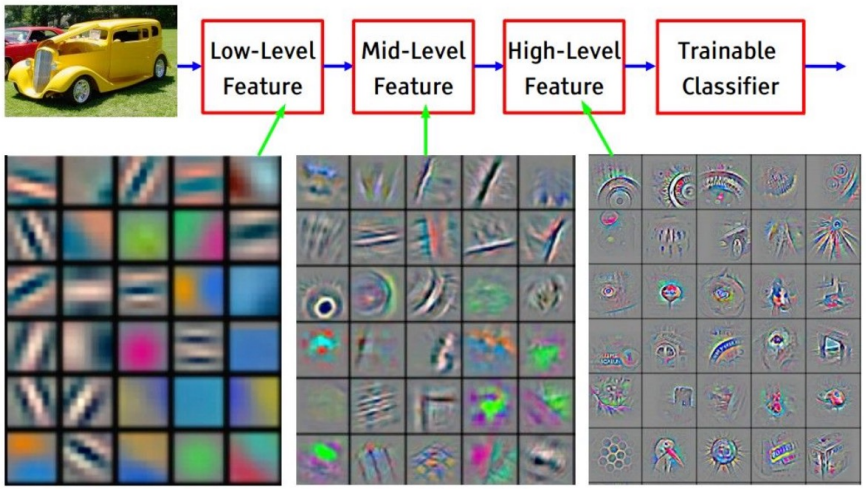


https://poloclub.github.io/cnn-explainer/

Convnet

ConvNet is a sequence of Convolutional Layers, interspersed with activation functions

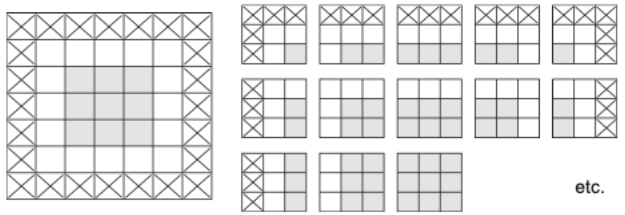


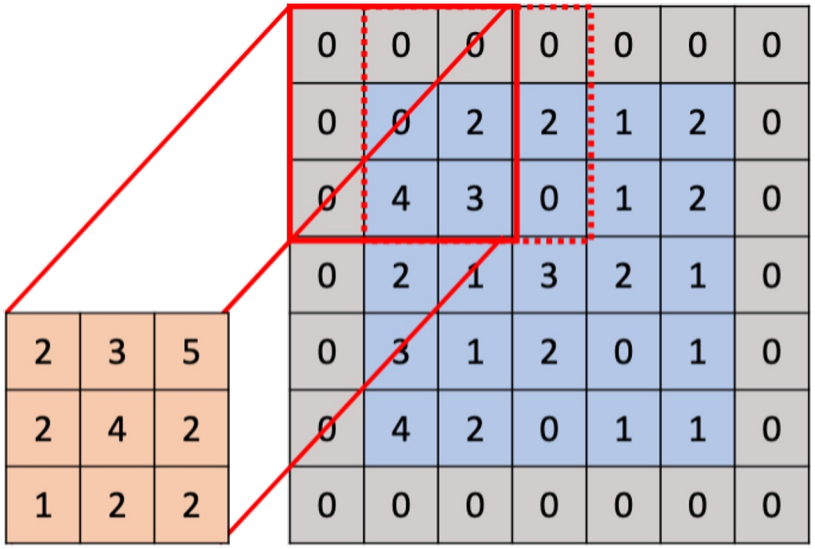


Zieler & Fergus 2013

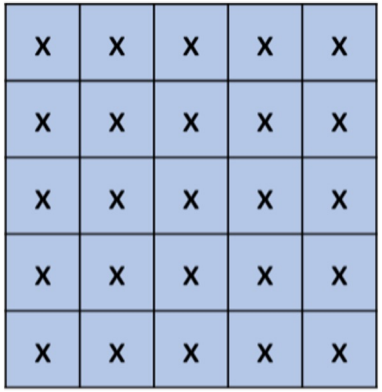

@txolerfx

Hyperparameters





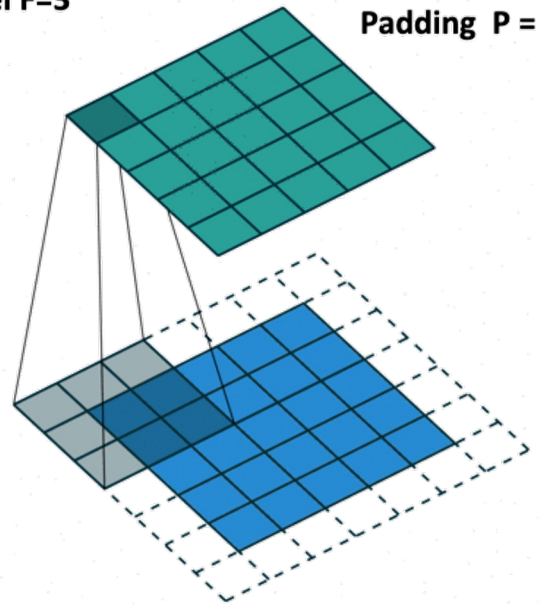
Conv

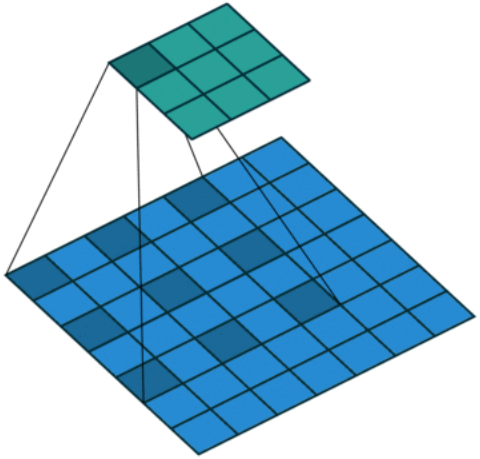


Kernel F=3

Padding P = 1

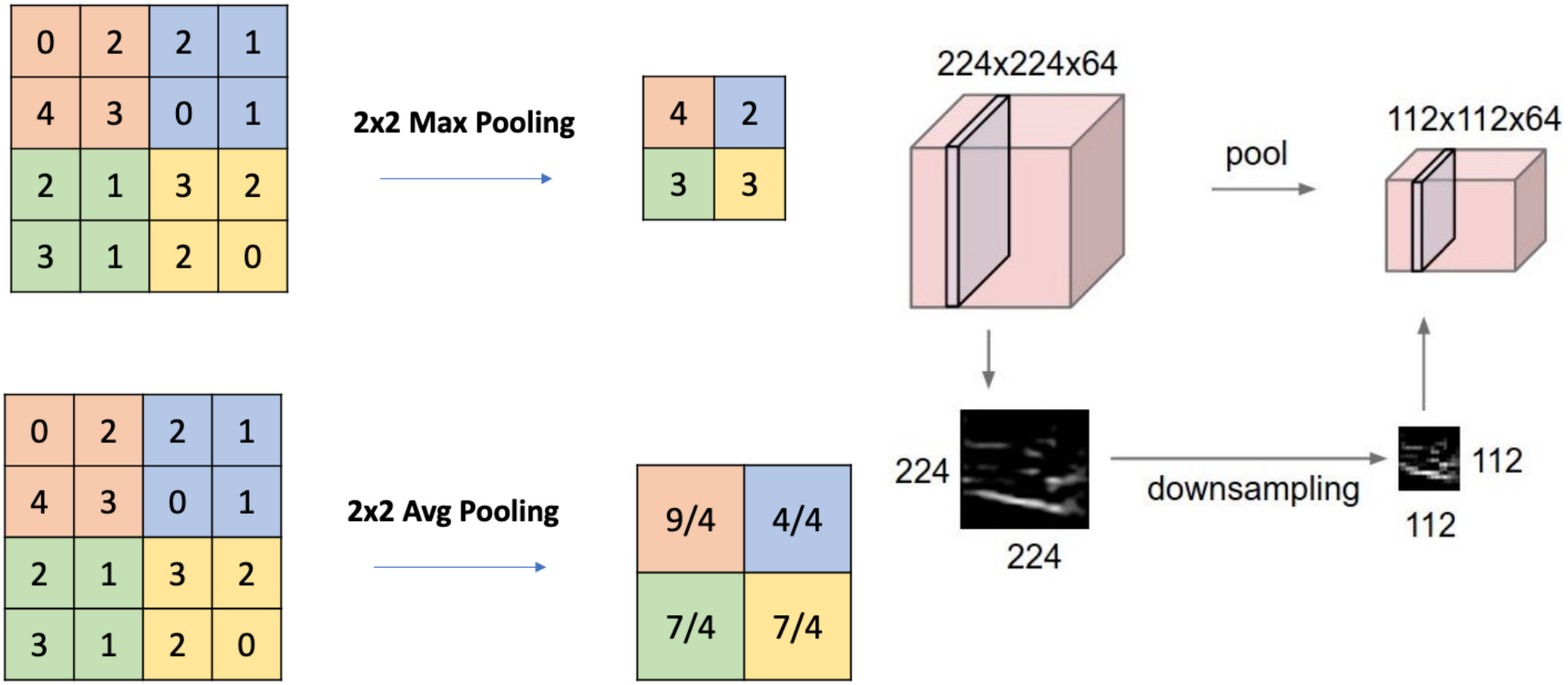
$$O = \frac{(W - F + 2P)}{S} + 1$$



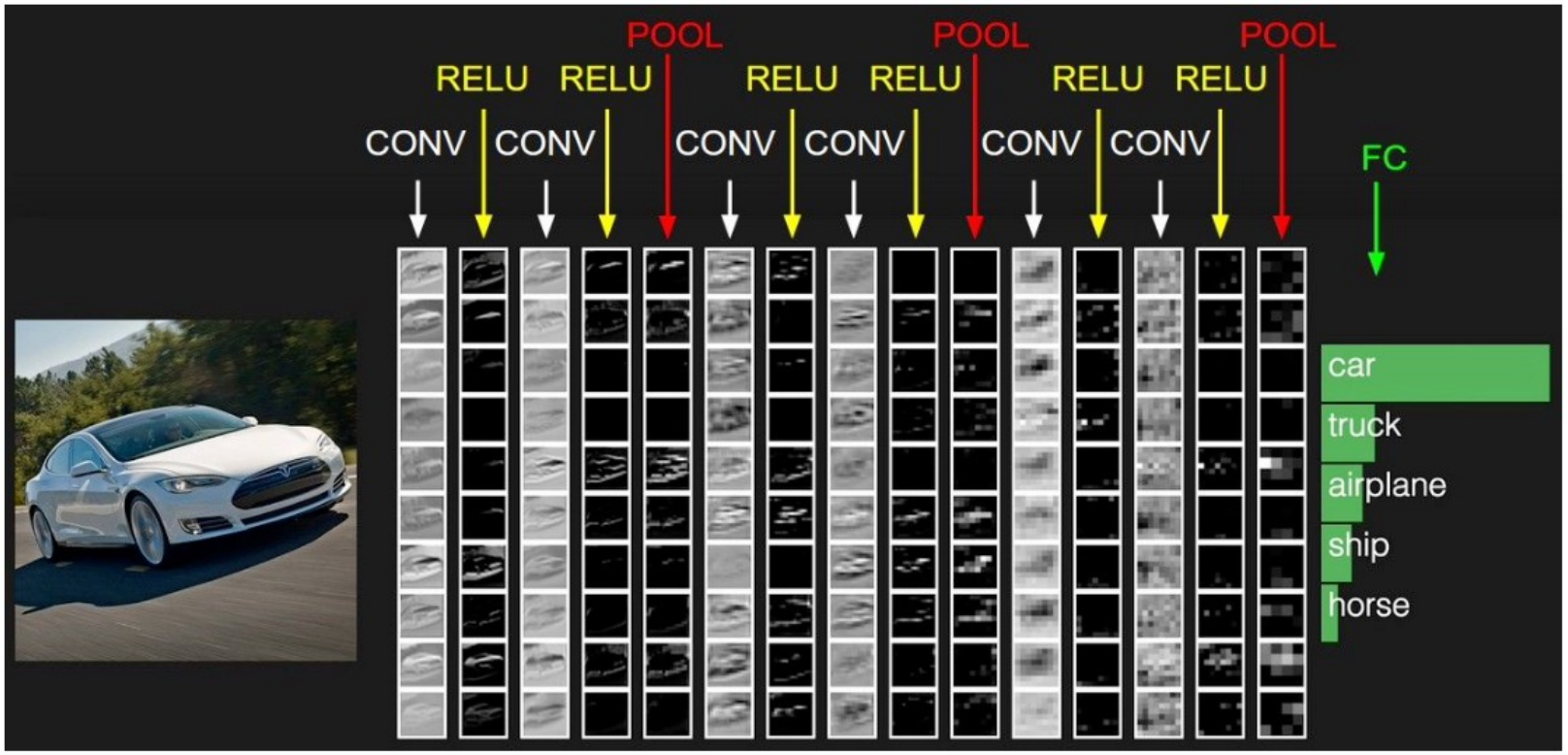


Pooling layer (POOL)

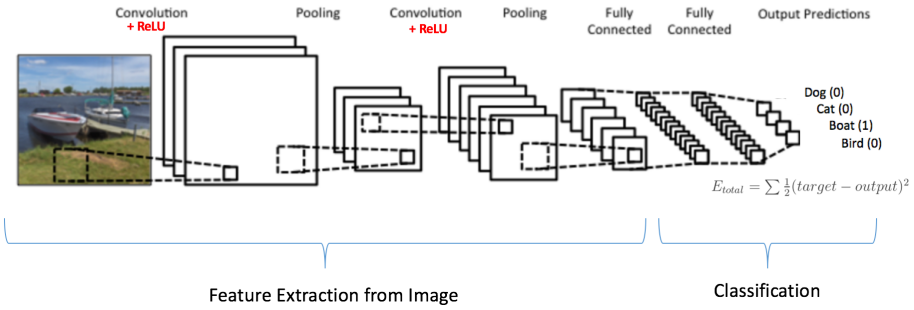
Subsampling, meant to reduce and condense information



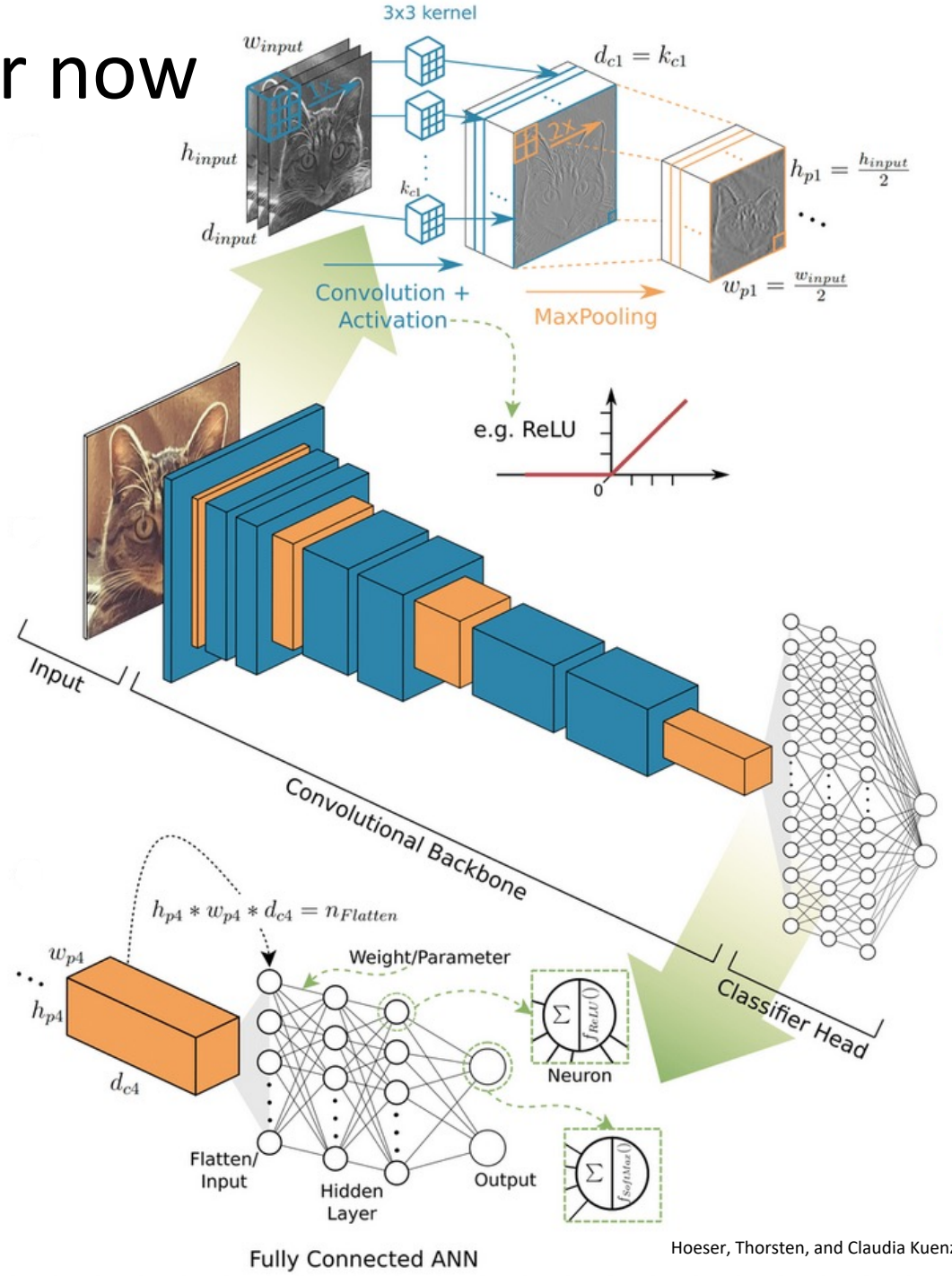
Fully connected layer (FC)



$$\text{softmax}(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{c=1}^{C} e^{z_c}}$$



All together now

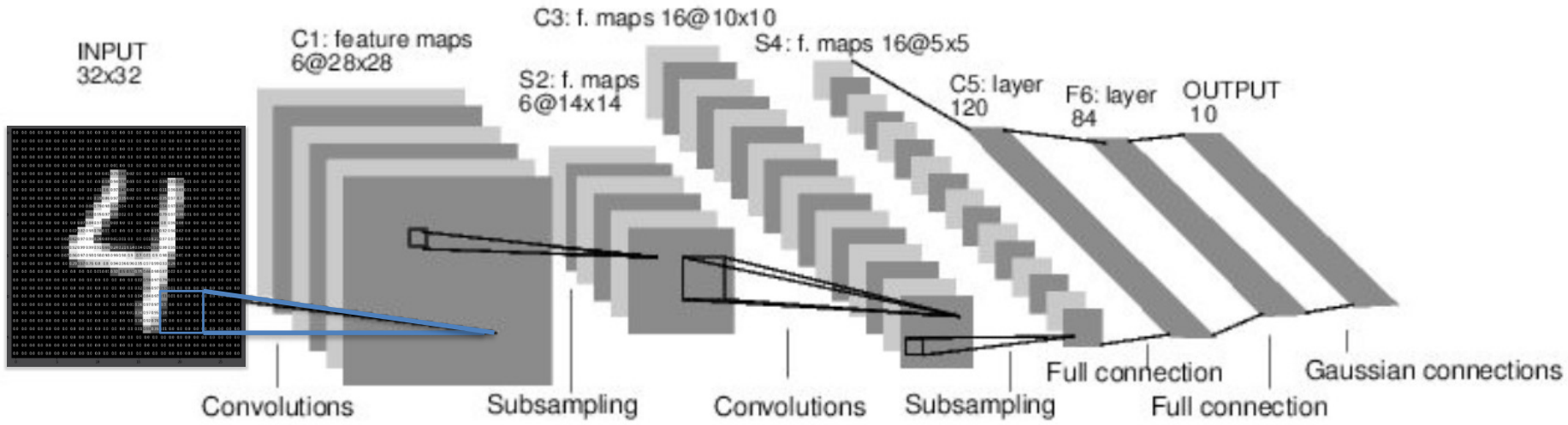


Hoeser, Thorsten, and Claudia Kuenzer. *Remote Sensing* 12.10 (2020): 1667.

LeNet (LeCun 1989)

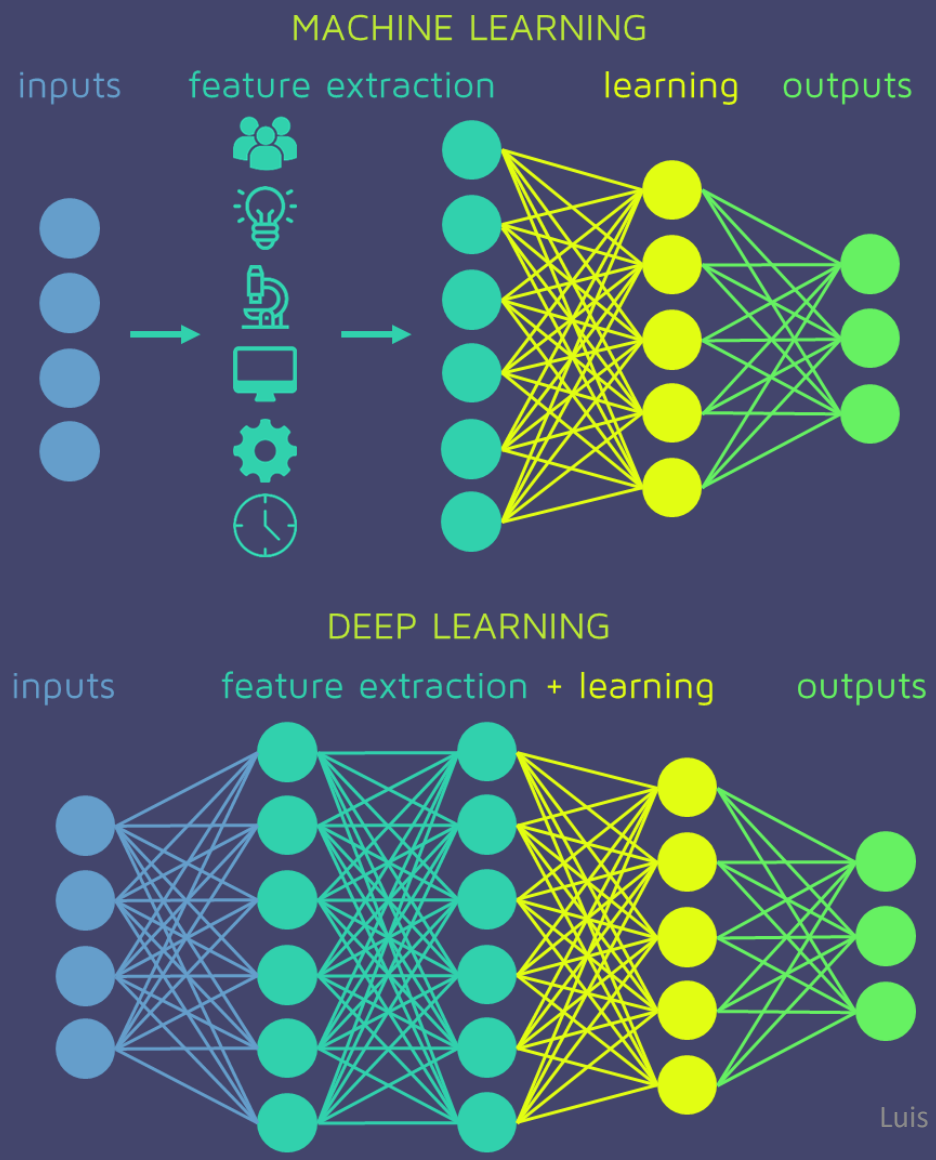
60000 parameters, 7 layers

$$O = \frac{(W - F + 2P)}{S} + 1$$

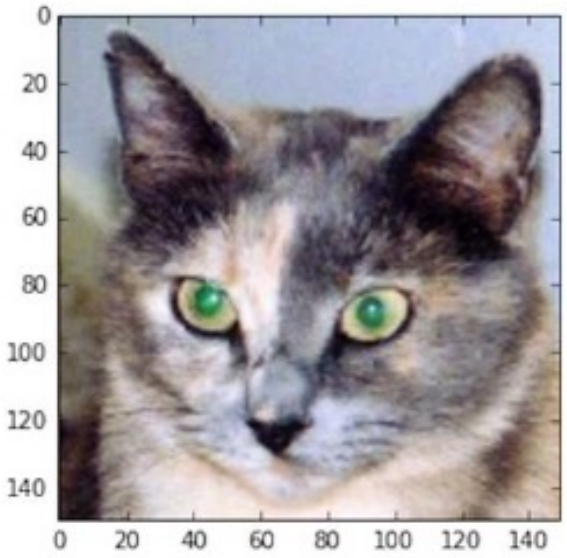


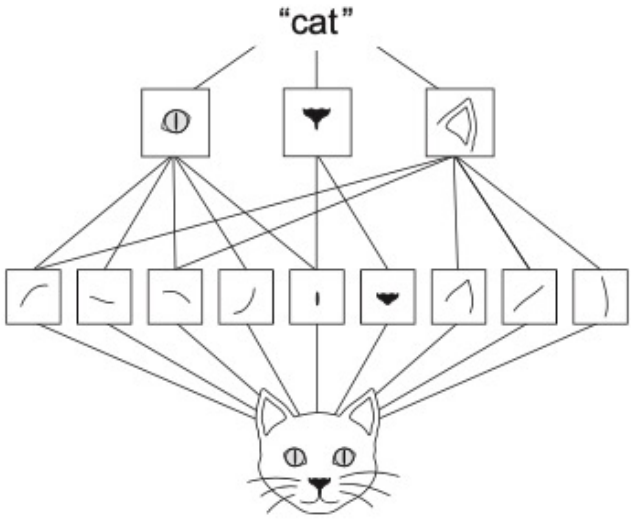
Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-CONV-FC]

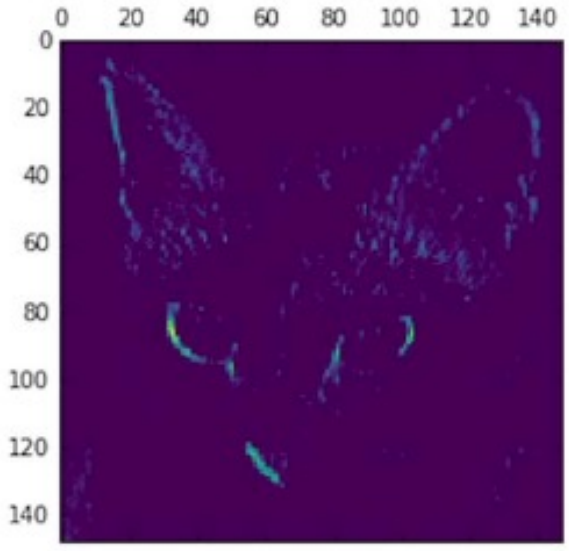
End to end learning

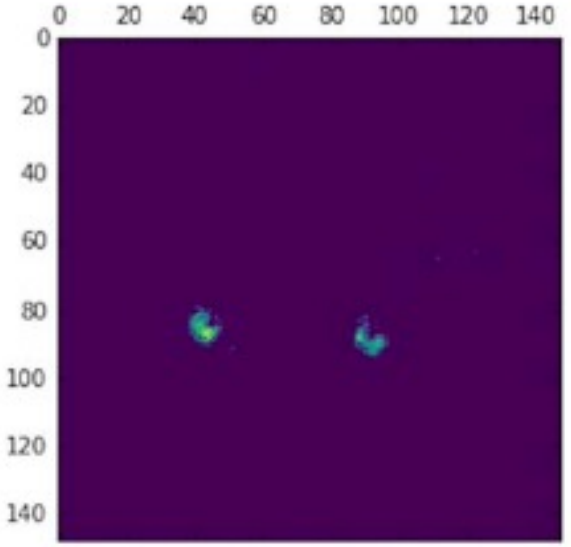


Automatic feature extraction

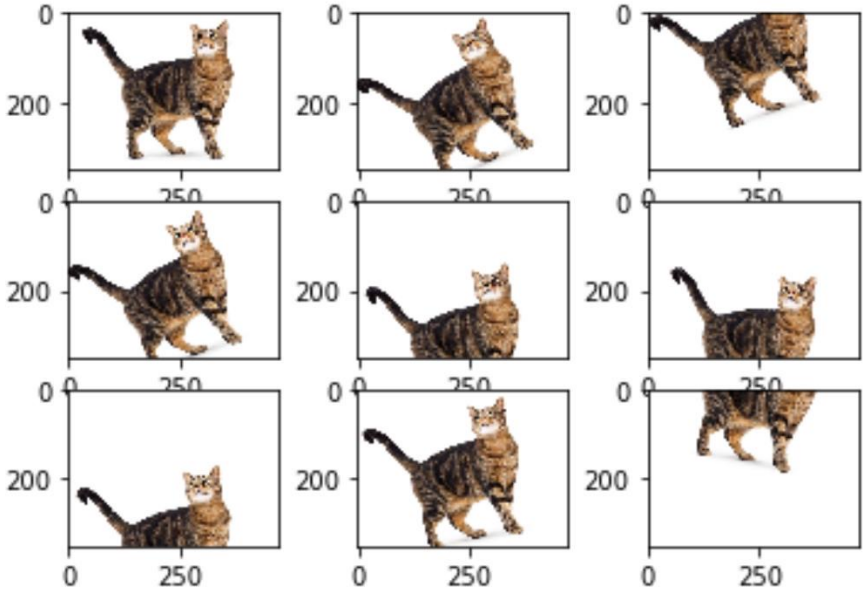


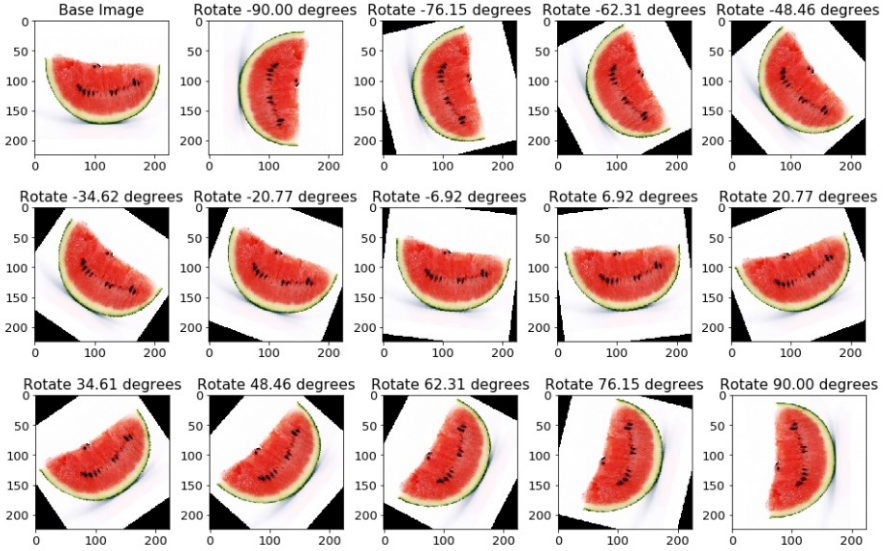


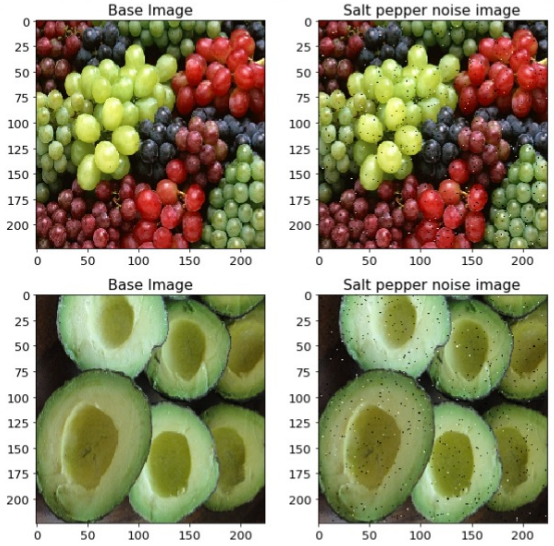




Data augmentation for training



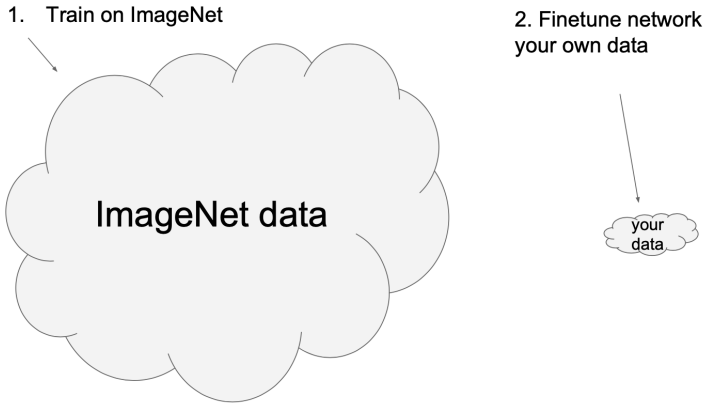




ConvNets are good for parsing images

- Naturally adapted to regular structure in images (through convolution operation)
- Invariant in regard to traslations
- Weight sharing, thus low memory requirement
- Efficient at test time
- Good generalization when traind with enough data
- Used also on sound data

Can I use DL in a small dataset?



Keras Applications

Keras Applications are deep learning models that are made available alongside pre-trained weights. These models can be used for prediction, feature extraction, and fine-tuning.

Weights are downloaded automatically when instantiating a model. They are stored at `~/.keras/models/`.

Upon instantiation, the models will be built according to the image data format set in your Keras configuration file at `~/.keras/keras.json`. For instance, if you have set `image_data_format=channels_last`, then any model loaded from this repository will get built according to the TensorFlow data format convention, "Height-Width-Depth".

Available models

Model	Size (MB)	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth	Time (ms) per inference step (CPU)	Time (ms) per inference step (GPU)
Xception	88	79.0%	94.5%	22.9M	81	109.4	8.1
VGG16	528	71.3%	90.1%	138.4M	16	69.5	4.2
VGG19	549	71.3%	90.0%	143.7M	19	84.8	4.4

Transfer Learning with CNNs

1. Train on ImageNet

2. If small dataset: fix all weights (treat CNN as fixed feature extractor), retrain only the classifier

i.e. swap the Softmax layer at the end

3. If you have medium sized dataset, **"finetune"** instead: use the old weights as initialization, train the full network or only some of the higher layers

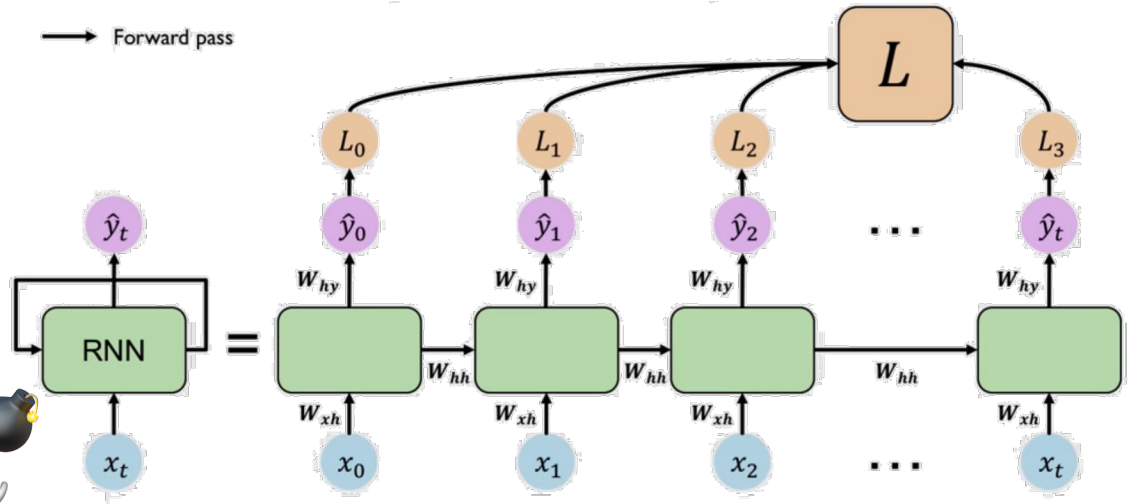
retrain bigger portion of the network, or even all of it.

Recurrent Neural Networks - RNNs

Vanilla RNNs

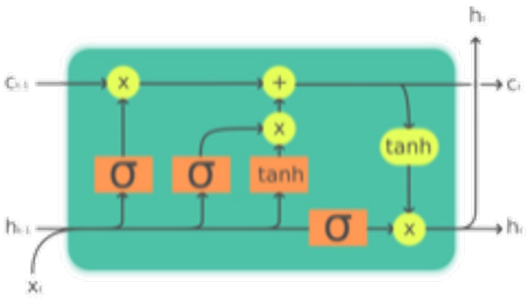
```
tf.keras.layers.SimpleRNN(rnn_units)
```

- For information flux or data sequences. E.g.:
  - voice recognition
  - handwritten text
  - video
  - data traffic
- But, exploding gradients, 💣
- and vanishing gradients 👻

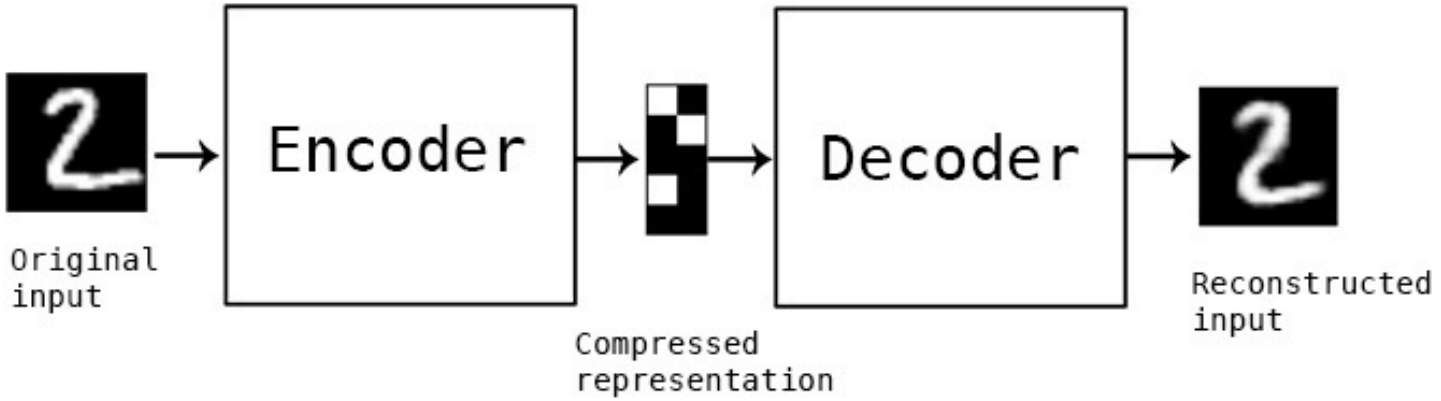


LSTM – Long-Short term memory

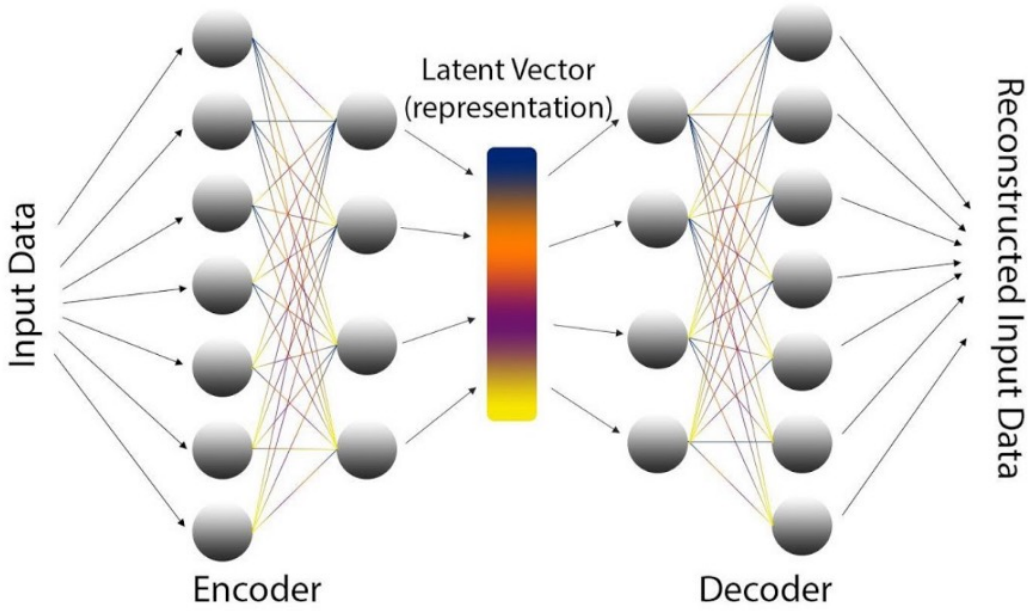
- Add interal structure through filters
- Proposed for solving vanishing gradients




```
tf.keras.layers.LSTM(num_units)
```



AUTOENCODERS



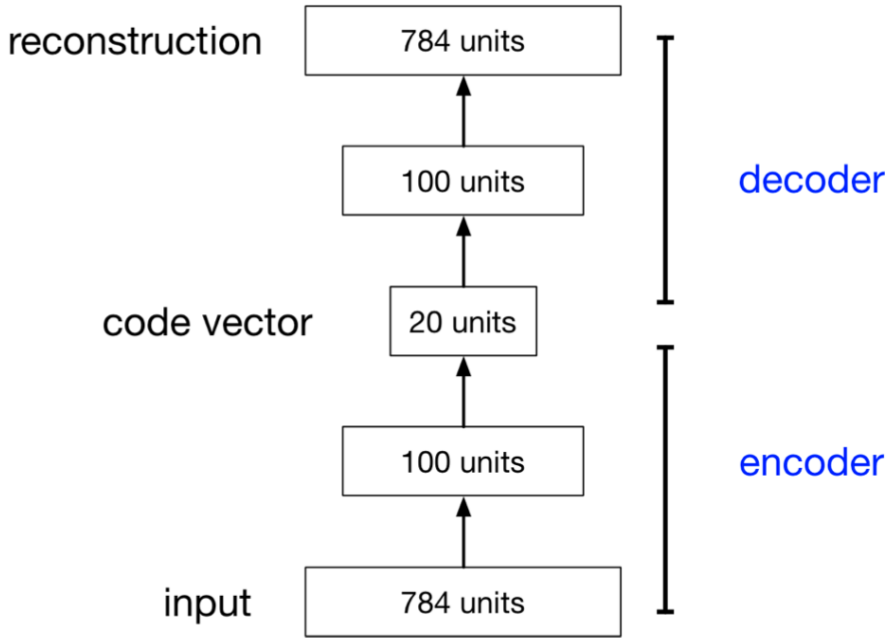
Autoencoders

(LeCun, 1987; Bourlard and Kamp, 1988; Hinton and Zemel, 1994)

- An autoencoder is a feed-forward neural net whose job it is to take an input x and predict x.
- Usually a *bottleneck* layer is added whose dimension is much smaller than the input.
- Unsupervised/Self-supervised

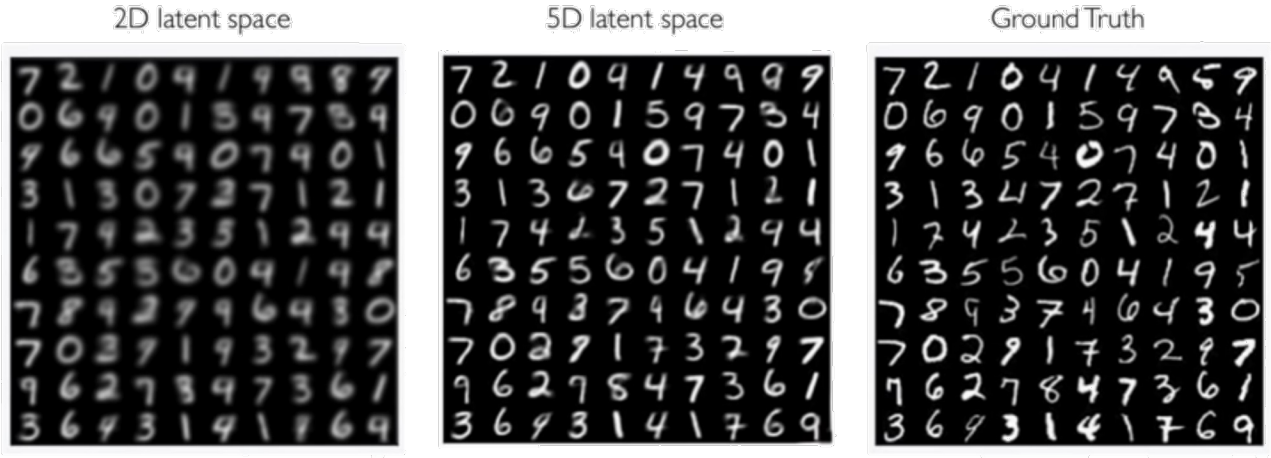
Why autoencoders?

- Map high-dimensional data to two dimensions for visualization
- Learn abstract features in an **unsupervised** way so you can apply them to a supervised task
- Unlabeled data can be much more available than labeled data



Hinton et al.,, Science 2006

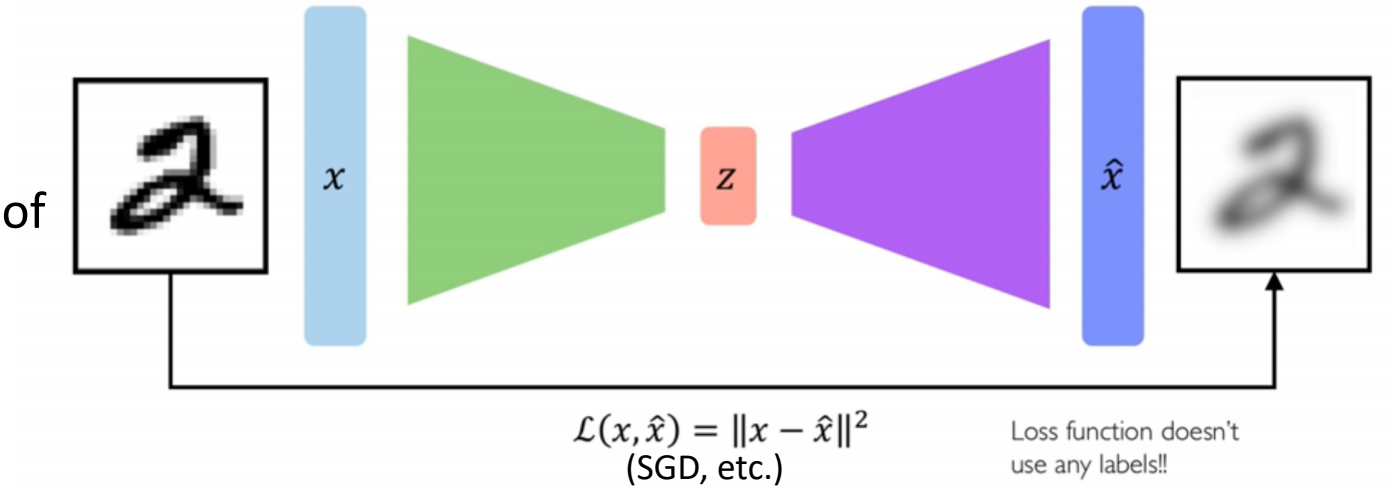
Autoencoders



Applications

- Dimensionality reduction (Hinton et al, 2006)
- May be used for other tasks, such as classification
- Feature learning
- Information retrieval

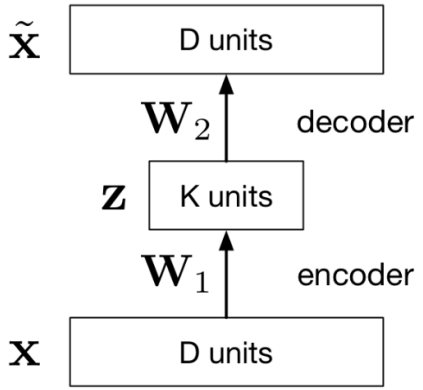
- Now at the forefront of **generative modeling**

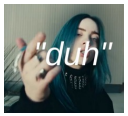


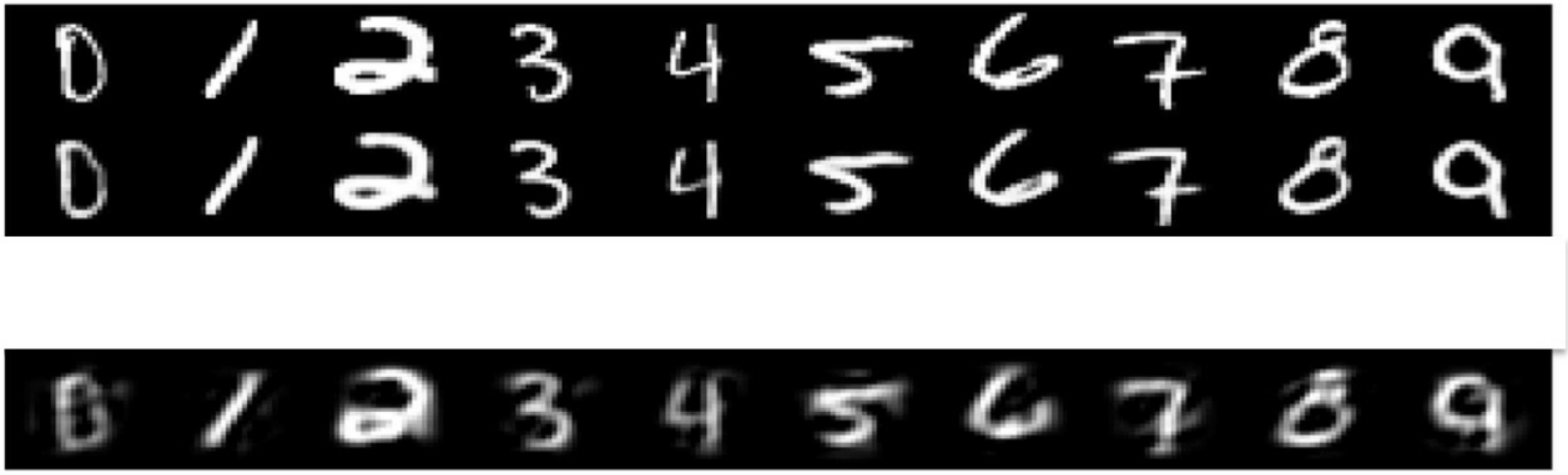
Autoencoders

(LeCun, 1987; Bourlard and Kamp, 1988; Hinton and Zemel, 1994)

- The simplest kind of autoencoder has one hidden layer, linear activations, and mean squared error loss.
- In this case, the optimal weights for a linear autoencoder are just the principal components
  - i.e. equivalent to PCA
- Nonlinear autoencoders can learn more powerful codes for a given dimensionality.





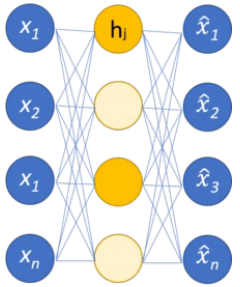


Regularized autoencoders

- Autoencoders need to prevent learning the Identity
  - Undercomplete/Overcomplete
- **Denoising autoencoders**
  - Learn to reconstruct corrupt data
  - Self-supervised

- **Sparse autoencoders**

$$L(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \Omega(\boldsymbol{h})$$ where $\Omega(\boldsymbol{h})$ KL, L1, L2



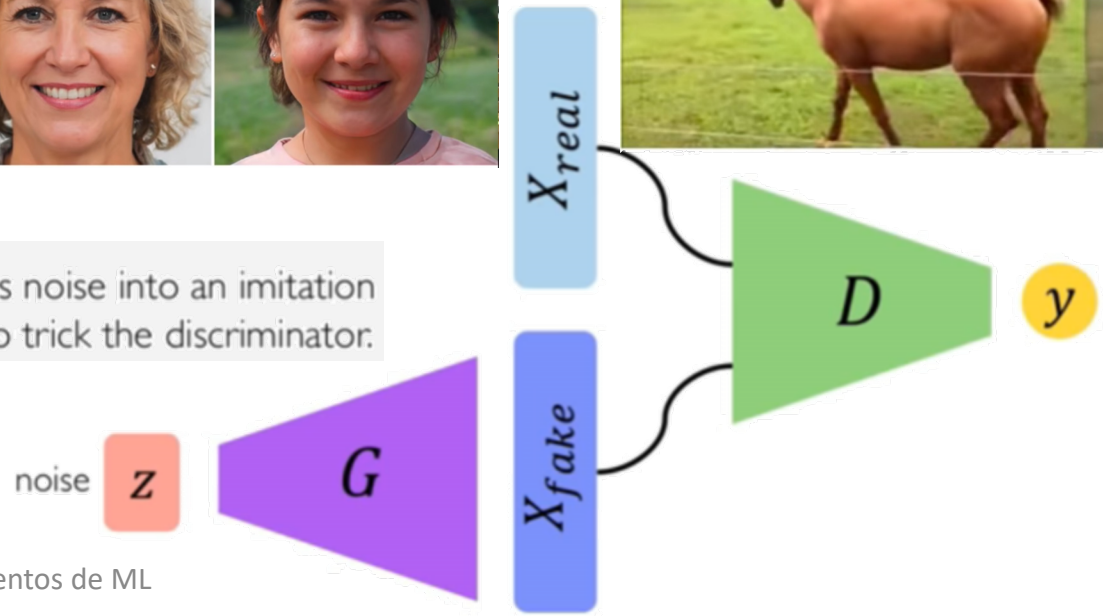
- **Variational autoencoders**
  - Generative models: they try to simulate how the data is generated in order to understand the underlying causal relations.

Generative Adversarial Networks (GANs)

Goodfellow, 2014

- Two NN competing with each other (generator and discriminator)
- Generator produces examples based on a reduced dataset, sends example as input to the discriminator, with the objective to augment its error
- The discriminator classifies the data as real or fake with the objective of decrease its error. The output informs the generator.
- Both use backpropagation to update weights
- Not easy to train

The **generator** turns noise into an imitation of the data to try to trick the discriminator.



ML Fundamentals – Lecture 10

- **Backpropagation**
- **CNNs**
- RNNs
- Autoencoders
- GANs

Next:
NNs Best practices