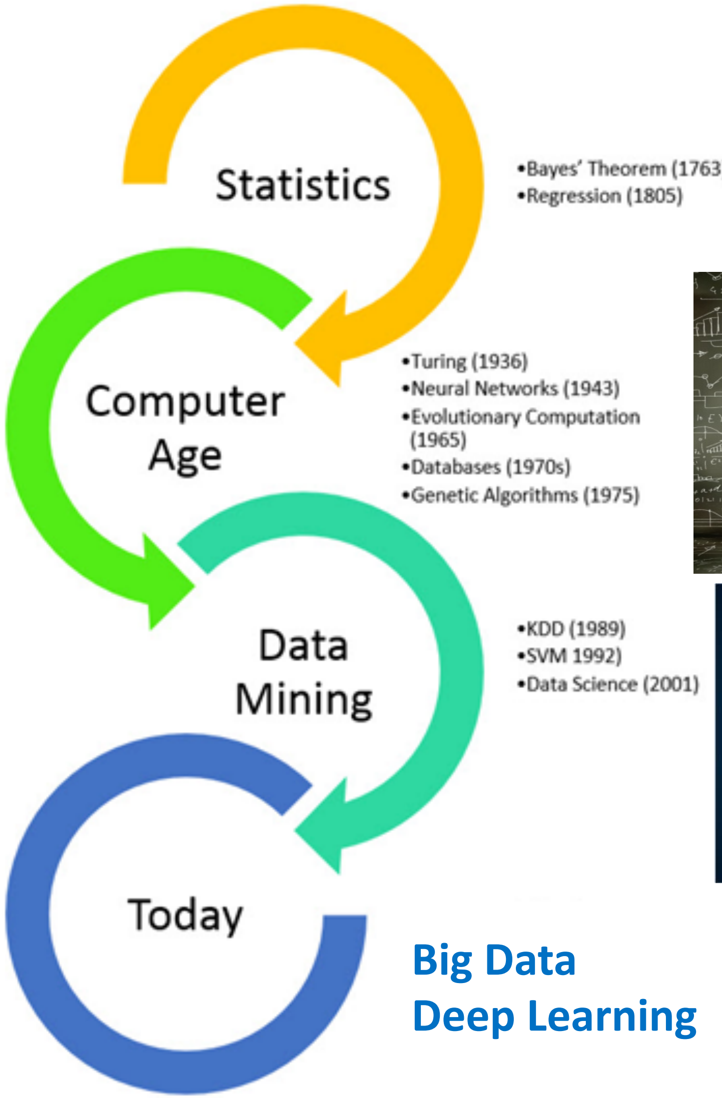
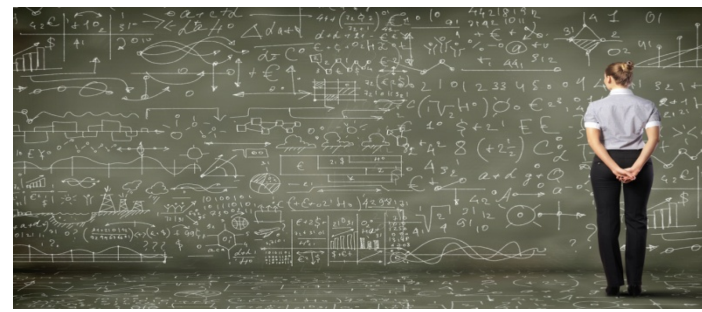
Ciencias de Datos y Deep Learning: Neuronas artificiales para aprender

Francisco Herrera

Ciencia de Datos y Deep Learning: Neuronas artificiales para aprender

History of Data Science

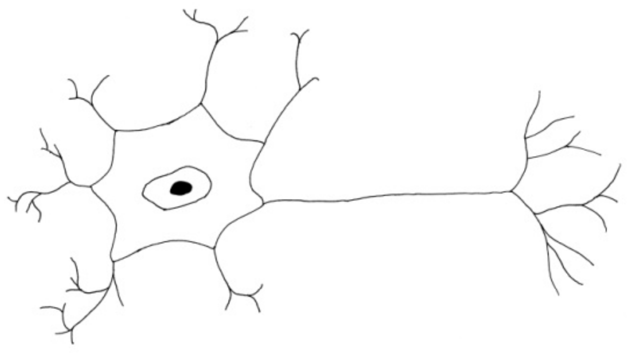


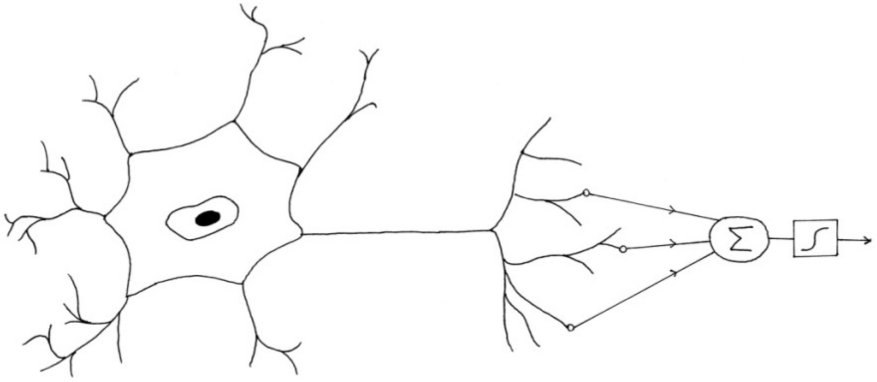


Big Data
Deep Learning

Universidad de Granada

Redes Neuronales

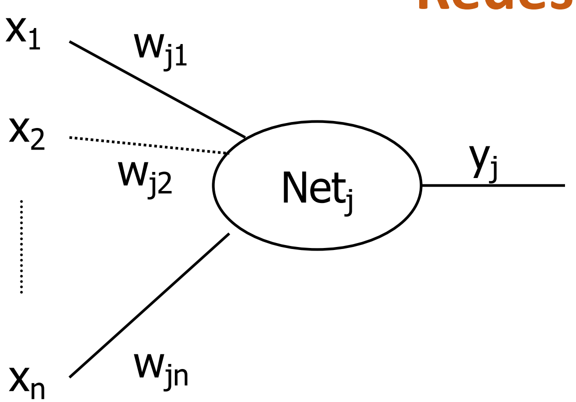




En 1943 Warren McCulloch y Walter Pitts presentaron su modelo de neurona artificial, y describieron los primeros fundamentos de lo que se llamaría posteriormente redes neuronales.

W. McCulloch and W. Pitts (1943). A Logical Calculus of ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics 5:115-133.

Redes Neuronales



$$Net_j = w_{j1} \cdot x_1 + w_{j2} \cdot x_2 + \ldots + w_{jn} \cdot x_n = \sum_{i=1}^{n} w_{ji} \cdot x_i$$

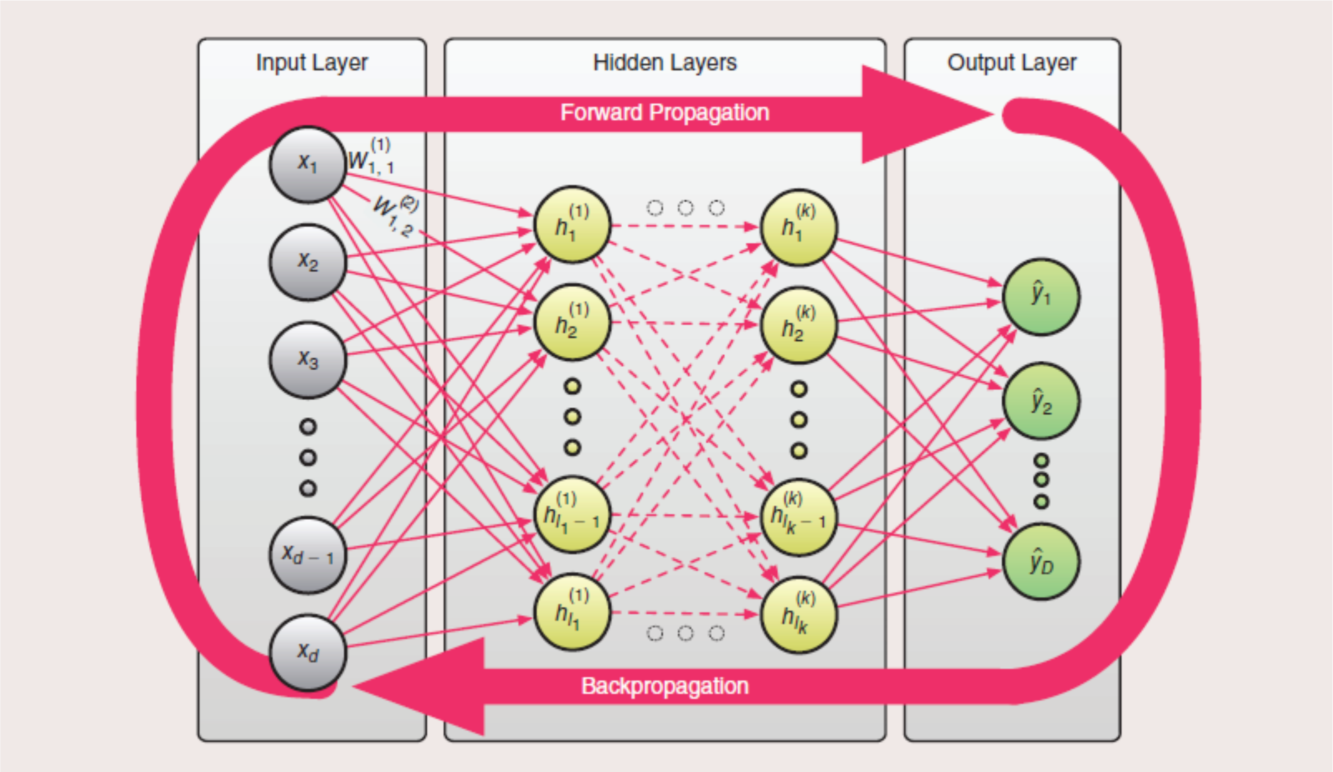
$$y_j = f(Net_j - \theta_j) = f\left(\sum w_{ji} \cdot x_i - \theta \right)$$

Una red neuronal se propone como un sistema inteligente que imita al sistema nervioso y a la estructura del cerebro, pero muy diferente en términos de su estructura y escala.

Al igual que las neuronas biológicas, las neuronas artificiales se interconectan para formar redes de neuronas artificiales.

Cada neurona artificial utiliza una **función procesamiento** que agrega la información de conexiones de entrada con otras neuronales artificiales, una **función de activación y una función de transferencia** para dar una salida de la neurona en sus conexiones de salida.

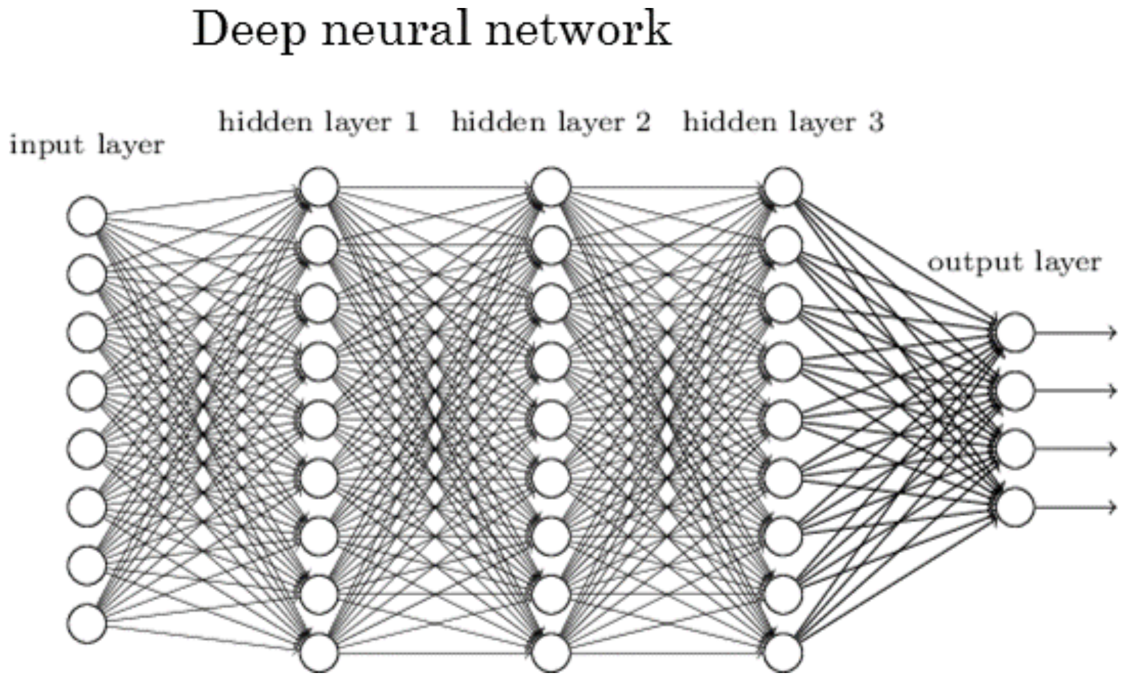
Red Neuronal Clásica: Backpropagation



Credits: [The Evolution of Neural Learning Systems: A Novel Architecture Combining the Strengths of NTs, CNNs, and ELMs. N Martinel](), [C Micheloni]()… - IEEE SMC Magazine, 2015 - ieeexplore.ieee.org

Deep Architecture (Redes Neuronales con muchas capas)

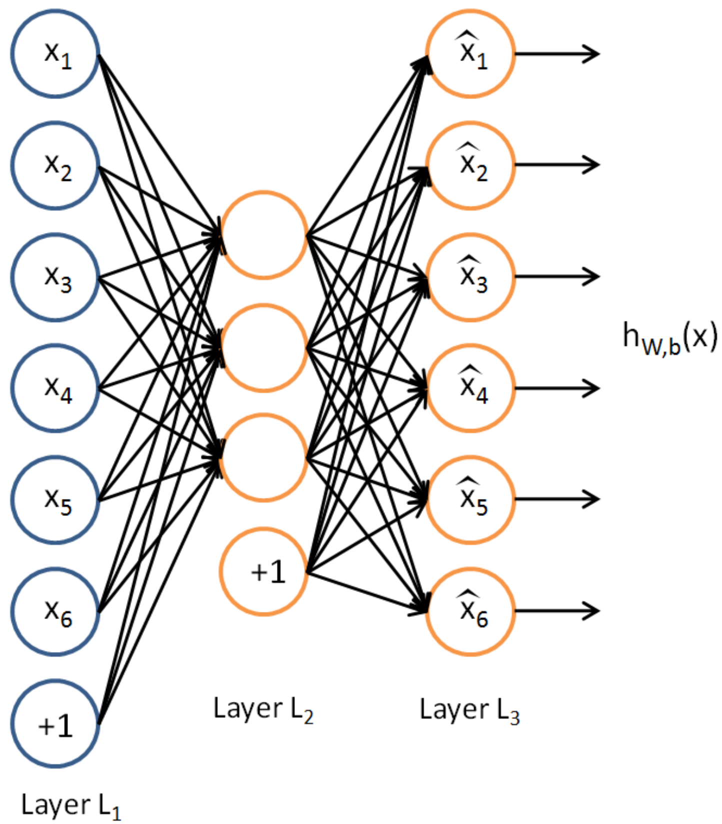
- Múltiples capas ocultas
- Aprendizaje jerárquico
  - Características cada vez más complejas
- Muy buen comportamiento en mútiples dominios: Vision, Audio, …



Unsupervised Deep Architecture : Autoencoder

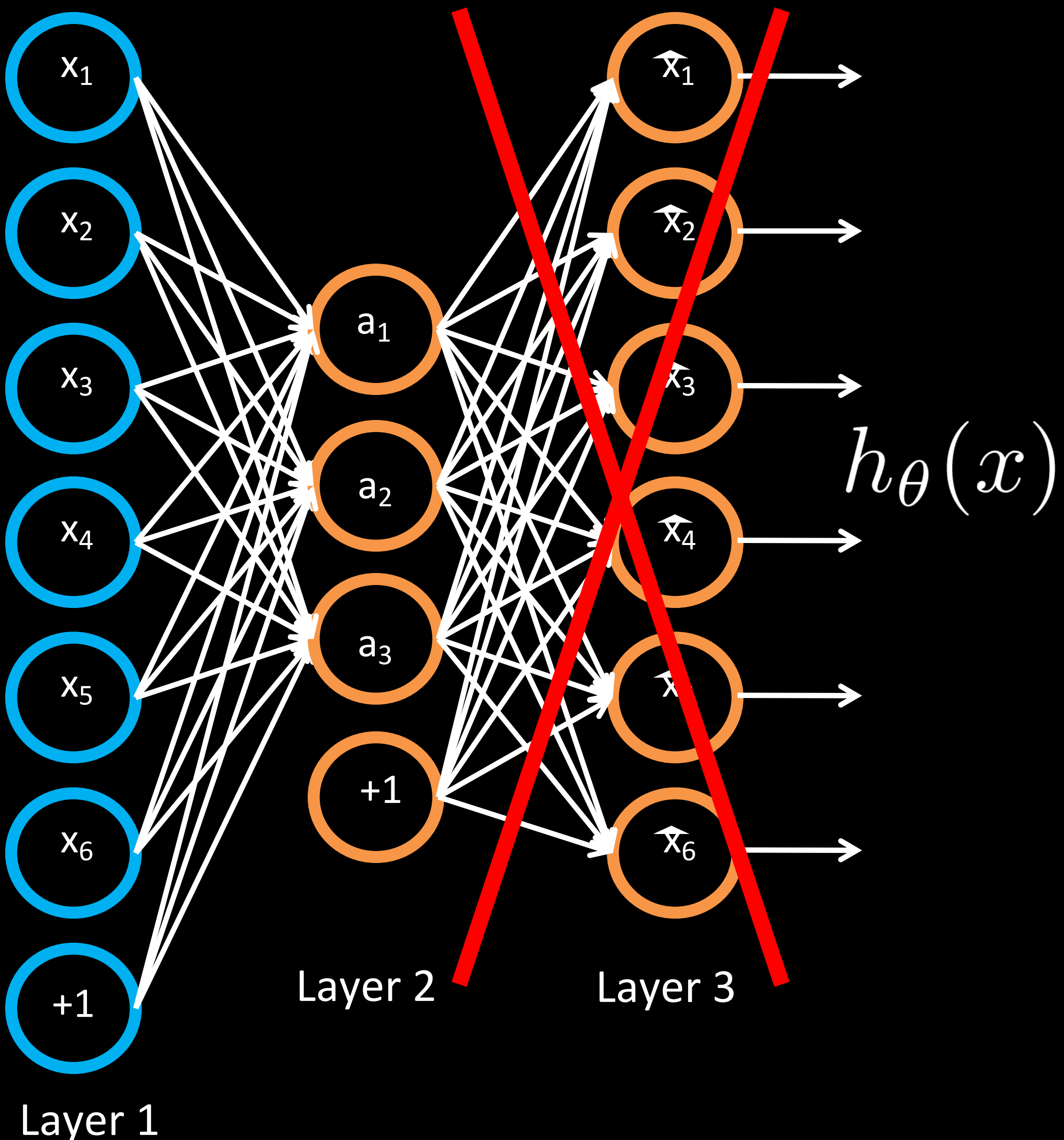
An autoencoder neural network is an unsupervised learning algorithm that applies backpropagation, setting the target values to be equal to the inputs.

The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction.



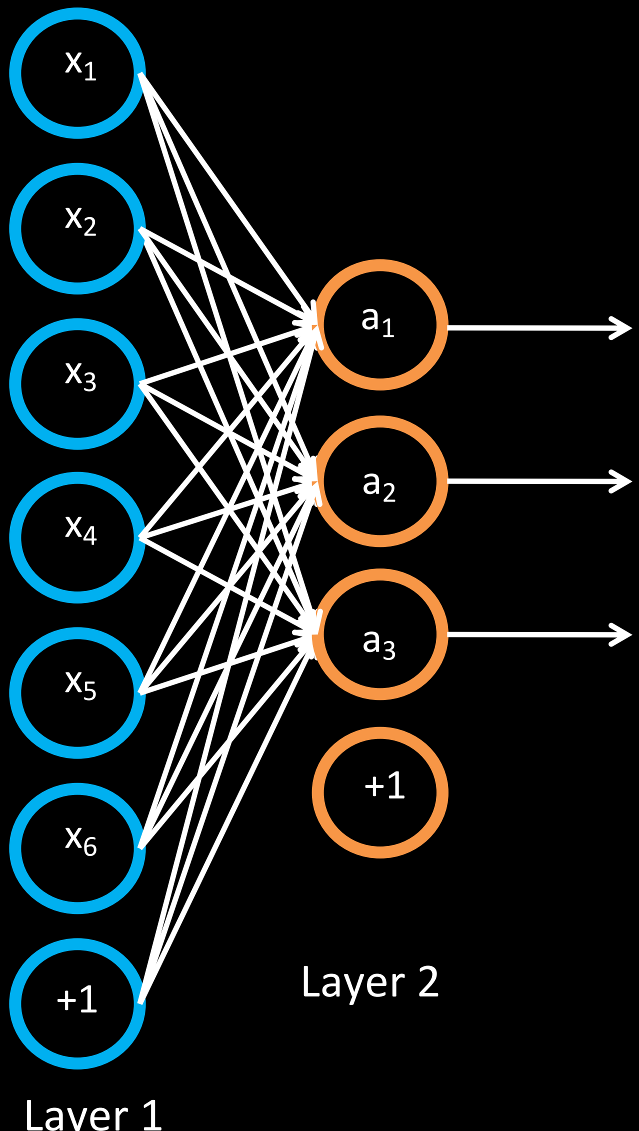
Universidad de Granada

Unsupervised feature learning with a neural network



Andrew Ng

Unsupervised feature learning with a neural network



$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

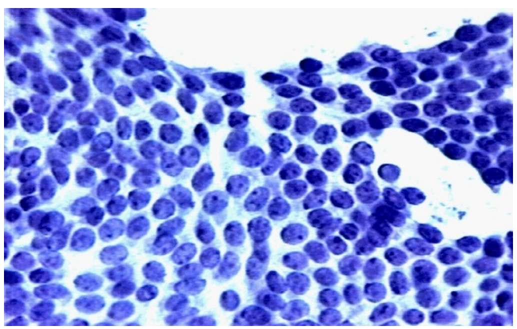
New representation for input.

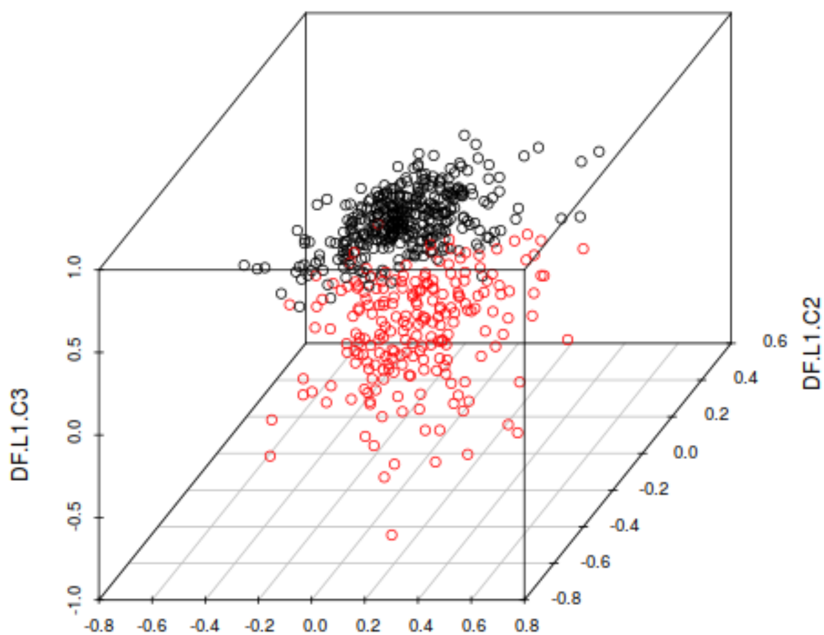
Andrew Ng

Deep Architecture: Autoencoder

- **Breast Cancer Wisconsin (Diagnostic) Data Set**

Autoencoder (f(x)=x) (una sola capa interna de 3 neuronas y 1000 "epochs". WDBC (569 instancias con 32 atributos de entrada)





Credito: D. Charte

Universidad de Granada

Deep Mind: Start up-2011
Demis Hassabis, Shane Legg y Mustafa Suleyman

Cornell University Library

arXiv.org > cs > arXiv:1312.5602

Search or Articl

Computer Science > Learning

Playing Atari with Deep Reinforcement Learning

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller

(Submitted on 19 Dec 2013)

We present the first deep learning model to successfully learn control policies directly from high-dimensional sensory input using reinforcement learning. The model is a convolutional neural network, trained with a variant of Q-learning, whose input is raw pixels and whose output is a value function estimating future rewards. We apply our method to seven Atari 2600 games from the Arcade Learning Environment, with no adjustment of the architecture or learning algorithm. We find that it outperforms all previous approaches on six of the games and surpasses a human expert on three of them.

Comments: NIPS Deep Learning Workshop 2013
Subjects: **Learning (cs.LG)**
Cite as: arXiv:1312.5602 **[cs.LG]**
(or **arXiv:1312.5602v1** **[cs.LG]** for this version)

Submission history

From: Volodymyr Mnih [view email]
[v1] Thu, 19 Dec 2013 16:00:08 GMT (221kb,D)

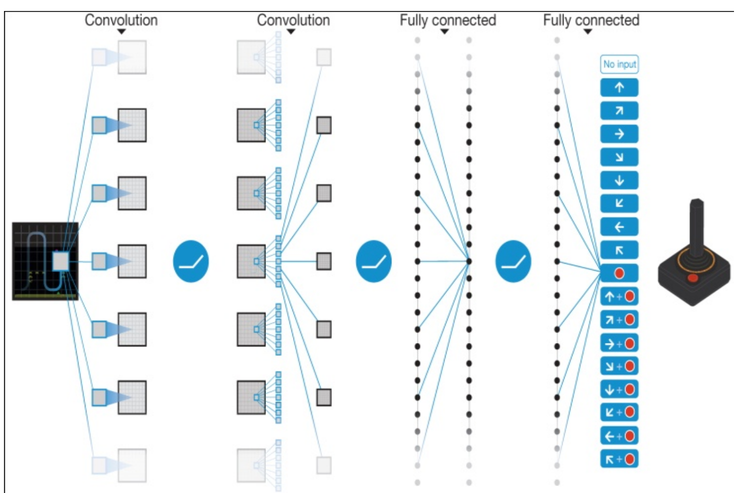
http://arxiv.org/abs/1312.5602

Ciencia de Datos y Deep Learning: Neuronas artificiales para aprender

Google DeepMind

El algoritmo se enfrentó a 49 juegos Arcade de los años 80 superando a otros algoritmos y a un jugador humano profesional

Juegos Arcade (Breakout)



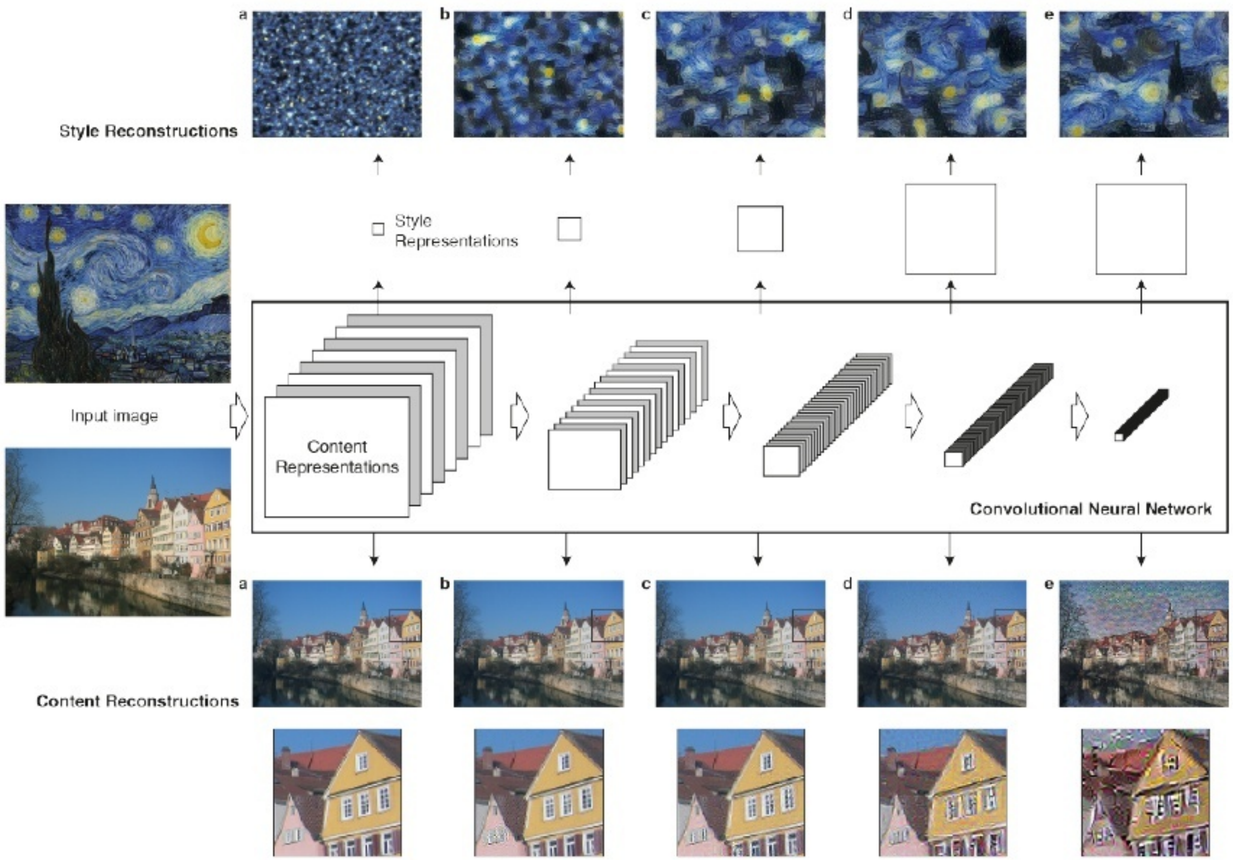
La inteligencia artificial hace historia: AlphaGo, de Google, derrotó al campeón del juego chino go

Publicado: 9 mar 2016 14:35 GMT

En un hito de la inteligencia artificial, la unidad DeepMind consiguió una victoria en la primera partida ante el surcoreano Lee Se-dol. "Estoy muy sorprendido, no esperaba perder", reconoció el jugador.

Ciencia de Datos y Deep Learning:
Neuronas artificiales para aprender

Deep Learning
Retos en la "pintura"



Deep Learning: DeepART

DEEPART.io Latest images Create image About Register Sign in

HOW IT WORKS

Our algorithm is inspired by the human brain. It uses the stylistic elements of one image to draw the content of another. Get your own artwork in just three steps.

1 Upload photo

2 Choose style

3 Submit

http://www.deepart.io/

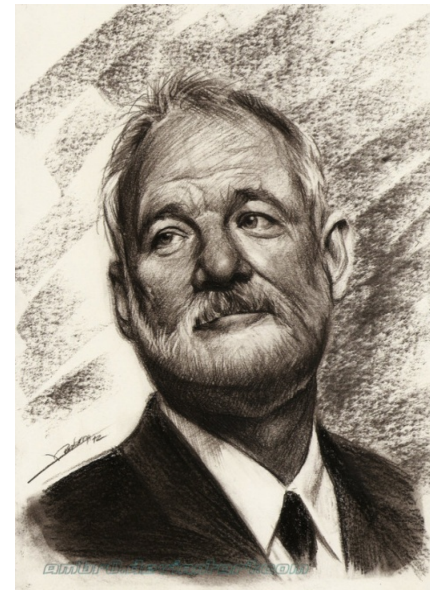
Universidad de Granada

Ejemplos del resultado de DeepART

van Goth

http://www.deepart.io/

Ejemplos del resultado de DeepART



http://www.deepart.io/

Modelo de DL utilizado y descripción de la metodología

Cornell University Library

arXiv.org > cs > arXiv:1508.06576

Computer Science > Computer Vision and Pattern Recognition

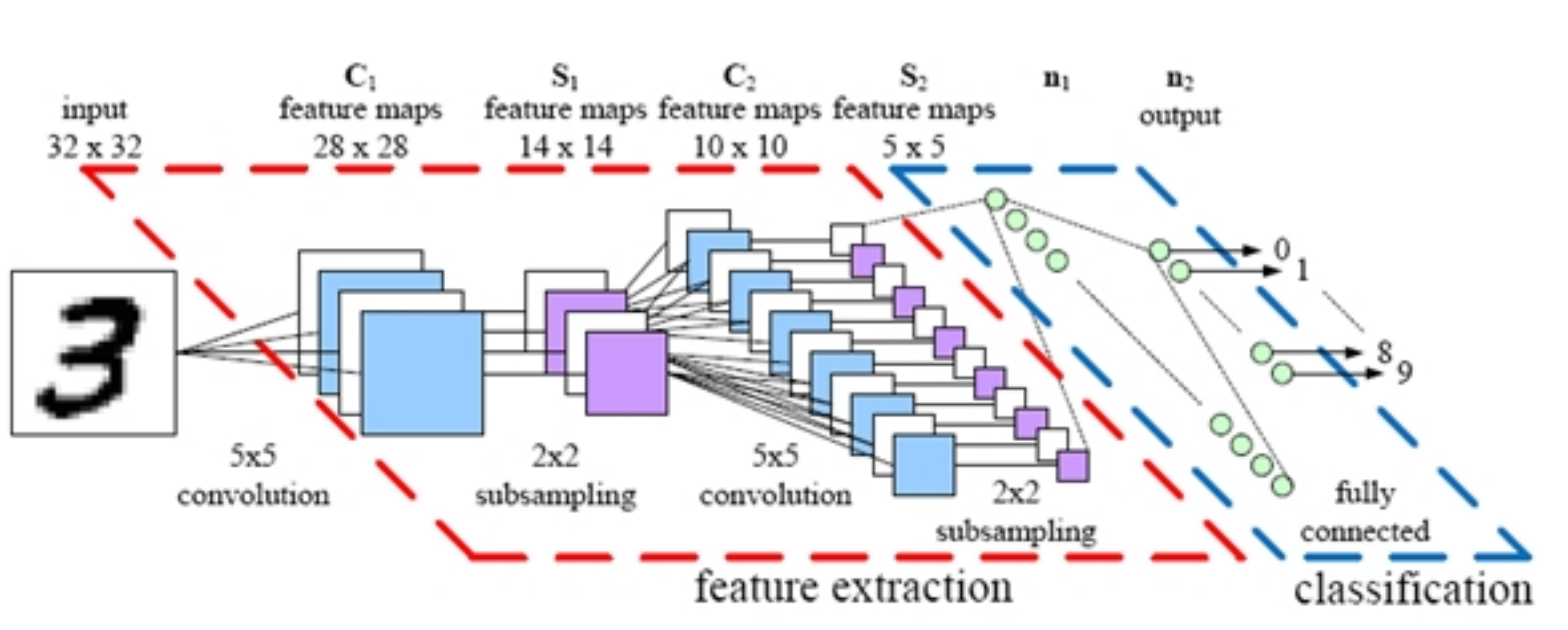
A Neural Algorithm of Artistic Style

Leon A. Gatys, Alexander S. Ecker, Matthias Bethge

(Submitted on 26 Aug 2015 (v1), last revised 2 Sep 2015 (this version, v2))

http://arxiv.org/abs/1508.06576

Supervised Deep Learning: Convolutional Neural Networks



http://parse.ele.tue.nl/cluster/2/CNNArchitecture.jpg

Supervised Deep Learning: Convolutional Neural Networks

Each module consists of a convolutional layer and a pooling layer.

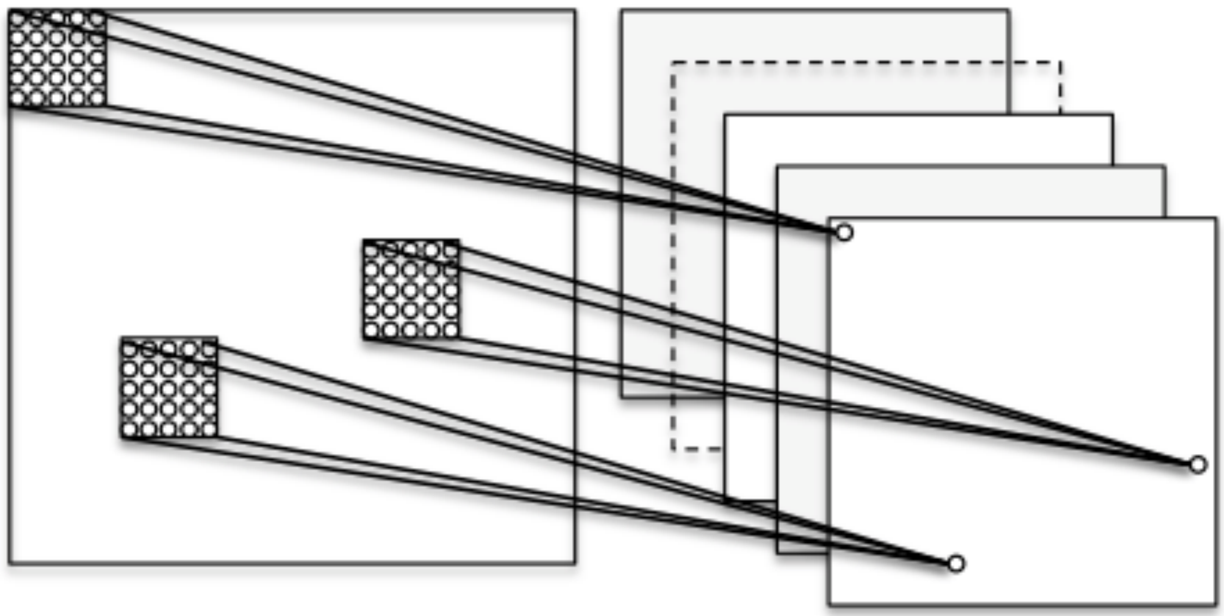
Typically tries to compress large data (images) into a smaller set of robust features, based on local variations.

Basic convolution can still create many features.

CNNs have been found highly effective and been commonly used in computer vision and image recognition.

Deep Learning

Convolutional steps



https://github.com/rasbt/python-machine-learning-book/blob/master/faq/difference-deep-and-normal-learning/convolution.png

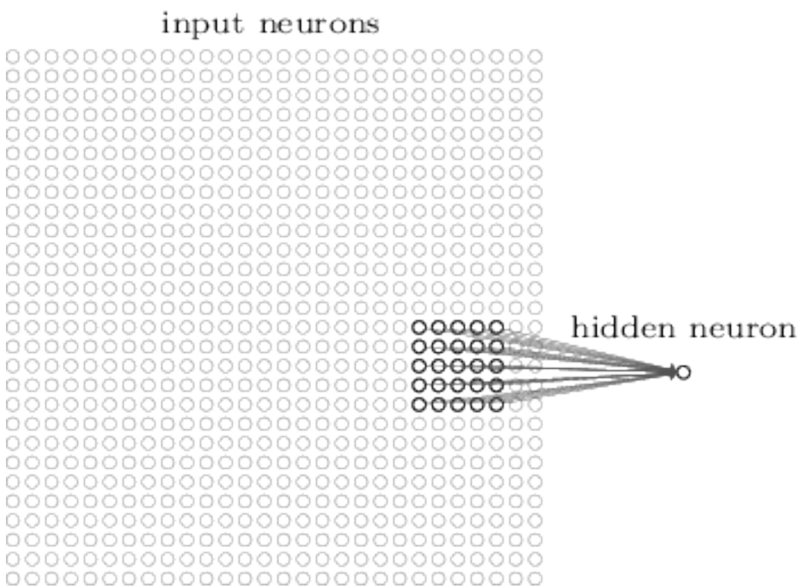
Deep Learning

Digit Recognizer and Convolutional NN

http://neuralnetworksanddeeplearning.com/chap6.html

Convolutional neural networks use three basic ideas: *local receptive fields*, *shared weights*, and *pooling*.

Local receptive fields: To be more precise, each neuron in the first hidden layer will be connected to a small region of the input neurons, say, for example, a 5×5 region, corresponding to 25 input pixels. So, for a particular hidden neuron, we might have connections that look like this:



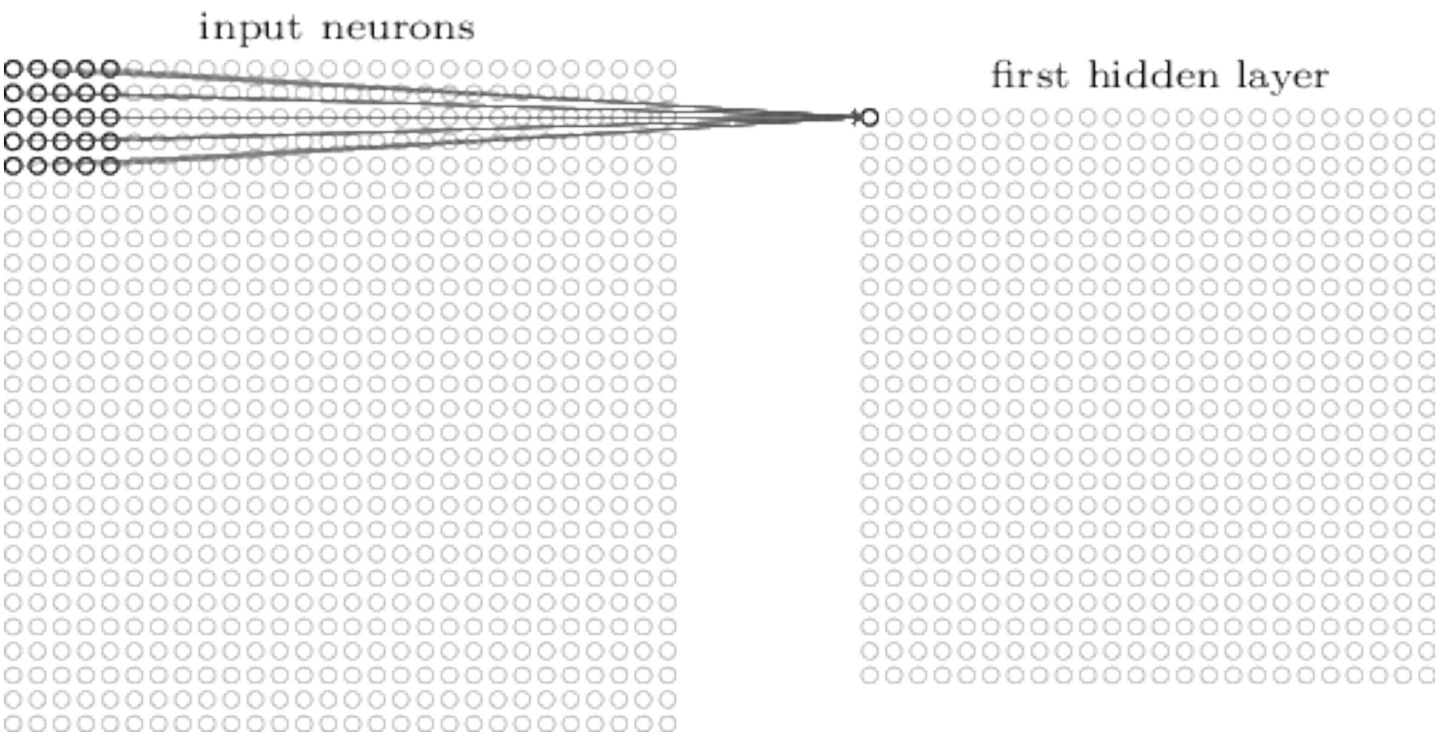
Deep Learning

Digit Recognizer and Convolutional NN

http://neuralnetworksanddeeplearning.com/chap6.html

Local receptive fields:

24×24 neurons



28×28 input image

Deep Learning

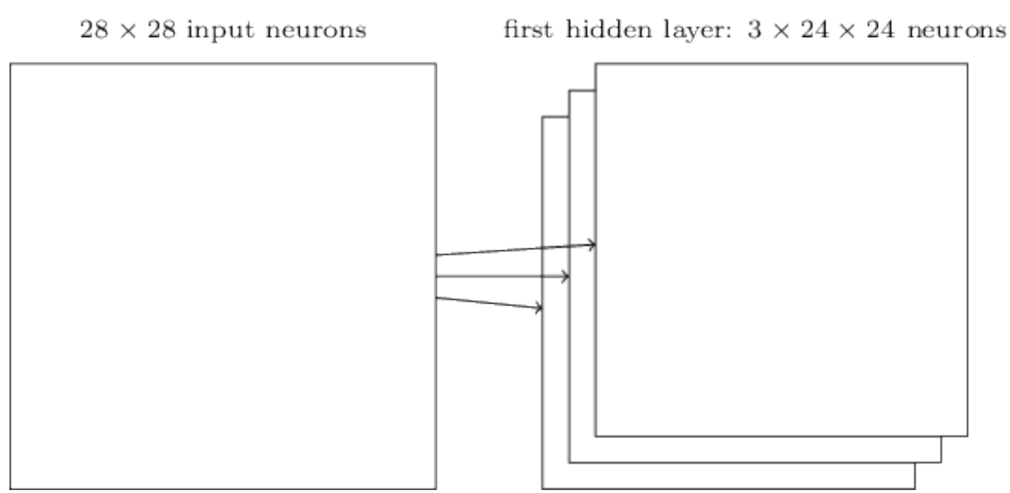
Digit Recognizer and Convolutional NN

http://neuralnetworksanddeeplearning.com/chap6.html

Shared weights and biases: the *same* weights and bias for each of the 24×24 hidden neurons (sigmoide function)

$$\sigma\left(b + \sum_{l=0}^{4}\sum_{m=0}^{4} w_{l,m}\, a_{j+l,k+m}\right). \qquad (125)$$

The map from the input layer to the hidden layer a *feature map*.

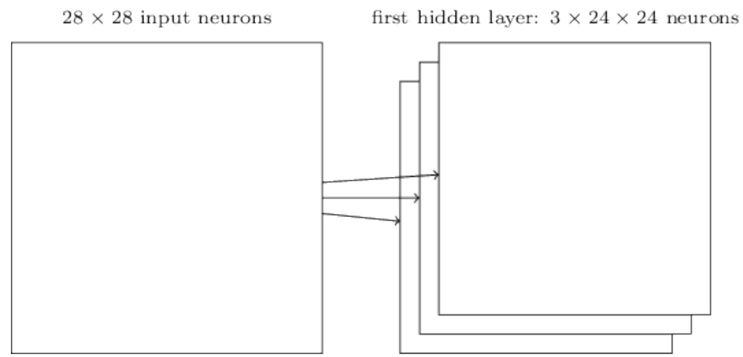


Deep Learning

Digit Recognizer and Convolutional NN

http://neuralnetworksanddeeplearning.com/chap6.html

Shared weights and biases:



In the example shown, there are 3 feature maps.

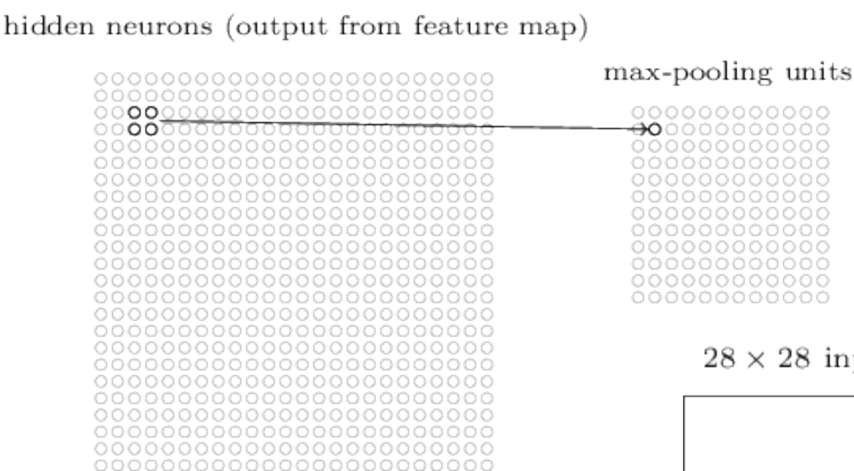
If we have 20 feature maps that's a total of 20×26=520 parameters defining the convolutional layer.
By comparison, suppose we had a fully connected first layer, with 784=28×28 input neurons, 30 hidden neurons, 23,550 parameters.

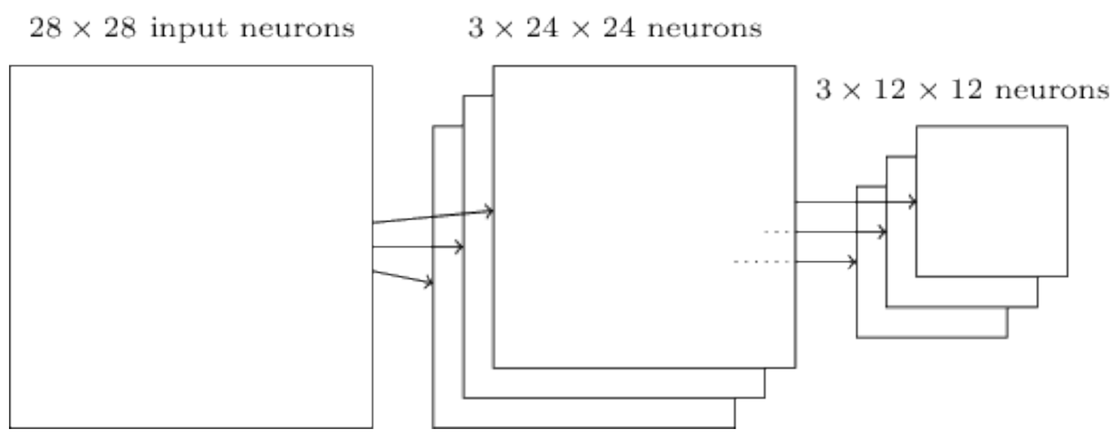
Deep Learning

Digit Recognizer and Convolutional NN

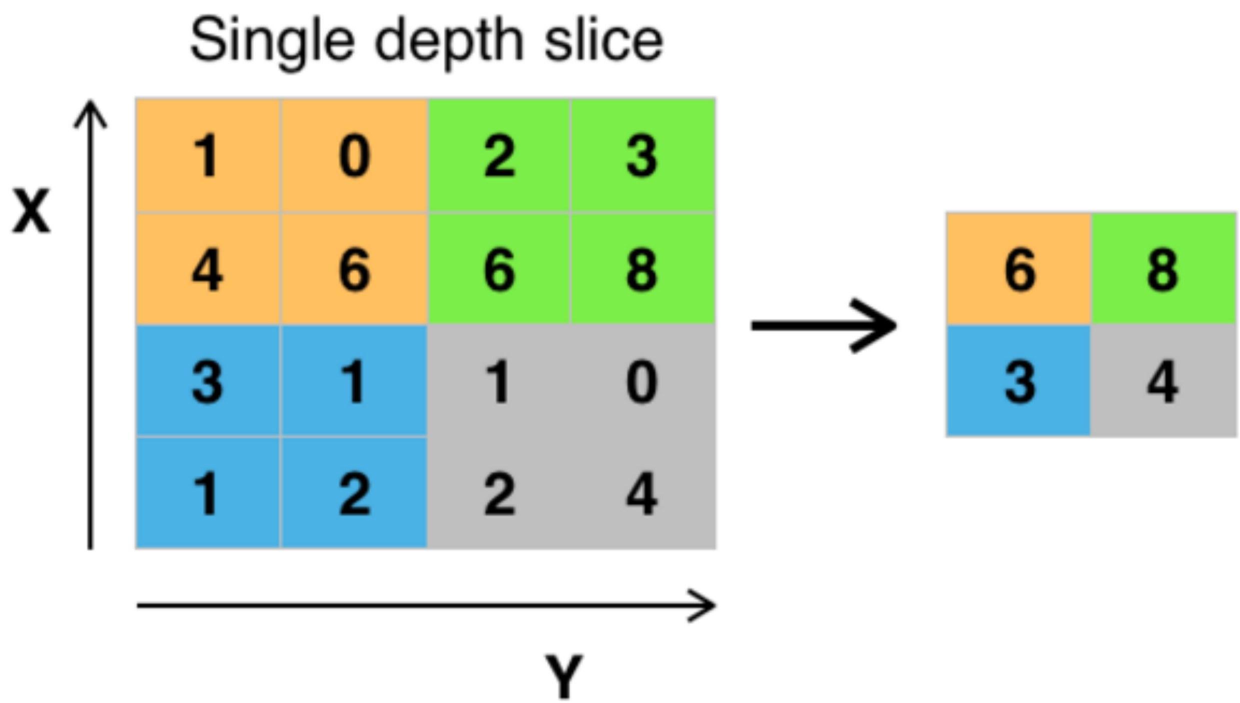
http://neuralnetworksanddeeplearning.com/chap6.html

Pooling layers: the pooling layers do is simplify the information in the output from the convolutional layer, one common procedure for pooling is known as *max-pooling*, in the 2x2 region input.





Pooling layers:

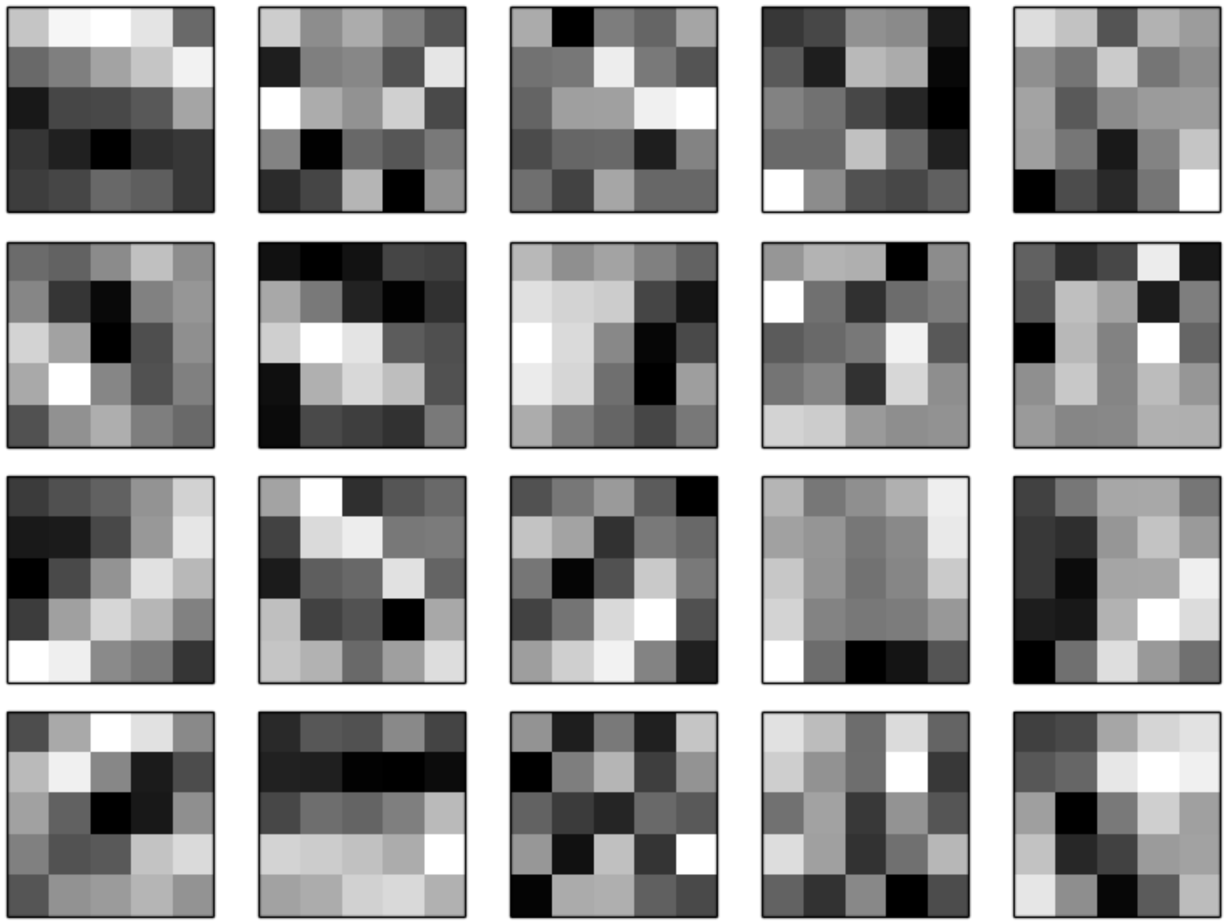


Max pooling with a 2x2 filter and stride = 2
(source: Wikipedia)

Deep Learning

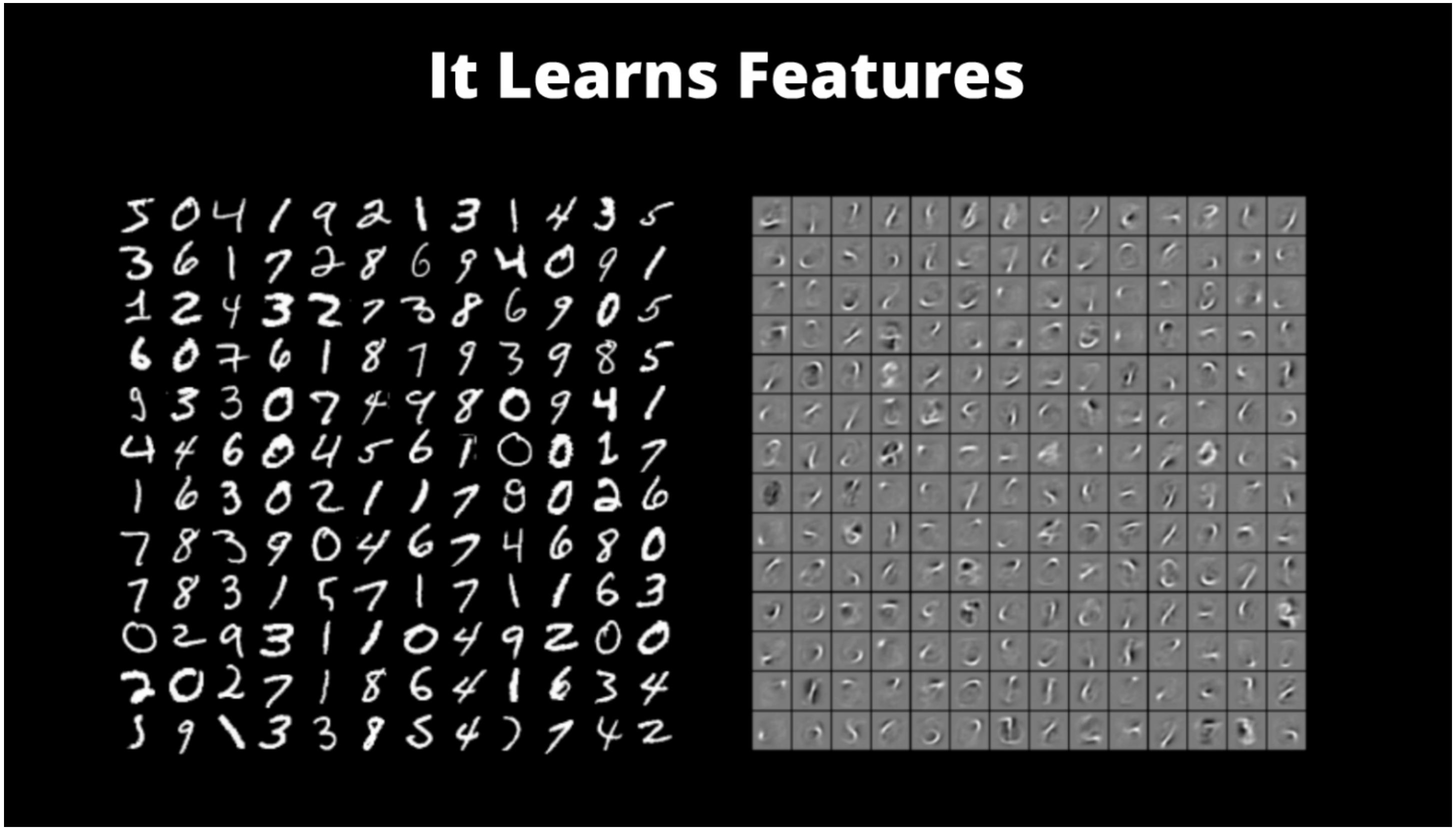
Digit Recognizer and Convolutional NN

http://neuralnetworksanddeeplearning.com/chap6.html



The 20 images correspond to 20 different feature maps

Deep Learning: MNIST data



A la izquierda, los dígitos de entrada sin procesar.
A la derecha, representaciones gráficas de las características aprendidas. En esencia, la red aprende a "ver" líneas y bucles.

Credits: https://www.datarobot.com/blog/a-primer-on-deep-learning/

Deep Learning

Digit Recognizer and preprocessing

A snapshot of image pre-processing for convolutional neural networks: case study of MNIST

Siham Tabik, Daniel Peralta, Andrés Herrera-Poyatos, Francisco Herrera

International Journal of Computational Intelligence Systems, Vol. 10 (2017) 555–568

99.72 accuracy

CNN models: LeNet, Network3, DropConnet

Preprocessing and augmentation

Ensembles

CNN models: LeNet [4]

Table 1. Topology of LeNet. Columns 2 and 3 show the configuration for processing original and cropped input images respectively.

Layer	Filter size for		*Stride*	Activation
	28x28-input	20x20-input		
conv1	$5 \times 5 \times 20$	$7 \times 7 \times 20$	1/3	–
maxpool1	2×2	2×2	2	–
conv2	$5 \times 5 \times 50$	$5 \times 5 \times 50$	1	–
maxpool2	2×2	2×2	2	–
fc1	500	500	–	ReLU
fc2	10	10	–	SoftMax

[4] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE, 86(11):2278*– 2324, 1998.

CNN models: Network3 [13]

Table 2. Topology of Network3 [13]. Columns 2 and 3 show the configuration for processing original and cropped input images respectively.

Layer	Filter size for 28x28-input	Filter size for 20x20-input	*Stride*	Activation
conv1	$5 \times 5 \times 20$	$3 \times 3 \times 20$	1	relu
maxpool1	2×2	2×2	2	–
conv2	$5 \times 5 \times 40$	$2 \times 2 \times 40$	1	relu
maxpool2	2×2	2×2	2	–
fc1	640	640	–	relu & dropout rate=0.5
fc2	1000	1000	–	relu & dropout rat =0.5
fc3	10	10	–	SoftMax

[13] Michael A Nielsen. Neural networks and deep learning. *URL: http://neuralnetworksanddeeplearning.com, 2015.*

CNN models: DropConnet [14]

Table 3. Topology of DropConnect [14]. Columns 2 and 3 show the configuration for processing original and cropped input images respectively.

Layer	Filter size for 28x28-input	Filter size for 20x20-input	*Stride*	Activation
conv1	$5 \times 5 \times 32$	$7 \times 7 \times 32$	1/3	–
maxpool1	2×2	2×2	2	–
conv2	$5 \times 5 \times 64$	$5 \times 5 \times 64$	1	–
maxpool2	2×2	2×2	2	–
fc1	150	150	–	relu & drop-connect rate: 0.5
fc2	10	10	–	softMax

[14] Li Wan, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *Proceedings of the 30th International Conference on Machine Learning (ICML- 13), pages 1058–1066, 2013.*

Preprocessing and augmentation

Translation. The image is translated a number of pixels toward a given direction.

Centering. To eliminate white columns/rows, and to resize by scaling.

Rotation. The image is rotated to a given angle θ.

Elastic deformation. Image pixels are slightly moved in random directions, keeping the image’s topology.

Preprocessing and augmentation

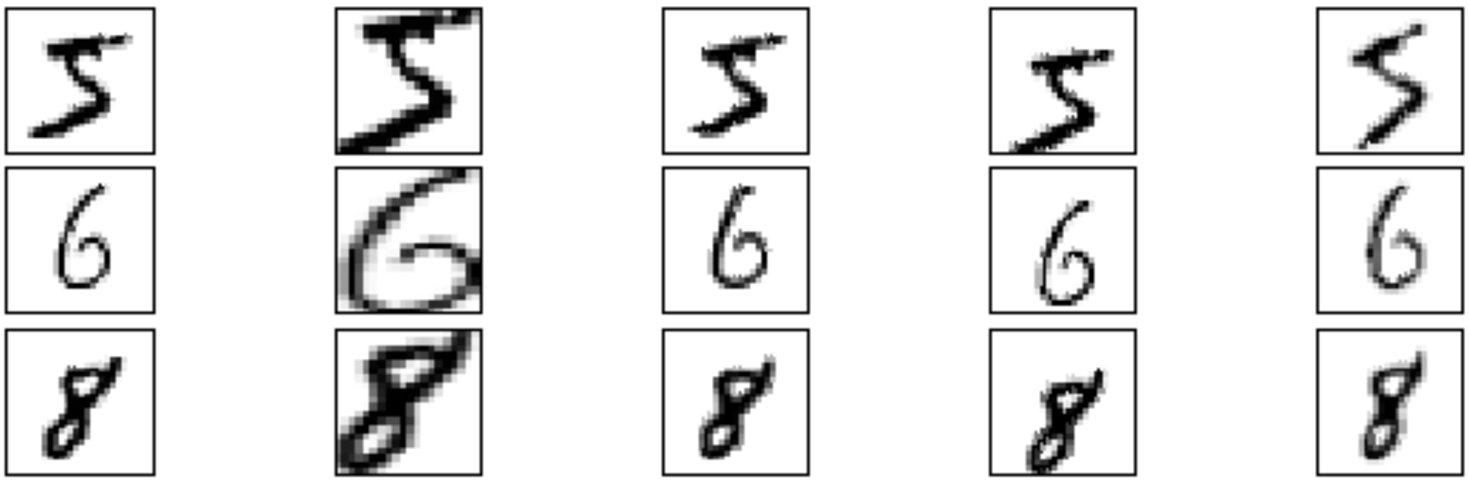


Fig. 2. Three original MNIST images (1st column) and the obtained images after centering (2nd column), elastic deformation (3rd column), translation (4th column) and rotation (5th column).

Deep Learning

Digit Recognizer and preprocessing

Preprocessing and augmentation

Table 4. The combinations of transformations analyzed in this study. All the combinations, from dataset 3 to 12, include the original dataset.

Dataset	Combination	# of training instances
1	Original	60,000
2	Centering	60,000
3	Elastic	300,000
4	Translation	300,000
5	Rotation	300,000
6	Elastic-centering	300,000
7	Rotation-centering	300,000
8	Translation-elastic	1,500,000
9	Translation-rotation	1,500,000
10	Rotation-elastic	1,500,000
11	Rotation-elastic-centering	1,500,000
12	Elastic-elastic	1,500,000

Deep Learning

Digit Recognizer and preprocessing

Preprocessing and augmentation

Table 6. Parameters of the used learning models and pre-processing algorithms

Algorithm	Parameter	Value
LeNet (SGD)	Number of iterations	10,000 / 50,000
	Batch size	64 / 256
	Learning rate	$lr_0(1+\gamma * iter)^{-b}$
	Initial learning rate (lr_0)	0.01
	γ	0.0001
	b	0.75
	Momentum	0.9
	Regularization Coefficient L2	0.0005
Network3 (SGD)	Number of epochs	10 / 20
	minibatch size	10
	Learning rate	0.03
	γ	0.0001
DropConnect (SGD)	Number of epochs	100 / 200
	minibatch size	128
	Learning rate	0.01
	Momentum	0.9
Elastic transformation	Typical deviation	6
	Number of transformations	4
	Strength	4
Translation	Magnitude	± 3 pixels
	Direction	Vertical y horizontal
Rotation	Angles	± 8 y ± 16 degrees
Centering	Final size	20×20 pixels

Deep Learning

Digit Recognizer and preprocessing

Preprocessing and augmentation

Table 7. Average and best test accuracies obtained by LeNet on each one of the preprocessed datasets using 10,000 and 50,000 iterations. Time (columns 5 and 9) is the time taken to execute the training-testing process on each dataset. The top five accurate models are labeled as [1], [2], [3], [4] and [5].

	LeNet (10,000 iter.)				LeNet (50,000 iter.)			
Dataset	**Average**	**Best**	**Epochs**	**Time(s)**	**Average**	**Best**	**Epochs**	**Time(s)**
Original	99.08%	99.18%	10.67	267.91	99.05%	99.21%	213.33	1070.29
Centered	98.85%	99.06%	10.67	203.52	98.95 %	98.09%	213.33	926.38
Elastic	99.09%	99.19%	2.13	232.75	99.36%	99.44%	42.67	1065.38
Translation	99.09%	99.32%	2.13	268.75	99.30%	99.41%	42.67	1065.38
Rotation	99.05%	99.10%	2.13	268.03	99.25%	99.37%	42.67	1065.38
Elastic-centered	[5]99.17%	99.26%	2.13	267.20	99.27%	99.36%	42.67	925.51
Rotation-centered	98.90%	99.07%	2.13	232.73	99.19%	99.33%	42.67	950.38
Translation-elastic	[4]99.18%	99.32%	0.43	267.43	[5]99.39%	99.54%	8.53	1050.38
Translation-rotation	99.16%	99.40%	0.43	267.41	[3]99.40%	97.55%	8.53	1045.38
Rotation-elastic	[1]**99.31%**	99.39%	0.43	268.14	[1]**99.47%**	**99.57%**	8.53	1046.25
Rotation-elastic-centered	[3]99.19%	99.24%	0.43	232.30	[2]99.43%	99.52%	8.53	925.68
Elastic-elastic	[2]99.27%	**99.45%**	0.43	268.10	[4]99.40%	99.50%	8.53	1047.64

Preprocessing and augmentation

Table 10. Test accuracies of LeNet with 500, 1000 and 1000 with dropout.

Dataset	500 neurons	1000 neurons	1000 neurons + dropout
Original	99,05%	99,05%	99,24%
Centered	98,95%	98,98%	99,16%
Elastic	99,36%	99,35%	99,46%
Translation	99,30%	99,29%	99,45%
Rotation	99,25%	99,26%	99,37%
Elastic-centered	99,27%	99,33%	99,41%
Rotation-centered	99,19%	99,15%	99,37%
Translation-elastic	99,39 %	99,39%	99,49%
Translation-rotation	99,40%	99,40%	99,49%
Rotation-elastic	99,47%	99,50%	99,55%
Rotation-elastic-centered	99,43%	99,48%	99,48%
Elastic-elastic	99,40%	99,50%	99,53%

Deep Learning

Digit Recognizer and preprocessing

Preprocessing and augmentation

Table 8. Average and best test accuracies obtained by Network3 on the twelve datasets using 10 and 20 epochs. Time (columns 5 and 9) is the time taken to train Network3 on each dataset. The top five accurate models are labeled as [1], [2], [3], [4] and [5].

	Network3(10 epochs)			**Network3(20 epochs)**		
Dataset	**Average**	**Best**	**Time(s)**	**Average**	**Best**	**Time(s)**
Original	99.01%	99.07%	124.45	99.25%	99.25%	205,21
Centered	98.73%	98.80%	118.32	98.97%	99.01%	196.92
Elastic	99.49%	99.54%	656,85	[3]99.61%	99.67%	1200,33
Translation	[5]99.49%	99.55%	631.53	[4]99.59%	99.63%	1228,71
Rotation	99.44%	99.50%	636.25	99.44%	99.50%	1256,95
Elastic-centered	99.32%	99.39%	566.44	99.57%	99.60%	1109,43
Rotation-centered	98.88%	98.94%	569.04	99.31%	99.32%	1167,32
Translation-elastic	[4]99.54%	99.57%	3647.78	[5]99.58%	99.63%	7111,65
Translation-rotation	[3]99.57%	99.61%	3650.66	99.58%	99.60%	7149,25
Rotation-elastic	[2]99.62%	**99.67%**	3642,85	[2]99.67%	99.69%	6996,23
Rotation-elastic-centered	99.43%	99.51%	3054,43	99.51%	99.52%	6908,70
Elastic-elastic	[1]**99.65%**	99.66%	3607.32	[1]**99.67** %	**99.70%**	7189,16

Deep Learning

Digit Recognizer and preprocessing

Preprocessing and augmentation

Table 9. Average and best test accuracies obtained by DropConnect on the twelve datasets using 100 and 200 epochs. Time (columns 5 and 9) is the time taken to execute the training-testing process on each dataset. The top five accurate models are labeled as [1], [2], [3], [4] and [5].

	DropConnet(100 epochs)			DropConnet(200 epochs)		
Dataset	**Average**	**Best**	**Time(s)**	**Average**	**Best**	**Time(s)**
Original	98,32%	98,83%	7803.43	98.98%	98,99%	18748.53
Centered	95.35%	94,46%	6659.31	95.13%	98,85%	18635.54
Elastic	99.33 %	99,35%	7512.25	99.36%	99,36%	18606.15
Translation	[5]99.43%	99,46%	7736.41	[5]99.47%	99,47%	18710.45
Rotation	99.18%	99,29%	7151.73	99.37%	99,47%	18729.29
Elastic-centered	96.58%	96,69%	6969.89	97.08%	97,09%	18661.80
Rotation-centered	98.30%	98,41%	6974.23	98.55%	98,63%	18668.05
Translation-elastic	99.40%	99,57%	7162.37	[3]99.58%	99,58%	18745.93
Translation-rotation	[2]99.57%	99,59%	7410.32	[1]**99.69%**	**99,69%**	18772.40
Rotation-elastic	[3]99.54%	**99,60%**	7397.40	[4]99.56%	99,56%	18724.38
Rotation-elastic-centered	[4]99.47%	99,49%	7803.73	99,44%	99,46%	18220.50
Elastic-elastic	[1]**99,58%**	99,59%	7911.30	[2]99,59%	99,61%	18712.22

Preprocessing and augmentation

Ensembles

Table 11. Test accuracies obtained by ensemble-5 and ensemble-3, using three configurations of the 1st fully connected layer of Lenet, based on three decision strategies, the most voted strategy (MV), the maximum of probability summation (MPS) and the maximum probability (MP).

	LeNet								
	500 neurons			**1000 neurons**			**1000 neurons with dropout**		
	MV	PSM	MP	MV	PSM	MP	MV	PSM	MP
Ensemble-5	99,57%	99,56%	99,59%	99,60%	99,62%	99,63%	99,61%	**99,64%**	99,63%
Ensemble-3	99,54%	99,58%	99,62%	99,61%	99,66%	99,64%	99,65%	**99,68%**	99,64%

Table 12. Test accuracy of ensemble-5 and ensemble-3 of LeNet, Network3 and DropConnect, for two training periods and using the most voted strategy.

	LeNet(500 neurons)		**Network3**		**DropConnect**	
	10,000 iter	**50,000 iter**	**10 epochs**	**20 epochs**	**100 epochs**	**200 epochs**
Ensemble-5	99,55%	**99,57%**	**99,72%**	**99,69%**	**99,72%**	99,66%
Ensemble-3	99,43%	**99,54%**	99,69%	99,67%	99,69%	**99,68%**

Deep Learning

Digit Recognizer and preprocessing

Preprocessing and augmentation

The 28 handwritten digits misclassified by ensemble-5 of Network3

The digit between () represents the correct class.

The 13 digits labeled with asterisks are also misclassified by DropConnect

Deep Learning: MNIST data

A snapshot of image pre-processing for convolutional neural networks: case study of MNIST

Siham Tabik, Daniel Peralta, Andrés Herrera-Poyatos, Francisco Herrera

International Journal of Computational Intelligence Systems, Vol. 10 (2017) 555–568

99.72 accuracy

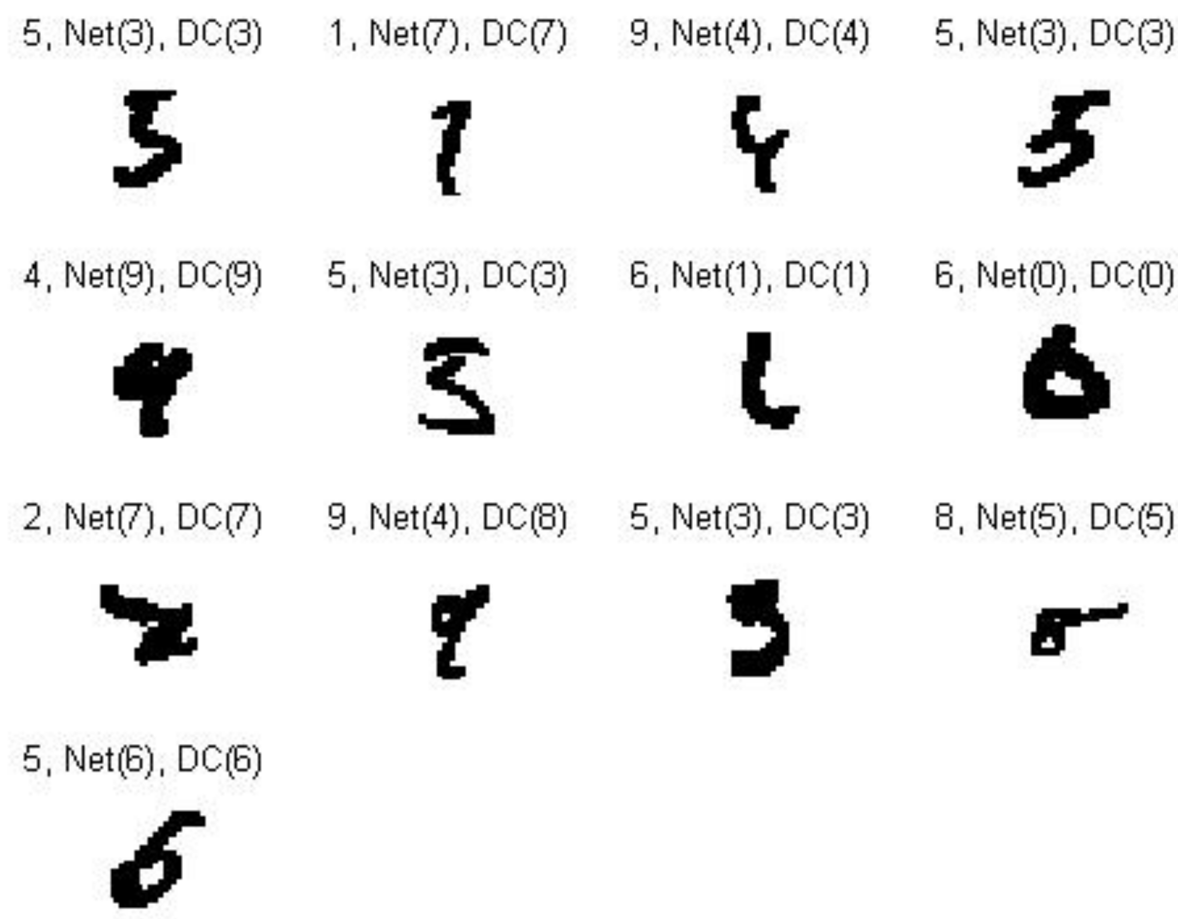
The 28 handwritten digits misclassified by ensemble-5 of DropConnet
The digit between () represents the correct class.
The 13 digits labeled with asterisks are also misclassified by Network3

Deep Learning

Digit Recognizer and preprocessing

Preprocessing and augmentation

The 13 handwritten digits misclassified by ensemble-5 of DropConnet and Network3



Deep Learning
MXNet

http://mxnet.io/get_started/index.html

MXNet: A Scalable Deep Learning Framework

MXNet is an open-source deep learning framework that allows you to define, train, and deploy deep neural networks on a wide array of devices, from cloud infrastructure to mobile devices. It is highly scalable, allowing for fast model training, and supports a flexible programming model and multiple languages. MXNet allows you to mix symbolic and imperative programming flavors to maximize both efficiency and productivity. MXNet is built on a dynamic dependency scheduler that automatically parallelizes both symbolic and imperative operations on the fly. A graph optimization layer on top of that makes symbolic execution fast and memory efficient. The MXNet library is portable and lightweight, and it scales to multiple GPUs and multiple machines.

Please choose the programming language of your choice for the rest of this document.

Python | R | Scala | Julia | Perl

Deep Learning
MXNet

http://mxnet.io/how_to/finetune.html

Fine-tune with Pre-trained Models

Many of the exciting deep learning algorithms for computer vision require massive datasets for training. The most popular benchmark dataset, ImageNet, for example, contains one million images from one thousand categories. But for any practical problem, we typically have access to comparatively small datasets. In these cases, if we were to train a neural network's weights from scratch, starting from random initialized parameters, we would overfit the training set badly.

One approach to get around this problem is to first pretrain a deep net on a large-scale dataset, like ImageNet. Then, given a new dataset, we can start with these pretrained weights when training on our new task. This process commonly called "fine-tuning". There are anumber of variations of fine-tuning. Sometimes, the initial neural network is used only as a *feature extractor*. That means that we freeze every layer prior to the output layer and simply learn a new output layer. In another document, we explained how to do this kind of feature extraction. Another approach is to update all of networks weights for the new task, and that's the appraoch we demonstrate in this document.

To fine-tune a network, we must first replace the last fully-connected layer with a new one that outputs the desired number of classes. We initialize its weights randomly. Then we continue training as normal. Sometimes it's common use a smaller learning rate based on the intuition that we may already be close to a good result.

Deep Learning

Learning from scratch vs fine-tuning (GoogLeNet y VGG-16)

VGG16

http://www.robots.ox.ac.uk/~vgg/research/very_deep/

K. Simonyan, A. Zisserman
Very Deep Convolutional Networks for Large-Scale Image Recognition
arXiv technical report, 2014
https://arxiv.org/pdf/1409.1556.pdf

VGG is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet , which is a dataset of over 14 million images belonging to 1000 classes.

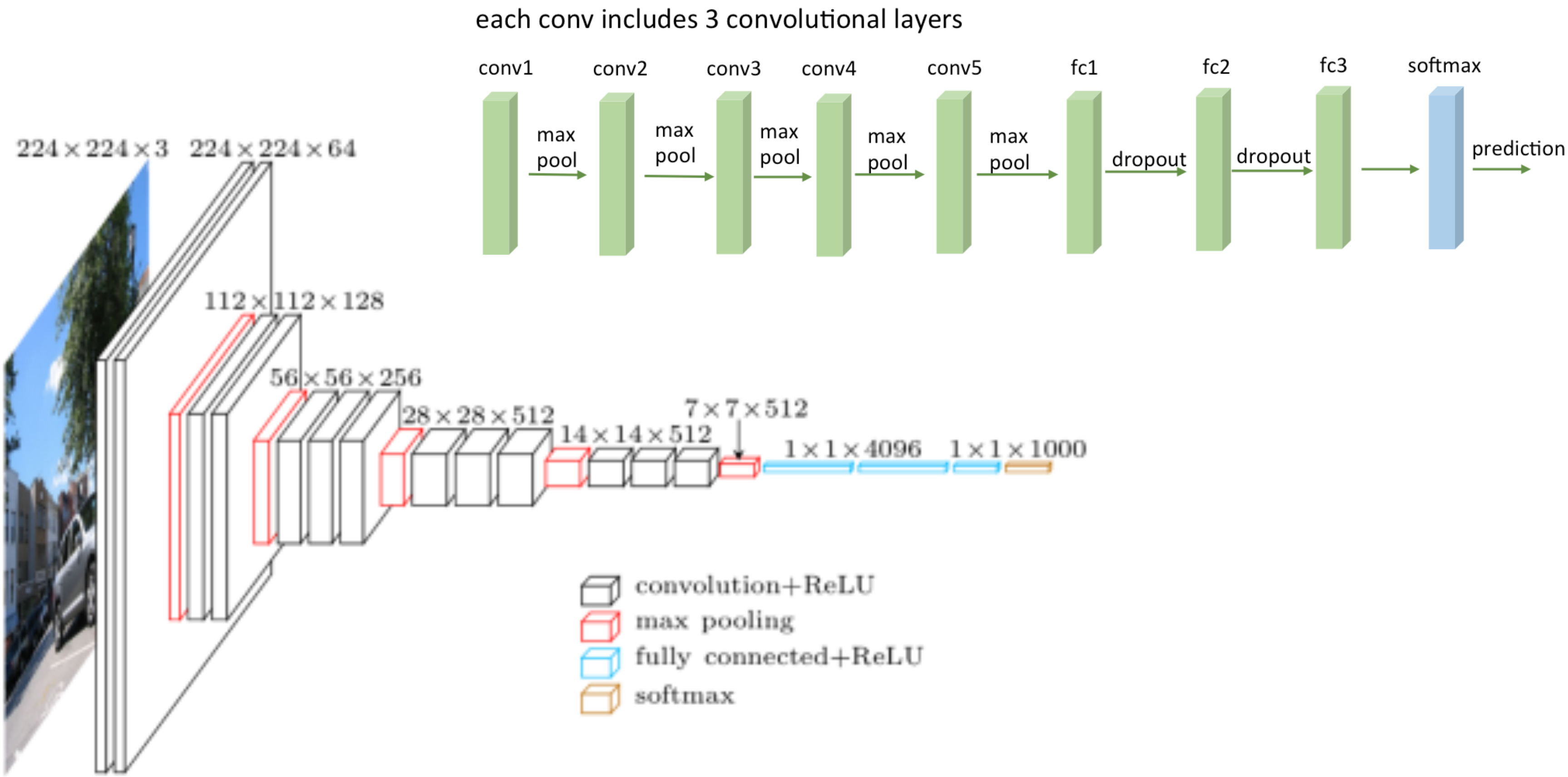
https://gist.github.com/ksimonyan/211839e770f7b538e2d8#file-readme-md

##Information
name: 16-layer model from the arXiv paper: "Very Deep Convolutional Networks for Large-Scale Image Recognition"
caffemodel: VGG_ILSVRC_16_layers
caffemodel_url: http://www.robots.ox.ac.uk/~vgg/software/very_deep/caffe/VGG_ILSVRC_16_layers.caffemodel
license: see http://www.robots.ox.ac.uk/~vgg/research/very_deep/
caffe_version: trained using a custom Caffe-based framework

Deep Learning

Learning from scratch vs fine-tuning (GoogLeNet y VGG-16)

VGG16



Source: https://www.cs.toronto.edu/~frossard/post/vgg16/

Deep Learning

Learning from scratch vs fine-tuning (GoogLeNet y VGG-16)

VGG16

Model and pre-trained parameters for VGG16 in TensorFlow
https://www.cs.toronto.edu/~frossard/post/vgg16/

Keras: https://keras.io/
https://keras.io/applications/
Deep Learning library for Theano and TensorFlow
Keras is a high-level neural networks API, written in Python and capable of running on top of either TensorFlow or Theano. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.*

Models for image classification with weights trained on ImageNet

- Xception
- VGG16
- VGG19
- ResNet50
- InceptionV3

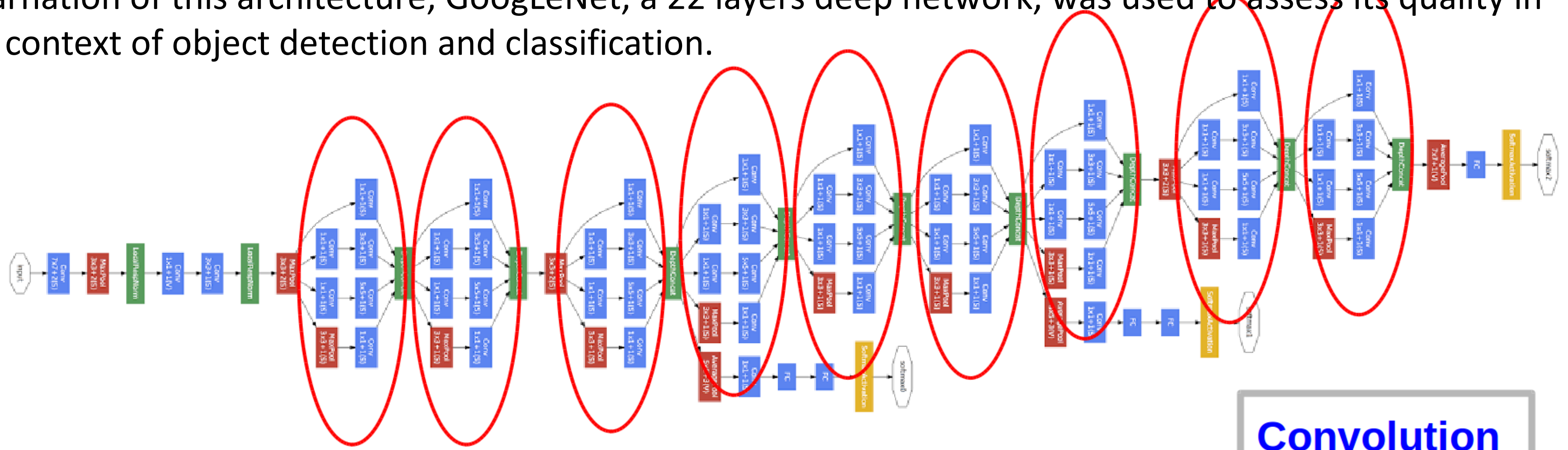
Deep Learning

Learning from scratch vs fine-tuning (GoogLeNet y VGG-16)

GoogLeNet

https://research.google.com/pubs/pub43022.html

We propose a deep convolutional neural network architecture codenamed Inception that achieves the new state of the art for classification and detection in the ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC2014). The main hallmark of this architecture is the improved utilization of the computing resources inside the network. By a carefully crafted design, we increased the depth and width of the network while keeping the computational budget constant. To optimize quality, the architectural decisions were based on the Hebbian principle and the intuition of multi-scale processing. One particular incarnation of this architecture, GoogLeNet, a 22 layers deep network, was used to assess its quality in the context of object detection and classification.



Convolution
Pooling
Softmax
Concat/Normalize

Going Deeper with Convolutions

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich

arXiv technical report, 2014

Deep Learning

Learning from scratch vs fine-tuning (GoogLeNet y VGG-16)

GoogLeNet

http://www.robots.ox.ac.uk/~vgg/research/very_deep/

GooglLeNet in Caffe
https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet

GooglLeNet in Keras
https://gist.github.com/joelouismarino/a2ede9ab3928f999575423b9887abd14
http://joelouismarino.github.io/blog_posts/blog_googlenet_keras.html

Deep Learning: Detección de Armas en Video

arXiv:submit/1806117 [cs.AI] 16 Feb 2017

Automatic Handgun Detection Alarm in Videos Using Deep Learning

Video: Skyfall, 2012

R. Olmos, S. Tabik, F. Herrera (UGR)

Universidad de Granada

Deep Learning: Detección de Armas en Video

R. Olmos, S. Tabik, F. Herrera (UGR)

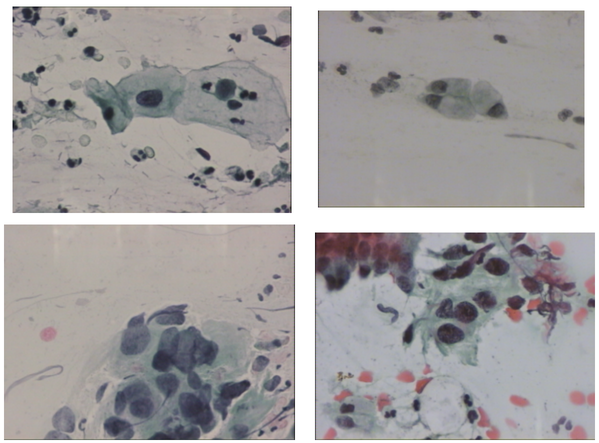
Video: Skyfall, 2012

Universidad de Granada

Ciencia de Datos y Deep Learning:
Neuronas artificiales para aprender

Ejemplos de aplicaciones

Cell classification from cervix smears



Fingerprint identification/classification

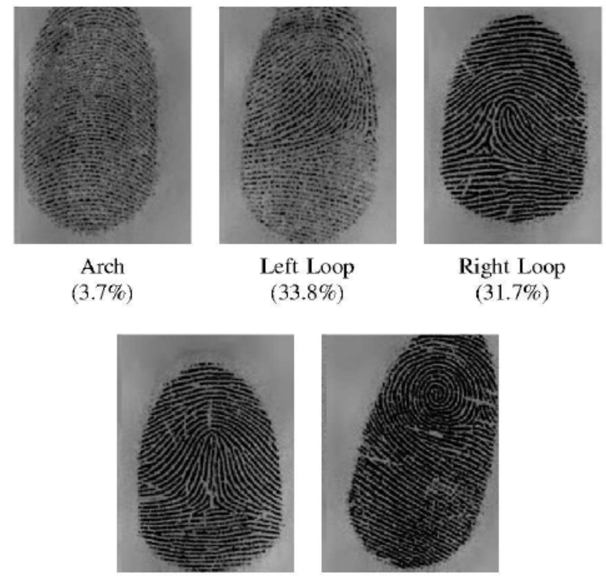
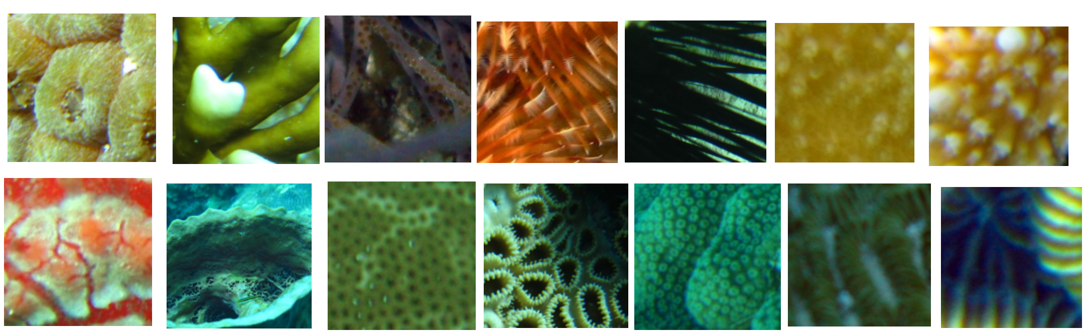


Fig. 1. Five fingerprint classes defined by Henry [10] and their frequencies.

Underwater image patches of corals



Universidad de Granada

Deep Learning

Lectura: Recent Overview

nature International weekly journal of science

Menu ▸ Advance

archive ▸ volume 521 ▸ issue 7553 ▸ insights ▸ reviews ▸ article

NATURE | INSIGHT | REVIEW

Deep learning

Yann LeCun, Yoshua Bengio & Geoffrey Hinton

Affiliations | Corresponding author

Nature **521**, 436–444 (28 May 2015) | doi:10.1038/nature14539
Received 25 February 2015 | Accepted 01 May 2015 | Published online 27 May 2015

http://www.nature.com/nature/journal/v521/n7553/full/nature14539.html

LeCum, Bengio y Hinton, 2015: "Ultimately, major progress in artificial intelligence will come about through systems that **combine representation learning with complex reasoning.** Although deep learning and simple reasoning have been used for speech and handwriting recognition for a long time, **new paradigms are needed to replace rule-based manipulation of symbolic expressions by operations on large vectors**"

Los avances en el conocimiento del cerebro y el razonamiento humano permitirán diseñar nuevos paradigmas de representación y razonamiento complejo.