

Introduction to Deep Learning

Bootcamp IID 2023

Mathieu Godbout et Alexandre Bouras

{mathieu.godbout.3, alexandre.bouras.1}@ulaval.ca

15 mai 2023



1 Introduction

2 Neural Networks

3 Training

4 Overfitting

Who Are We?



1 Introduction

2 Neural Networks

3 Training

4 Overfitting

Deep Neural Networks



[Krizhevsky2012ImageNetCW, Krizhevsky2012ImageNetCW]

Deep Neural Networks

The image shows a user interface for a deep learning model, likely a neural network trained on the ImageNet dataset. At the top, there are four small images with their predicted labels below them:

- A red mite with the label "mite" and a confidence score of 0.999.
- A large container ship with the label "container ship" and a confidence score of 0.999.
- A person on a motor scooter with the label "motor scooter" and a confidence score of 0.999.
- A leopard with the label "leopard" and a confidence score of 0.999.

Below these are two rows of predicted labels for other objects:

mite	container ship	motor scooter	leopard
mite	container ship	motor scooter	leopard
black widow	lifeboat	go-kart	jaguar
cockroach	amphibian	moped	cheetah
tick	fireboat	bumper car	snow leopard
starfish	drilling platform	golfcart	Egyptian cat

grille	mushroom	dalmatian	squirrel monkey
convertible	agaric	dalmatian	squirrel monkey
grille	mushroom	grape	spider monkey
pickup	jelly fungus	elderberry	titi
beach wagon	gill fungus	ffordshire bullterrier	indri
fire engine	dead-man's-fingers	currant	howler monkey

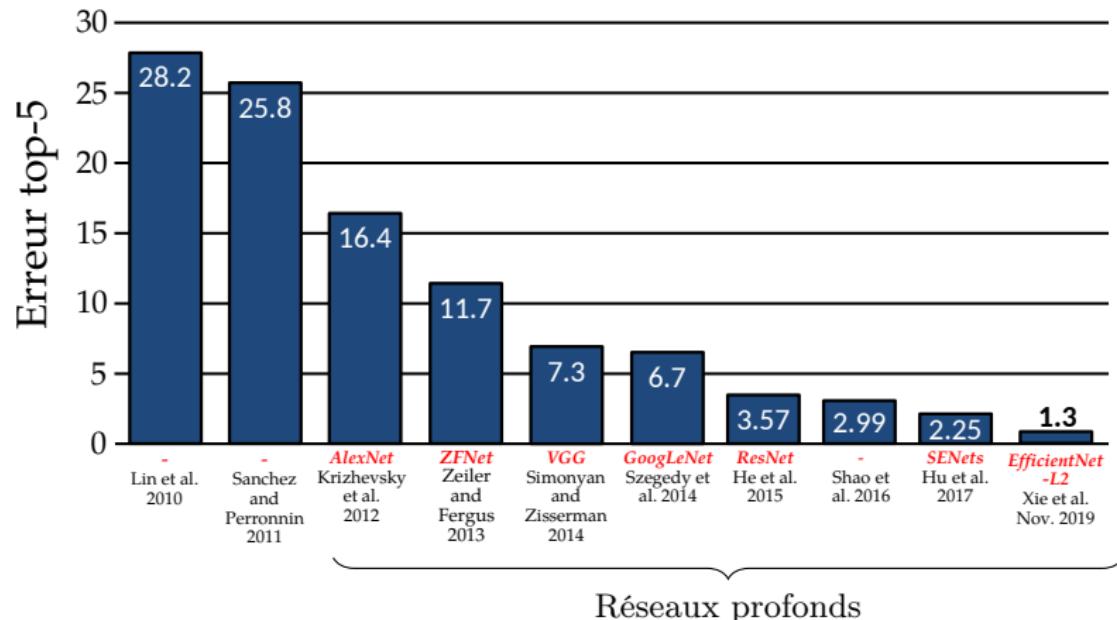
The interface includes language detection and translation features. It shows "FRANÇAIS ANGLAIS ARABE" and "DETECTOR LA LANGUE". Below the interface, there is a text input field with the placeholder "Deep learning is awesome!" and a translated message "L'apprentissage en profondeur est génial!" with a star icon.

[Krizhevsky2012ImageNetCW, Krizhevsky2012ImageNetCW]

Computer Vision

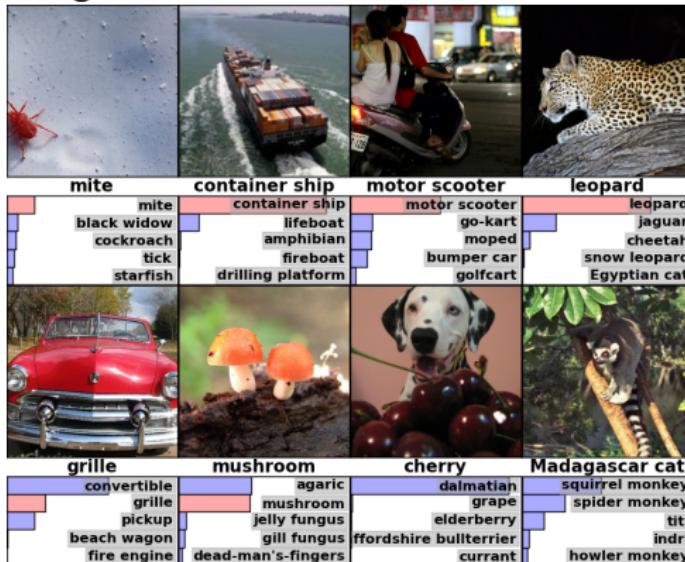
Large Scale Visual Recognition Challenge (LSVRC)

Image classification challenge on the 1,000 image classes of ImageNet



Computer Vision

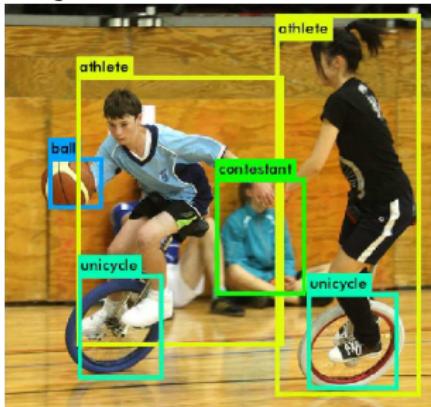
■ Image classification



[Krizhevsky2012ImageNetCW, Krizhevsky2012ImageNetCW]

Computer Vision

- Image classification
- Object detection



[redmon2017yolo9000, redmon2017yolo9000]

Computer Vision

- Image classification
- Object detection
- Image segmentation



[cordts2016cityscapes, cordts2016cityscapes]

Computer Vision

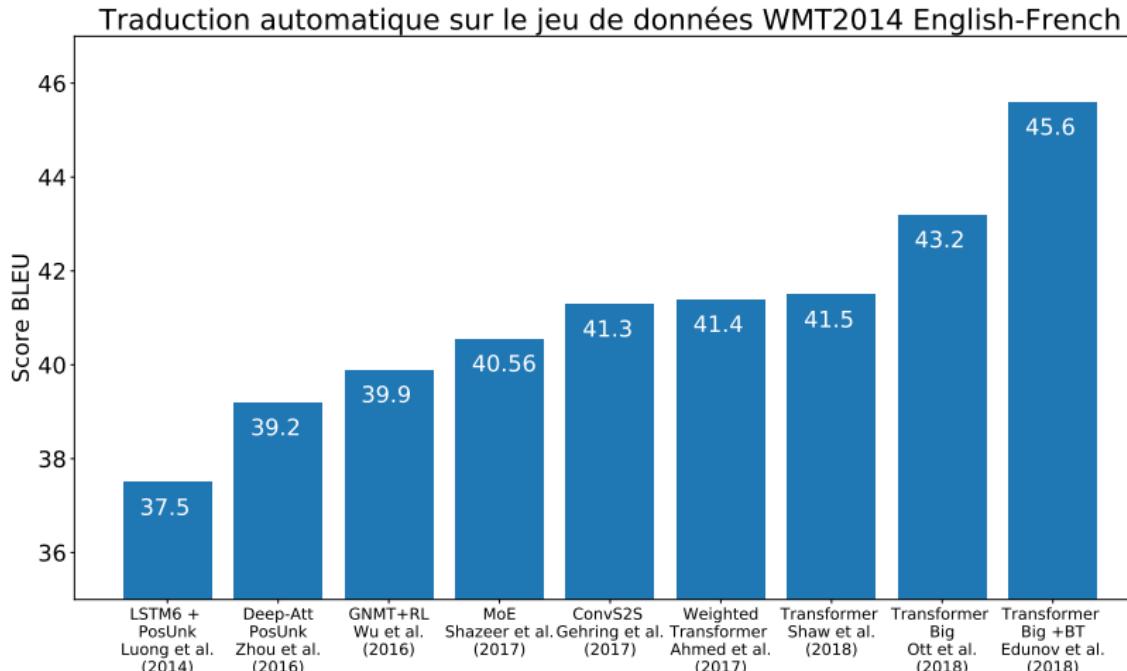
- Image classification
- Object detection
- Image segmentation
- Image generation



Computer Vision

- Image classification
- Object detection
- Image segmentation
- Image generation
- etc.

Natural Language Processing (NLP)



<https://paperswithcode.com/sota/machine-translation-on-wmt2014-english-french>

Natural Language Processing

■ Translation

The screenshot shows a translation interface with two main sections. The left section displays the English input "Deep learning is awesome!" with a small 'x' icon to its right. The right section shows the French translation "L'apprentissage en profondeur est génial!" followed by a star icon. Above these sections are language selection menus: "DETECTOR LA LANGUE ANGLAIS FRANCAIS ARABE" on the left and "FRANCAIS ANGLAIS ARABE" on the right. Below the input and output fields are various icons for file operations and settings. A progress bar at the bottom indicates "25 / 5000".

Natural Language Processing

- Translation
- Text classification (e.g. topic, sentiment)



John Doe

★★★★★ **Awesome product**

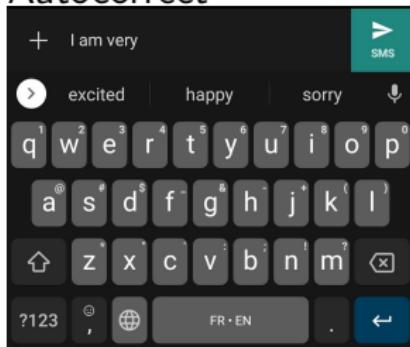
Reviewed in Canada on May 04, 2021

This is an awesome product. It couldn't be better!!!

16 people found this helpful

Natural Language Processing

- Translation
- Text classification (e.g. topic, sentiment)
- Autocorrect



Natural Language Processing

- Translation
- Text classification (e.g. topic, sentiment)
- Autocorrect
- Chatbot



Natural Language Processing

- Translation
- Text classification (e.g. topic, sentiment)
- Autocorrect
- Chatbot
- etc.

1 Introduction

2 Neural Networks

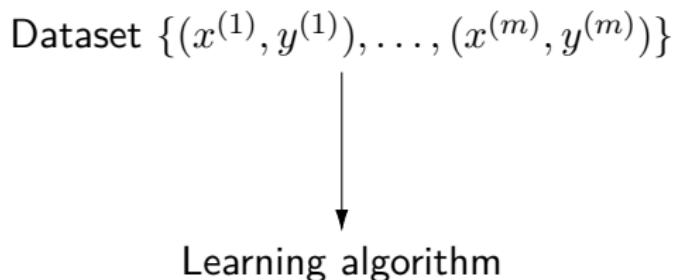
3 Training

4 Overfitting

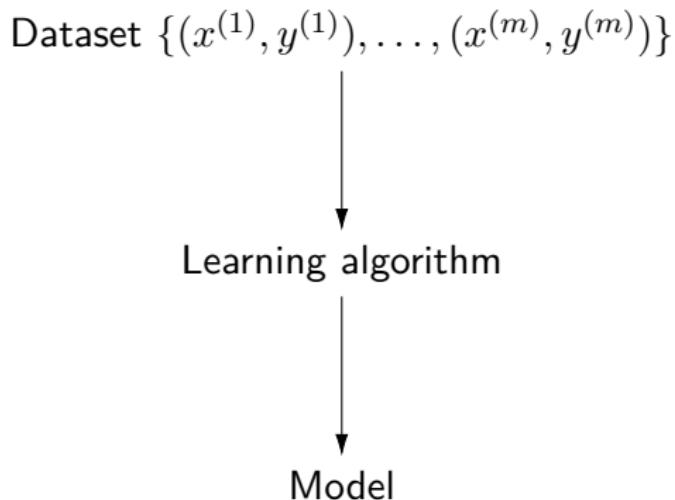
Machine Learning

Dataset $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$

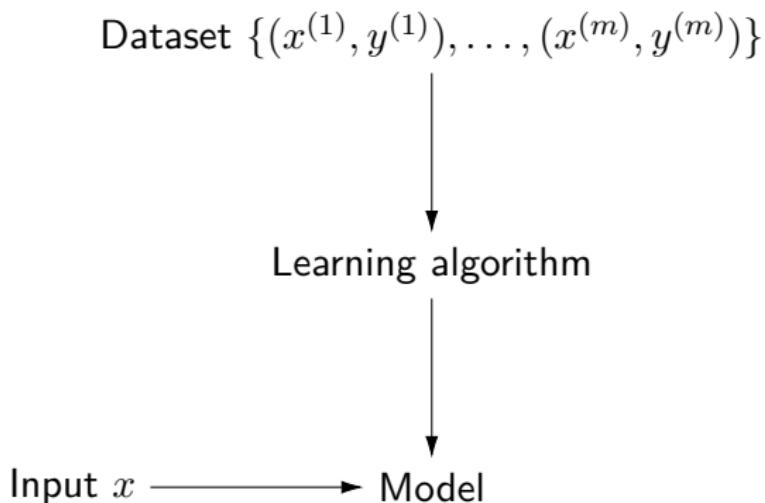
Machine Learning



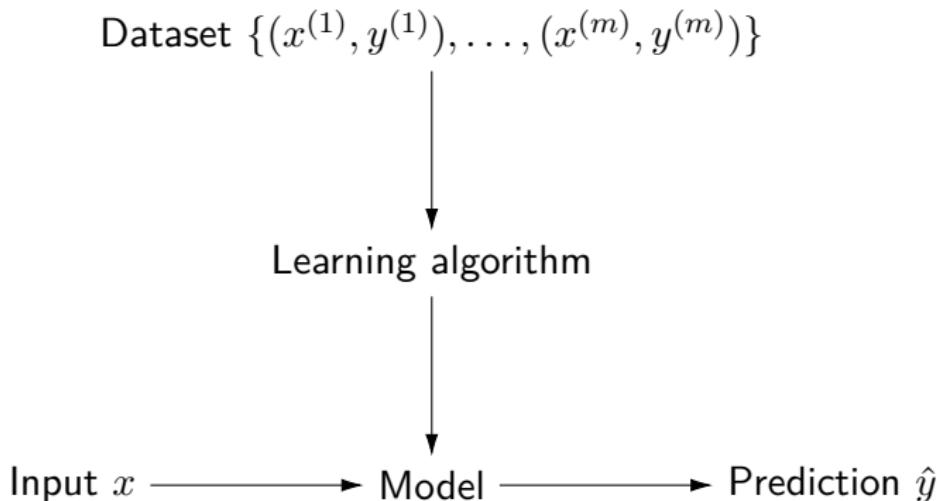
Machine Learning



Machine Learning



Machine Learning

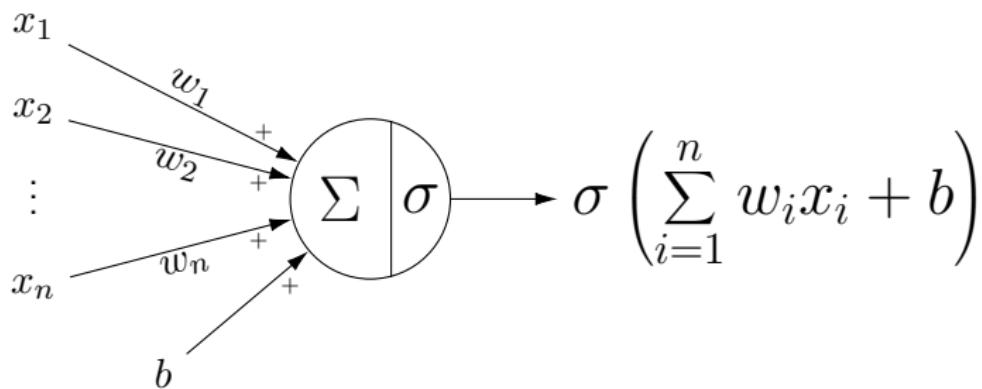


The Basis of Neural Networks: The Neuron

Let $x = (x_1, x_2, \dots, x_n)$ be n features (or variables in statistics).

The Basis of Neural Networks: The Neuron

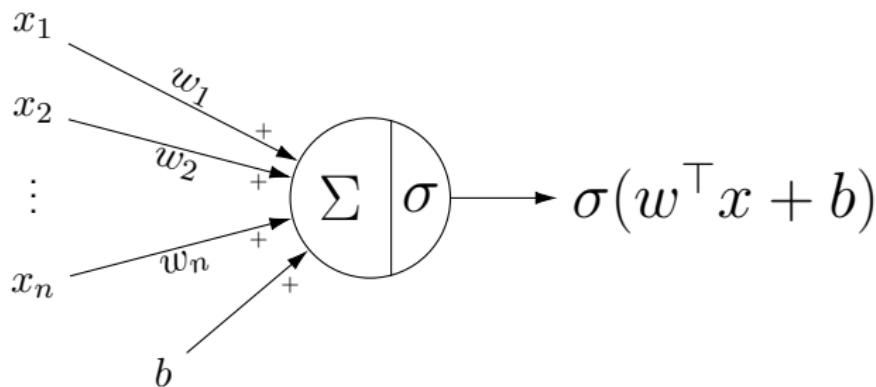
Let $x = (x_1, x_2, \dots, x_n)$ be n features (or variables in statistics). A neuron computes the following.



where w are called the weights of the neuron and $\sigma(\cdot)$ is a non-linear activation function.

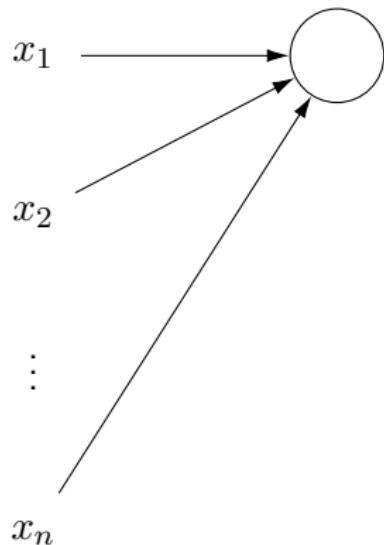
The Basis of Neural Networks: The Neuron

Let $x = (x_1, x_2, \dots, x_n)$ be n features (or variables in statistics). A neuron computes the following.

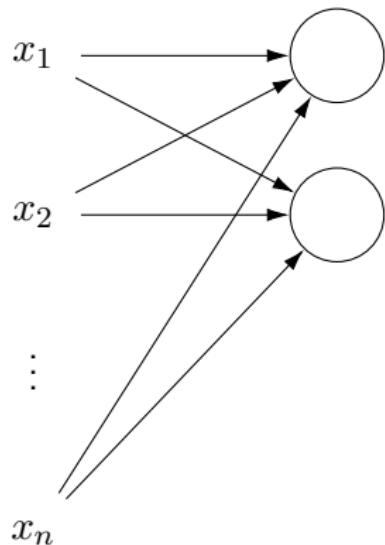


where w are called the weights of the neuron and $\sigma(\cdot)$ is a non-linear activation function.

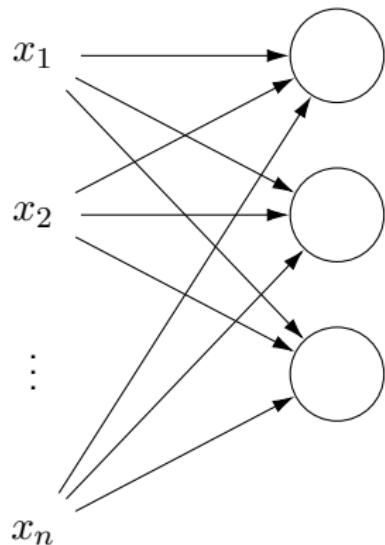
Neural Network



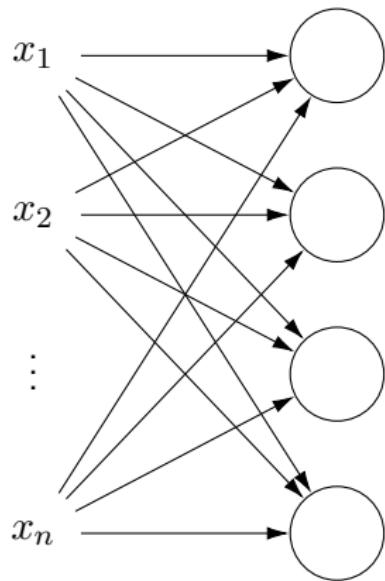
Neural Network



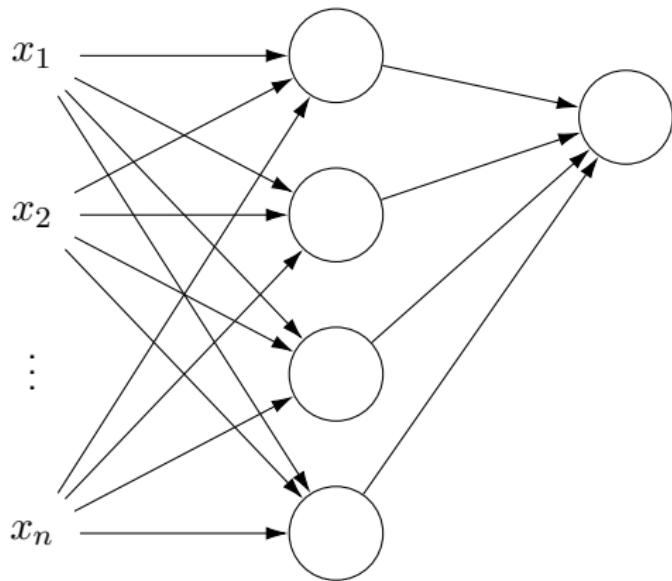
Neural Network



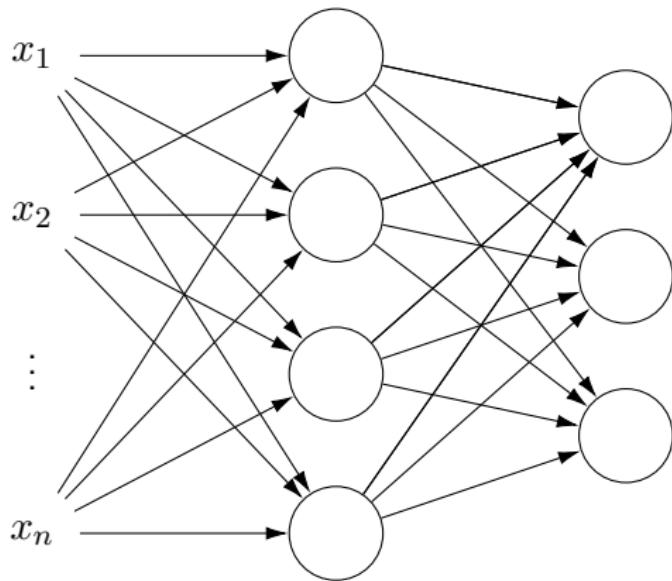
Neural Network



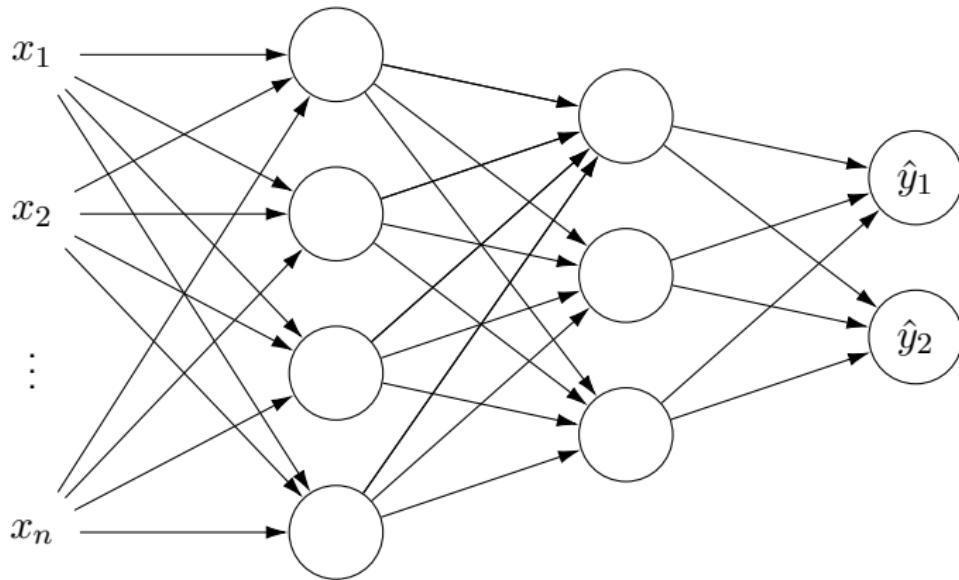
Neural Network



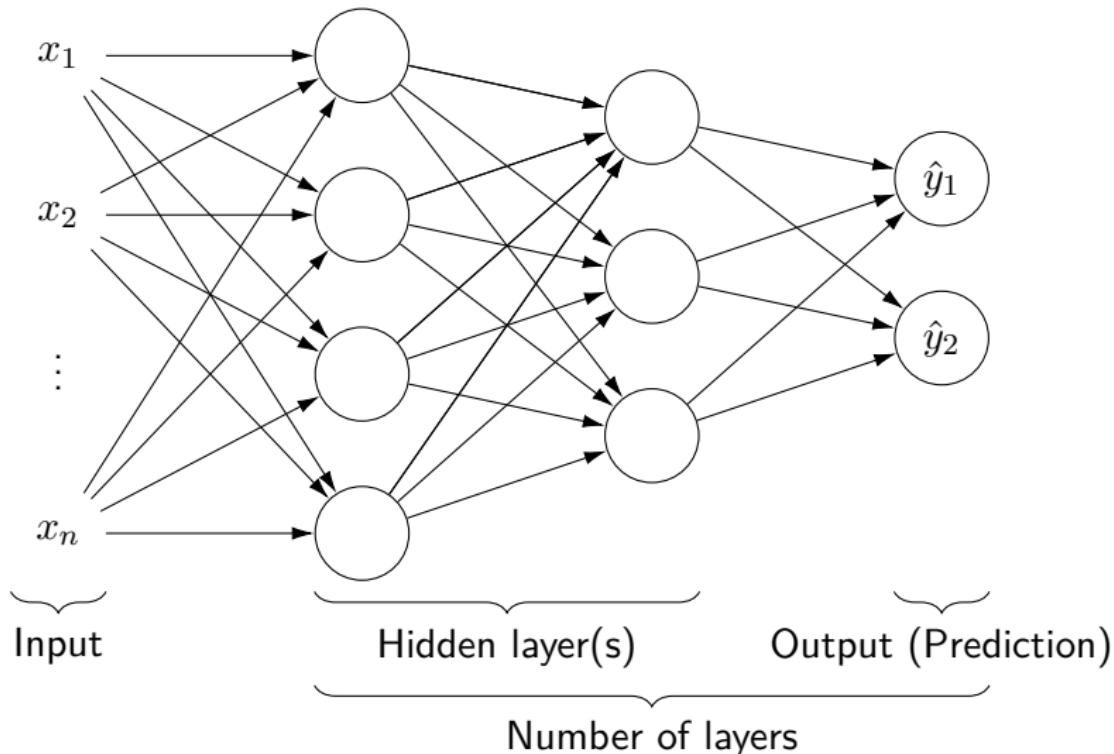
Neural Network



Neural Network



Neural Network



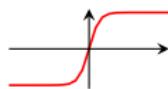
Activation Functions

Given a non-activated output $z = w^\top x + b$, then $\sigma(z) = \dots$

Activation Functions

Given a non-activated output $z = w^\top x + b$, then $\sigma(z) = \dots$

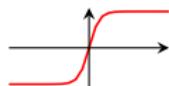
- $\tanh(z)$



Activation Functions

Given a non-activated output $z = w^\top x + b$, then $\sigma(z) = \dots$

- $\tanh(z)$



- The ReLU function:

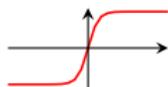
$$\text{ReLU}(z) = \max(0, z)$$



Activation Functions

Given a non-activated output $z = w^\top x + b$, then $\sigma(z) = \dots$

- $\tanh(z)$

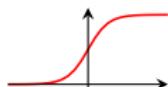


- The ReLU function:



- The logistic function (also called sigmoid function):

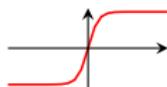
$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



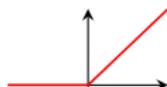
Activation Functions

Given a non-activated output $z = w^\top x + b$, then $\sigma(z) = \dots$

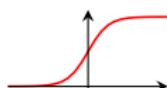
- $\tanh(z)$



- The ReLU function:



- The logistic function (also called sigmoid function):

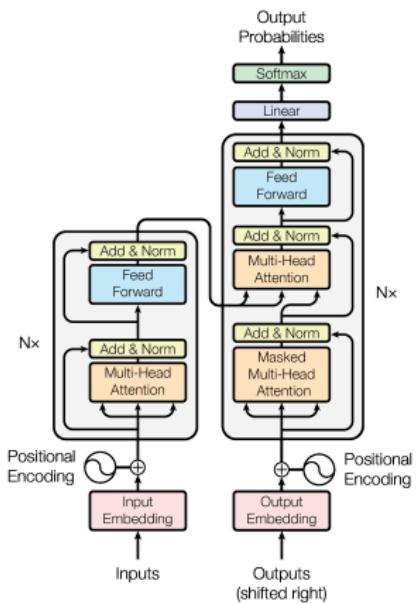
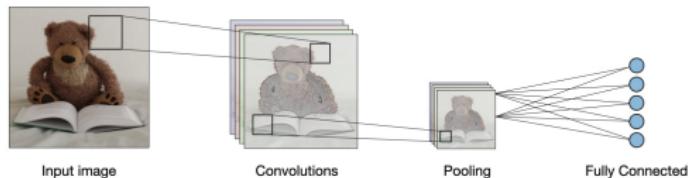


- The softmax function:

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_{j=1}^m \exp(z_j)}$$

for $z = Wx + b \in \mathbb{R}^m$.

Other Architectures



Loss Function

Measure of error between a prediction $f(x)$ and a ground truth y .

Goal: minimize the loss function!

Loss Function

Measure of error between a prediction $f(x)$ and a ground truth y .

Goal: minimize the loss function!

- Mean squared error for regression
 - $y, f(x) \in \mathbb{R}$

$$\mathcal{L}_{\text{MSE}}(f(x), y) = (y - f(x))^2$$

Loss Function

Measure of error between a prediction $f(x)$ and a ground truth y .

Goal: minimize the loss function!

- Mean squared error for regression
 - $y, f(x) \in \mathbb{R}$

$$\mathcal{L}_{\text{MSE}}(f(x), y) = (y - f(x))^2$$

- Cross-entropy loss for classification with C classes ($\mathcal{Y} = \{1, \dots, C\}$)
 - $y \in \mathcal{Y}$
 - $f(x) = [f(x)_1, \dots, f(x)_C] = \text{softmax}([z_1, \dots, z_C])$

Loss Function

Measure of error between a prediction $f(x)$ and a ground truth y .

Goal: minimize the loss function!

- Mean squared error for regression
 - $y, f(x) \in \mathbb{R}$

$$\mathcal{L}_{\text{MSE}}(f(x), y) = (y - f(x))^2$$

- Cross-entropy loss for classification with C classes ($\mathcal{Y} = \{1, \dots, C\}$)
 - $y \in \mathcal{Y}$
 - $f(x) = [f(x)_1, \dots, f(x)_C] = \text{softmax}([z_1, \dots, z_C])$

$$\mathcal{L}_{\text{CE}}(f(x), y) = -\log f(x)_y$$

Loss Function

Measure of error between a prediction $f(x)$ and a ground truth y .

Goal: minimize the loss function!

- Mean squared error for regression

- $y, f(x) \in \mathbb{R}$

$$\mathcal{L}_{\text{MSE}}(f(x), y) = (y - f(x))^2$$

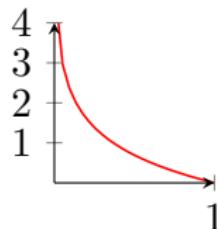
- Cross-entropy loss for classification with C classes

$$(\mathcal{Y} = \{1, \dots, C\})$$

- $y \in \mathcal{Y}$

- $f(x) = [f(x)_1, \dots, f(x)_C] = \text{softmax}([z_1, \dots, z_C])$

$$\mathcal{L}_{\text{CE}}(f(x), y) = -\log f(x)_y$$



1 Introduction

2 Neural Networks

3 Training

4 Overfitting

Training Objective

Goal: minimize the loss function for all examples (pairs (x, y))!

$$\mathcal{L}(f(x), y)$$

Training Objective

Goal: minimize the loss function for all examples (pairs (x, y))!

- The only thing we control are the parameters θ .

$$\mathcal{L}(f(x; \theta), y)$$

Training Objective

Goal: minimize the loss function for all examples (pairs (x, y))!

- The only thing we control are the parameters θ .

$$\mathbb{E}_{(x,y) \in \mathcal{D}} \mathcal{L}(f(x; \theta), y)$$

Training Objective

Goal: minimize the loss function for all examples (pairs (x, y))!

- The only thing we control are the parameters θ .

$$J(\theta) = \mathbb{E}_{(x,y) \in \mathcal{D}} \mathcal{L}(f(x; \theta), y)$$

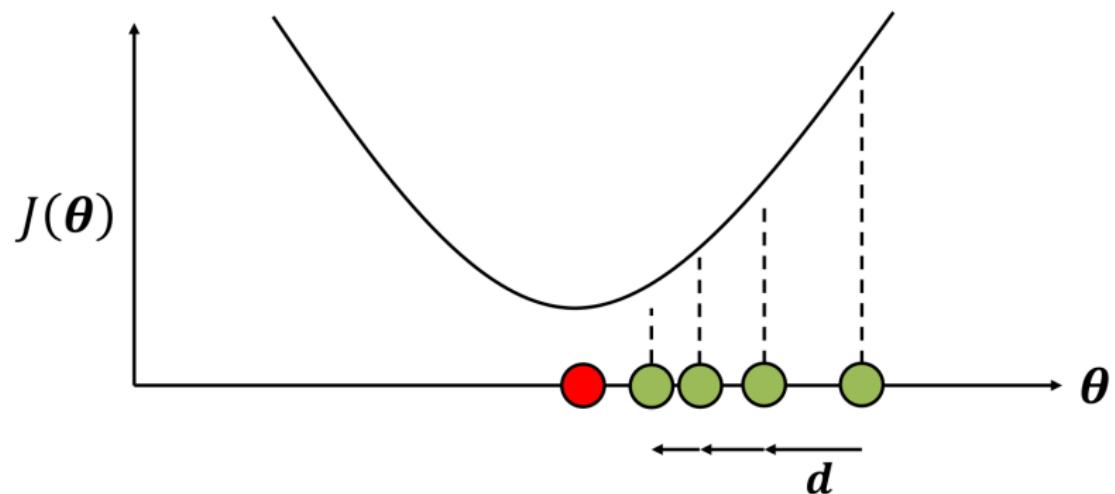
Training Objective

Goal: minimize the loss function for all examples (pairs (x, y))!

- The only thing we control are the parameters θ .

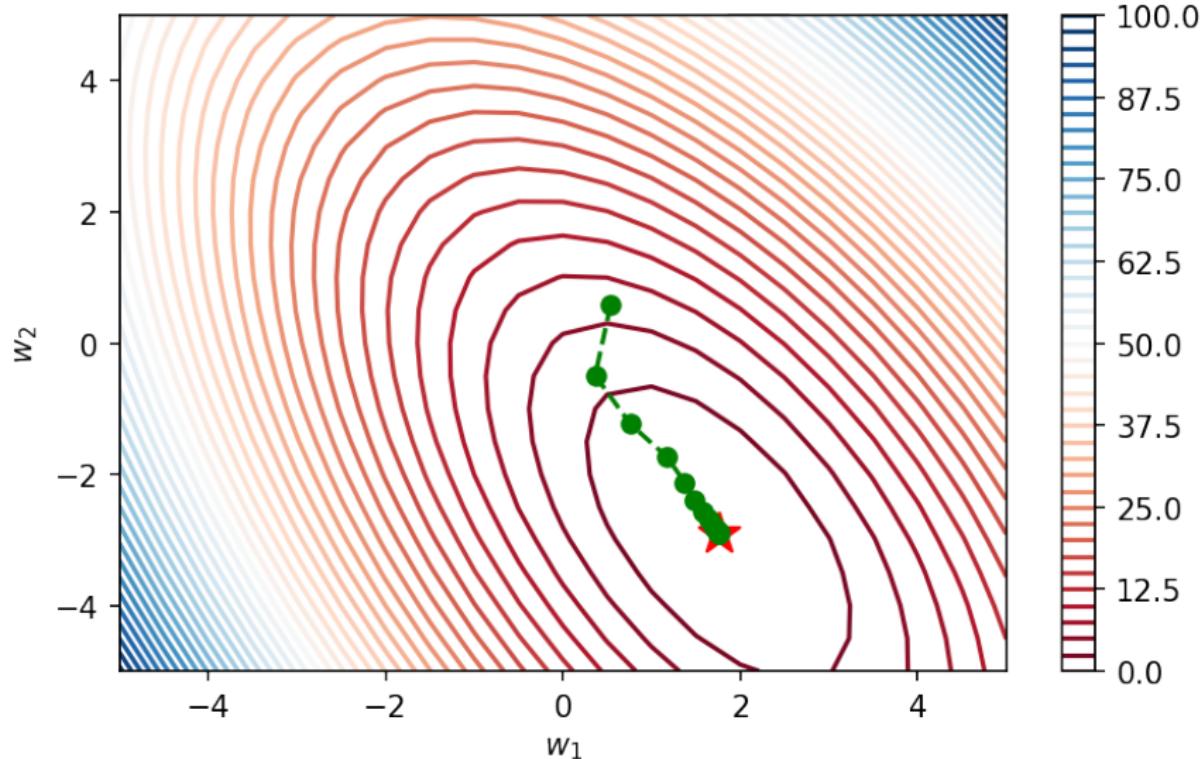
$$\begin{aligned} J(\theta) &= \mathbb{E}_{(x,y) \in \mathcal{D}} \mathcal{L}(f(x; \theta), y) \\ &\approx \frac{1}{N} \sum_{i=1}^N \mathcal{L}(f(x_i; \theta), y_i) \end{aligned}$$

Gradient Descent



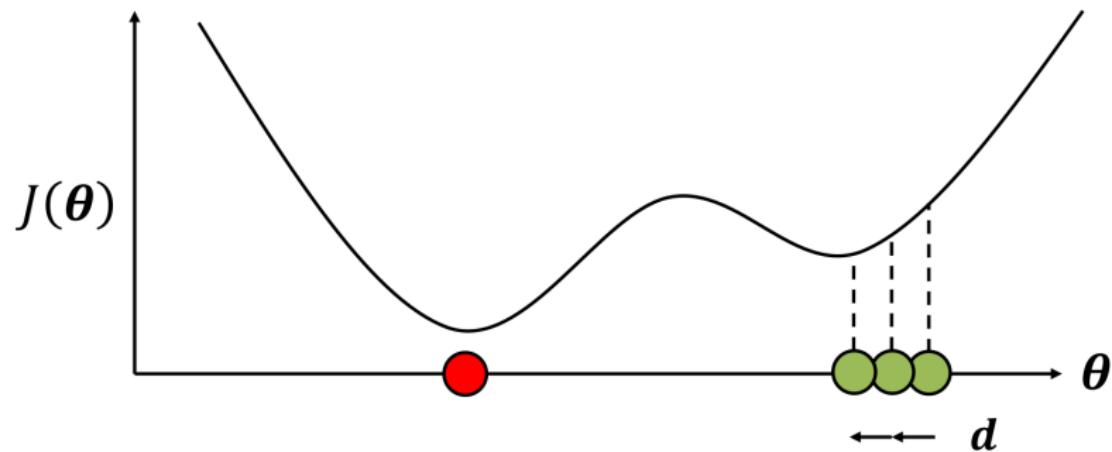
From Université Laval GLO-7030 by Ludovic Trottier

Gradient Descent



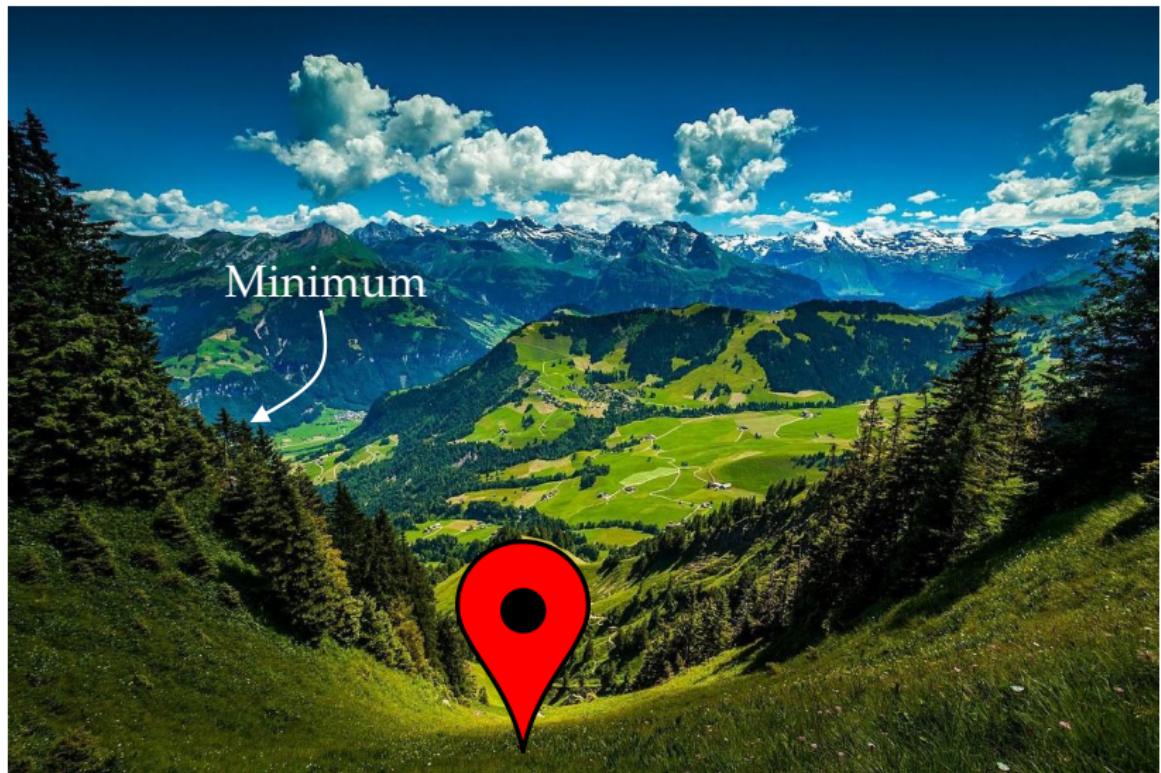
From Université Laval GLO-7030 by Pascal Germain

Gradient Descent For Deep Neural Networks



From Université Laval GLO-7030 by Ludovic Trottier

Gradient Descent For Deep Neural Networks



Adapted by Philippe Giguère for Université Laval GLO-7030 from Standford CS231N

Gradient Descent For Deep Neural Networks

Réalité : trouver le fond de la vallée embrumée, à tâtons



Adapted by Philippe Giguère for Université Laval GLO-7030 from Standford CS231N

Computation of Gradient

Using "**backpropagation**", we compute the **partial derivative** of each parameter **with respect to the loss**. The vector containing all partial derivatives is called the **gradient**.

$$d = \nabla_{\theta} J(\theta) = \left[\frac{\partial J(\theta)}{\partial \theta_1}, \dots, \frac{\partial J(\theta)}{\partial \theta_n} \right]$$

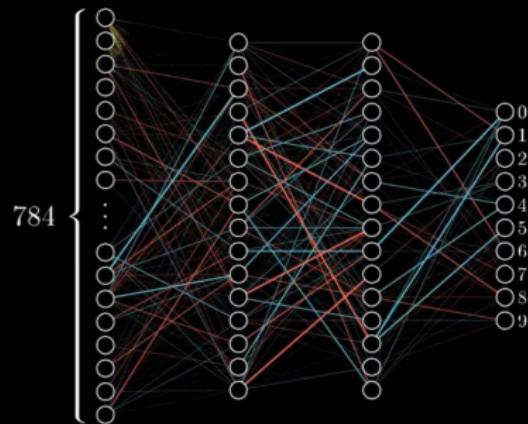
To update the parameters:

$$\theta \leftarrow \theta - \epsilon d$$

ϵ is called the learning rate.

Gradient

Training in
progress. . .



Extrait de <https://youtu.be/IHZwWFHwa-w> 3Blue1Brown

Optimization Algorithms

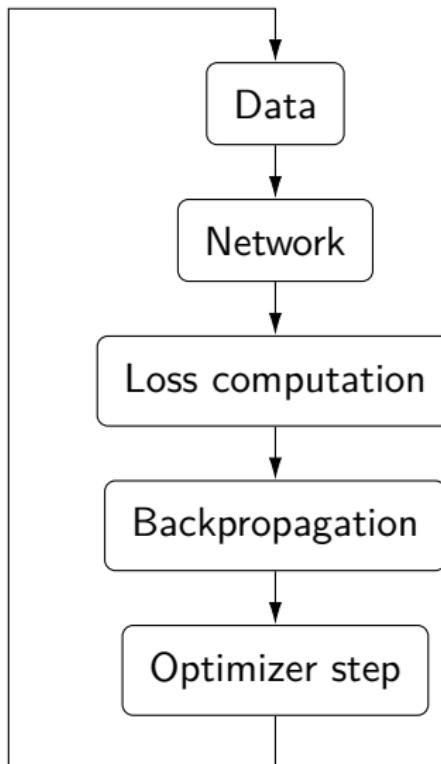
Simplest update rule:

$$\theta \leftarrow \theta - \epsilon d$$

Many types exist:

- **SGD**
- SGD with momentum
- SGD with momentum Nesterov
- Adagrad
- RMSprop
- **Adam**

Training Procedure



Training Procedure

procedure TRAIN($f(\cdot; \theta)$, S)

input: Neural network f parameterized by θ

input: Dataset $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$

end procedure

Training Procedure

```
procedure TRAIN( $f(\cdot; \theta)$ ,  $S$ )
    input: Neural network  $f$  parameterized by  $\theta$ 
    input: Dataset  $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$ 
    for  $n$  epochs do
        ...
    end for
end procedure
```

Training Procedure

```
procedure TRAIN( $f(\cdot; \theta)$ ,  $S$ )
    input: Neural network  $f$  parameterized by  $\theta$ 
    input: Dataset  $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$ 
    for  $n$  epochs do
        Let  $S' = S$ 
        while  $S' \neq \emptyset$  do
            Draw a batch  $B \subseteq S'$  of  $b$  examples.
            ...
        end while
    end for
end procedure
```

Training Procedure

```
procedure TRAIN( $f(\cdot; \theta)$ ,  $S$ )
    input: Neural network  $f$  parameterized by  $\theta$ 
    input: Dataset  $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$ 
    for  $n$  epochs do
        Let  $S' = S$ 
        while  $S' \neq \emptyset$  do
            Draw a batch  $B \subseteq S'$  of  $b$  examples.
             $\ell = \frac{1}{b} \sum_{(x,y) \in B} \mathcal{L}(f(x; \theta), y)$ 
        end while
    end for
end procedure
```

Training Procedure

```
procedure TRAIN( $f(\cdot; \theta)$ ,  $S$ )
    input: Neural network  $f$  parameterized by  $\theta$ 
    input: Dataset  $S = \{(x_1, y_1), \dots, (x_N, y_N)\}$ 
    for  $n$  epochs do
        Let  $S' = S$ 
        while  $S' \neq \emptyset$  do
            Draw a batch  $B \subseteq S'$  of  $b$  examples.
             $\ell = \frac{1}{b} \sum_{(x,y) \in B} \mathcal{L}(f(x; \theta), y)$ 
             $d = \nabla_{\theta} \ell$ 
            Update  $\theta$  with  $d$  using chosen optimizer
        end while
    end for
end procedure
```

1 Introduction

2 Neural Networks

3 Training

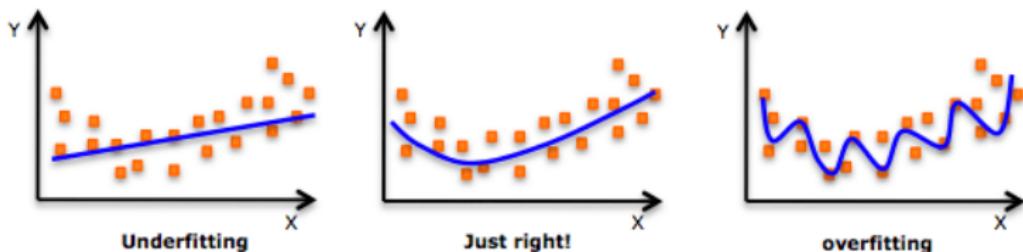
4 Overfitting

Neural Networks are Powerful

- Universal Approximation Theorem

Neural Networks are Powerful

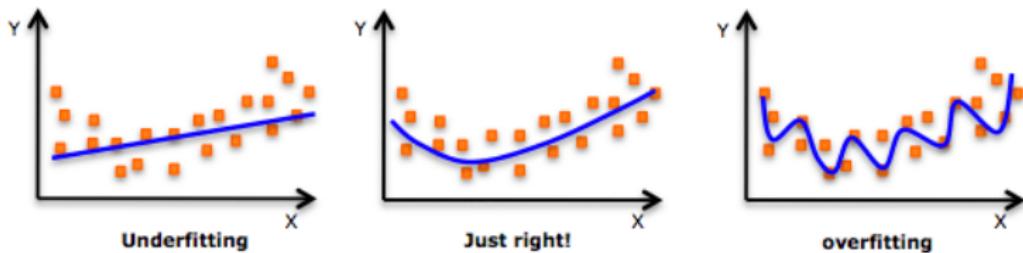
- Universal Approximation Theorem
- ... is a *blessing!*



An example of overfitting, underfitting and a model that's "just right!"

Neural Networks are Powerful

- Universal Approximation Theorem
- ... is a *blessing!*
- ... and a *curse!*



An example of overfitting, underfitting and a model that's "just right!"

Neural Networks are Powerful

- Universal Approximation Theorem
- ... is a *blessing!*
- ... and a *curse!*
- How can we avoid overfitting?

Regularization

Add a penalty term to the loss¹:

$$\mathcal{L}(f(x), y) = \mathcal{L}_{\text{MLE}}(f(x), y) + \lambda \Omega(f)$$

¹MLE stands for Maximum Likelihood Estimation.

Regularization

Add a penalty term to the loss¹:

$$\mathcal{L}(f(x), y) = \mathcal{L}_{\text{MLE}}(f(x), y) + \lambda \Omega(f)$$

- L2 regularization: $\Omega(f) = \|\theta\|_2^2$, where θ are the learnable parameters of the neural network f .

¹MLE stands for Maximum Likelihood Estimation.

Regularization

Add a penalty term to the loss¹:

$$\mathcal{L}(f(x), y) = \mathcal{L}_{\text{MLE}}(f(x), y) + \lambda \Omega(f)$$

- L2 regularization: $\Omega(f) = \|\theta\|_2^2$, where θ are the learnable parameters of the neural network f .
- L1 regularization (or LASSO): $\Omega(f) = \|\theta\|_1$

¹MLE stands for Maximum Likelihood Estimation.

Regularization

Add a penalty term to the loss¹:

$$\mathcal{L}(f(x), y) = \mathcal{L}_{\text{MLE}}(f(x), y) + \lambda \Omega(f)$$

- L2 regularization: $\Omega(f) = \|\theta\|_2^2$, where θ are the learnable parameters of the neural network f .
- L1 regularization (or LASSO): $\Omega(f) = \|\theta\|_1$
- etc.

¹MLE stands for Maximum Likelihood Estimation.

Dropout

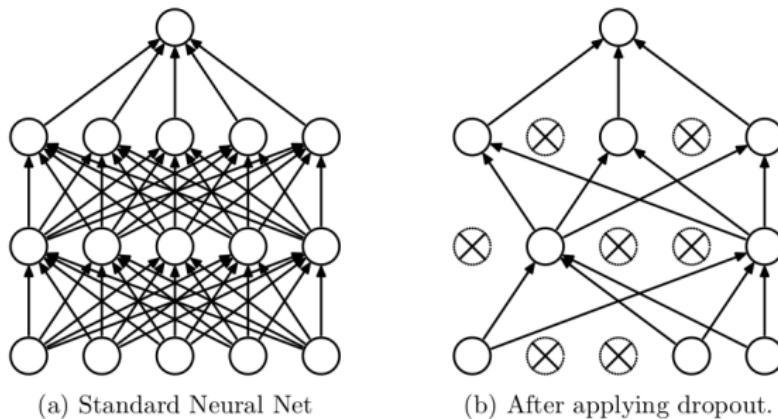
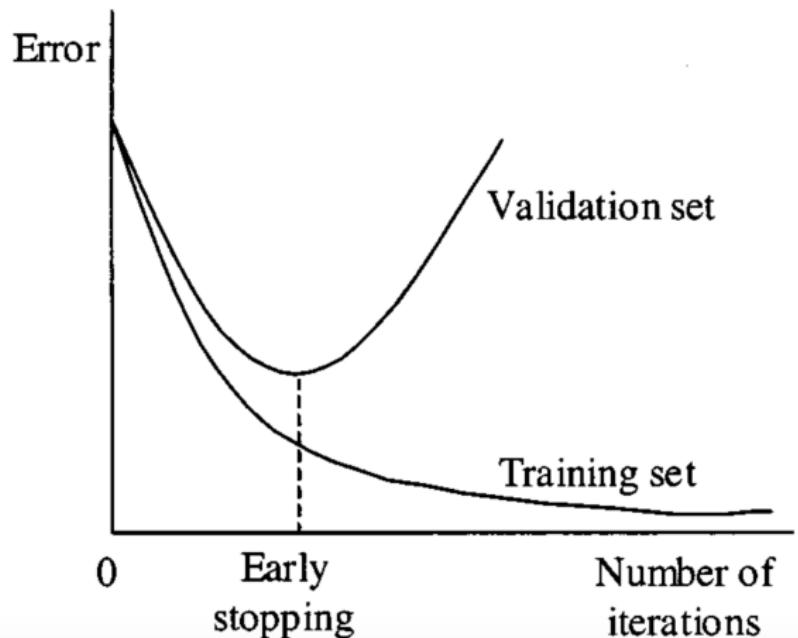


Figure 1: Dropout Neural Net Model. **Left:** A standard neural net with 2 hidden layers. **Right:** An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

Early Stopping

Extract a validation set \mathcal{V} from training set. Use it to assess generalization potential.



Deep Learning Libraries



TensorFlow

PyTorch

Deep Learning Libraries



TensorFlow



PyTorch Demo

Liens de référence

- Cours GLO-4030/7030 Apprentissage par réseaux de neurones profonds:
 - Slides: <https://ulaval-damas.github.io/glo4030/>
 - Laboratoire:
<https://github.com/ulaval-damas/glo4030-labs>
- Vidéos du cours CS231n:
<https://www.youtube.com/watch?v=vT1JzLTH4G4&list=PLC1qU-LWwrF64f4QKQT-Vg5Wr4qEE1Zxk>
- Tutoriels et documentation de PyTorch
 - <https://pytorch.org/tutorials/> (pas tout le temps les meilleures pratiques)
 - <https://pytorch.org/docs/stable/index.html>
- Documentation de Poutyne: <https://poutyne.org/>

The End.

Questions?

Bibliography I

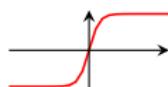
Activation Functions

Given a non-activated output $z = w^\top x + b$, then $\sigma(z) = \dots$

Activation Functions

Given a non-activated output $z = w^\top x + b$, then $\sigma(z) = \dots$

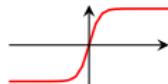
- $\tanh(z)$



Activation Functions

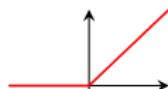
Given a non-activated output $z = w^\top x + b$, then $\sigma(z) = \dots$

- $\tanh(z)$



- The ReLU function:

$$\text{ReLU}(z) = \max(0, z)$$



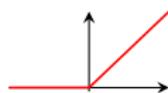
Activation Functions

Given a non-activated output $z = w^\top x + b$, then $\sigma(z) = \dots$

- $\tanh(z)$

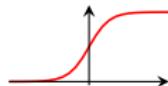


- The ReLU function:



- The logistic function (also called sigmoid function):

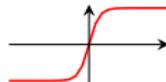
$$\sigma(z) = \frac{1}{1 + \exp(-z)}$$



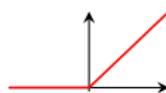
Activation Functions

Given a non-activated output $z = w^\top x + b$, then $\sigma(z) = \dots$

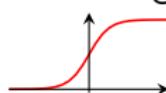
- $\tanh(z)$



- The ReLU function:



- The logistic function (also called sigmoid function):



- The softmax function:

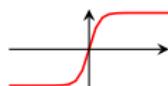
$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_{j=1}^m \exp(z_j)}$$

for $z = Wx + b \in \mathbb{R}^m$.

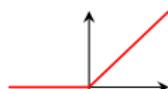
Activation Functions

Given a non-activated output $z = w^\top x + b$, then $\sigma(z) = \dots$

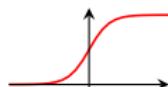
- $\tanh(z)$



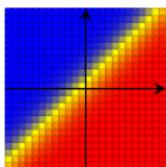
- The ReLU function:



- The logistic function (also called sigmoid function):



- The softmax function:



$$\text{softmax}(z)_1 = \frac{\exp(x)}{\exp(x)+\exp(y)} \text{ for } z = [x, y]$$

Loss Function

Measure of error between a prediction $\hat{y} = f(x)$ and a ground truth y .

Goal: minimize the loss function!

- Squared error (SE) for regression:

$$\mathcal{L}_{\text{SE}}(\hat{y}, y) = (\hat{y} - y)^2$$

for $\hat{y}, y \in \mathbb{R}$.

Loss Function

Measure of error between a prediction $\hat{y} = f(x)$ and a ground truth y .

Goal: minimize the loss function!

- Cross-entropy loss for classification with C classes ($\mathcal{Y} = \{1, \dots, C\}$)

Loss Function

Measure of error between a prediction $\hat{y} = f(x)$ and a ground truth y .

Goal: minimize the loss function!

- Cross-entropy loss for classification with C classes ($\mathcal{Y} = \{1, \dots, C\}$)
 - $y \in \mathcal{Y}$
 - $\hat{y} = [\hat{y}_1, \dots, \hat{y}_C] = \text{softmax}([z_1, \dots, z_C])$

Loss Function

Measure of error between a prediction $\hat{y} = f(x)$ and a ground truth y .

Goal: minimize the loss function!

- Cross-entropy loss for classification with C classes ($\mathcal{Y} = \{1, \dots, C\}$)
 - $y \in \mathcal{Y}$
 - $\hat{y} = [\hat{y}_1, \dots, \hat{y}_C] = \text{softmax}([z_1, \dots, z_C])$

Based on the cross entropy:

$$H(q, p) = - \sum_{c \in \mathcal{Y}} p(c) \log q(c)$$

Loss Function

Measure of error between a prediction $\hat{y} = f(x)$ and a ground truth y .

Goal: minimize the loss function!

- Cross-entropy loss for classification with C classes ($\mathcal{Y} = \{1, \dots, C\}$)
 - $y \in \mathcal{Y}$
 - $\hat{y} = [\hat{y}_1, \dots, \hat{y}_C] = \text{softmax}([z_1, \dots, z_C])$

Based on the cross entropy:

$$H(q, p) = - \sum_{c \in \mathcal{Y}} p(c) \log q(c)$$

where $p(c) = \mathbb{1}(y = c)$ and $q(c) = \hat{y}_c$.

Loss Function

Measure of error between a prediction $\hat{y} = f(x)$ and a ground truth y .

Goal: minimize the loss function!

- Cross-entropy loss for classification with C classes ($\mathcal{Y} = \{1, \dots, C\}$)
 - $y \in \mathcal{Y}$
 - $\hat{y} = [\hat{y}_1, \dots, \hat{y}_C] = \text{softmax}([z_1, \dots, z_C])$

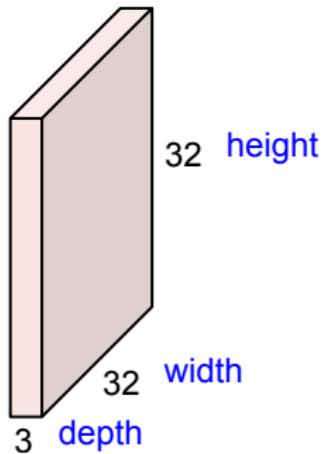
Based on the cross entropy:

$$H(q, p) = - \sum_{c \in \mathcal{Y}} p(c) \log q(c)$$

where $p(c) = \mathbb{1}(y = c)$ and $q(c) = \hat{y}_c$. This simplifies to:

$$\mathcal{L}_{\text{CE}}(\hat{y}, y) = - \sum_{c=1}^C \mathbb{1}(y = c) \log \hat{y}_c = - \log \hat{y}_y$$

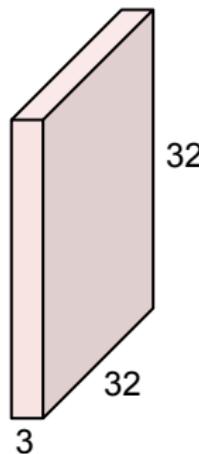
Convolution Layer



Slides of Fei-Fei Li, Ranjay Krishna, Danfei Xu; Standford CS231n.

Convolution Layer

32x32x3 image

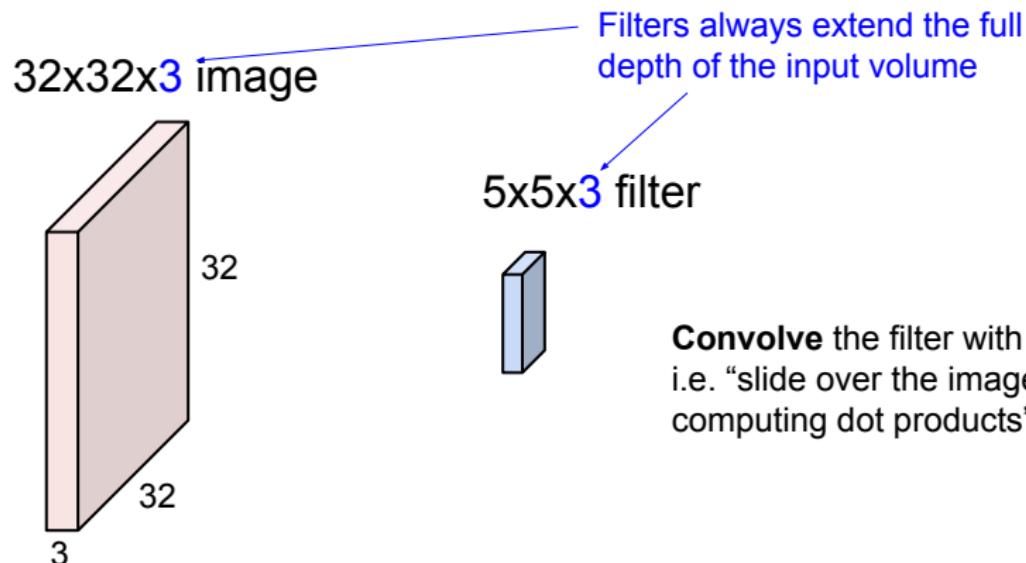


5x5x3 filter



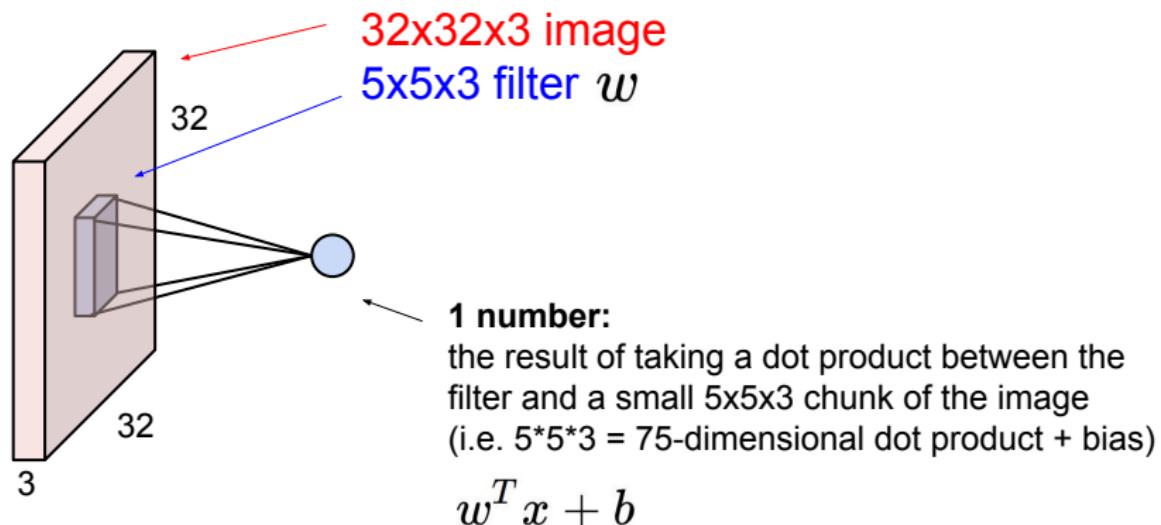
Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Convolution Layer



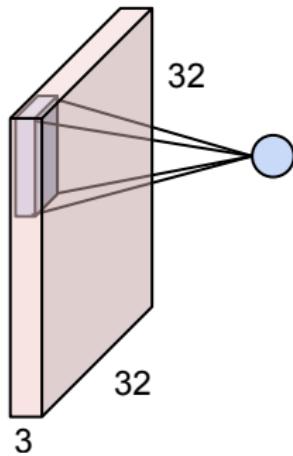
Slides of Fei-Fei Li, Ranjay Krishna, Danfei Xu; Standford CS231n.

Convolution Layer



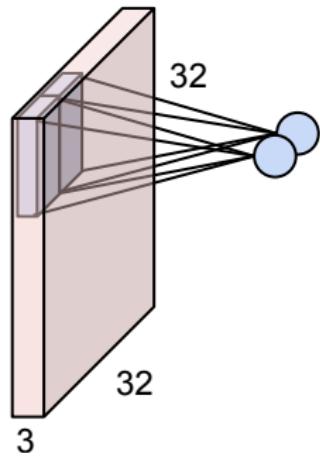
Slides of Fei-Fei Li, Ranjay Krishna, Danfei Xu; Standford CS231n.

Convolution Layer



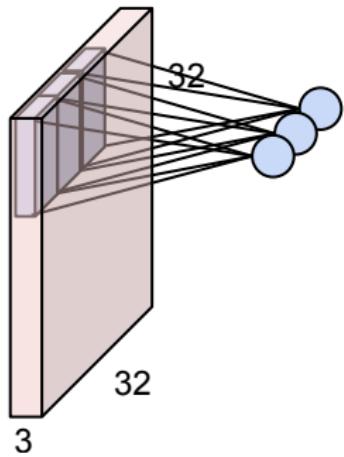
Slides of Fei-Fei Li, Ranjay Krishna, Danfei Xu; Standford CS231n.

Convolution Layer



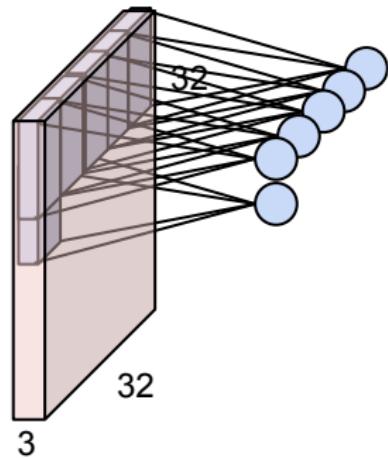
Slides of Fei-Fei Li, Ranjay Krishna, Danfei Xu; Standford CS231n.

Convolution Layer



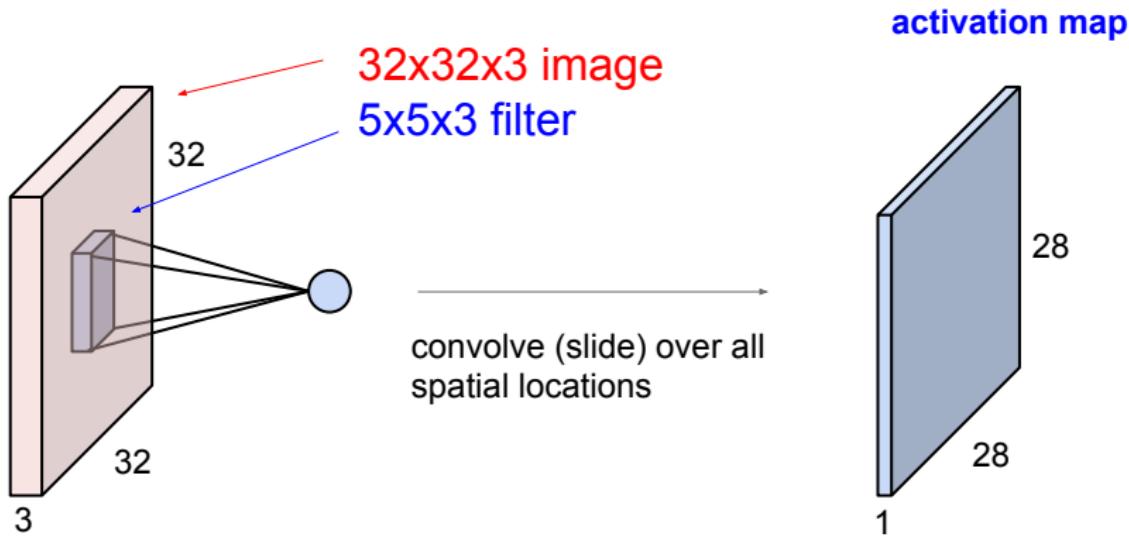
Slides of Fei-Fei Li, Ranjay Krishna, Danfei Xu; Standford CS231n.

Convolution Layer



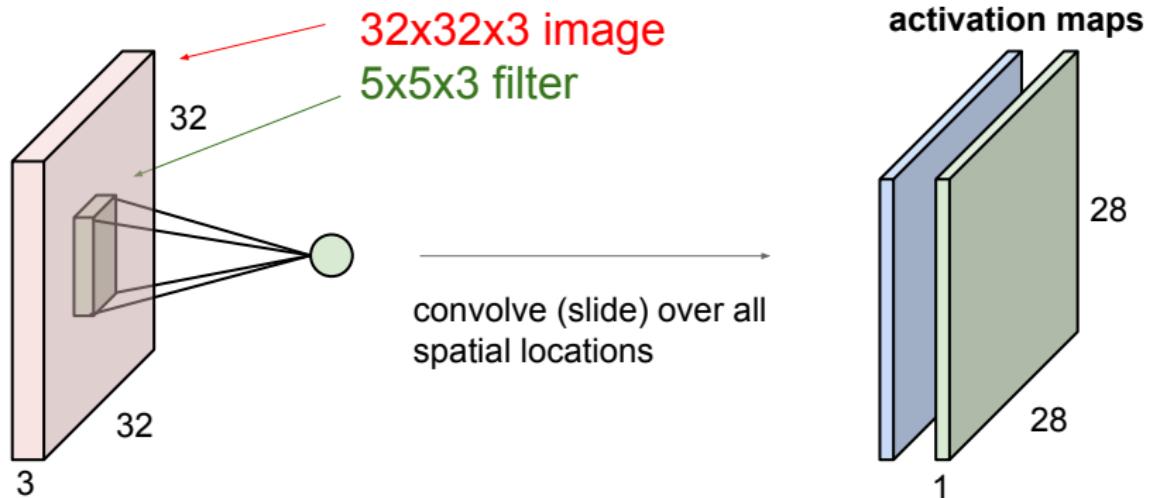
Slides of Fei-Fei Li, Ranjay Krishna, Danfei Xu; Standford CS231n.

Convolution Layer



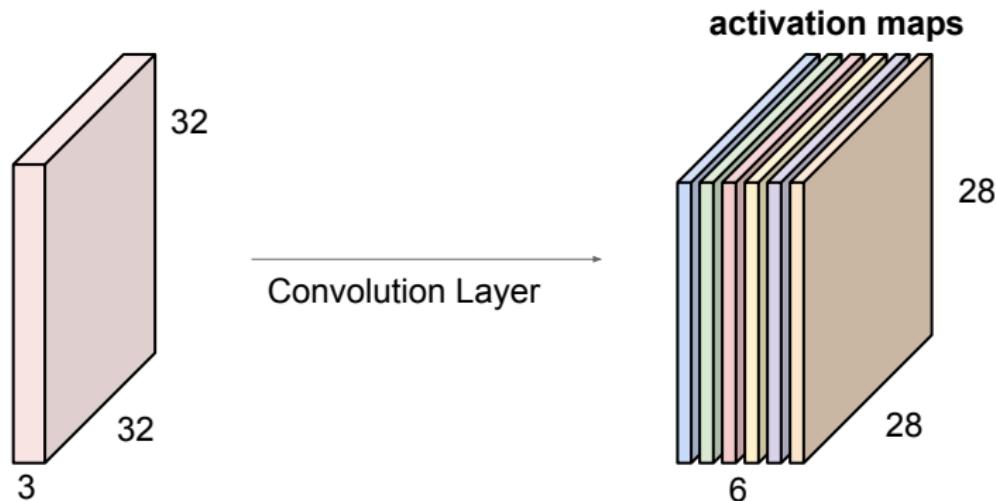
Slides of Fei-Fei Li, Ranjay Krishna, Danfei Xu; Standford CS231n.

Convolution Layer



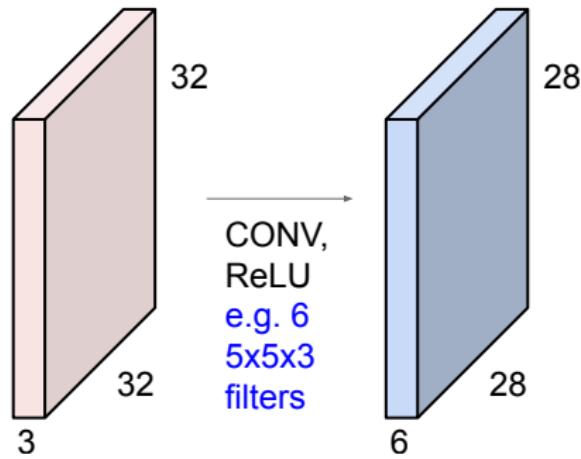
Slides of Fei-Fei Li, Ranjay Krishna, Danfei Xu; Standford CS231n.

Convolution Layer



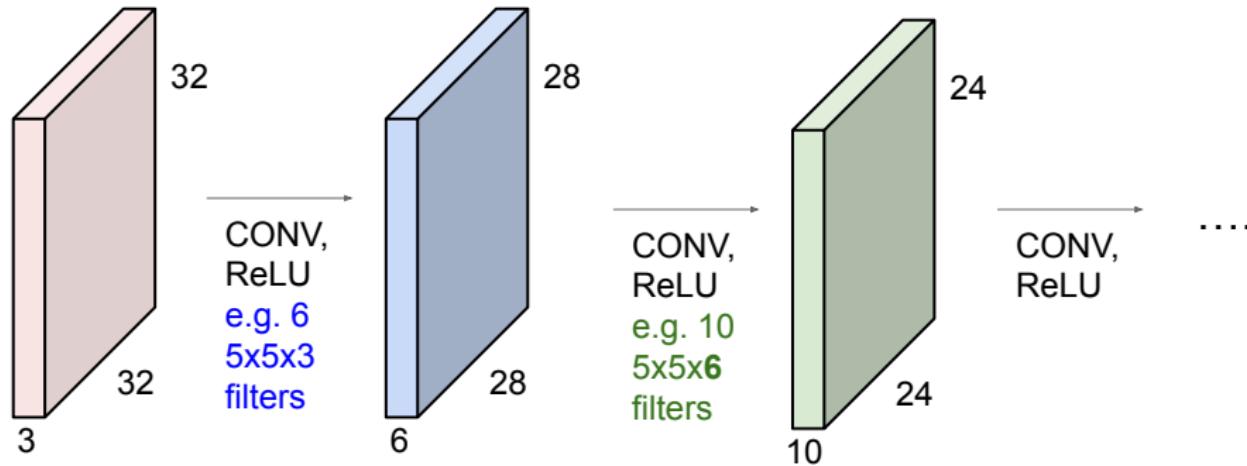
Slides of Fei-Fei Li, Ranjay Krishna, Danfei Xu; Standford CS231n.

Convolution Layer



Slides of Fei-Fei Li, Ranjay Krishna, Danfei Xu; Standford CS231n.

Convolution Layer



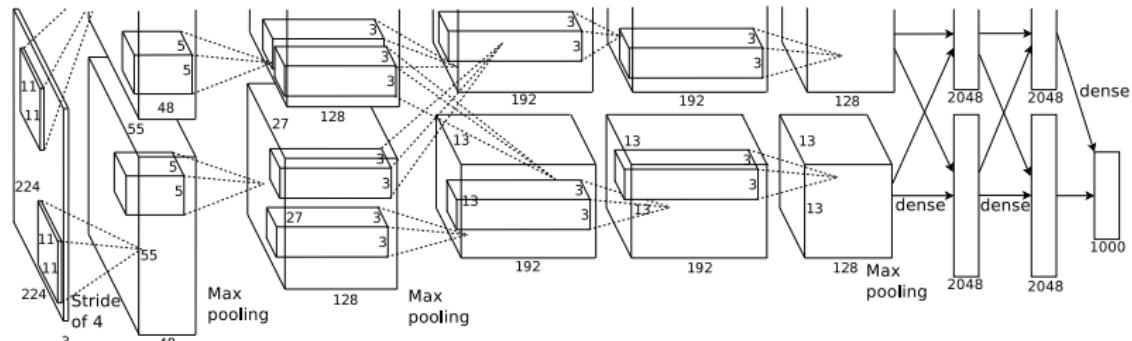
Slides of Fei-Fei Li, Ranjay Krishna, Danfei Xu; Standford CS231n.

Other Types of Layers

Many types of layers exist. Here is a few.

- Max pooling/average pooling
- Batch normalization
- Dropout

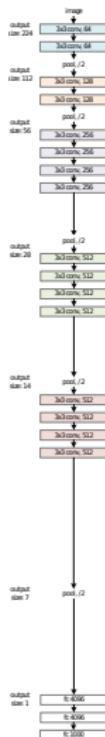
Deep Neural Network Architectures



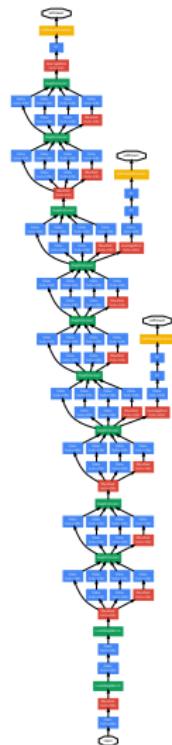
AlexNet

[Krizhevsky2012ImageNetCW, Krizhevsky2012ImageNetCW]

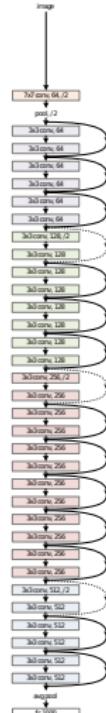
Deep Neural Network Architectures



VGG



GoogLeNet (or Inception)



ResNet

[simonyan2014very, simonyan2014very]
[szegedy2015going, szegedy2015going]
[he2016deep, he2016deep]