Aula18

DATA SCIENCE IPT

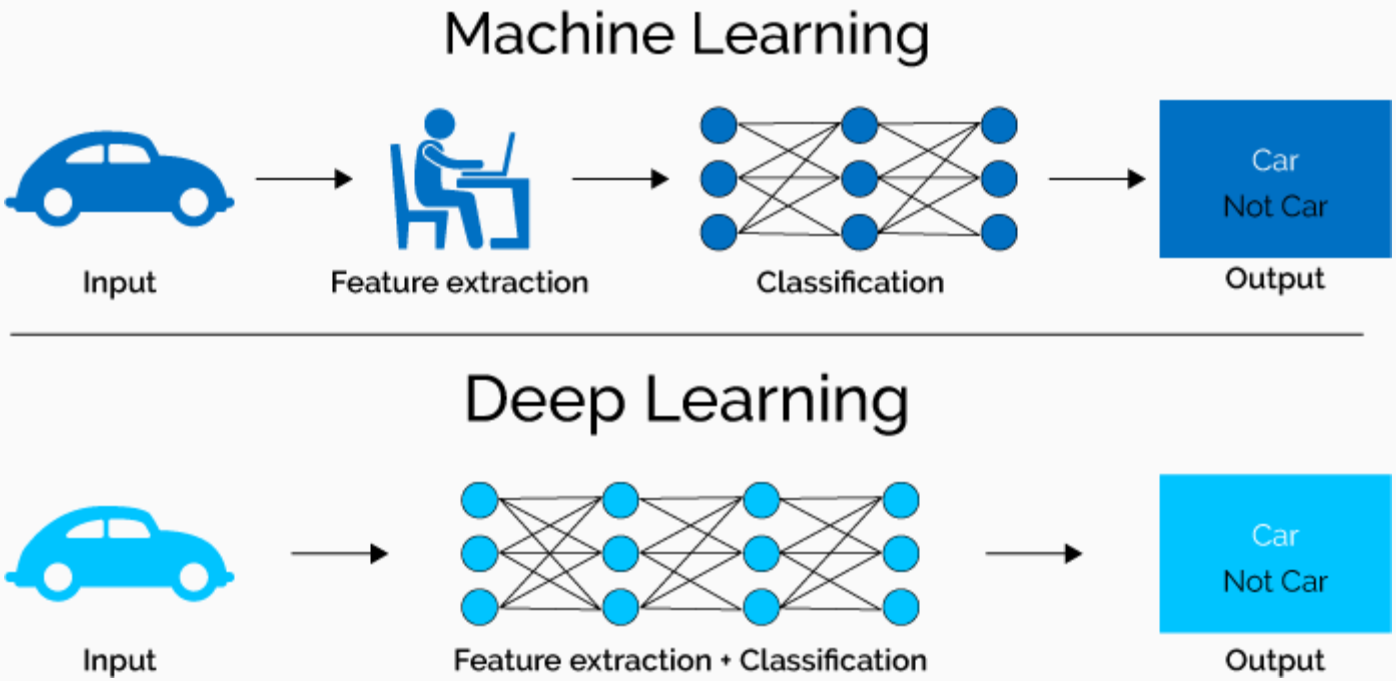
LINKED EDUCATION

V2019

DEEP LEARNING

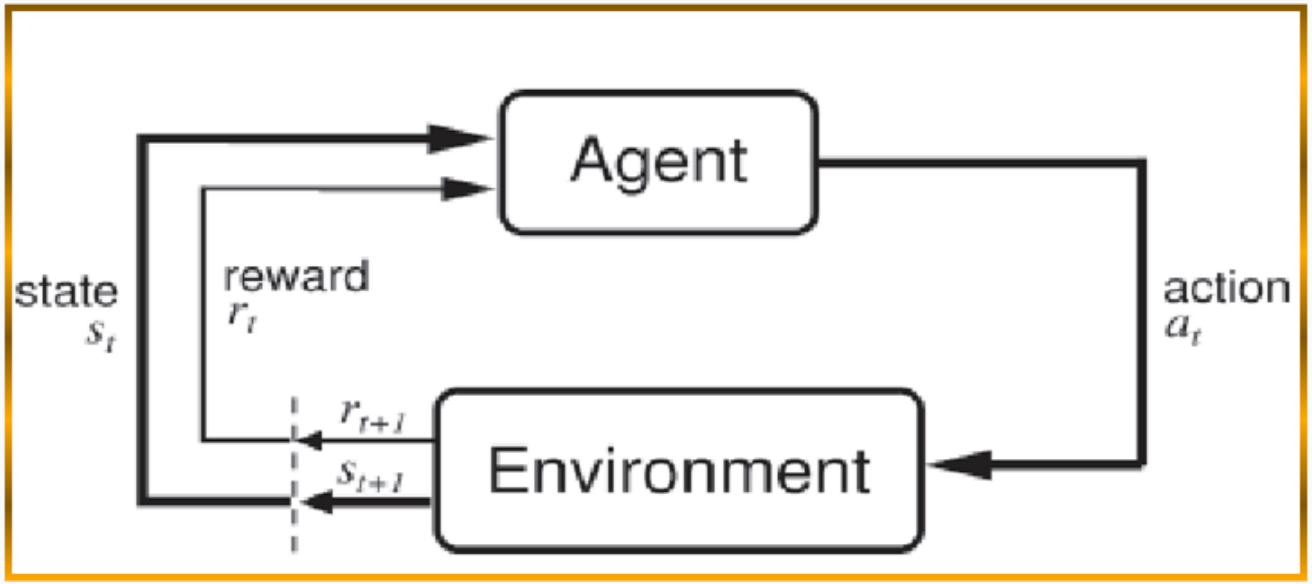
Deep Learning

Conteúdo

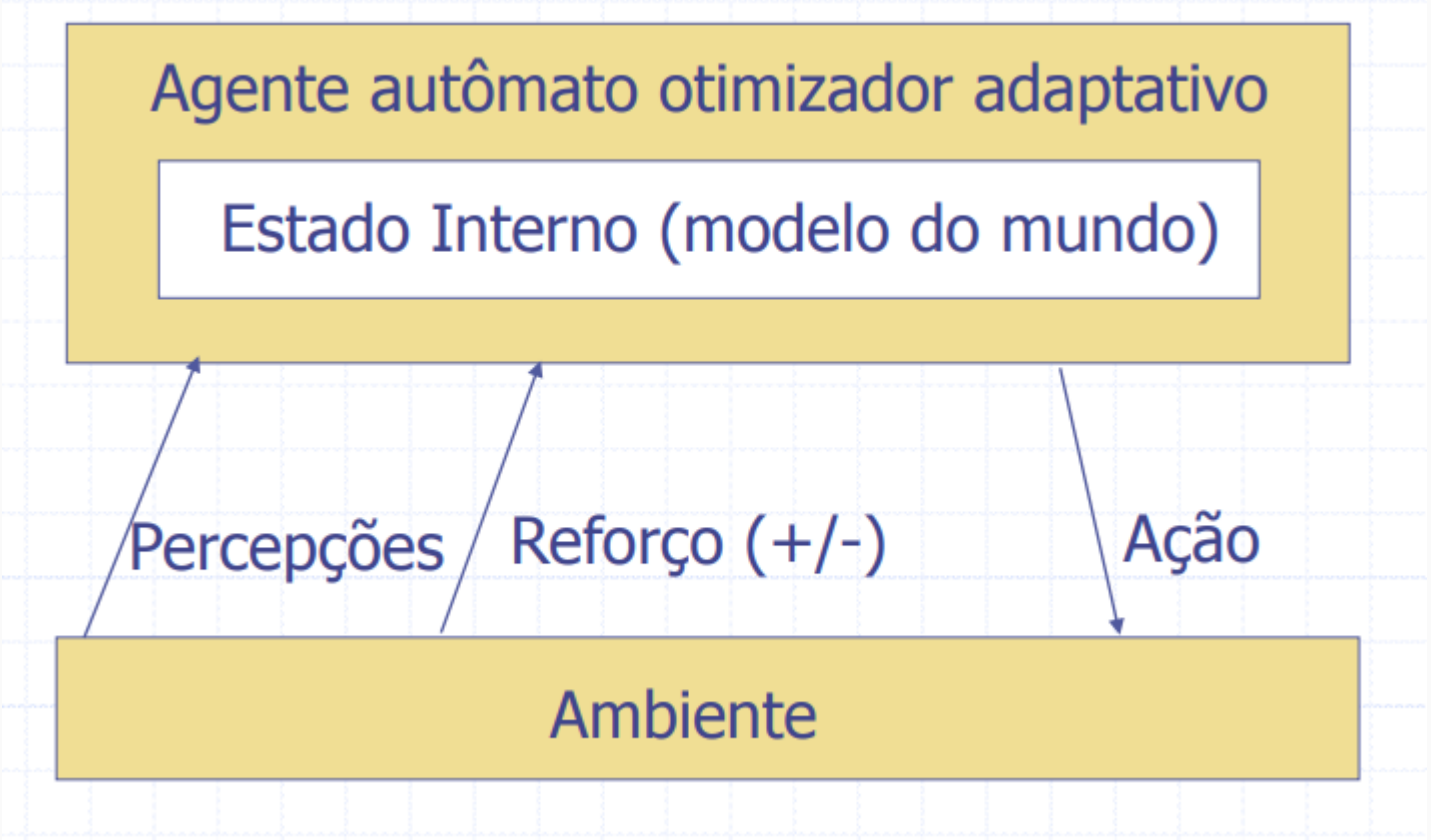


APRENDIZADO POR REFORÇO

Reinforcement Learning



Reinforcement Learning



CNN - Convolutional Neural Networks

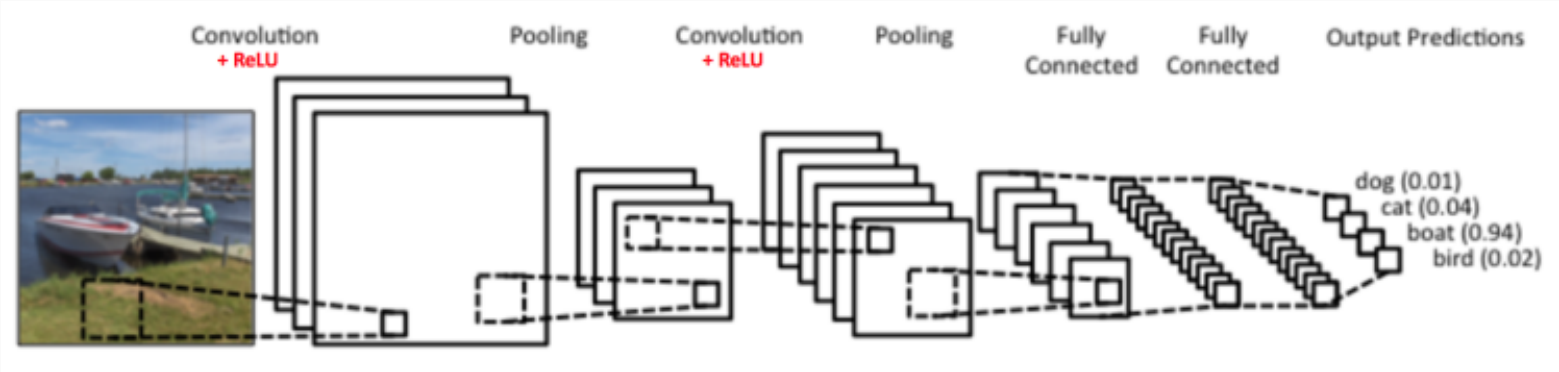
Conteúdo

CNNs são redes Feed Forward Multilayer com um arranjo arquitetural específico. As CNNs funcionam muito bem para classificação de imagens, por exemplo.

A figura abaixo traz as duas etapas típicas até chegar em uma MLP fully connected usual :

Convolution + Relu

Pooling



Fonte :https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets

CNN - Convolutional Neural Networks

Conteúdo

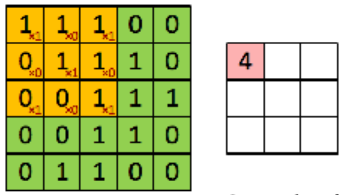
1- Convolution : O filtro vai percorrendo toda a imagem (input) e gerando o produto dos valores do input pelos do filtro. Esses valores são somados, resultando na feature já com a convolução.

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

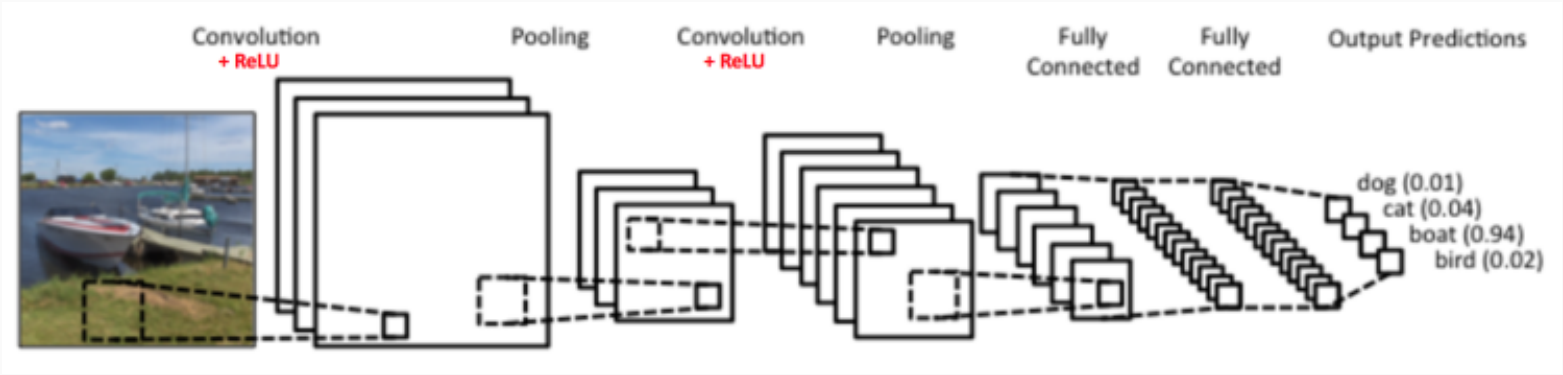
Input (image)

1	0	1
0	1	0
1	0	1

Filter (Kernel)



Animação em :
https://ujwlkarn.files.wordpress.com/2016/07/convolution_schematic.gif?w=268&h=196



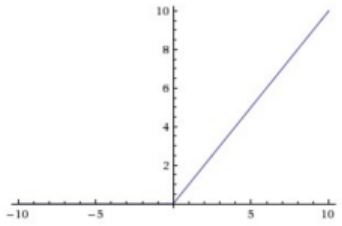
Vamos testar alguns filtros em
http://setosa.io/ev/image-kernels/

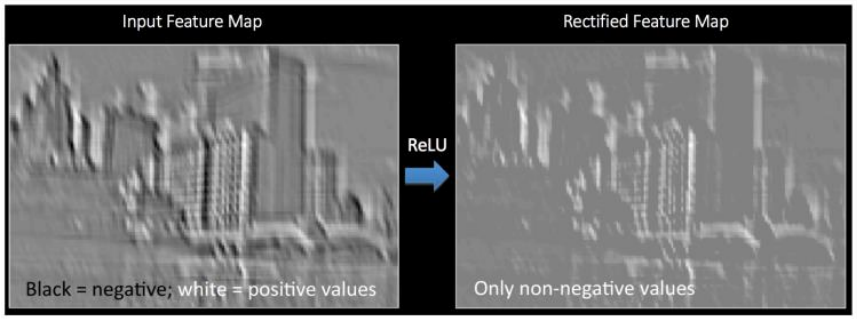
CNN - Convolutional Neural Networks

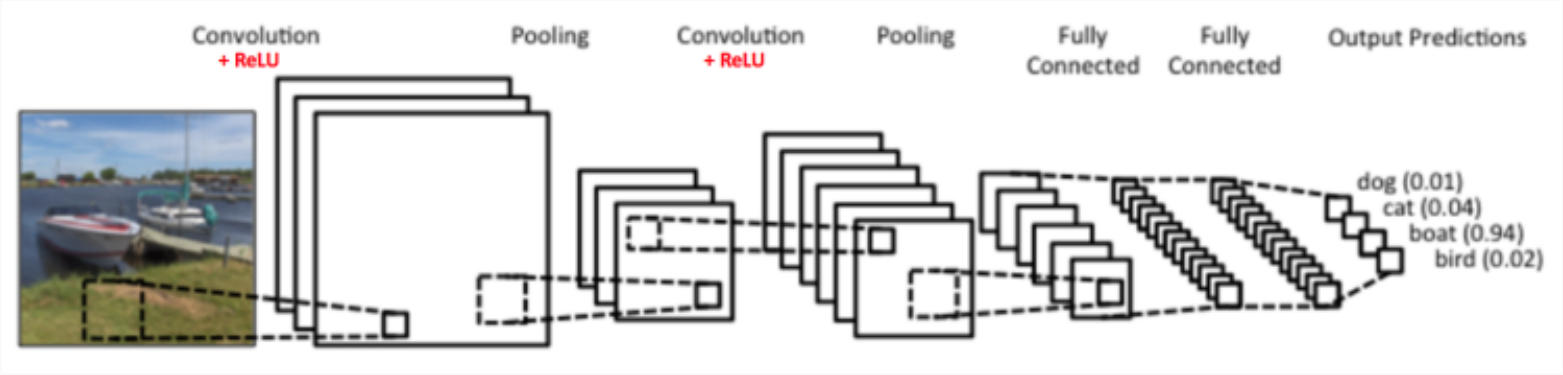
Conteúdo

1- Convolution (Relu) : Depois da convolução (que é linear), ocorre a Relu (Rectified Linear Unit), que não é linear e é dada pelo máximo entre 0 e o input (elimina valores negativos)

Output = Max(zero, Input)



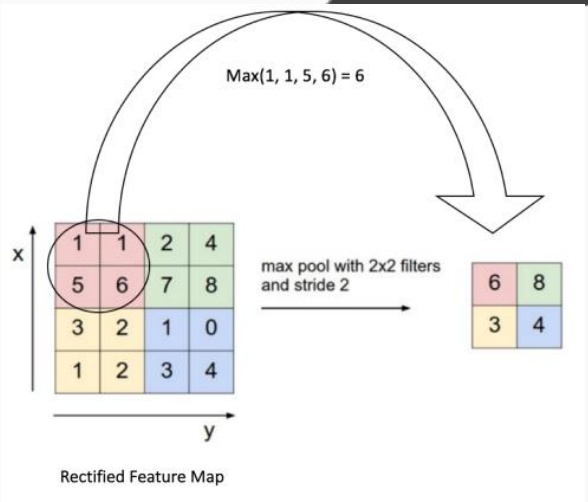


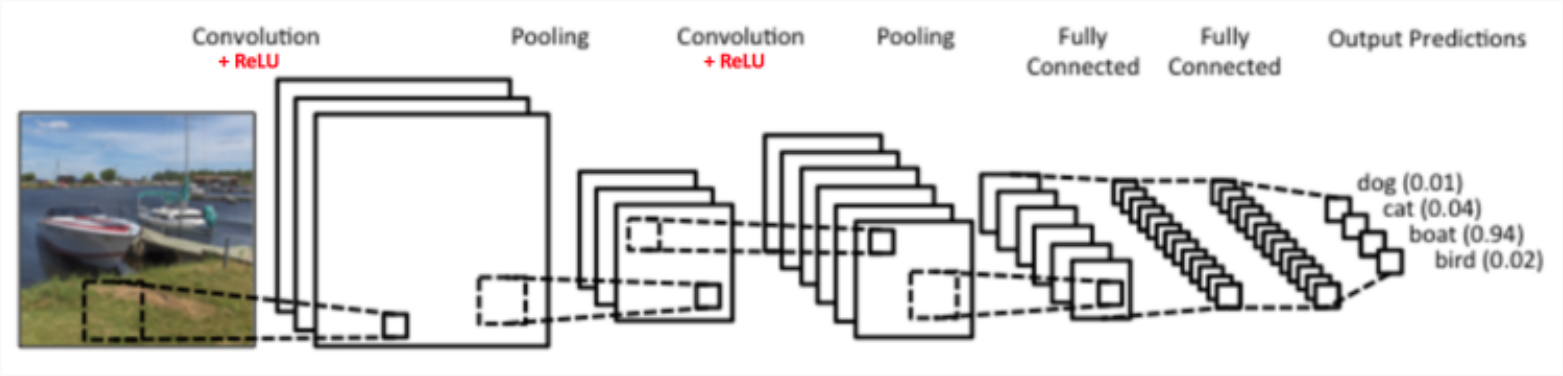


CNN - Convolutional Neural Networks

Conteúdo

2- Pooling : O Pooling (subsampling ou downsampling) reduz dimensionalmente (menos pixels na saída) por transformar um conjunto de pixels na sua “Média” ou “Máximo” por exemplo. A ideia é simplificar (menos features) sem perder informações importantes.

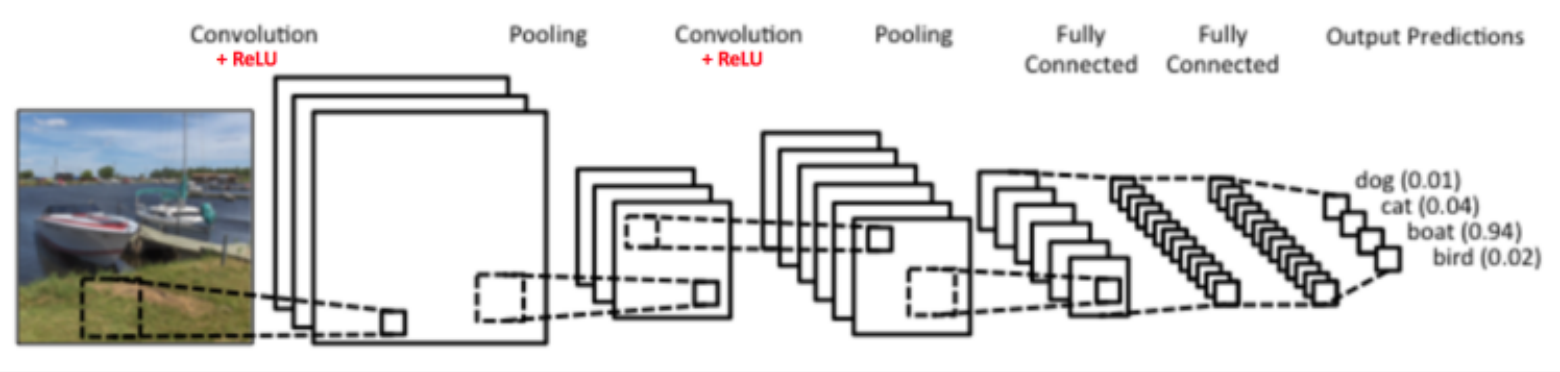




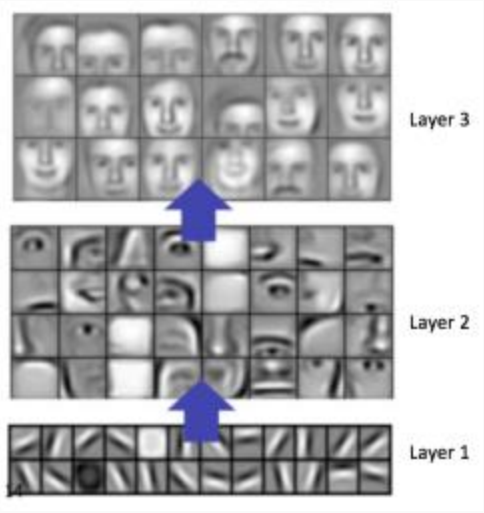
CNN - Convolutional Neural Networks

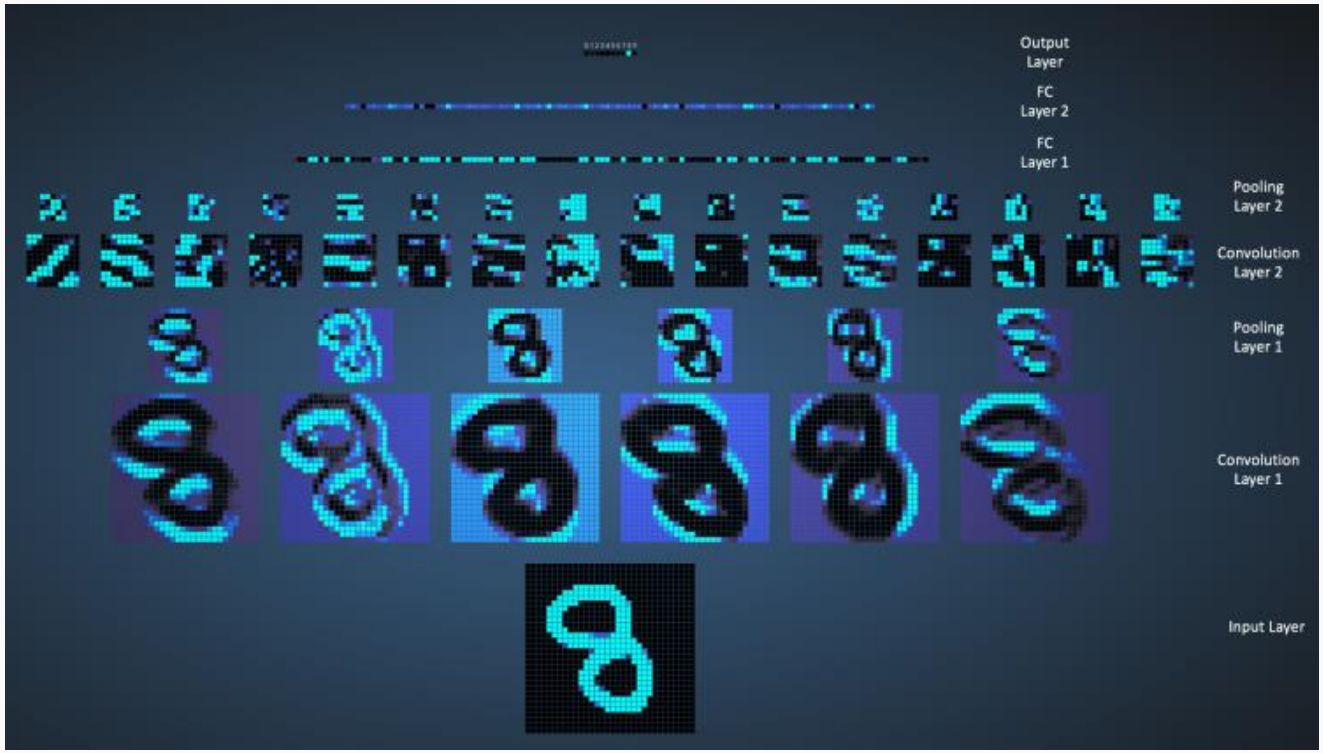
Conteúdo

Depois de passar pelas camadas de Convolution+Relu + Poolling... .as features entram em um MLP fully connected usual.



Exemplos do processamento de ConvNets





Alguns detalhes importantes :

Os pesos (incluindo os filtros de convolução) são obtidos via aprendizado.

O aprendizado pode ser realizado com backpropagation

A inspiração veio do processo de visão (Biologia).

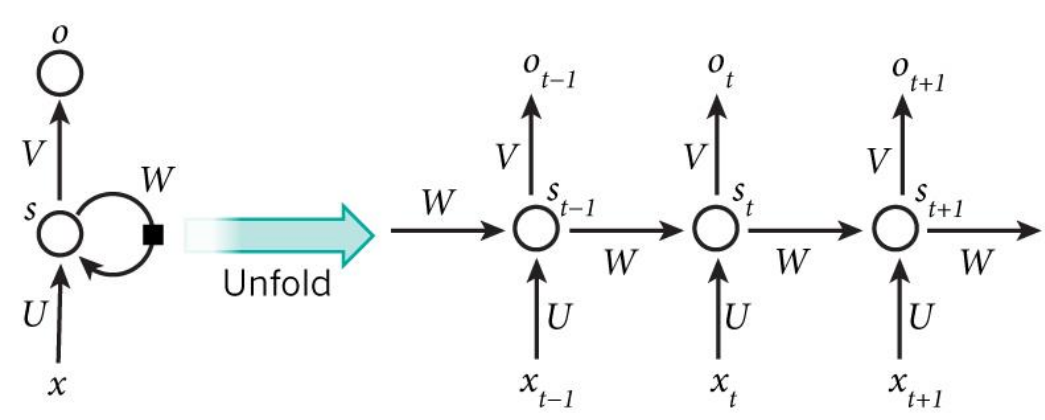
RNN – Recurrent Neural Netwoks

Conteúdo

Nas redes neurais recorrentes, além do input (x) em um tempo “t” como entrada, há também um “input extra”, vindo do tempo t-1, o “hidden state” do tempo t-1, denominado $s_{t-1.}$

Em cada t, o hidden state s é dado por $s_t=f(Ux_t+Ws_{t-1})$, com pesos “U” para a entrada x_t e peso W para o hidden state de t-1. f é uma função ativadora. O output da rede em cada t é dado pelo peso V multiplicado por s_t (com ativação posterior).

O hidden state é como uma memória, com o passado influenciando o presente (*le passé qui ne passe pas*).



RNN – Recurrent Neural Netwoks

Nas redes neurais recorrentes, os mesmos parâmetros (pesos) são utilizados em todos os processamentos ao longo do tempo...se fosse uma rede de muitas camadas (ao invés de muitas iterações no tempo), haveria muitos pesos...

As RNN são muito utilizadas no processamento de sequências (um texto, séries temporais etc.)

Pode ser que apenas o último output seja relevante em alguns casos, e também o input pode ser desnecessário em todos os “t”.

RNN – Recurrent Neural Netwoks

Nas redes neurais recorrentes, os mesmos parâmetros (pesos) são utilizados em todos os processamentos ao longo do tempo...se fosse uma rede de muitas camadas (ao invés de muitas iterações no tempo), haveria muitos pesos...

As RNN são muito utilizadas no processamento de sequências (um texto, séries temporais, Speech Recognition etc.)

Pode ser que apenas o último output seja relevante em alguns casos, e também o input pode ser desnecessário em todos os “t”.

Como treinar uma RNN? Podemos pensar uma RNN como uma rede feed-forward com muitas camadas (uma para cada tempo, com pesos iguais). Assim, poderíamos utilizar o algoritmo backpropagation. Porém seria BPTT (Backpropagation Through Time). Para sequências muito longas, porém, essa estratégia pode gerar problemas (não tratar bem dependências de longo prazo) e, para tratar esses problemas, algumas variações das RNNs foram criadas...entre elas, as LSTM.

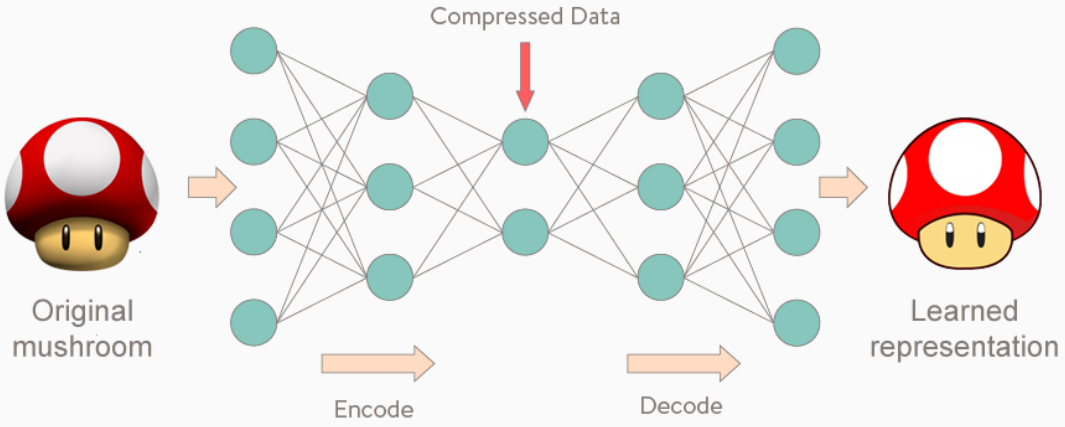
Muitos processamentos podem ser mais simples ou menos simples conforme os dados são representados. Quando nos pedem para fazer cálculos com algarismos romanos, nossa estratégia é converter os algarismos romanos em arábicos e realizar os cálculos.

Representações compartilhadas podem facilitar o reúso dos algoritmos. Por exemplo, se uma imagem ou um texto são transformados em um vetor de floats, podem, em princípio, compartilhar algoritmos que processam vetores em ML.

Autoencoders

Autoencoders são redes neurais que são treinadas para copiar o input no output. Há pelo menos duas grandes utilizações dessas redes :

a)Compactação de dados
b)Obter uma nova representação dos dados



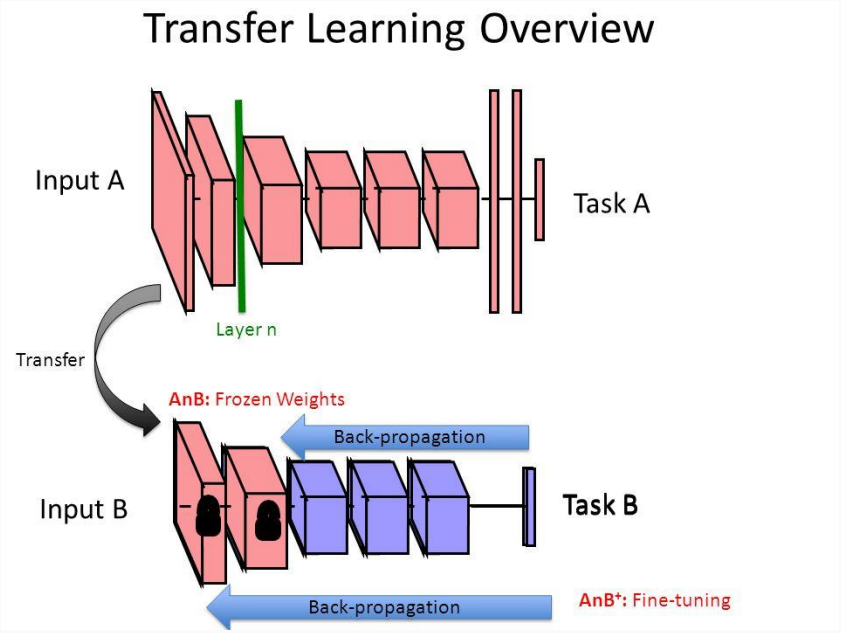
Transfer Learning

É uma técnica de aprendizado de máquina em que um modelo treinado em uma tarefa é redirecionado em uma segunda tarefa relacionada.

Exemplo : Partir de uma CNN treinada para classificar imagens em “n” classes e

1) treinar apenas a última camada (sem mexer nos pesos da CNN) com novo dataset (só de raças de cães, por exemplo)...estamos, nesse caso, reusando a “preparação das features” da CNN.

2) Treinar a última camada da CNN e também as outras, com o novo dataset (tuning)



Deep Learning na prática...bibliotecas

Biblioteca de software de código aberto para computação numérica usando grafos de fluxo de dados

Originalmente desenvolvido pela Google Brain Team para realizar a aprendizagem de máquina e pesquisa de redes neurais profundas

Genérica o suficiente para ser aplicável em uma grande variedade de outros domínios, o TensorFlow fornece um extenso conjunto de funções e classes que permitem que os usuários construam vários modelos a partir do zero.

Por que Tensorflow?

API do Python

Portabilidade: roda em uma ou mais CPUs ou GPUs em um desktop, servidor ou dispositivo móvel com uma única API

Flexibilidade: de Raspberry Pi, Android, Windows, iOS, Linux para farms de servidores

Visualização

Checkpoints (para gestão de experimentos)

Auto-diferenciação autodiff

grande comunidade

Projetos relevantes usando TensorFlow

O que é um tensor?

n-dimensional array

0-d tensor: scalar (number)

1-d tensor: vector

2-d tensor: matrix *and so on*

Lazy Computing: TensorFlow é uma maneira de representar a computação sem realmente realizá-la até que seja feita a solicitação. A primeira etapa para aprender Tensorflow é compreender sua característica chave principal, a aproximação do “grafo computacional". Basicamente, todos os códigos Tensorflow contêm duas partes importantes: **graphs** e **sessions**

Um **graph** define a computação. Ele não calcula nada, ele não mantém quaisquer valores, ele apenas define as operações que você especificou em seu código. Uma **session** permite executar **graphs** ou parte deles. Ela aloca recursos (em uma ou mais máquinas) para isso e mantém os valores reais de resultados intermediários e variáveis.

Conteúdo

```
[23]  import tensorflow as tf
```

Hello World!

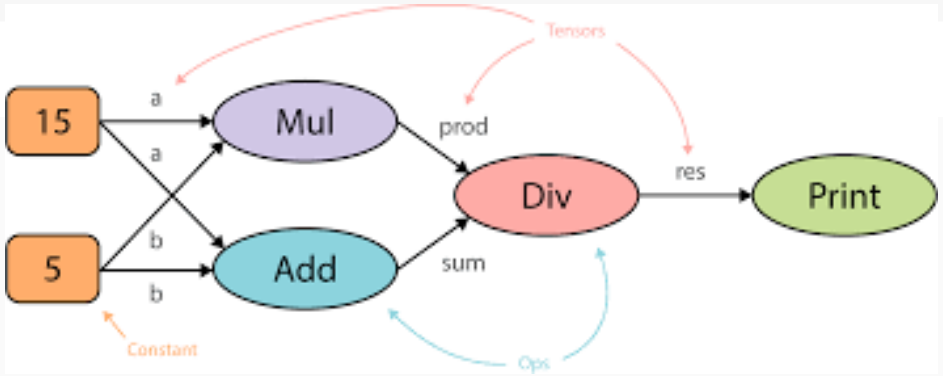
Vamos usar o Google Colab

```
[24]  a=tf.Variable(15,dtype=tf.int32)
      b=tf.Variable(5,dtype=tf.int32)
      dv=tf.divide(tf.multiply(a,b),tf.add(a,b))
      print('Antes da Session ',dv)
```

```
Antes da Session  Tensor("truediv_7:0", shape=(), dtype=float64)
```

```
with tf.Session() as sess:
  sess.run(tf.initializers.global_variables())
  print(sess.run(a),sess.run(b),sess.run(dv))
```

```
15 5 3.75
```



Os **placeholders** no TensorFlow são semelhantes às variáveis e podem ser declarados usando tf.placeholder. Não é necessário fornecer um valor inicial (especificado em tempo de execução com o argumento feed_dict dentro de Session.run), enquanto que em tf.Variable um valor inicial é declarado.

Partindo de **perceptron-tensorflow_res.ipynb**,

Analise o código (entenda o placeholder)

Atividade: crie perceptron1-tensorflow.ipynb onde o bias é o primeiro elemento de w (sem a constante bias)

Conteúdo

Partindo de **logreg-tensorflow.ipynb**,

Analisar código para verificar:

A função custo (loss)

O otimizador (Gradiente Descent)

A otimização em épocas

Atividade 1: gerar gráfico das amostras

Atividade 2: calcular acurácia

Atividade 3: fazer a otimização em batches de 5 amostras

Conteúdo

Partindo de **MLP-Tensorflow-digits.ipynb**,

Analisar código para verificar:

One hot encoding

Função rede (monta a rede com 1 hidden layer)

A ativação da rede é softmax(?)

Otimizador é Adam(?)

Atividade 1: calcular acurácia no treino e teste (70-30)

Atividade 2: colocar mais uma camada hidden

Cursos com Alta Performance de Aprendizado