UFG
UNIVERSIDADE FEDERAL DE GOIÁS

Aula 5

Deep Learning aplicado na classificação de imagens de satélite

Construção de uma Rede Neural Densa

Priscila M. Kai

Roteiro

01 **Aula 1**
Introdução ao Sensoriamento Remoto

02 **Aula 2**
Aquisição de Imagens de Sensoriamento Remoto

03 **Aula 3**
Extração de características

04 **Aula 4**
Construção de Modelos para a Classificação de Culturas

05 **Aula 5**
Construção de uma Rede Neural Densa

05

Construção de uma Rede Neural Densa

O que veremos?

Roteiro

1. Redes Neurais
    - Perceptron
    - Perceptron Multicamadas
    - Rede Densa
        - Exemplos em Python
    - Dropout
        - Exemplos em Python

Redes Neurais Artificiais

Muito se tem falado sobre as Redes Neurais Artificiais (RNA), principalmente pela aplicação do Deepfake em figuras famosas, transferindo a face de uma pessoa para outra, com resultado cada vez mais realista.

Redes Neurais Artificiais

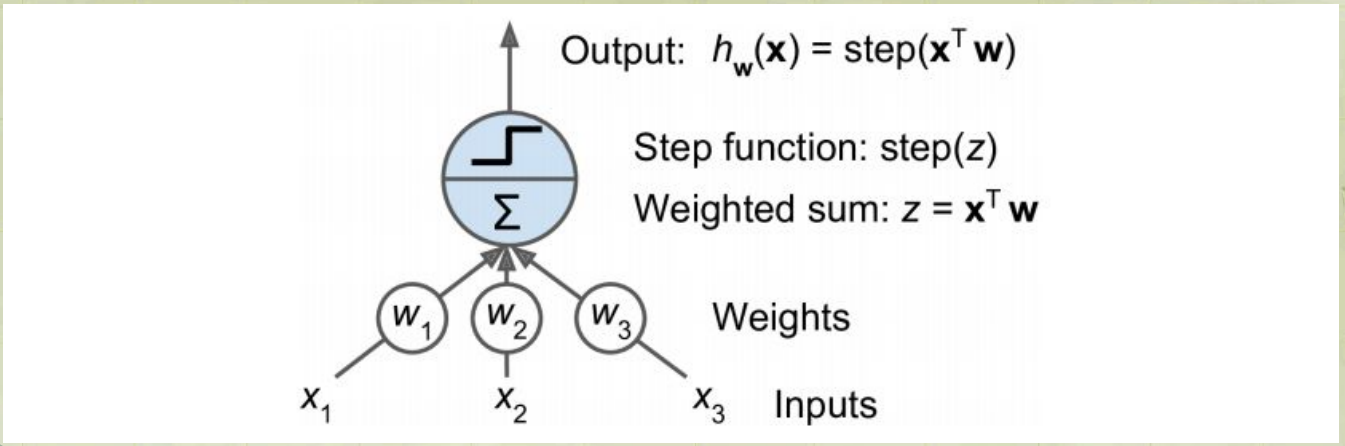
A presença das RNAs também está presente na geração de áudio, classificação de imagens, identificação de plantas, além de diversas outras finalidades. Assim, podemos dizer que as RNAs possuem grande importância no desenvolvimento de tecnologias, mostrando seu grande potencial nas mais variadas áreas.

Redes Neurais Artificiais

Redes neurais artificiais funcionam como sistemas de computação inspirados em **redes neurais biológicas**, por nós conectados em uma única direção (***feedforward***), no qual sinais de entrada são processados, com propagação ou não, resultando em combinações de sinais de saída.

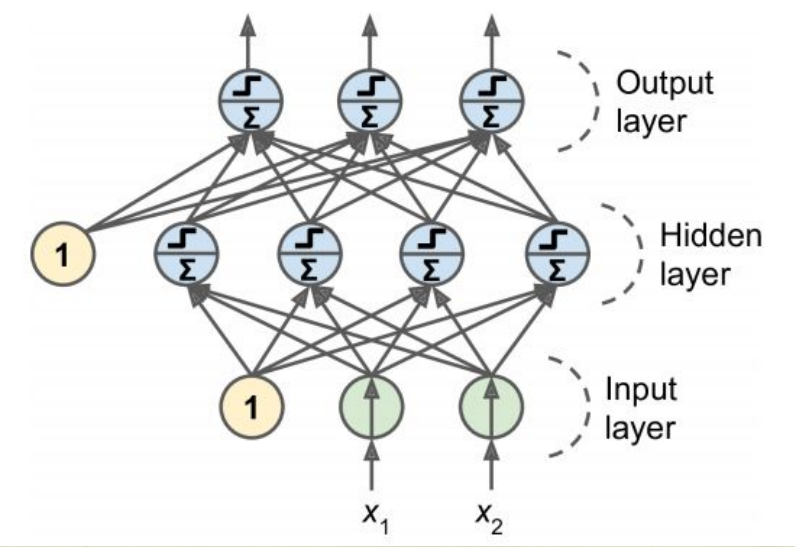
Perceptron

O **Perceptron** é composto por uma única camada. Como não temos processamento realizado na camada de entrada, conta-se apenas a camada de saída.



Perceptron Multicamadas

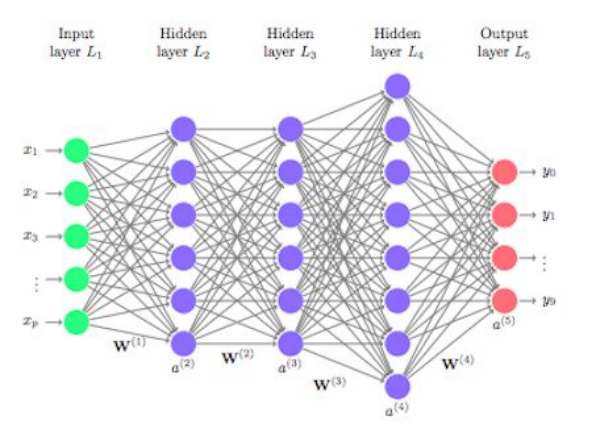
Uma rede formada por vários Perceptrons é chamada de **Perceptron Multicamadas** (MLP).

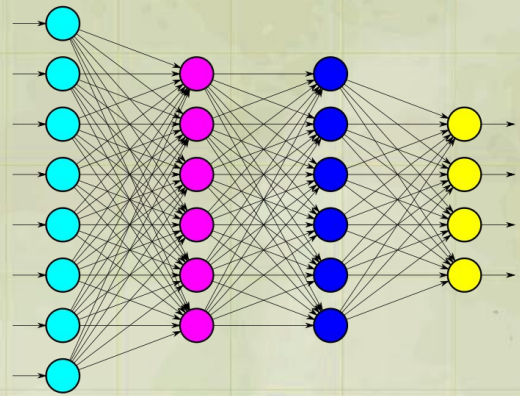


Rede Densa

Em uma rede em que todos os neurônios de uma camada estão conectados a todos os neurônios da camada anterior, temos então uma camada totalmente conectada, também chamada de **densa**.

Quando uma rede neural possui várias camadas ocultas, ela é chamada de **rede neural profunda** (DNN - Deep Neural Network).





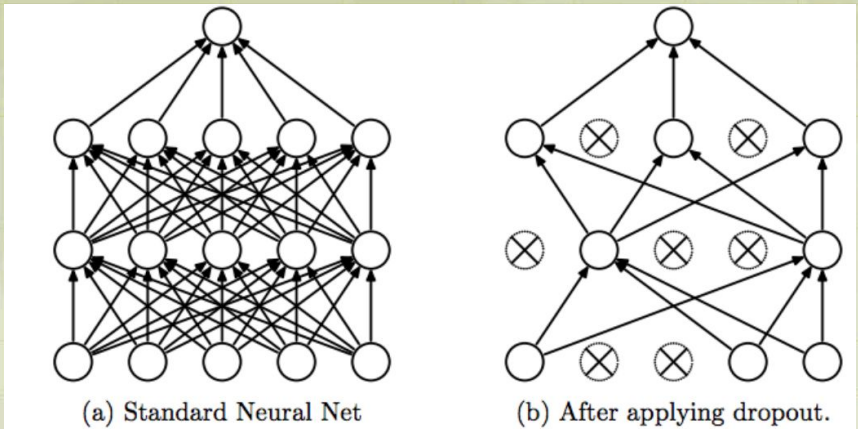
```
#Biblioteca para construção do modelo
from keras.models import Sequential
from keras import layers

#Modelo com 1 camada
model = Sequential([
            layers.Dense(num_class, activation = "softmax", input_shape=(19,))
#camada que retorna um array com as probabilidades, com soma = 1
            ])

model2 = Sequential([
            layers.Dense(5, activation = "relu", input_shape=(19,)),
           layers.Dense(num_class, activation = "softmax")
            ])

model3 = Sequential([
            layers.Dense(10, activation = "relu", input_shape=(19,)),
          layers.Dense(5, activation = "relu"),
          layers.Dense(num_class, activation = "sigmoid")
            ])
```

Dropout

A técnica de **Dropout** consiste em "desligar" temporariamente um ou mais neurônios em uma camada para um ciclo de processamento.

No ciclo seguinte de processamento estes neurônios são "ligados" e em novos ciclos, outros neurônios escolhidos aleatoriamente são "desligados".

Dessa maneira, diferentes conjuntos de neurônios são obtidos a cada ciclo, como se estivéssemos treinando redes neurais diferentes.



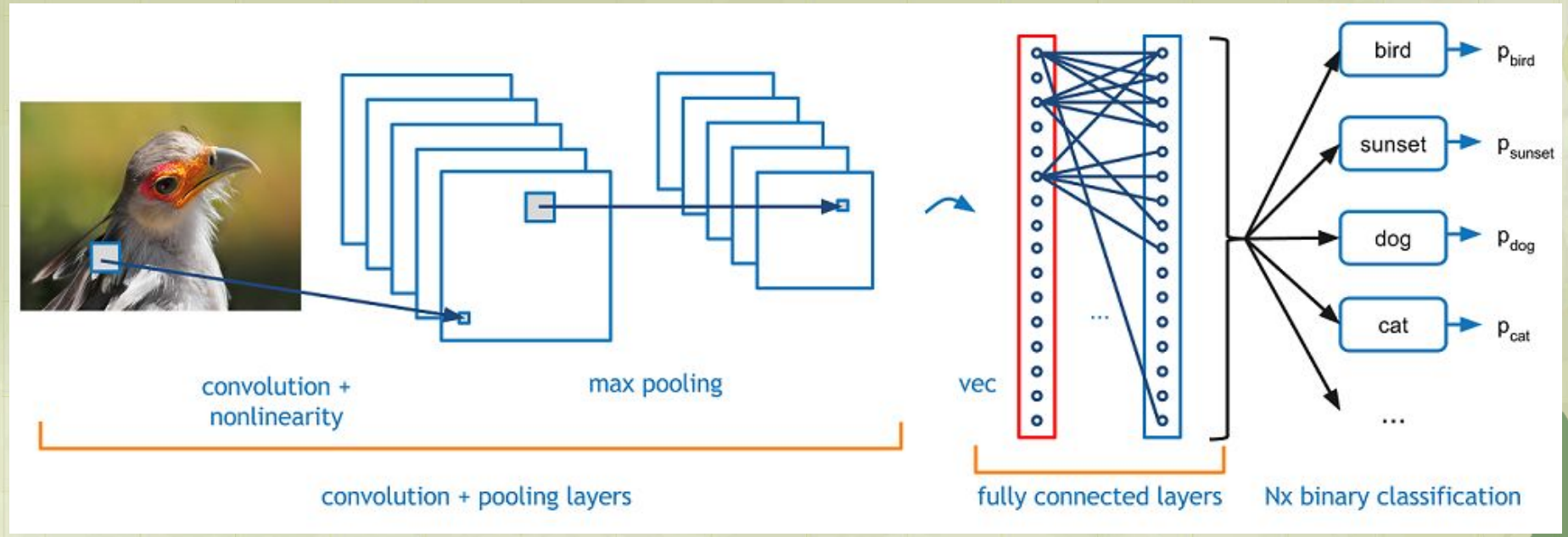
(a) Standard Neural Net

(b) After applying dropout.

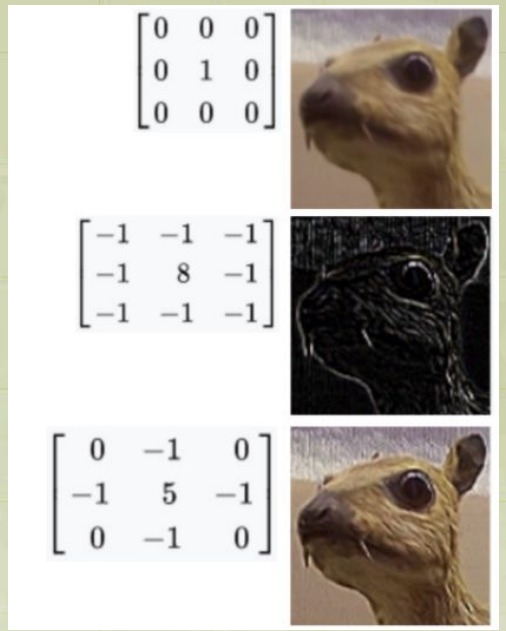
Exemplos em Python

```
#Biblioteca para construção do modelo
from keras.models import Sequential
from keras import layers

model = Sequential([
            layers.Dense(10, activation="relu",input_shape=(19,)),
            layers.Dropout((0.1)),
            layers.Dense(8, activation="relu"),
            layers.Dropout((0.1)),
            layers.Dense(4, activation="sigmoid")
            ])
```

Redes Neurais Convolucionais



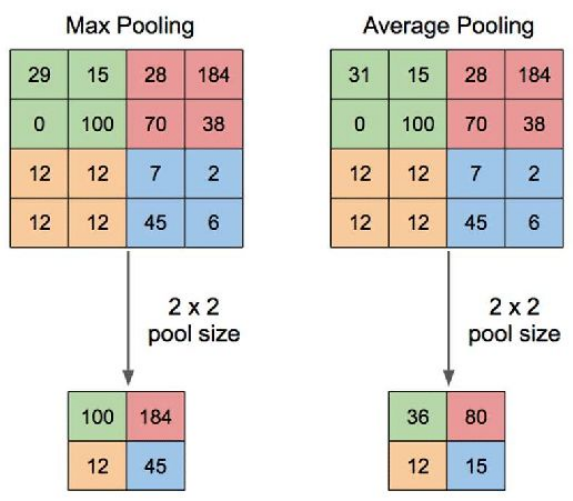
Redes Neurais Convolucionais



Exemplos de Kernel's utilizados nas convoluções:

Em uma CNN este kernel não possui valores fixos, eles são parâmetros treinados pelo algoritmo.

Redes Neurais Convolucionais




```
#Exemplo: LENET
model = Sequential([
              layers.Conv2D(6, (5, 5), activation=' relu',
input_shape=(250, 250, 1)),
            layers.MaxPooling2D((2,2)),
            layers.Conv2D(16, (5, 5), activation=' relu'),
            layers.MaxPooling2D((2,2)),
            layers.Flatten(),
            layers.Dense(120, activation = ' relu'),
            layers.Dense(84, activation = ' relu'),
                    layers.Dense(num_class,  activation  =
'softmax')
            ])
```