



*Inatel*

# Redes Neurais e Deep Learning

Prof. Eng. Ranyeri do Lago Rocha  
e-mail [ranyeri.rocha@inatel.br](mailto:ranyeri.rocha@inatel.br)

***Inatel***

## Agenda

- Modelos de Classificação e Detecção

Todo o conteúdo deste documento está relacionado a direito autoral e é de circulação restrita, porquanto de propriedade exclusiva da Fundação Instituto Nacional de Telecomunicações (CNPJ 24.492.886/0001-04), protegido por força das disposições da Lei n.º 9.610/1998. A utilização deste material sem prévia e expressa autorização da proprietária constituirá infração à lei, com repercussões tanto na esfera civil quanto criminal.

# Modelos de Classificação e Detecção

# Modelos de Classificação e Detecção

- Uma breve introdução à Redes Neurais Artificiais
- CNN para Visão Computacional

## Uma breve introdução à Redes Neurais Artificiais

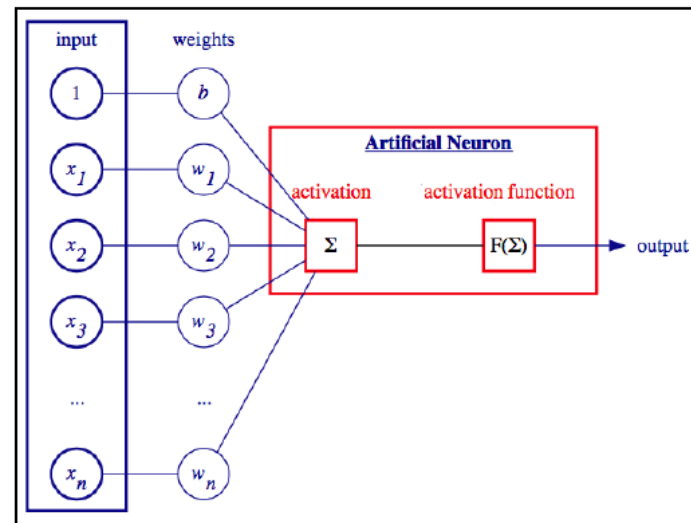
### Da Perceptron à Multi Layer Perceptron

Perceptron foi proposto baseado no estudo de comportamento do neurônio biológico, conforme modelo matemático de 1943.

- *Limitação*: resolve apenas problemas lineares

MLP foi proposta baseada no perceptron com a intenção de dar flexibilidade ao perceptron: aprender pesos das conexões e ter conexões com outros neurônios

- *Vantagem*: boa capacidade para resolver problemas complexos

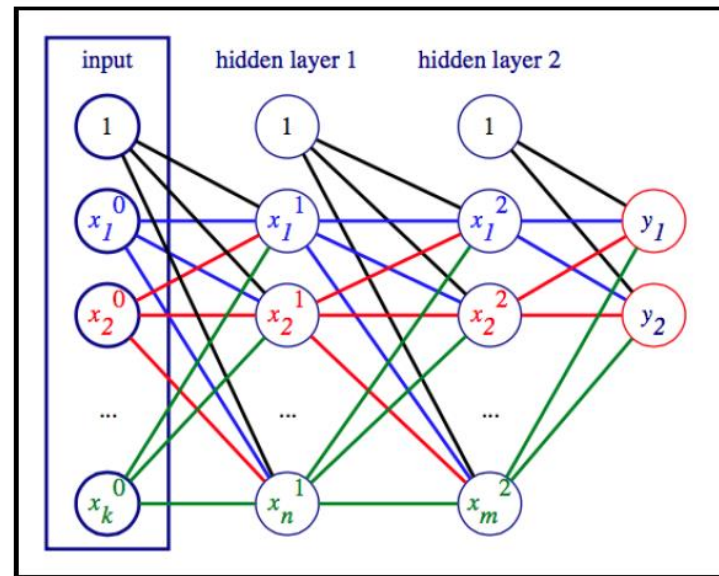


## Uma breve introdução à Redes Neurais Artificiais

### Da Perceptron à Multi Layer Perceptron

A imagem ao lado é uma típica representação de uma rede neural, MLP.

- Totalmente conectada
- Os neurônios são ativados pela combinação linear dos valores de peso e dados da camada anterior
- Em todas as camadas há um termo de *bias*
- Os neurônios intermediários podem ter diferentes ativações
- Os neurônios de saída podem ter diferentes ativações

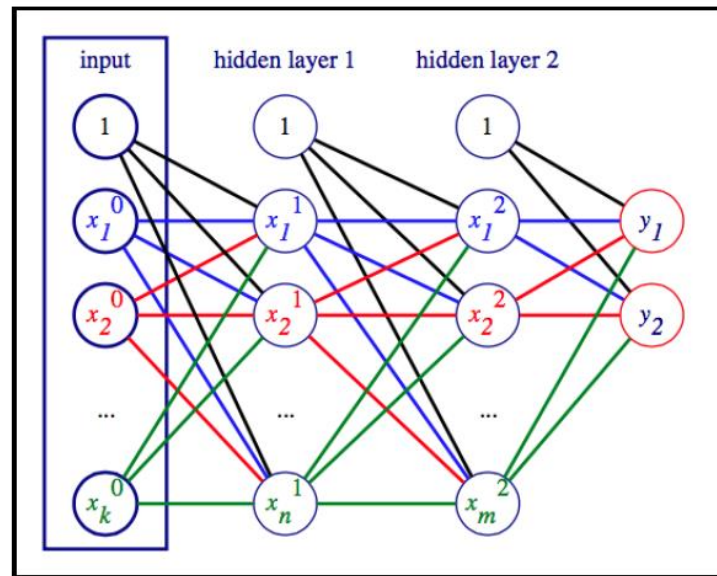


## Uma breve introdução à Redes Neurais Artificiais

### Da Perceptron à Multi Layer Perceptron

A imagem ao lado é uma típica representação de uma rede neural, MLP.

- Atualização dos pesos sinápticos
- Treinamento vs. Teste
- Dimensão do dado de entrada impacta na primeira camada intermediária
- Quantidade de pesos a serem atualizados



## Uma breve introdução à Redes Neurais Artificiais

### Além da Multi Layer Perceptron

Outras duas estruturas associas à **Redes Neurais** (NN), são as conhecidas como **Redes Neurais Recorrentes** (RNN) e a **Rede Neural Convolucional** (CNN). De forma bastante geral, a MLP geralmente é mais associada a problemas de regressão, as RNNs mais associadas a problemas com dados sequenciais, pelo fator recorrente tratar bem das dependências temporais e, CNNs comumente associadas a problemas com dados como imagem e vídeo.

Embora cada estrutura tenha seu papel, no conceito de **Deep Learning**, estas estruturas podem, e geralmente são, combinadas para tratar o problema com diferentes “abordagens”.



# Modelos de Classificação e Detecção

- Uma breve introdução à Redes Neurais Artificiais
- CNN para Visão Computacional

## CNN para Visão Computacional

### CNN ou Deep Learning

**Deep Learning** é uma classe de técnicas de aprendizado de máquina em que a informação é processada em camadas hierárquicas. A hierarquia está associada à representação e extração de características dos dados em níveis de complexidade alto.

**Convolutional Neural Networks (CNN)** é uma rede neural com várias camadas especiais para extração de característica. São, idealmente, organizadas a mesma maneira que as células biológicas são organizadas no córtex visual do cérebro. Camadas especiais são as chamadas **camadas convolucionais que aplicam um filtro à imagem de entrada**. É, sem dúvida, a melhor abordagem para tarefas de visão computacional, até agora.

## CNN para Visão Computacional

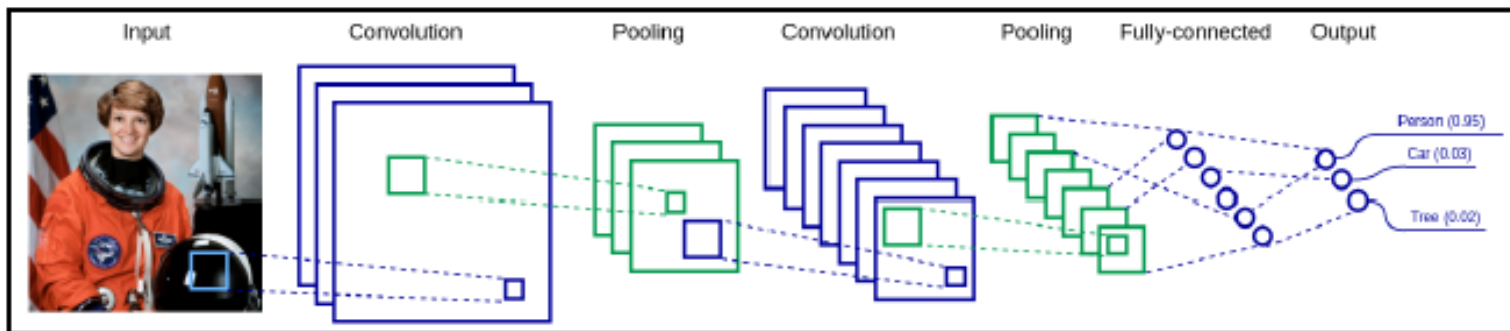
### Visão Computacional

Visão é, sem dúvida, o mais importante sentido dos humanos. Quando tentamos reproduzir, a tarefa de **reconhecimento de imagens** foi por um longo tempo um dos **problemas mais difíceis de resolver**. Historicamente era complicado “explicar” a uma máquina quais características pertenciam a cada objeto e, principalmente, como detectá-las.

Em Deep Learning, esta tarefa pode ser **aprendida por si só!**

## CNN para Visão Computacional

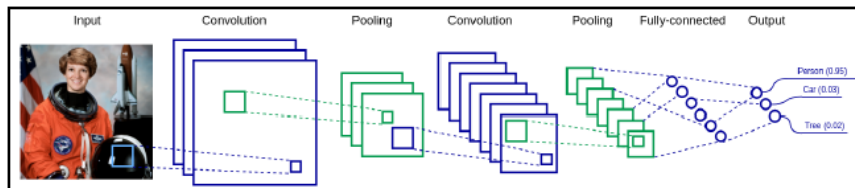
### Visão Computacional



## CNN para Visão Computacional

### Visão Computacional

- Entrada



CNN são redes especializadas que consideram a imagem de entrada como um *tensor*. Tensor é similar às matrizes ou vetores (*arrays*) em *numpy* e podem ser usados em GPUs melhorando o desempenho, tomando vantagem do processamento paralelo.

É importante entender os formatos dos tensores, que podem ser:

- escalar (0D *tensors*), vetor (1D *tensors*), matriz (2D *tensors*), 3D *tensors*, slicing *tensors*, 4D, 5D ou *tensor on GPU*

## CNN para Visão Computacional

### Visão Computacional

#### - Entrada

Escalar (0D *tensor*)

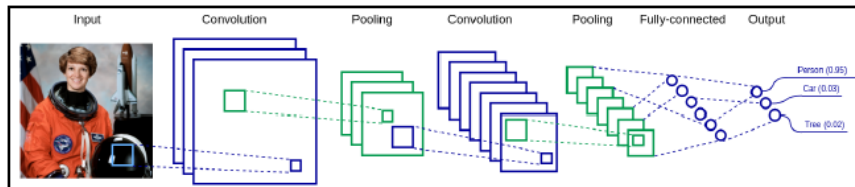
- Tensor contendo apenas um elemento *FloatTensor* ou *LongTensor*

Vetor (1D *tensor*)

- Tensor contendo um *array* de elementos

Matriz (2D *tensor*)

- Tensor que representa dados como matrizes ou tabelas, no formato [X,Y], onde X são as linhas e Y as colunas.



## CNN para Visão Computacional

### Visão Computacional

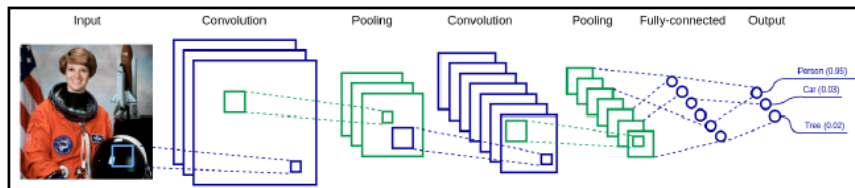
#### - Entrada

#### 3D *tensor*

- Quando adicionamos várias matrizes juntas forma-se um tensor em 3 dimensões. Quando representamos imagens, geralmente, são representadas como um 3D *tensor*. Uma imagem é representada pelas informações de altura, largura e canal (RGB), sendo um tensor [altura, largura, canal]. Por exemplo, uma imagem 200x200 colorida é representada por [200, 200, 3].

#### *Slicing Tensor*

- É a representação de uma porção de um tensor original, podendo ser 1D, 2D ou 3D. Uma imagem recortada, por exemplo, é um *slicing tensor*.



## CNN para Visão Computacional

### Visão Computacional

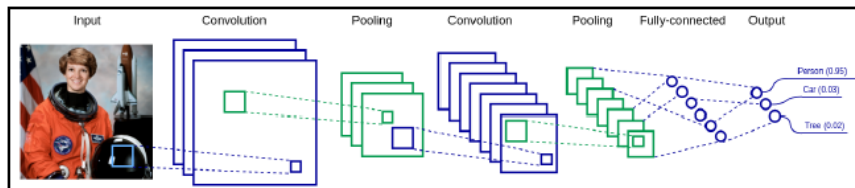
#### - Entrada

#### 4D tensor

- É comum a utilização de tensores de 4 dimensões em processamento de imagem em batelada. CPUs e, principalmente, GPUs são otimizadas para processar de forma rápida e eficiente múltiplos exemplos de entrada. Os *batches* geralmente utilizados são 16, 32 e 64. No mesmo exemplo da imagem [200,200,3], se tomarmos 64 imagens do conjunto, teremos um tensor 4D, [64,200,200, 3].

#### 5D tensor

- Para vídeo, um tensor é representado com 5 dimensões. A ideia é a mesma para o tensor 4D mas com a inserção do quinto elemento referente ao vídeo. Um vídeo com 30 frames, com as mesmas dimensões da imagem anterior, seria representado por [1,30,200,200,3]





## CNN para Visão Computacional

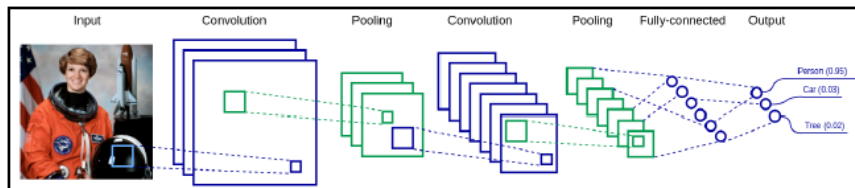
### Visão Computacional

#### - Camada convolucional

Este é o primeiro passo na extração de características de uma imagem.

O objetivo é manter o relacionamento entre pixels próximos aprendendo as características sobre uma pequena seção da imagem.

*É uma operação matemática.* Duas entradas (imagem e máscara do filtro) e uma saída (resultado da convolução)



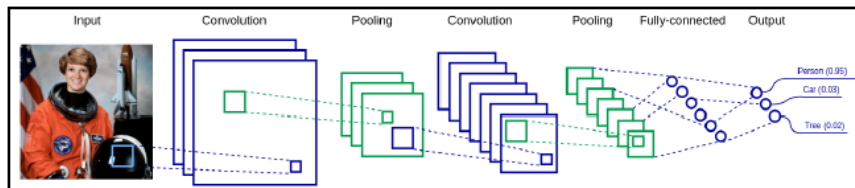
## CNN para Visão Computacional

### Visão Computacional

#### - Camada convolucional

Aplica-se um filtro, uma máscara/template, na imagem desde o pixel mais superior esquerdo até o pixel mais inferior direito. Máscaras podem ser de diferentes tamanhos, 1, 3, 5, 7, etc. Quanto maior a máscara maior a área de cobertura na imagem. É comum utilizar máscaras 7x7 ou 9x9 nas camadas iniciais.

Aqui é possível mover o filtro com uma passo maior que 1, chamado de **stride**. Neste caso, uma operação de convolução com *stride* maior que 1 é chamada, usualmente, de *convolução stride*



## CNN para Visão Computacional

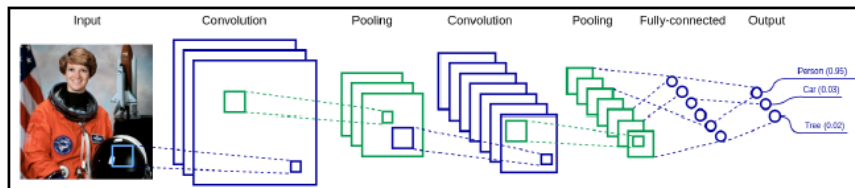
### Visão Computacional

#### - Camada convolucional

A matriz resultante após a convolução terá um formato específico:

$$\begin{aligned} altura\ saída &= h - f_h + 1 \\ largura\ saída &= w - f_w + 1 \\ profundidade &= 1 \end{aligned}$$

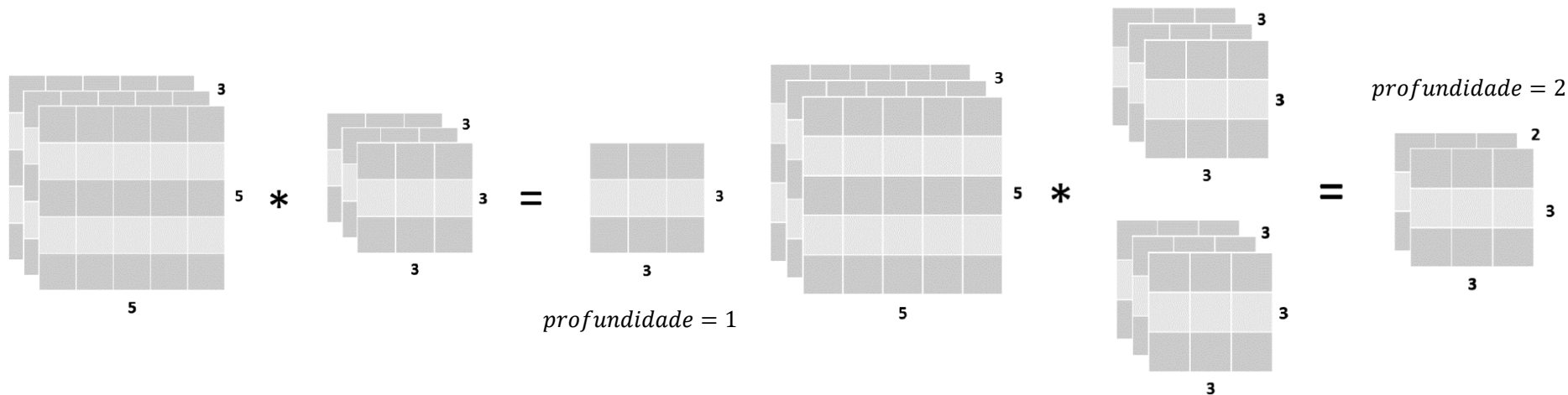
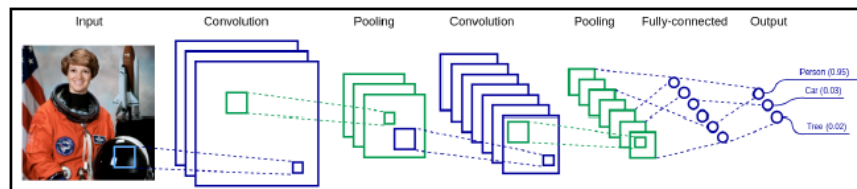
$h$  é a altura da imagem e  $w$  a largura, em pixel,  $f_h$  e  $f_w$  altura e largura da máscara, em pixel e a profundidade é a resultante da convolução. Na sequência este conceito de profundidade é explorado.



## CNN para Visão Computacional

Visão Computacional

### - Camada convolucional



## CNN para Visão Computacional

### Visão Computacional

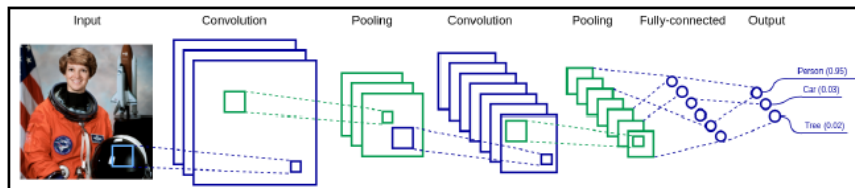
#### - Camada convolucional

É comum ter uma profundidade maior que 1. Ou seja, utilizar mais de um filtro (máscara) na etapa de convolução. O objetivo é claro: **Obter diferentes características da imagem!**

Um filtro para bordas diagonais, um filtro para bordas horizontais, etc...

Na prática, a quantidade de filtros é aumentada a cada camada. É comum duplicar, por exemplo. Primeira camada convolucional com 8, segunda 16, terceira 32, etc.

**Por último:** a configuração do filtro não precisa ser determinada. Faz parte do problema definir os parâmetros do filtro.



## CNN para Visão Computacional

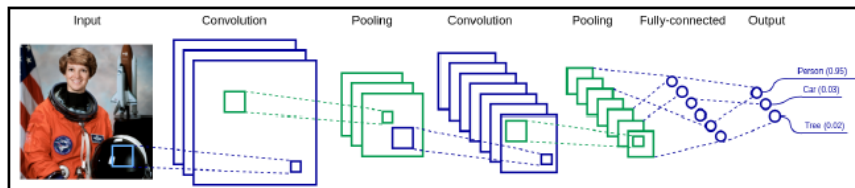
### Visão Computacional

#### - Camada convolucional

Transformação por vizinhança e “padding”

O parâmetro “padding” (como veremos em outra parte do material) diz respeito a inclusão de pixels de valor constante na borda externa da imagem. É geralmente usado valor 0 (zero). O objetivo é manter as dimensões da matriz de entrada quando processada com o filtro.

Nas primeiras camadas, a ideia é manter ao máximo a preservação de informação da imagem original.



## CNN para Visão Computacional

### Visão Computacional

#### - Camada convolucional

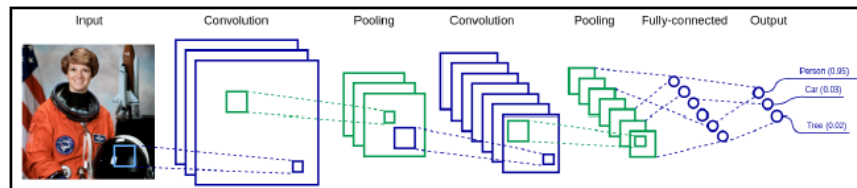
Transformação por vizinhança e “padding”

Assim, a saída pode ser calculada:

$$\begin{aligned} \text{altura saída} &= h - f_h + 2 * P + 1 \\ \text{largura saída} &= w - f_w + 2 * P + 1 \end{aligned}$$

E calcula-se o *padding*,  $P$

$$P = \frac{f_h \text{ ou } w - 1}{2}$$



## CNN para Visão Computacional

### Visão Computacional

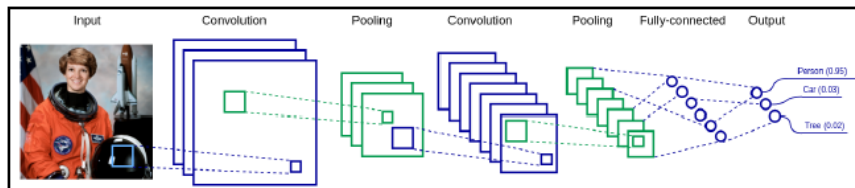
#### - Camada convolucional

O conceito do *stride* apresentado alguns slides atrás. Aqui mostramos na sequência como aparece na prática.

**Relembre:** número de pixels que um filtro irá deslocar sobre a matriz de entrada, tanto horizontalmente quanto verticalmente.

Por padrão, *stride* igual a 1. Entretanto, outros valores podem ser utilizados.

$$\text{altura saída} = \frac{h - f_h}{s} + 1$$
$$\text{largura saída} = \frac{w - f_w}{s} + 1$$





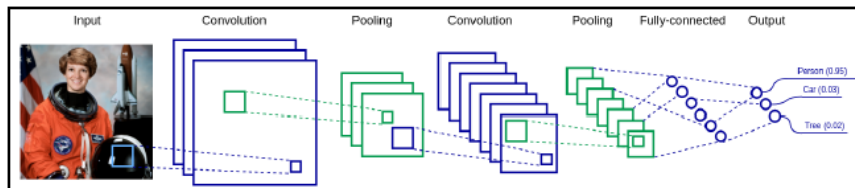
## CNN para Visão Computacional

### Visão Computacional

#### - Camada convolucional

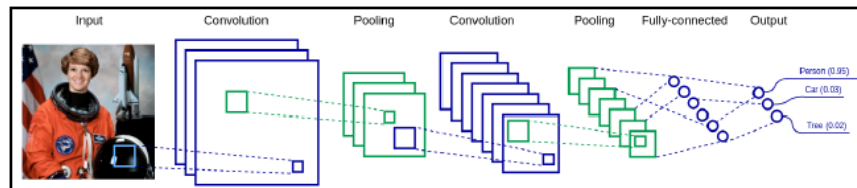
Para finalizar, podemos ter uma expressão que representa a saída depois de uma camada de convolução usando *padding* e *stride*:

$$\begin{aligned} \text{altura saída} &= \frac{(h - f_h) + 2 * P}{s} + 1 \\ \text{largura saída} &= \frac{(w - f_w) + 2 * P}{s} + 1 \end{aligned}$$



## CNN para Visão Computacional

### Visão Computacional



### - Camada convolucional

```
import torch.nn as nn
import torch.nn.functional as F

class CNN_network(nn.Module):
    def __init__(self):
        super(CNN_network, self).__init__()
        self.conv1 = nn.Conv2d(3, 18, 3, 1, 1)

    def forward(self, x):
        x = F.relu(self.conv1(x))
        return x
```

Customização de um módulo

Criação de uma classe

Definição da arquitetura

3 canais de entrada, 18 filtros, cada com tamanho 3, *stride* 1 e *padding* 1

Cálculo a ser efetuado ao passar a informação pelas camadas

## CNN para Visão Computacional

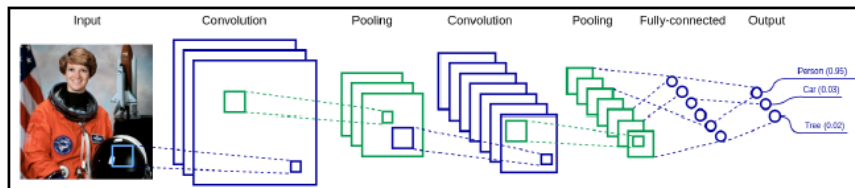
### Visão Computacional

#### - Camada de *Pooling*

É uma prática comum adicionar, após a camada convolucional, uma camada de *Pooling*. A ideia principal é reduzir a dimensão dos mapas de características das camadas de convolução. Duas características principais:

1. Redução dos dados para processamento
2. Forçar o algoritmo a não focar em pequenas mudanças na posição da imagem

A ideia é a mesma da camada convolucional. Tamanho das máscaras e o *stride* também é encontrado na camada de *Pooling*. Mas aqui, não temos pesos. A operação feita na resultante da camada convolucional pode ser MAX (MaxPooling) ou MÉDIA (AveragePooling).



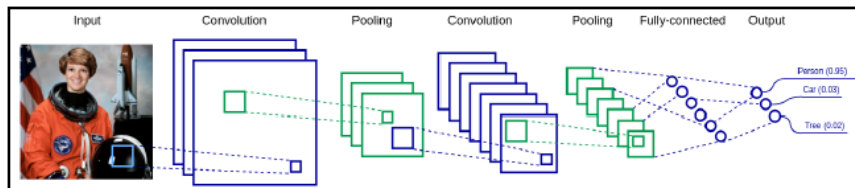
## CNN para Visão Computacional

### Visão Computacional

#### - Camada de Pooling

A camada de *pooling*, por convenção são as últimas partes do processo de seleção de características. A ideia é extrair as informações mais relevantes de subseções da imagem. As dimensões da camada de *pooling* é tipicamente 2, com stride igual ao seu tamanho (2).

Para tarefas de classificação de imagem, é mais comum o use de camadas MAX POOLING. Estas preservam as características mais relevantes.



## CNN para Visão Computacional

### Visão Computacional

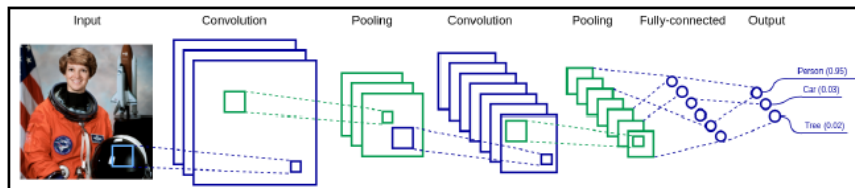
#### - Camada de Pooling

*MAX Pooling*

O resultante da camada de *pooling* é o pixel de maior intensidade dentro da máscara.

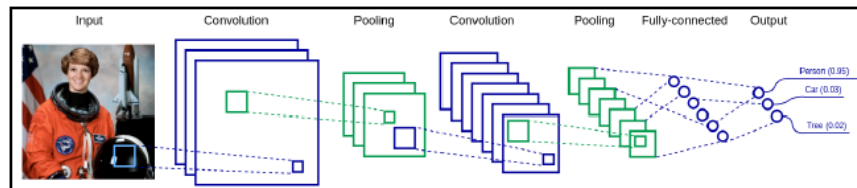
*AVERAGE Pooling*

O resultante da camada de *pooling* é a média dos valores de pixel dentro da máscara.



## CNN para Visão Computacional

### Visão Computacional



### - Camada de Pooling

```
import torch.nn as nn
import torch.nn.functional as F
```

```
class CNN_network(nn.Module):
```

```
    def __init__(self):
```

```
        super(CNN_network, self).__init__()
```

```
        self.conv1 = nn.Conv2d(3, 18, 3, 1, 1)
```

```
        self.pool1 = nn.MaxPool2d(2, 2)
```

```
    def forward(self, x):
```

```
        x = F.relu(self.conv1(x))
```

```
        x = self.pool1(x)
```

```
        return x
```

Customização de um módulo

Criação de uma classe

Definição da arquitetura

3 canais de entrada, 18 filtros, cada com tamanho 3, *stride* 1 e *padding* 1

Tamanho do filtro 2 e *stride* 2

Cálculo a ser efetuado ao passar a informação pelas camadas

## CNN para Visão Computacional

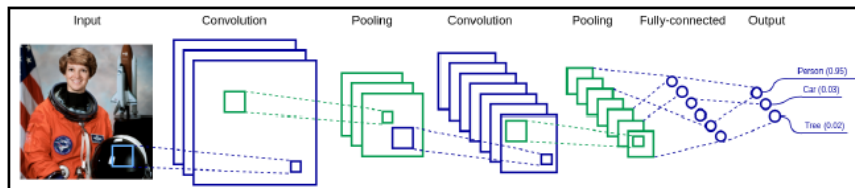
### Visão Computacional

#### - Ativação não linear - ReLU

É uma prática comum utilizar uma camada não linear após a camada de *Pooling* ou a convolução ser aplicada. A maioria das arquitetura de rede tendem a usar a ReLU ou variações. A intenção é aplicar uma função não linear a cada elemento do mapa de característica.

#### - Rede totalmente conectada

Ao final das camadas de convolução e *Pooling*, uma rede totalmente conectada é implementada a fim de desempenhar o papel principal, classificação, por exemplo.



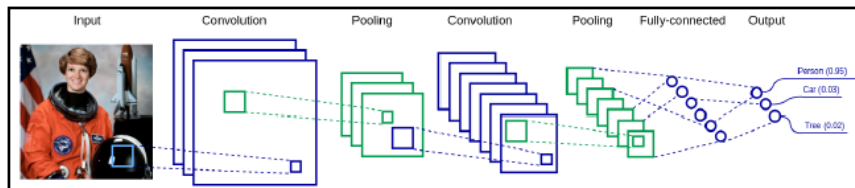
## CNN para Visão Computacional

### Visão Computacional

#### - Rede totalmente conectada

A última camada (*pooling*, geralmente), terá sua saída (matriz) convertida em um vetor e irá alimentar a rede totalmente conectada. A ideia aqui é a mesma de uma rede neural tradicional.

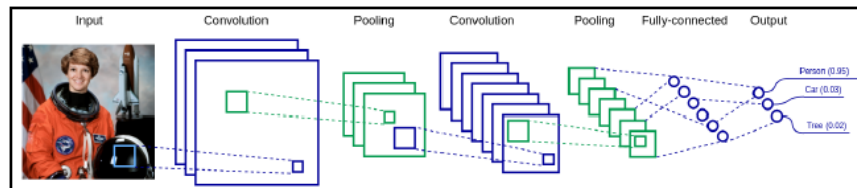
A proposta é simples. Considerar todas as características que foram detectadas pelas camadas anteriores com a função de classificar a imagem. Como comentado no tópico de introdução à Redes Neurais, as ativações utilizadas aqui são ReLU e na camada de saída, SoftMAX.





## CNN para Visão Computacional

### Visão Computacional



### - Rede Totalmente Conectada

```
import torch.nn as nn
import torch.nn.functional as F
```

```
class CNN_network(nn.Module):
```

```
    def __init__(self):
```

```
        super(CNN_network, self).__init__()
```

```
        self.conv1 = nn.Conv2d(3, 18, 3, 1, 1)
```

```
        self.pool1 = nn.MaxPool2d(2, 2)
```

```
        self.linear1 = nn.Linear(32*32*16, 64)
```

```
        self.linear2 = nn.Linear(64, 10)
```

3 canais de entrada, 18 filtros, cada com tamanho 3, *stride* 1 e *padding* 1

Tamanho do filtro 2 e *stride* 2

Neurônios entrada e intermediário

Neurônios intermediários e saída

```
    def forward(self, x):
```

```
        x = F.relu(self.conv1(x))
```

```
        x = self.pool1(x)
```

```
        x = x.view(-1, 32*32*16)
```

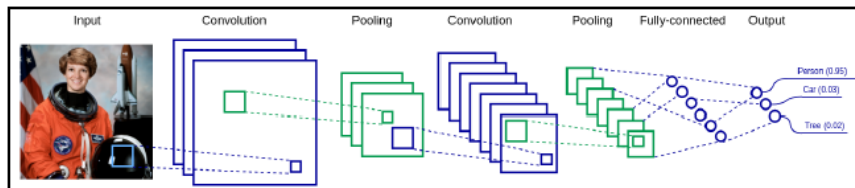
```
        x = F.relu(self.linear1(x))
```

```
        x = F.log_softmax(self.linear2(x), dim=1)
```

```
        return x
```

## CNN para Visão Computacional

### Visão Computacional



### Considerações sobre imagens em CNN

A forma mais simples de representar uma imagem (matriz) para inserir em uma rede neural tradicional é transformando ela em um vetor (*flatten*). É muito usual, mas, neste caso, perde-se a dependência espacial dos pixel, que, na forma matricial ainda permanecem. A proposta da CNN é usar um conjunto de pixel da imagem por vez, determinado pelo tamanho da máscara do filtro.

A ideia então é utilizar a capacidade dos filtros de produzir uma melhor representação da imagem.

3	4	1
2	3	5
1	2	3



3
4
1
2
3
5
1
2
3

## CNN para Visão Computacional

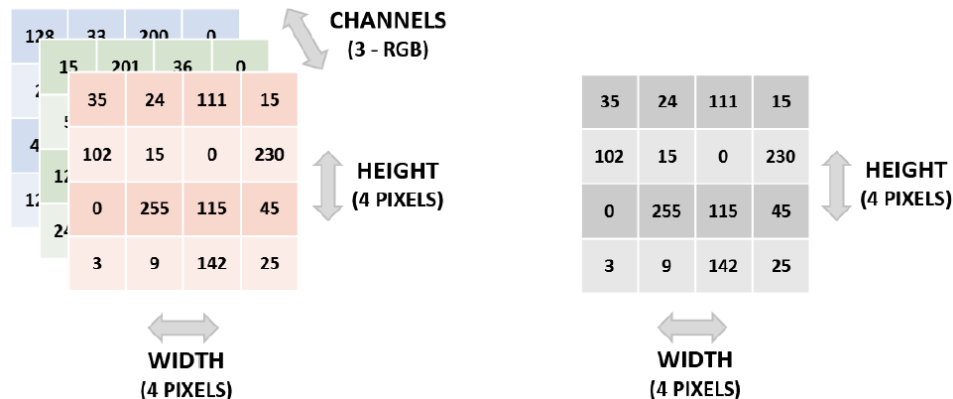
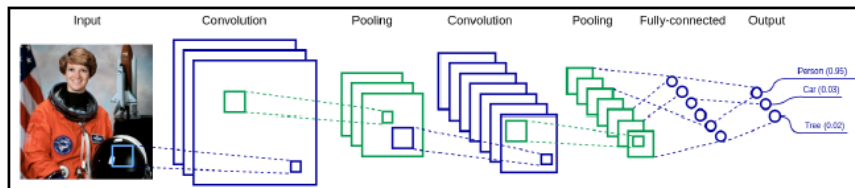
### Visão Computacional

### Considerações sobre imagens em CNN

A já conhecida representação de imagem, em canais RGB (para coloridas) e escala de cinza.

É importante ressaltar que, os pré-processamento comuns em imagens são:

- Ajustar as imagens ao mesmo tamanho
- Normalizar o valor dos pixels, considerando faixa de valores de 0 a 1, ou centralizados na média.



## CNN para Visão Computacional

### Visão Computacional

As tarefas geralmente associadas à visão computacional são, como já descrito:

1. Classificação
2. Localização
3. Detecção de objetos
4. Segmentação

Todas estas tarefas são possíveis com CNN, graças a capacidade de extração de características da rede.

## CNN para Visão Computacional

### Visão Computacional

#### Classificação

É a principal e mais comum tarefa conhecida em Visão Computacional. A ideia principal é classificar conteúdos de uma imagem em um conjunto de categorias, conhecido como *label* (rótulos). A tarefa de classificação, geralmente, está associada a probabilidade de o objeto na imagem pertencer a uma classe ou não.

Classificação é um típico problema de Aprendizado Supervisionado. É preciso conhecer o rótulo de cada instância de treinamento para guiar o ajuste dos parâmetros/pesos.

## CNN para Visão Computacional

### Visão Computacional

#### Classificação

Classificação não é só possível em CNN ou em tarefas de visão computacional. É importante ressaltar que, a maioria das aplicações de classificação são fora da área de visão computacional, como por exemplo, classificar transações fraudulentas, classificar cliente em grupos de comportamento, classificar *status* de um equipamento dado as condições de trabalho, etc.

Entretanto, como já comentado, em visão computacional é, sem dúvida, a maior aplicação.

## CNN para Visão Computacional

### Visão Computacional

#### Localização

O objetivo da tarefa de localização é gerar “*bounding box*” que descreva a localização de um objeto na imagem. A saída consiste de um rótulo (classe) e as informação de “*bounding box*”.

Pelo descrito, a tarefa de localização é exatamente como descrevemos a tarefa de detecção de objetos. Entretanto, algumas literaturas usam localização para referenciar apenas um (1) objeto na imagem. O fato é que, a definição de localização e detecção de objetos é a mesma: **encontrar a localização de um ou mais objetos em uma imagem e reportar as classes associadas.**

## CNN para Visão Computacional

### Visão Computacional

#### Deteccção de objetos

...“encontrar objetos de uma certa classe, como faces, carros, árvores, etc, em imagens ou vídeos.”

Importante: Deteccção de objeto pode detectar vários objetos em uma imagem e a sua localização

Um detector retorna:

- A classe do objeto
- Valor de confiança ou probabilidade na faixa de  $[0,1]$
- A coordenada da região da imagem – *bounding box*



## CNN para Visão Computacional

### Visão Computacional

#### Deteccção de objetos

Algumas abordagens são utilizadas para a tarefa de detecção de objetos. A saber:

1. *Deteccção em dois estágios*

Métodos bastante acurados porém relativamente lentos. Envolvem uma etapa de varredura na imagem e proposição de vários “*bounding boxes*” candidatos a localização do objeto, usando CNN. A segunda etapa é a de classificação das regiões de interesse.

2. *Deteccção em um estágio*

Uma única CNN produz o tipo de objeto e o *bounding box*. São abordagens mais rápidas, porém, menos acuradas quando comparadas com detecção em dois estágios.

## CNN para Visão Computacional

Visão Computacional

### Segmentação

A tarefa de segmentação consiste em ter na saída: classe do objeto e o seu contorno. A segmentação é uma tarefa geralmente associada a marcação de objetos para análise futura. Delimitar área de um tumor em uma imagem, por exemplo, é uma aplicação prática.

## CNN para Visão Computacional

### Visão Computacional

#### Segmentação Semântica

Semântica é contexto. A segmentação semântica visa dividir a imagem em regiões associadas à classe do objeto. Cada pixel da imagem é associado a uma classe. É a mesma ideia de uma tarefa de classificação, mas, a nível de pixel.

Neste caso é preciso um treinamento com dados reais e rotulados (segmentados), onde, os rótulos para cada imagem versões da imagem semanticamente segmentadas.

## CNN para Visão Computacional

### Visão Computacional

#### Segmentação Semântica

Como o nome sugere, está técnica marca partes de uma imagem com uma categoria, por exemplo, todos as árvores na cor verde, prédios em vermelho, carros em cinza e assim por diante.

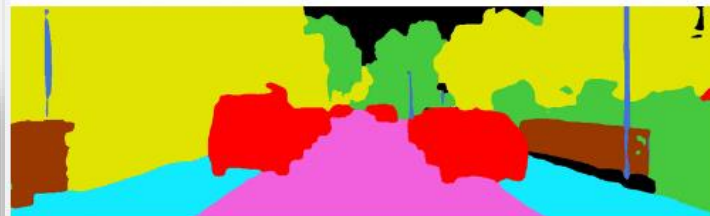
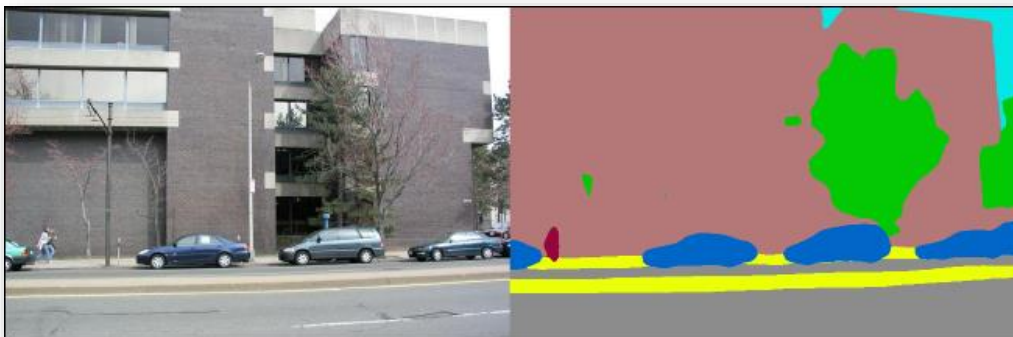
É um processo inteligente e importante para entender o que há na imagem, além de apenas uma estrutura ou região.









Veja que é a operação de reconhecimento e entendimento mais próximo do nível de pixel.  
Importante: Um *dataset* também é preciso, com os rótulos para treinamento.

## CNN para Visão Computacional

Visão Computacional

Segmentação Semântica



 Road	 Sidewalk	 Building	 Fence
 Pole	 Vegetation	 Vehicle	 Unlabel

## CNN para Visão Computacional

### Visão Computacional

#### Transferência de Aprendizado

Em cenários com modelos “pequenos” usando conjunto de dados também “pequenos”, a prática usada é a de criar o modelo do começo:

- *dataset*
- processamento dos dados, se necessário
- criação do modelo
- treinamento (mesmo que dure algumas horas)
- avaliação/uso.

Mas, e se o **conjunto de dados** for grande, **extremamente grande** e o **modelo** necessário é também **extremamente complexo**?

## CNN para Visão Computacional

### Visão Computacional

#### Transferência de Aprendizado

*Importante:* Conjunto de dados grandes não são sempre disponíveis para todas as tarefas.

*Importante:* Mesmo que tenha dados disponíveis, os rótulos são igualmente importantes.

Nestes cenários, usamos a Transferência de Aprendizado!

#### **O que é?**

É o processo de aplicar um modelo de Aprendizado de Máquina existente, já treinado, a um novo modelo. **É importante que o problema seja relacionado!**

## CNN para Visão Computacional

### Visão Computacional

#### Transferência de Aprendizado

1. Tudo começa com uma modelo pré-treinado;
  - Um cenário comum é o modelo treinado com ImageNet. Mas, qualquer outro que faça sentido para o problema pode ser usado.
2. Já comentamos que, a rede neural totalmente conectada ao fim de uma CNN faz a transferência das características extraídas para o que realmente importa, classificar!
3. A ideia é: Usar toda a estrutura de camadas convolucionais e pooling pré treinadas e, alterar a/as camada/as final/ais.



## CNN para Visão Computacional

### Visão Computacional

#### Transferência de Aprendizado

Duas abordagens:

- a. Usar a parte original da rede como extrator de características e apenas treinar a/as nova/as camada/as
  - Na atualização dos pesos é preciso travar os pesos da rede pré-treinada
- b. Ajuste fino na rede inteira
  - Faz-se um novo treinamento, com o conjunto de dados disponível, partindo dos pesos atuais da rede pré-treinada. É possível travar algumas camadas, como as iniciais, e deixar apenas as mais profundas (ao final da rede), para serem atualizadas. *Estas buscam por características mais específicas.*

*Inatel*



inatel



inateloficial



ascominatel



inatel.tecnologias



company/inatel

***Inatel***

Inatel - Instituto Nacional de Telecomunicações  
Campus em Santa Rita do Sapucaí - MG - Brasil  
Av. João de Camargo, 510 - Centro - 37540-000  
+55 (35) 3471 9200

Escritório em São Paulo - SP - Brasil  
WTC Tower, 18º andar - Conjunto 1811/1812  
Av. das Nações Unidas, 12.551 - Brooklin Novo - 04578-903  
+55 (11) 3043 6015