

Aprendizado Profundo 1

Aplicações de CNN

Professor: Lucas Silveira Kupssinskü

Agenda

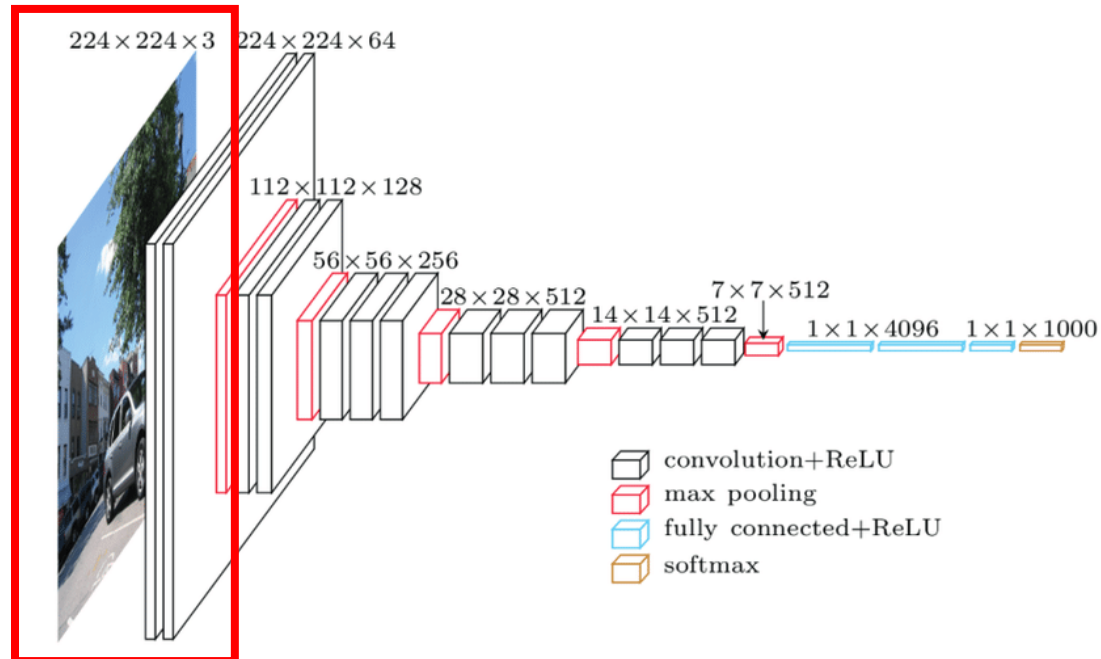
- VGG16
- Classificação de Imagens
- Classificação + Localização
- Segmentação Semântica
- Outras tarefas de visão computacional

VGG

- Uma arquitetura de CNN usada até hoje
- Proposta em 2013, com primeiro protótipo em 2014
- Taxa de erro ImageNet
 - Top-5: 6.8%
 - Top-1: 23.7%
- Ganhou 1º e 2º lugar no ILSVRC (*ImageNet Large Scale Visual Recognition Challenge*)
- $\pm 140M$ parâmetros treináveis

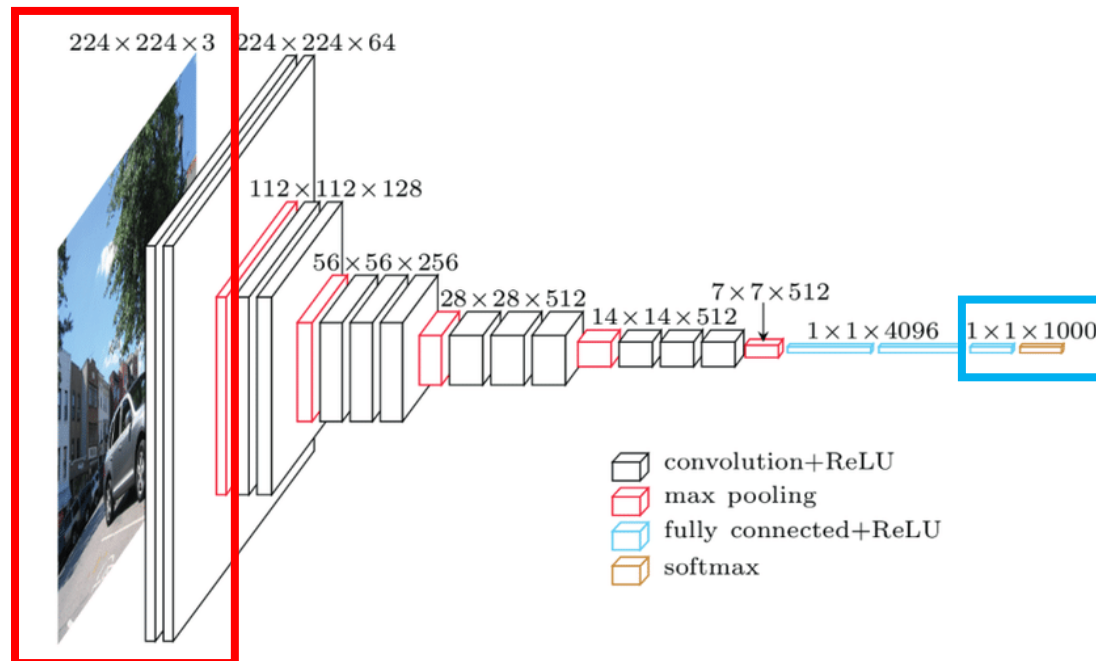
VGG

- Vamos compreender o problema
 - ImageNet contém imagens na resolução $224 \times 224 \times 3$ (RGB)
 - As entradas são tensores $(224, 224, 3)$



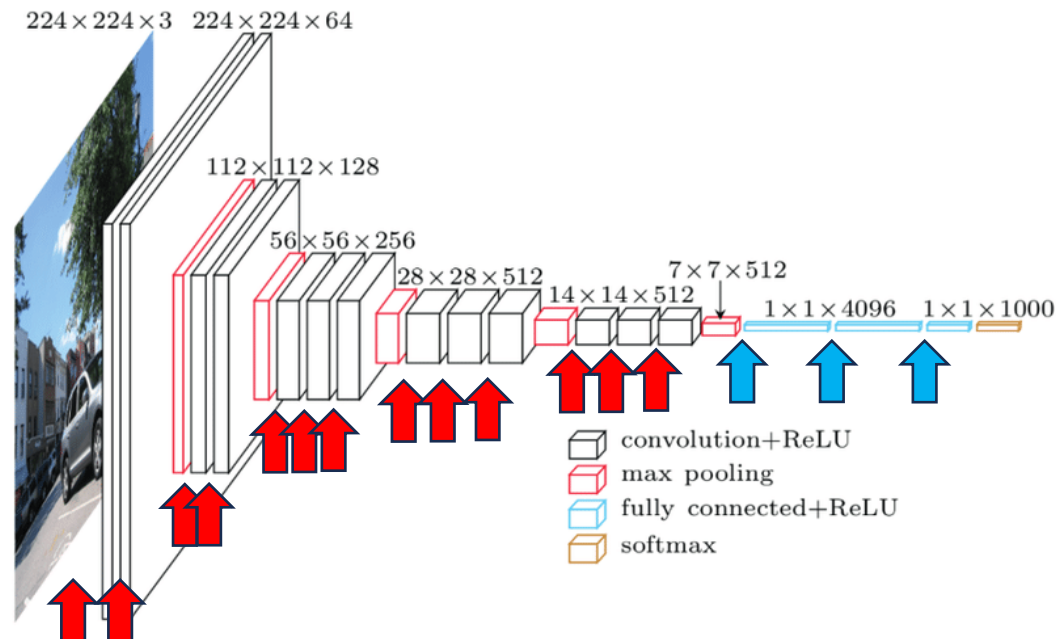
VGG

- Vamos compreender o problema
 - ImageNet contém imagens na resolução $224 \times 224 \times 3$ (RGB)
 - As entradas são tensores (224, 224, 3)
 - Objetivo é classificar cada imagem em uma das 1000 classes pré-definidas
 - Entropia Cruzada com os logits



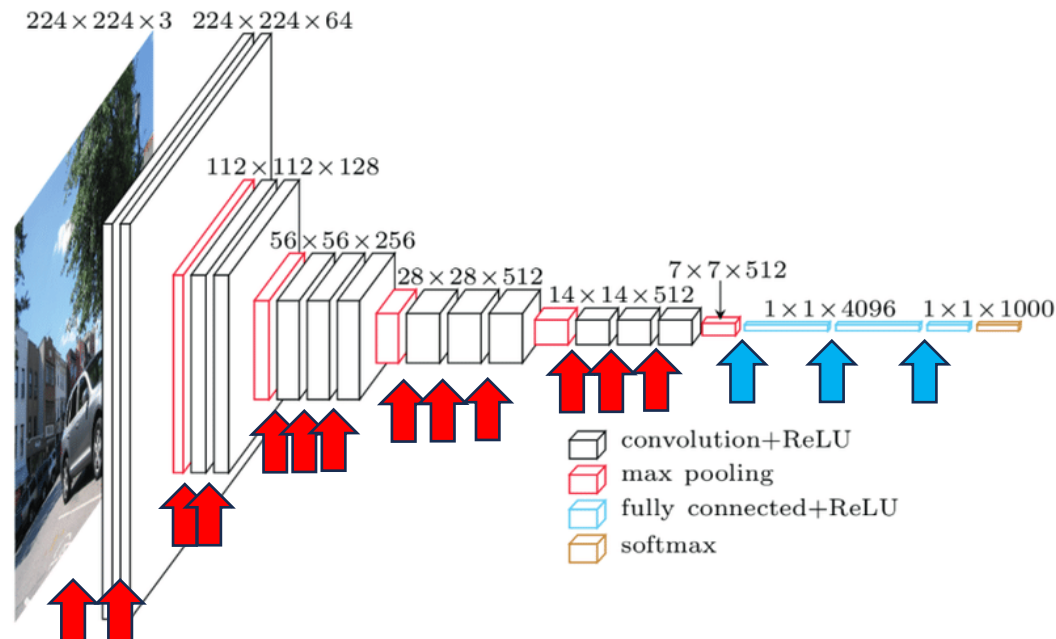
VGG

- Precisamos transformar uma imagem ($224 \times 224 \times 3$) para uma classe ($1 \times 1 \times 1000$)
 - Convoluções + ReLU: aumentam os canais
 - Agrupamentos (pooling): subamostram a dimensão espacial
 - Camadas densas: fazem a classificação



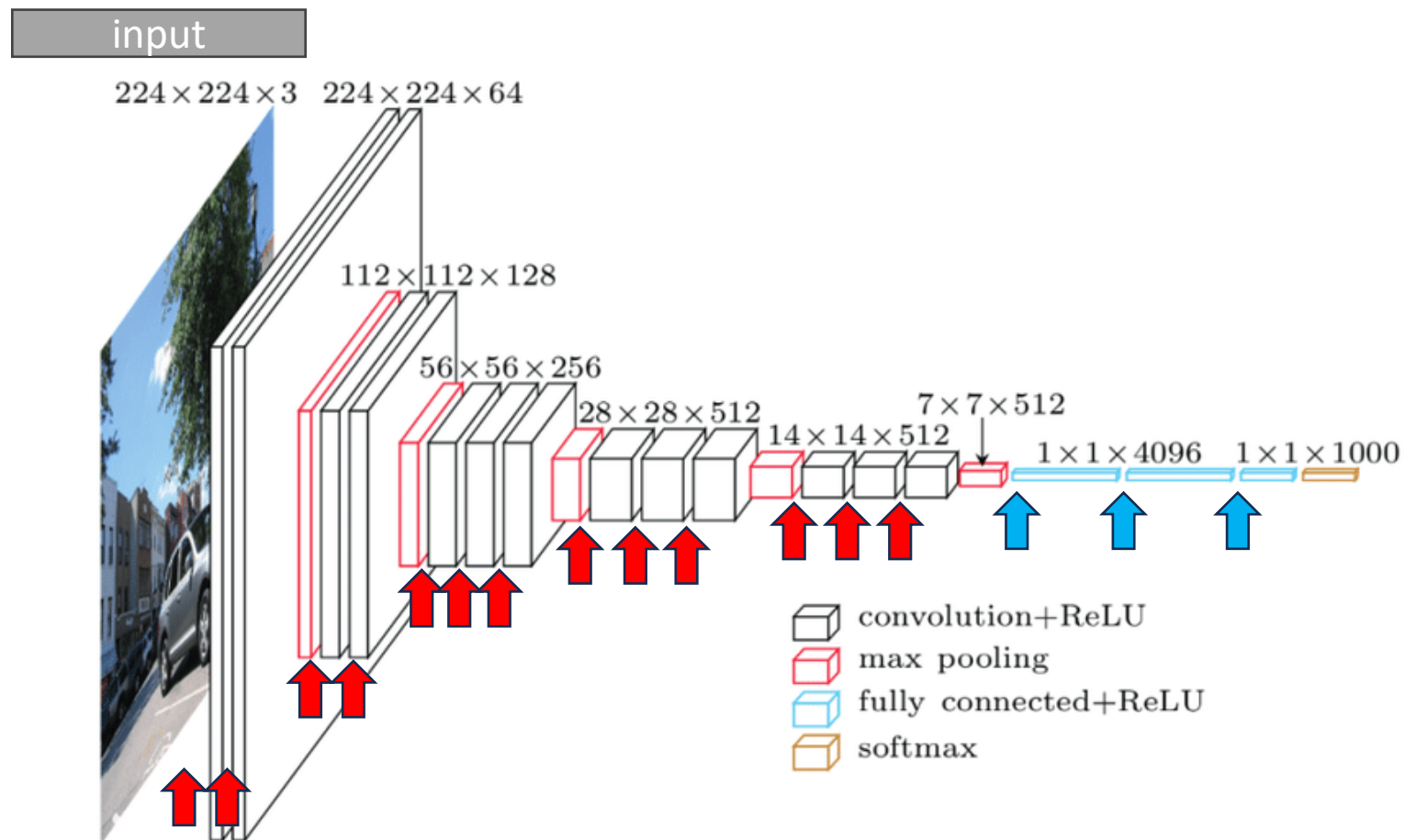
VGG

- VGG16 é a implementação mais comum
 - Equivalente a variante “D” descrita no artigo
 - 16 camadas com parâmetros treináveis (13 convs e 3 densas)
 - Repare todas as conv são 2D 3x3 com stride 1



VGG

Total de Parâmetros Treináveis: 0

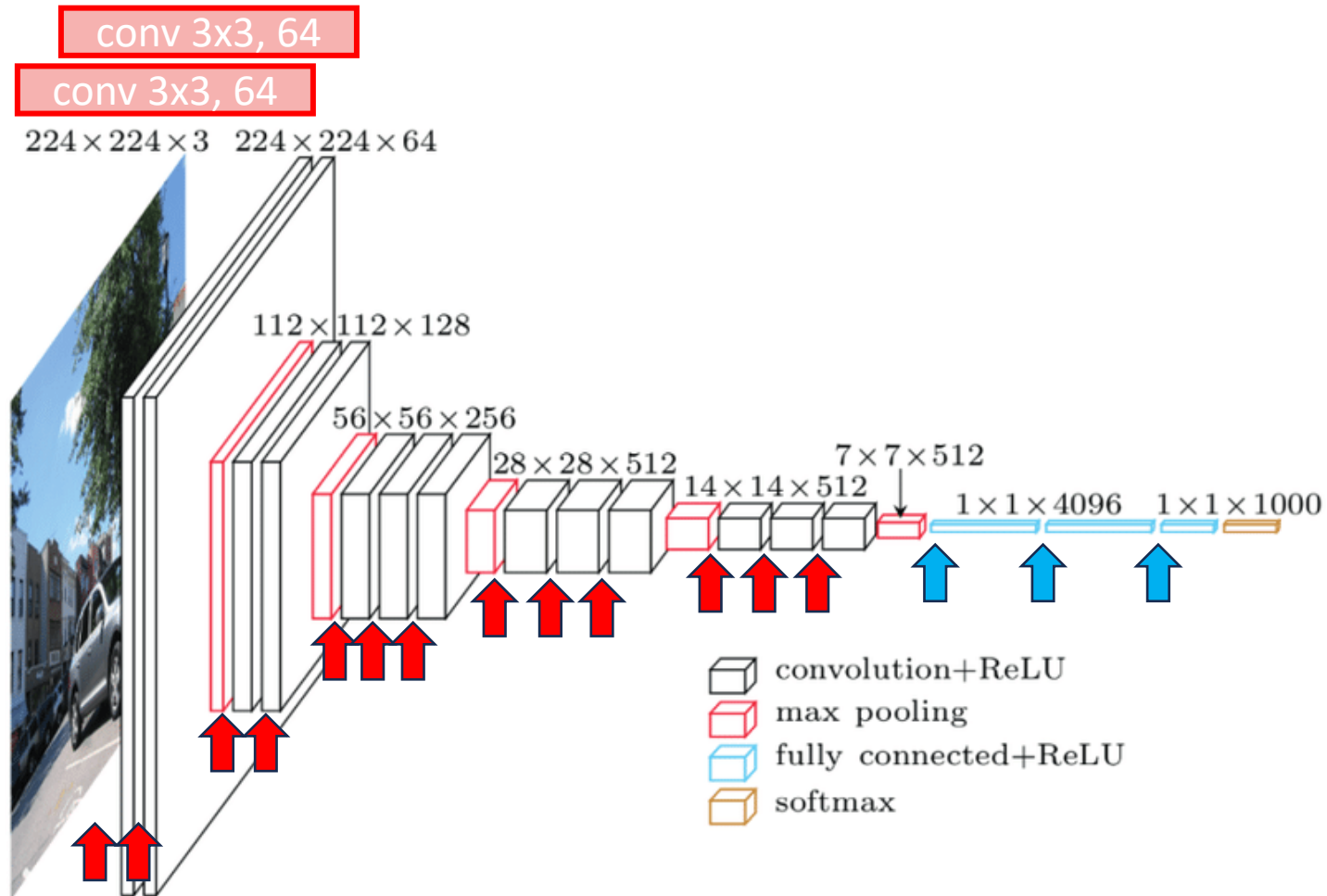


VGG

Total de Parâmetros Treináveis: 38.592

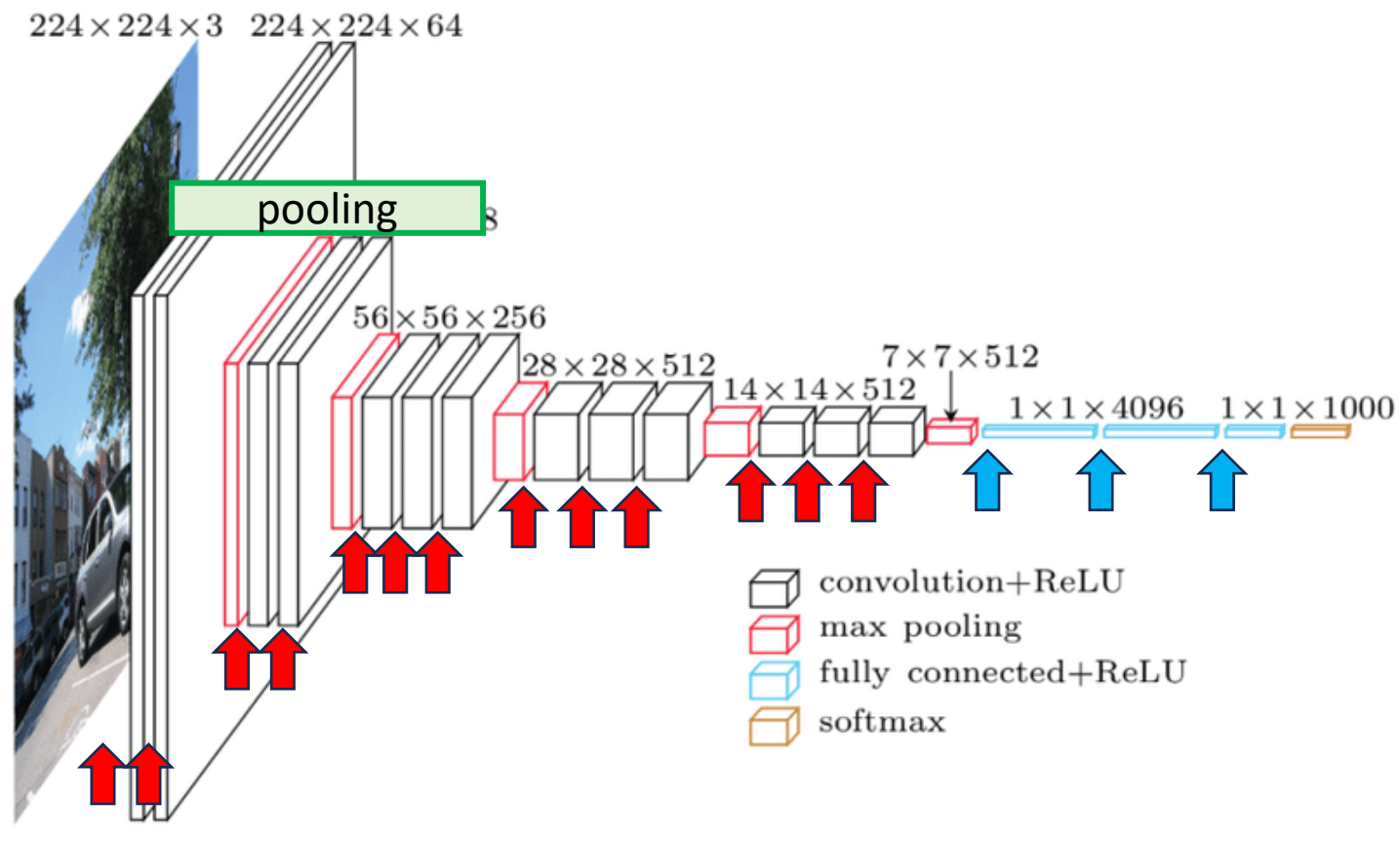
$3 \times 3 \times 3 \times 64 + 3 \times 3 \times 64 \times 64$

$1728 + 36864$



VGG

Total de Parâmetros Treináveis: 38.592
0

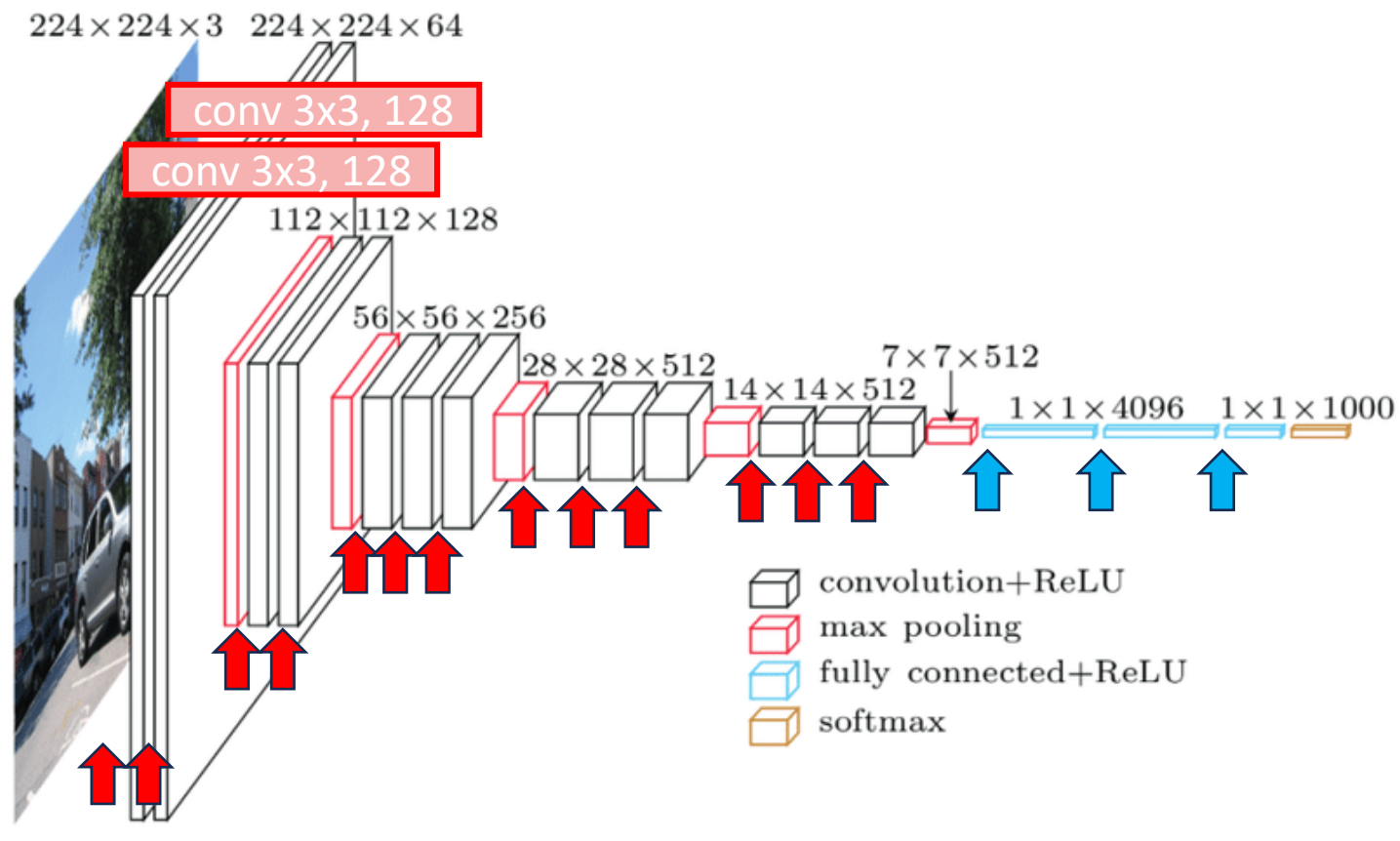


VGG

Total de Parâmetros Treináveis: 259.776

$$3 \times 3 \times 64 \times 128 + 3 \times 3 \times 128 \times 128$$

$$73728 + 147456$$



pooling

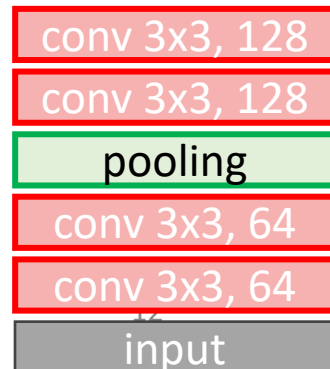
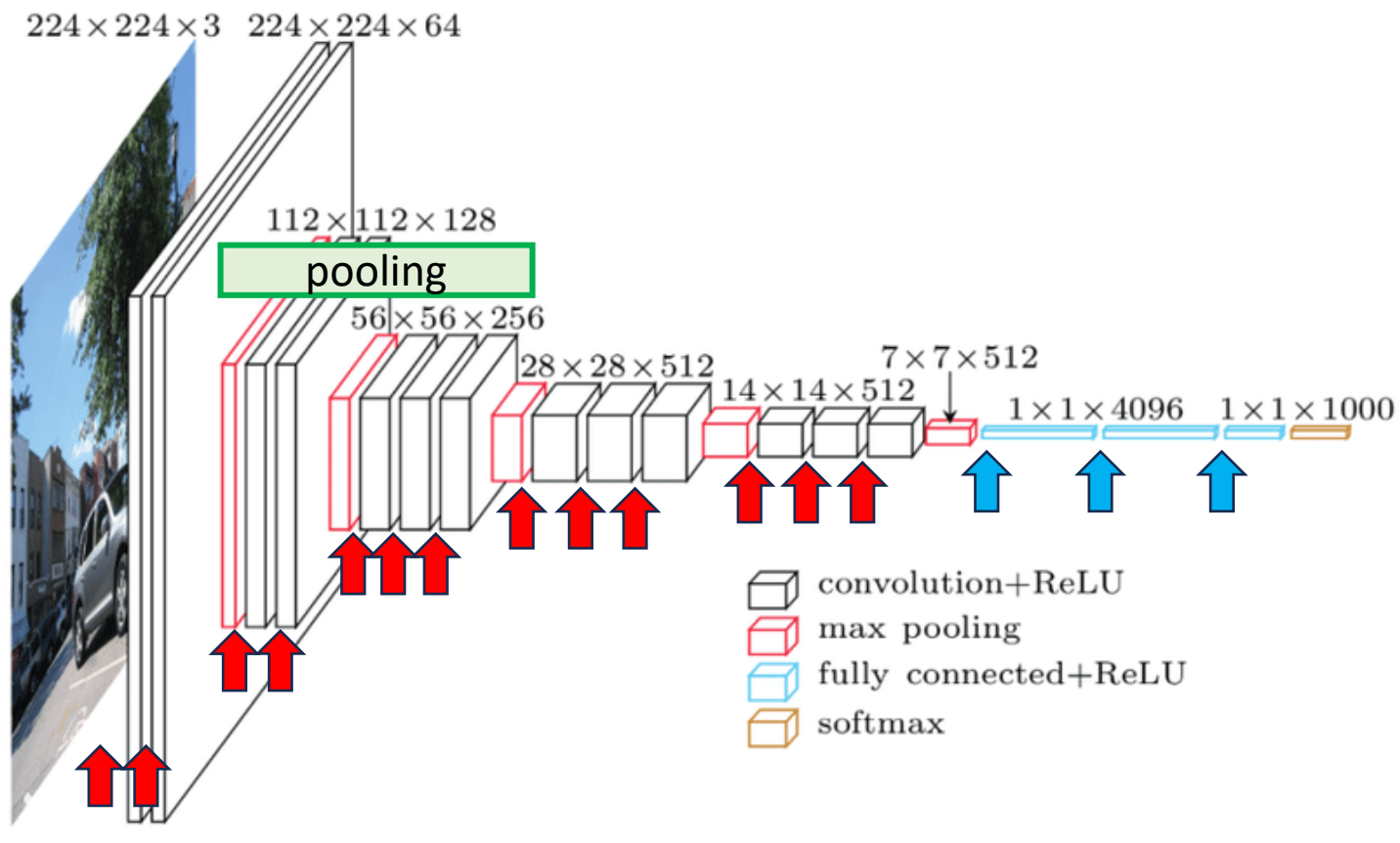
conv 3x3, 64

conv 3x3, 64

input

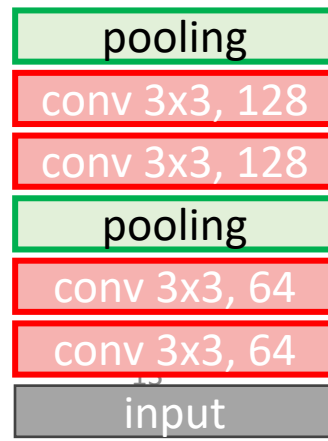
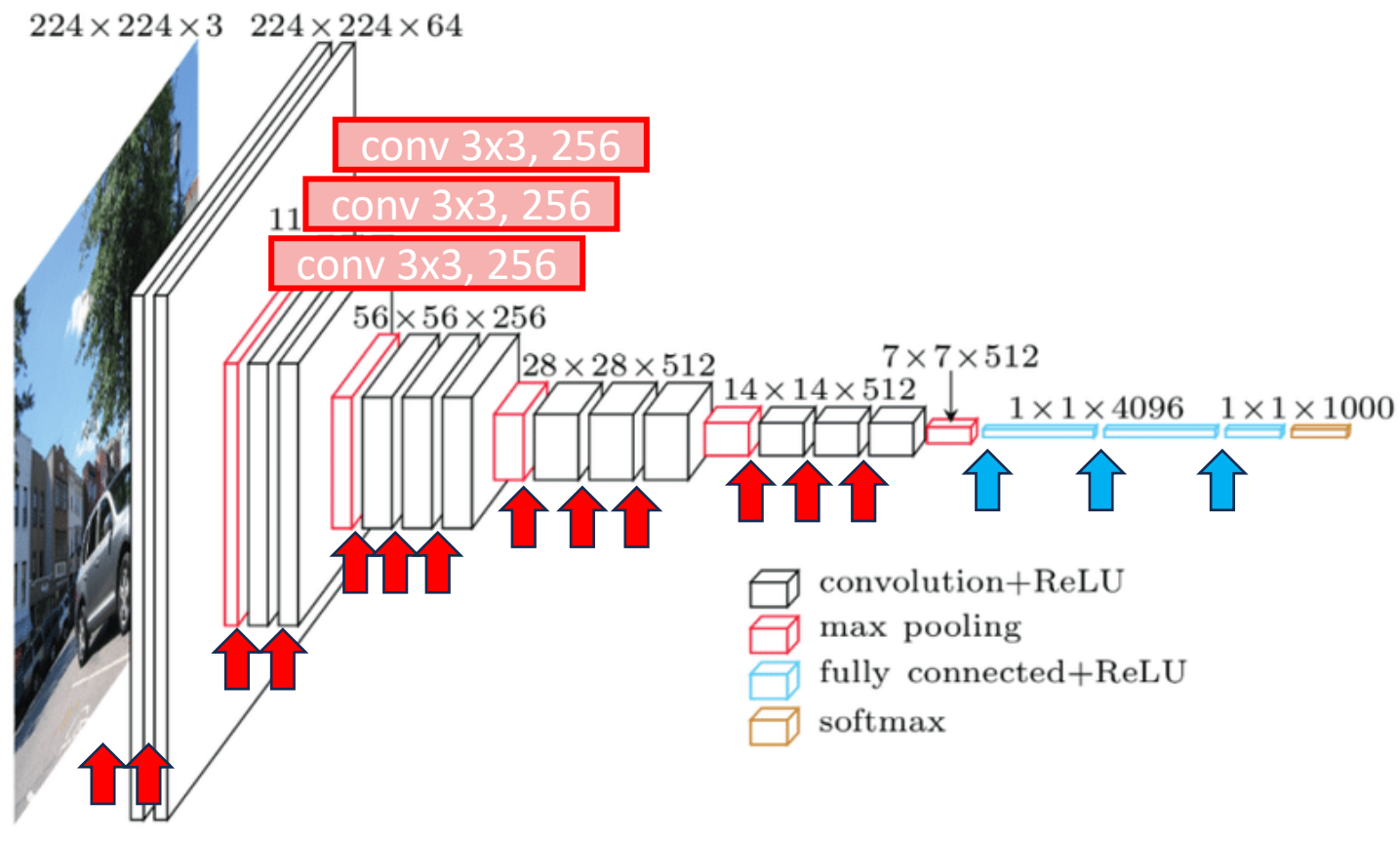
VGG

Total de Parâmetros Treináveis: 259.776
0



VGG

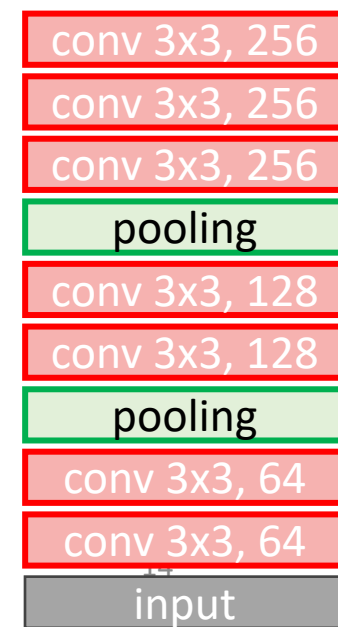
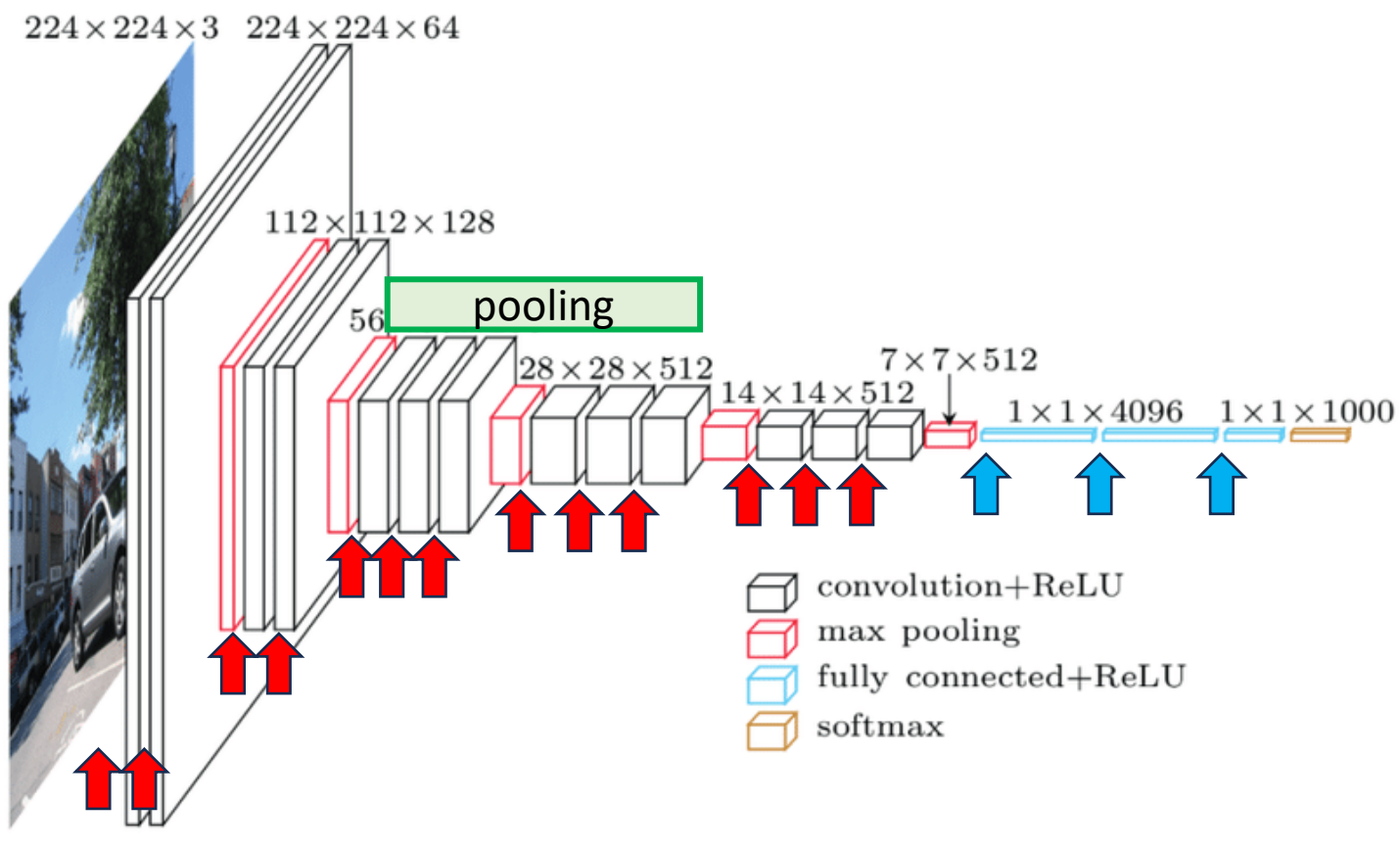
Total de Parâmetros Treináveis: 1.734.336
 $3 \times 3 \times 128 \times 256 + 3 \times 3 \times 256 \times 256 + 3 \times 3 \times 256 \times 256$
 $294912 + 589824 + 589824$



VGG

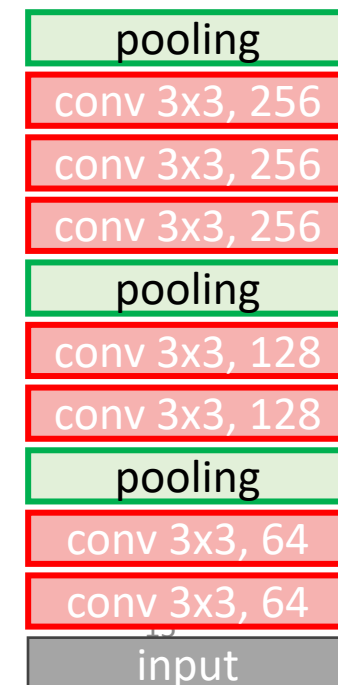
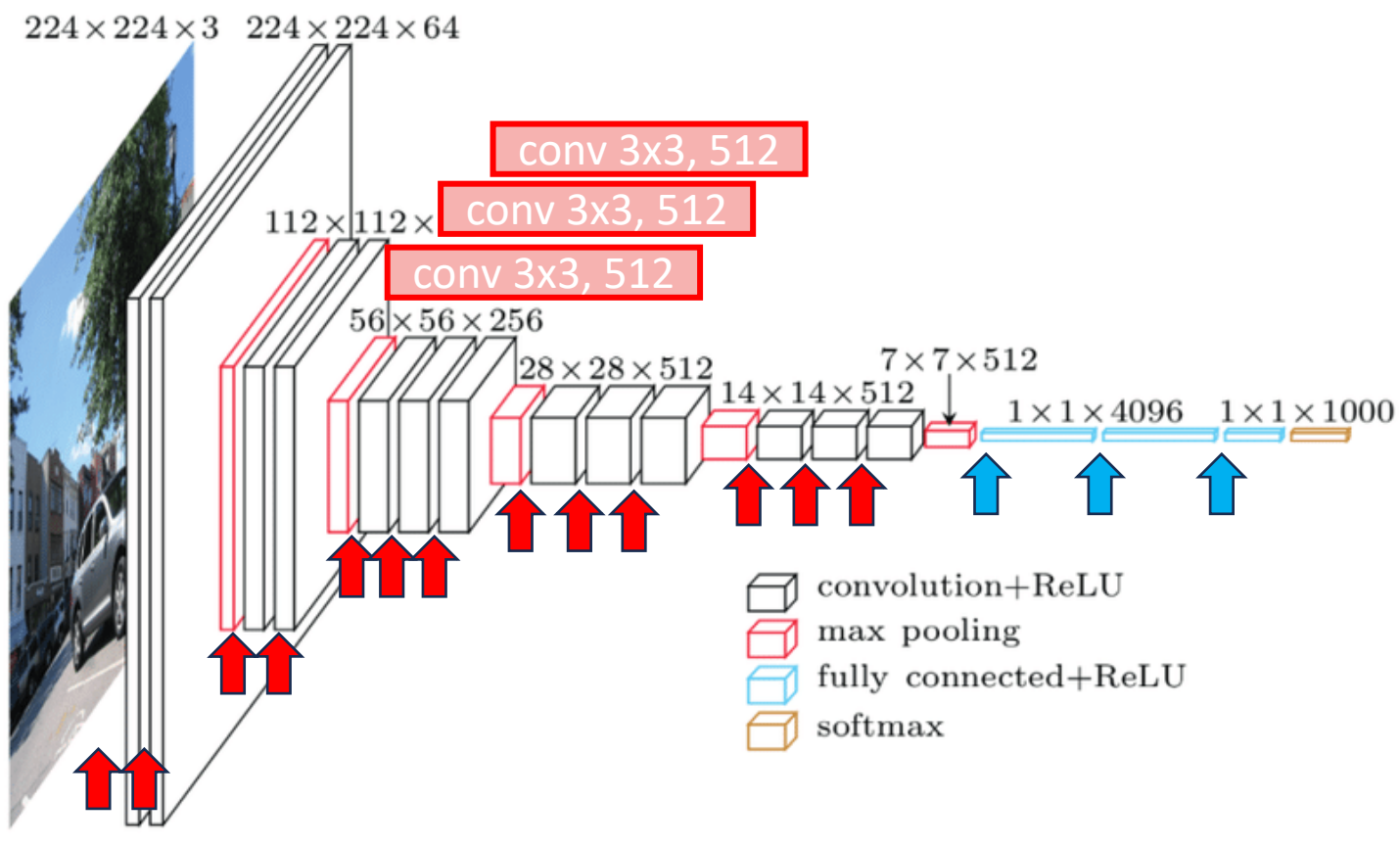
Total de Parâmetros Treináveis: 1.734.336

0



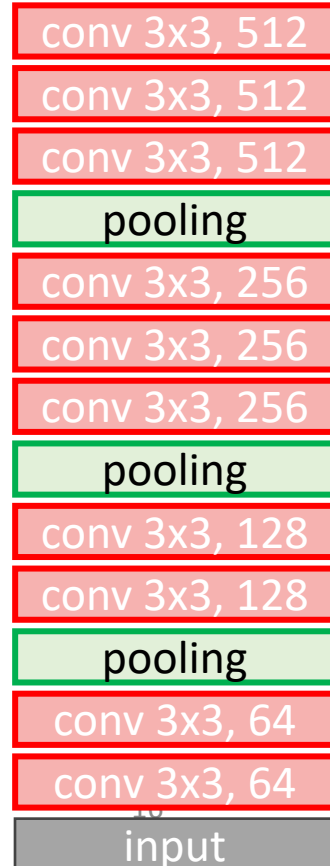
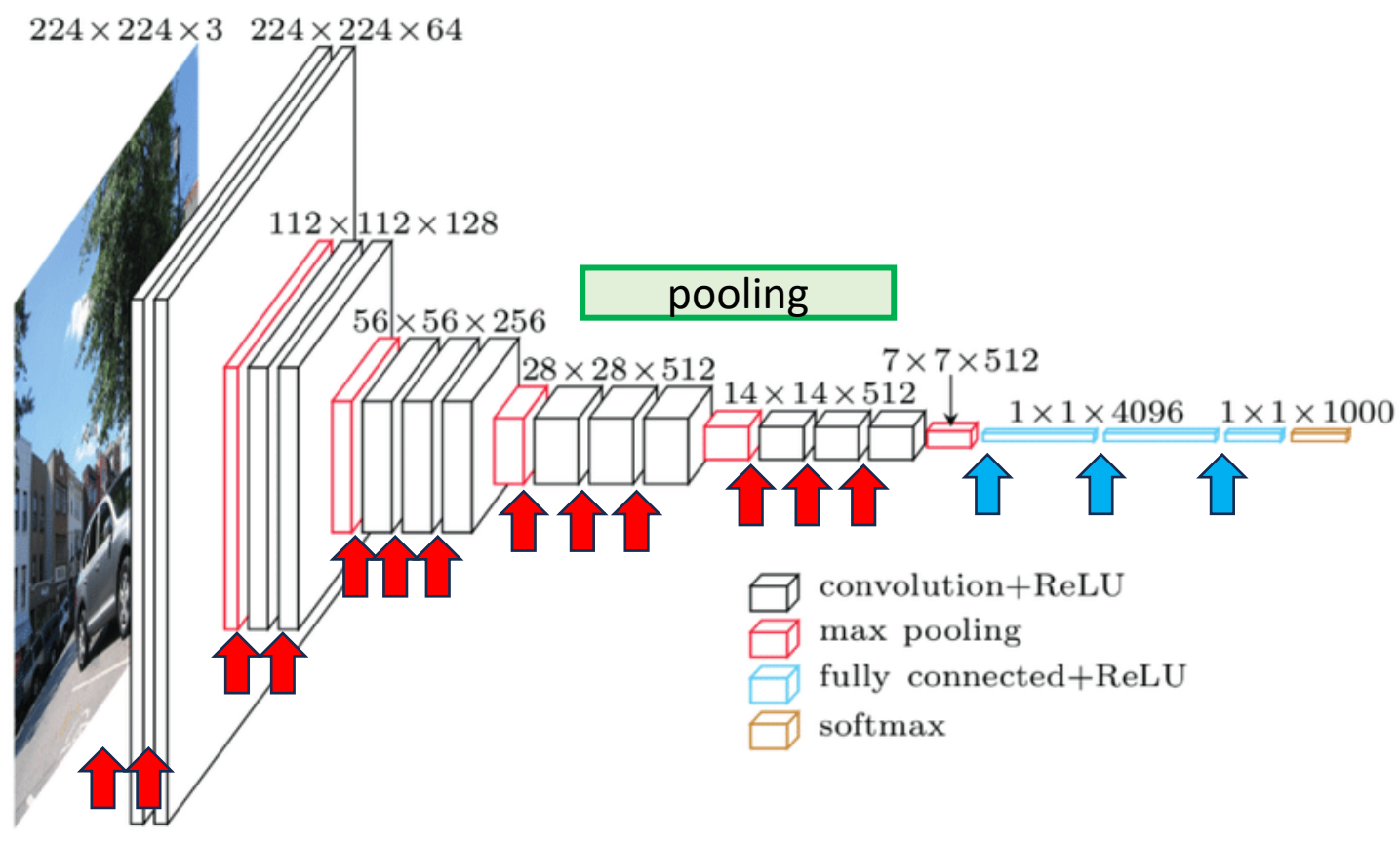
VGG

Total de Parâmetros Treináveis: 7.632.576
 $3 \times 3 \times 256 \times 512 + 3 \times 3 \times 512 \times 512 + 3 \times 3 \times 512 \times 512$
 $1179648 + 2359296 + 2359296$



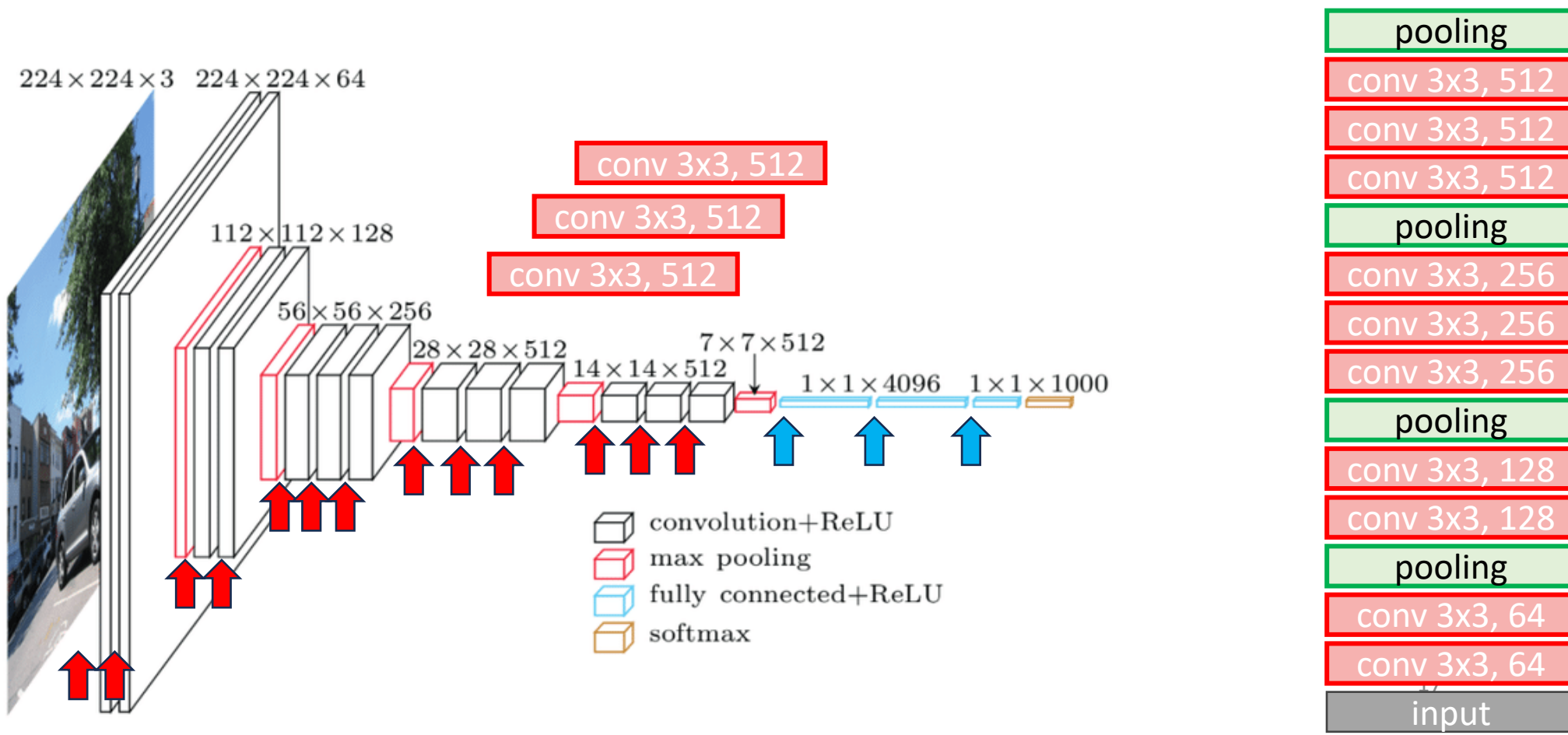
VGG

Total de Parâmetros Treináveis: 7.632.576
0



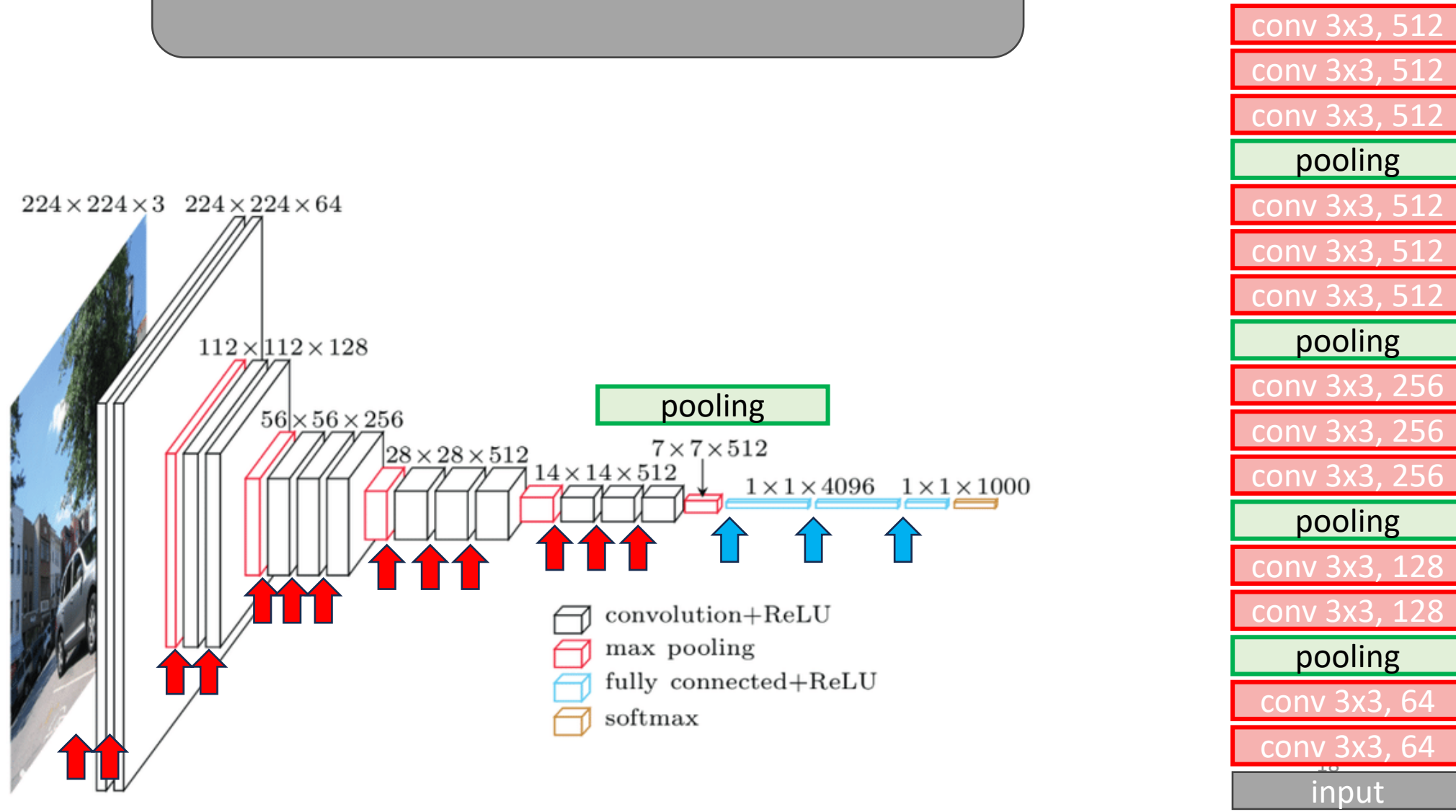
VGG

Total de Parâmetros Treináveis: 14.710.464
 $3 \times 3 \times 512 \times 512 + 3 \times 3 \times 512 \times 512 + 3 \times 3 \times 512 \times 512$
 $2359296 + 2359296 + 2359296$



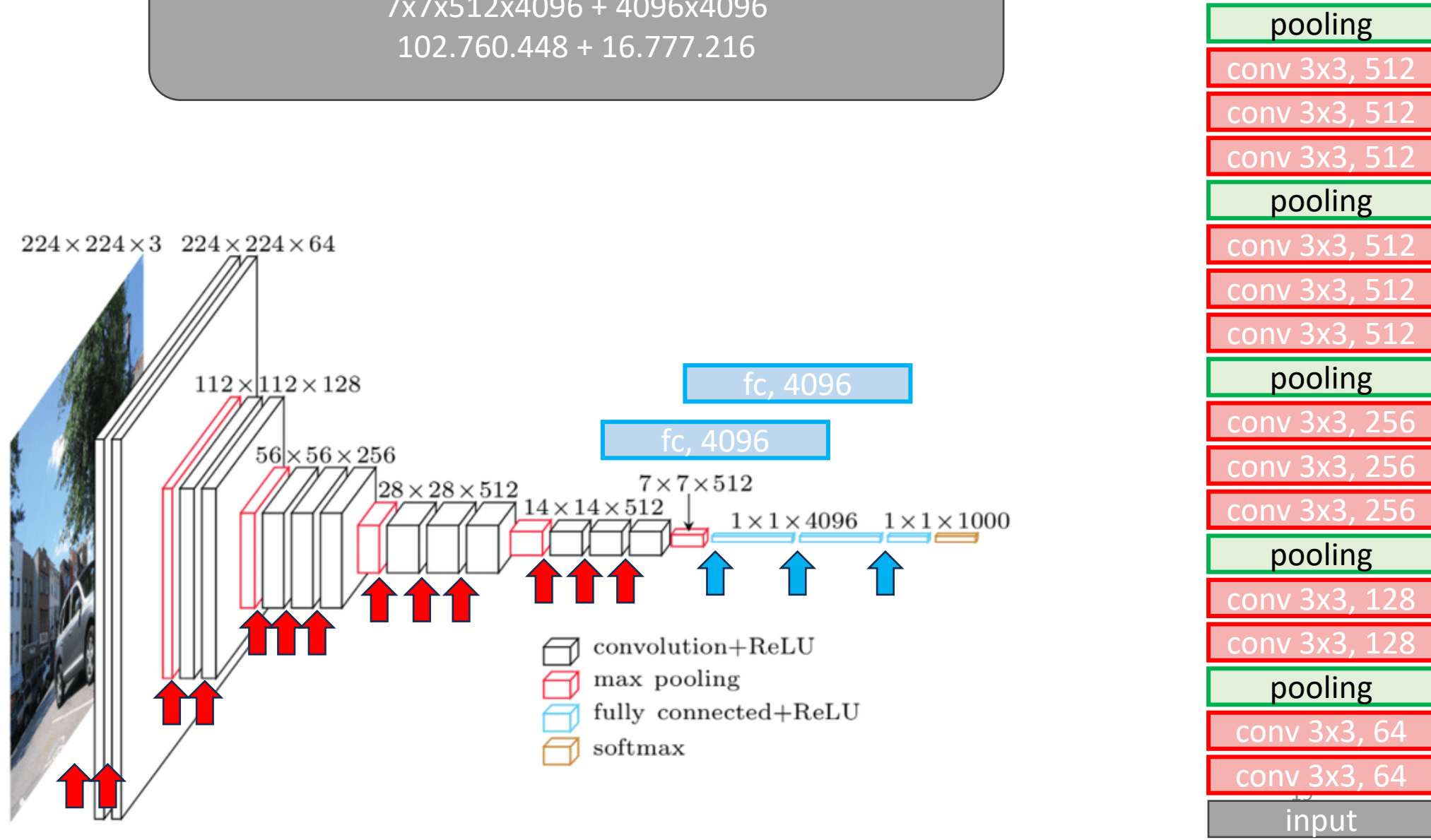
VGG

Total de Parâmetros Treináveis: 14.710.464
0



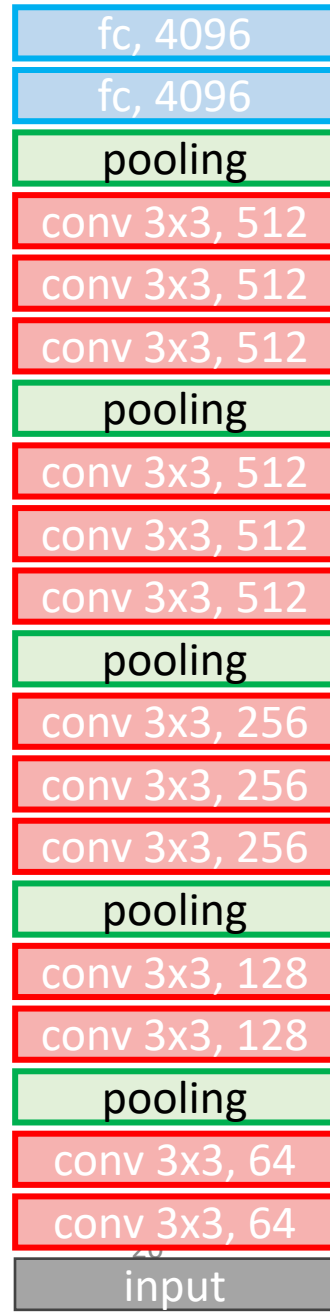
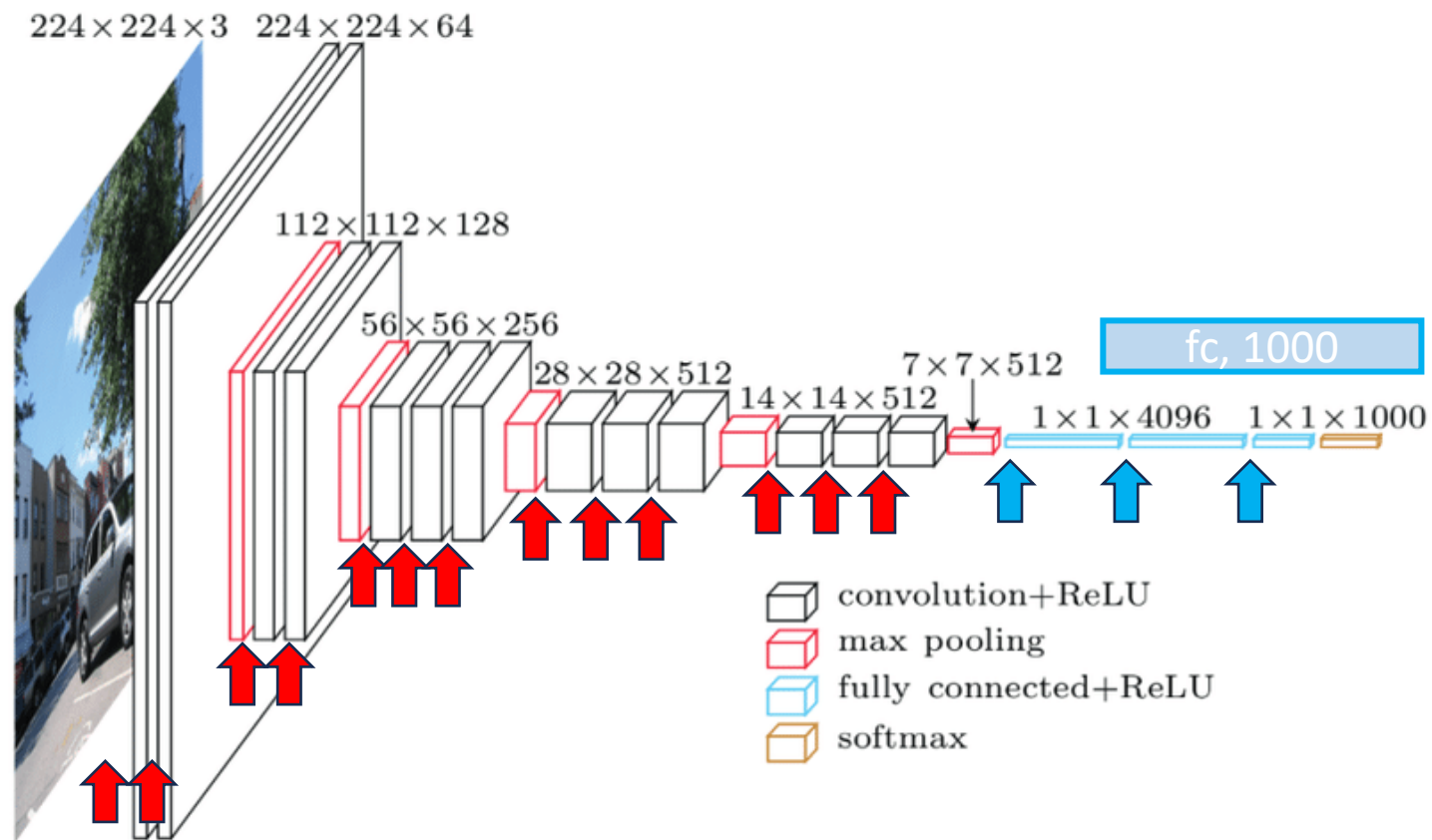
VGG

Total de Parâmetros Treináveis: 134.248.128
 $7 \times 7 \times 512 \times 4096 + 4096 \times 4096$
 $102.760.448 + 16.777.216$



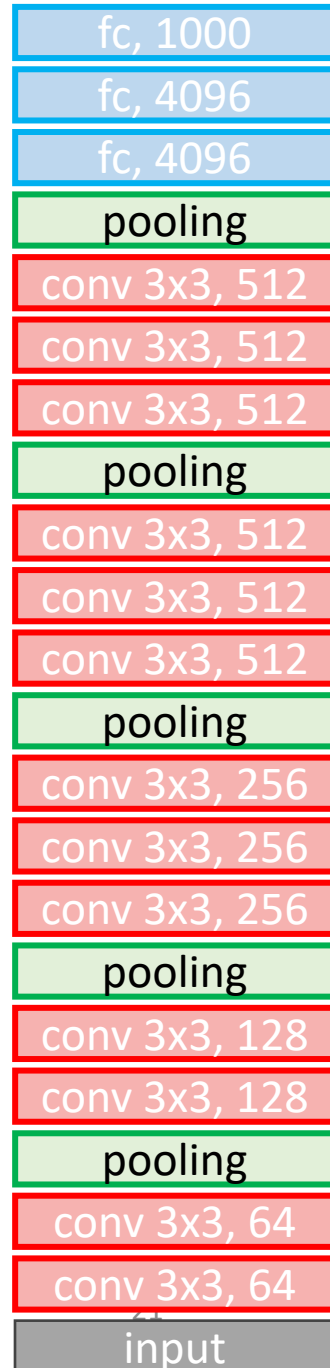
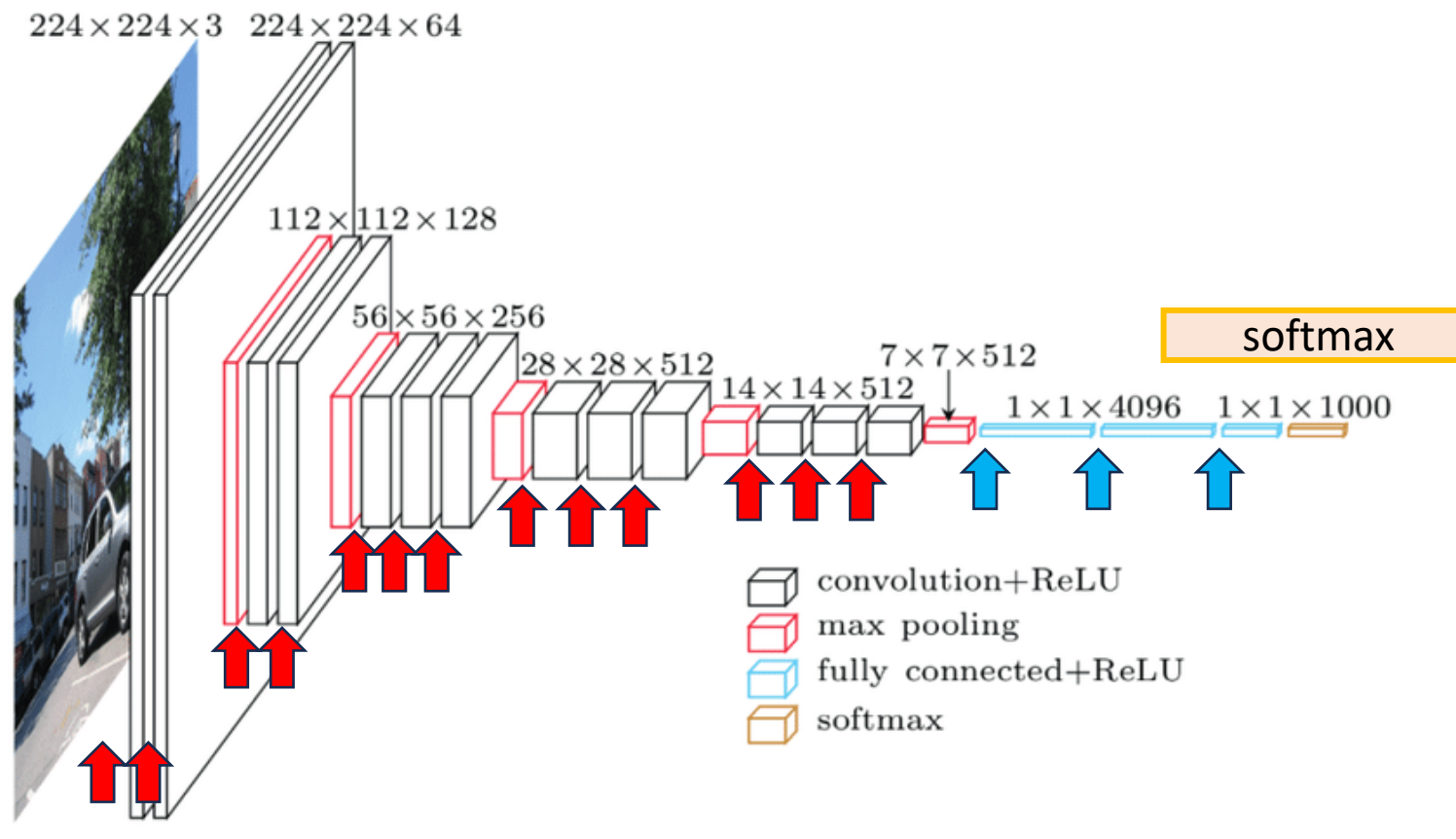
VGG

Total de Parâmetros Treináveis: 138.344.128
4096x1000
4.096.000



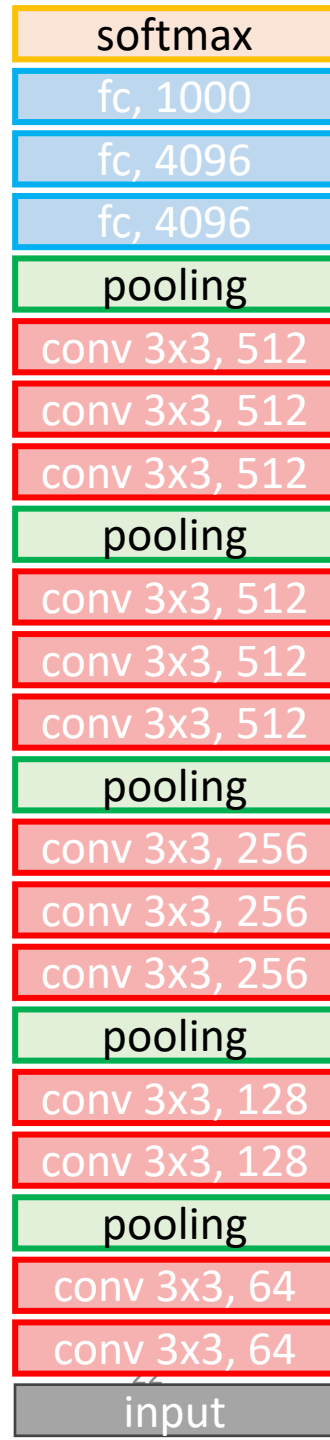
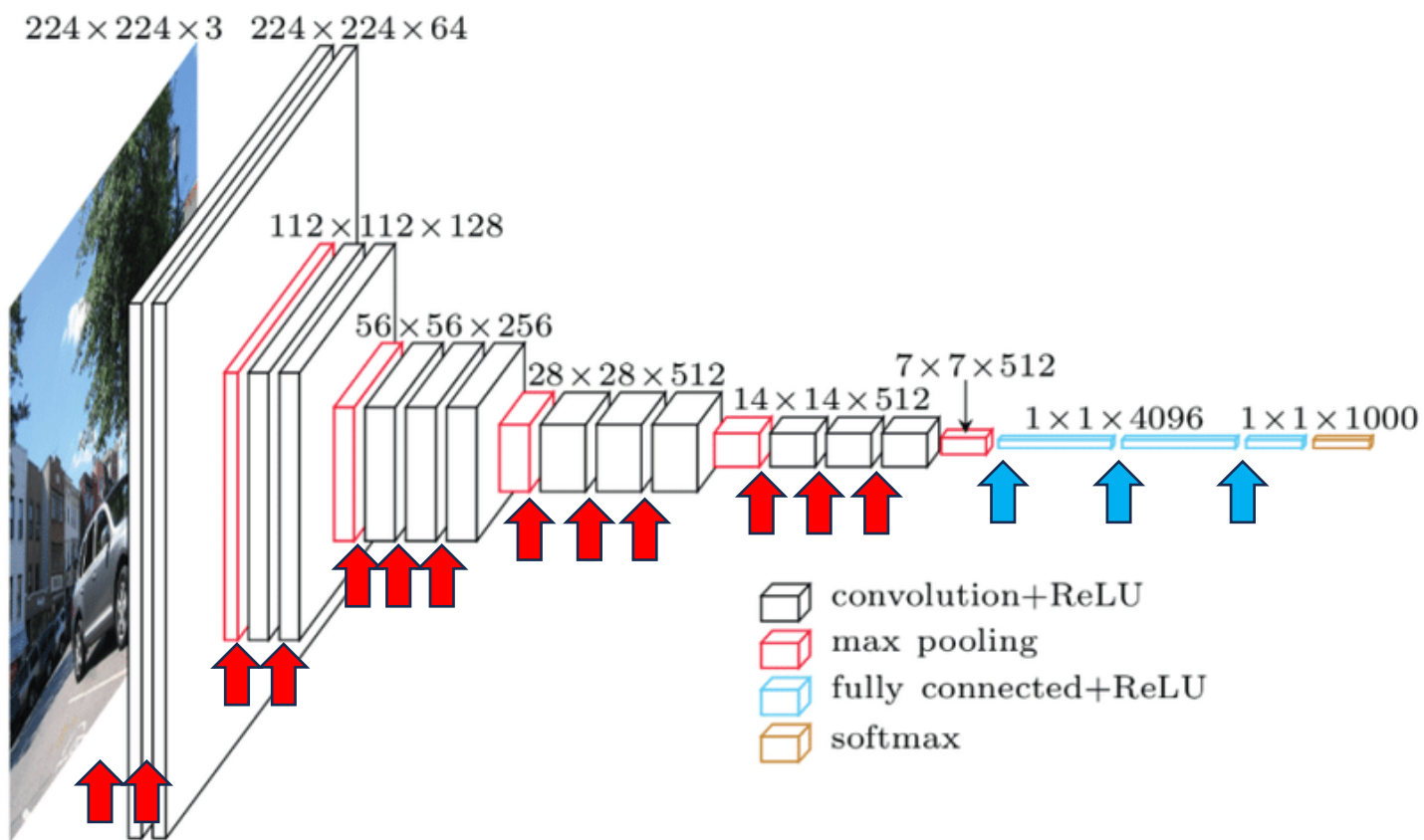
VGG

Total de Parâmetros Treináveis: 138.344.128
0



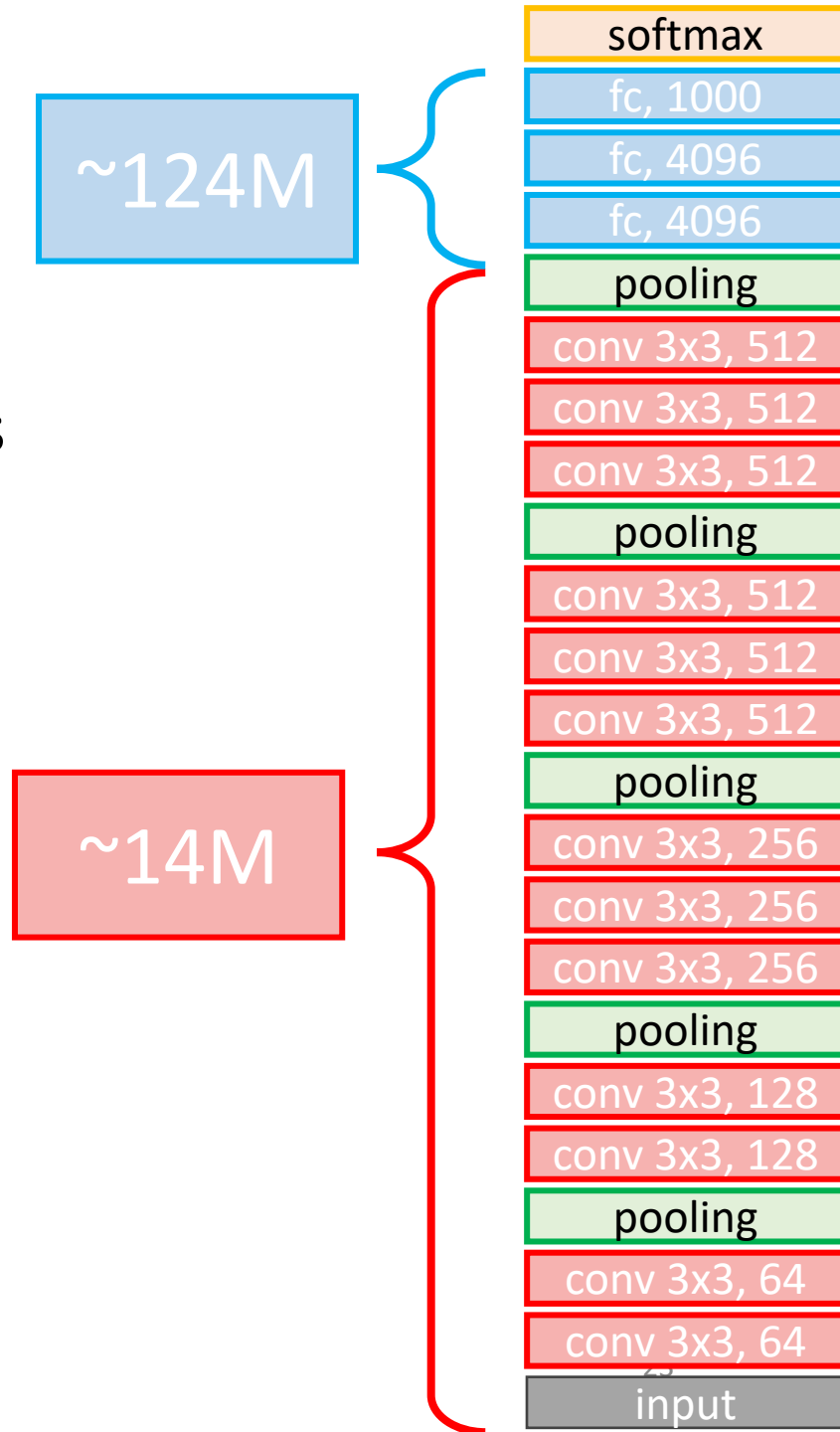
VGG

Total de Parâmetros Treináveis: 138.344.128
Observação: não contamos os *bias*



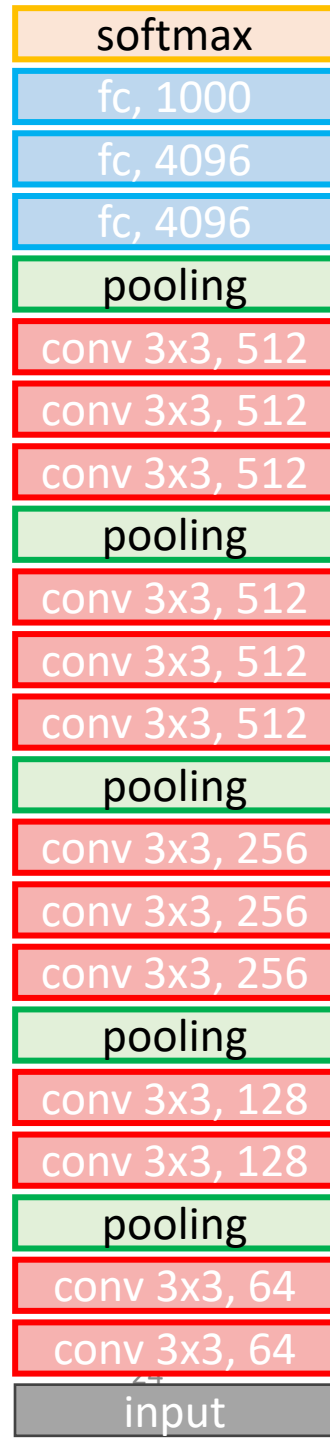
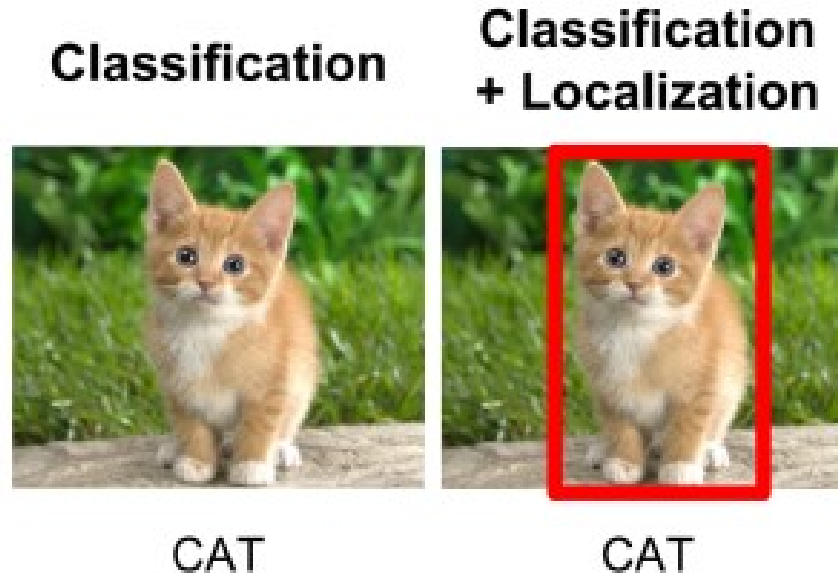
VGG

- Repare que a maioria dos parâmetros treináveis encontra-se nas camadas densas (MLP)



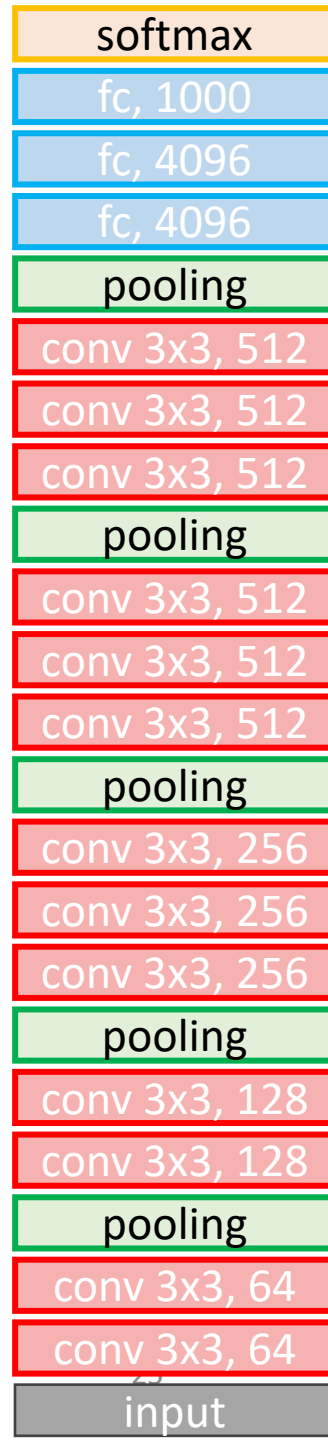
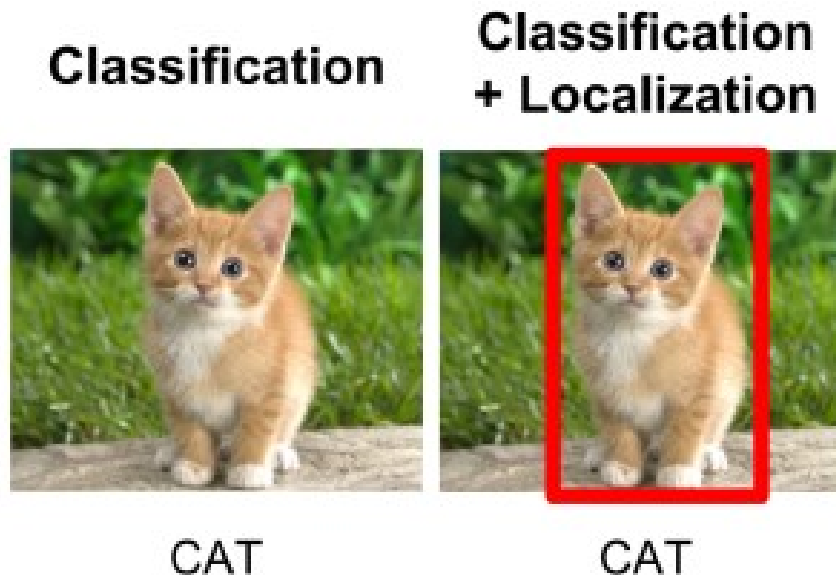
Localização

- Além de classificar a imagem podemos localizar o objeto
- Como?



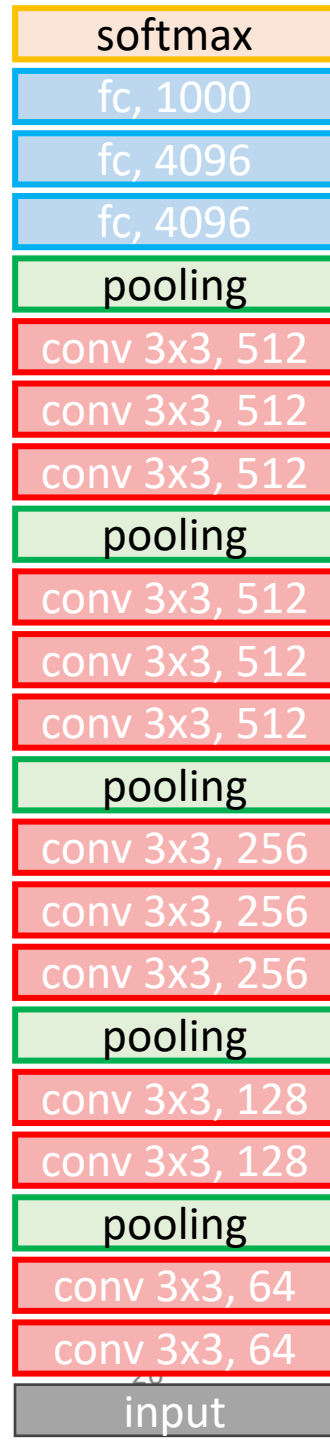
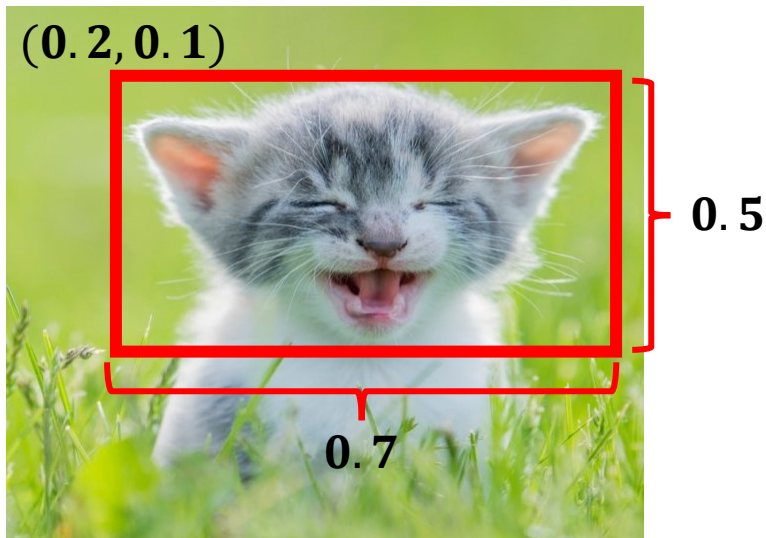
Localização

- Para desenhar uma *bounding box* precisamos de 4 valores
 - (x, y) do centro ou de um dos cantos
 - altura e largura
- Podemos modelar esses 4 valores como um problema de regressão multivariada



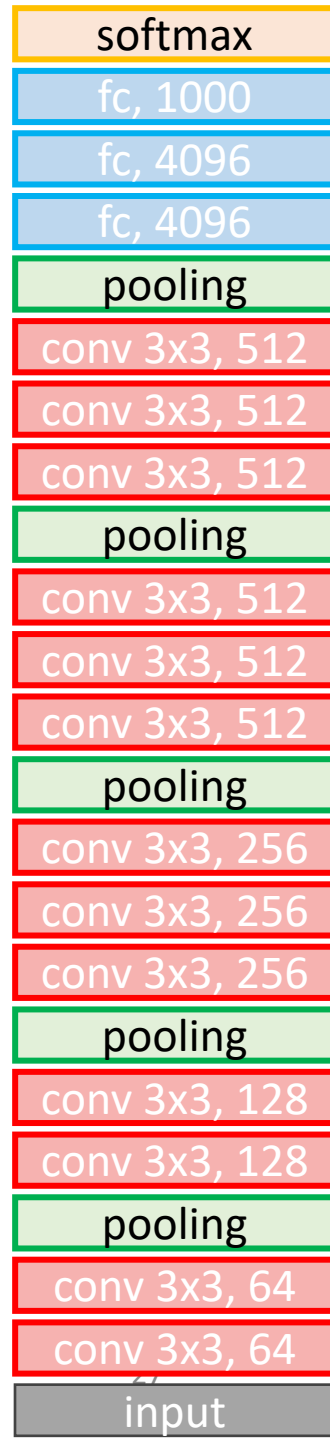
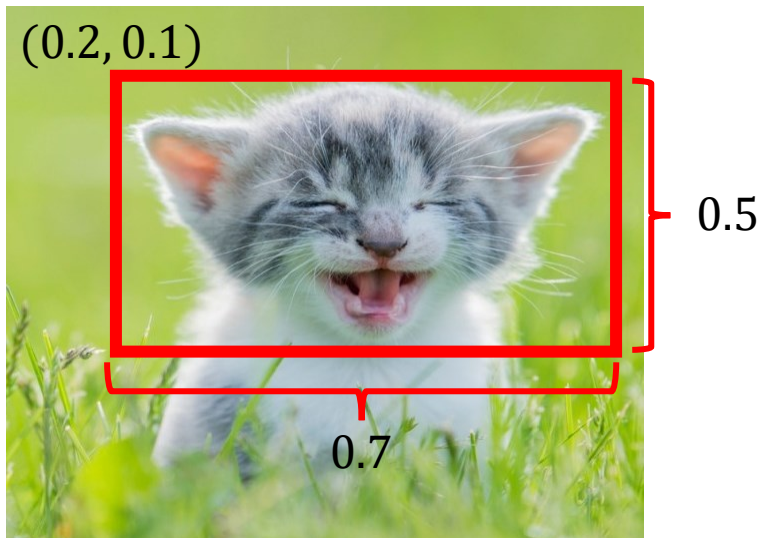
Localização

- Valor alvo $y = (x, y, h, w)$
 - Ponto superior esquerdo da imagem é (0,0)
 - Ponto inferior direito é (1,1)



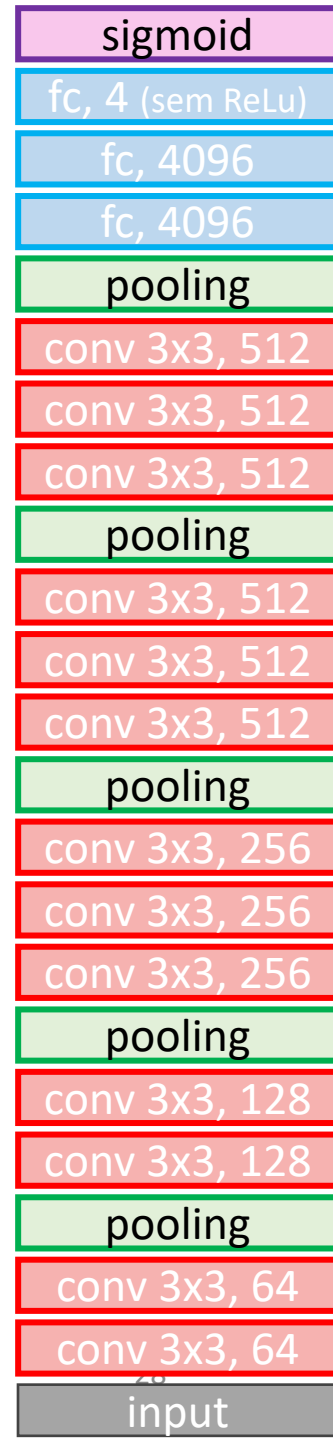
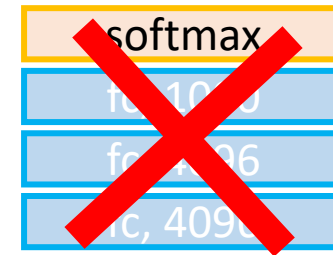
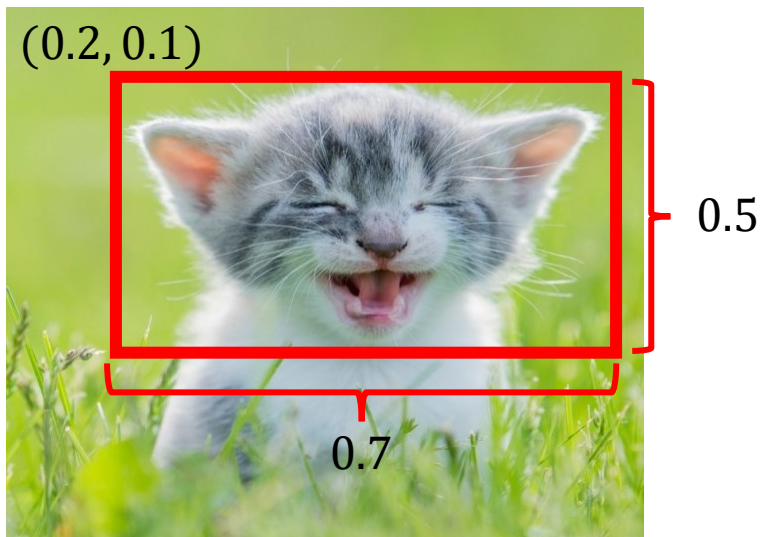
Localização

- Valor alvo $y = (x, y, h, w)$
 - Ponto superior esquerdo da imagem é (0,0)
 - Ponto inferior direito é (1,1)
- O *bounding box* abaixo pode ser representado por
 - (0.2, 0.1, 0.5, 0.7)



Localização

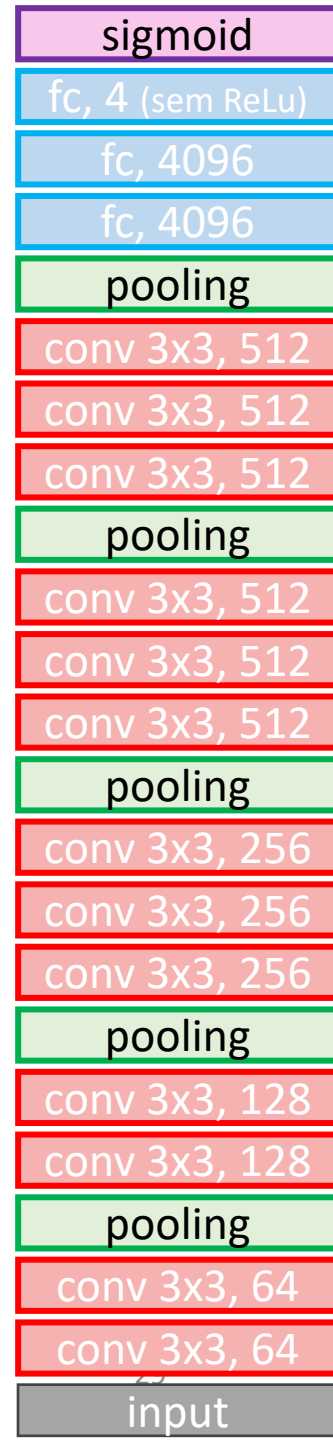
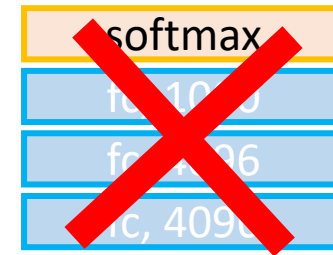
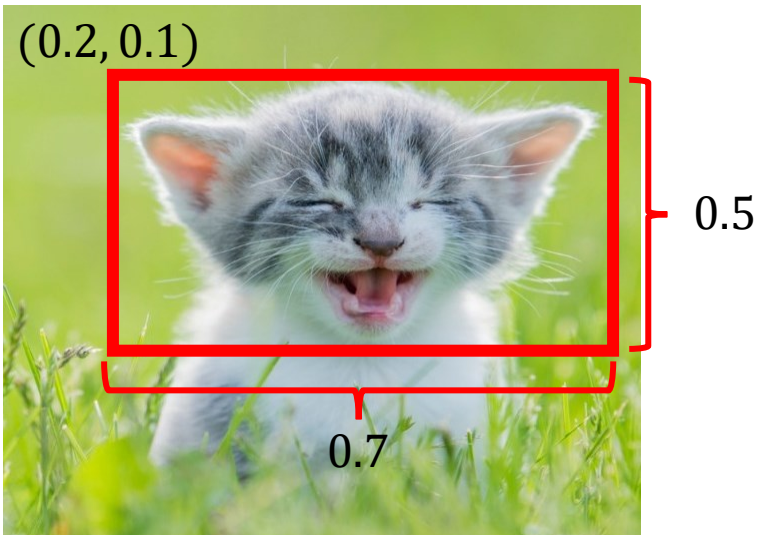
- O *bounding box* abaixo pode ser representado por
 - $(0.2, 0.1, 0.5, 0.7)$
- A saída da rede deve ter 4 unidades Sigmoid



Localização

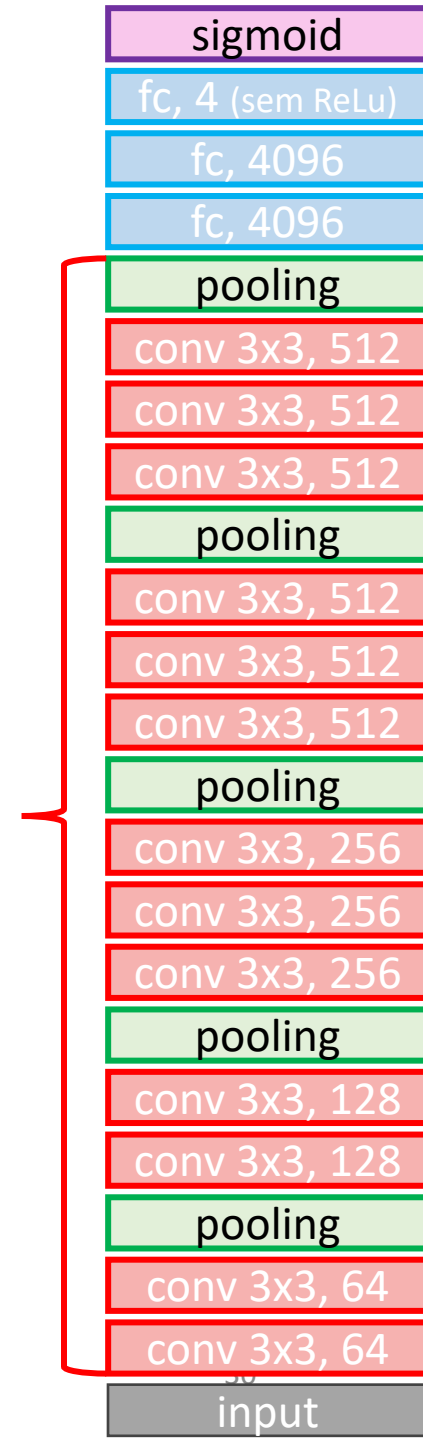
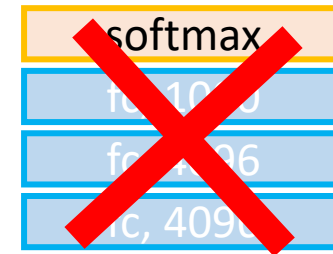
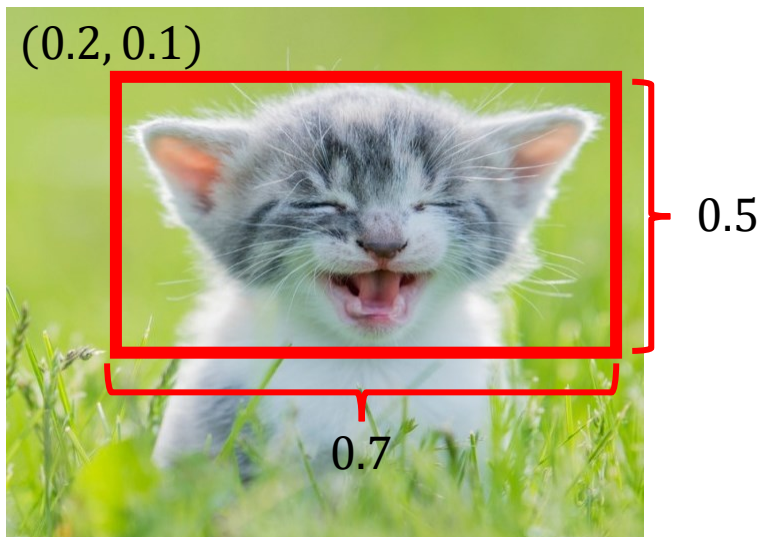
- O *bounding box* abaixo pode ser representado por
 - (0.2, 0.1, 0.5, 0.7)
- A saída da rede deve ter 4 unidades Sigmoid
- Podemos usar a loss MSE

$$\frac{1}{2N} \sum_{i=1}^N \left[\left(y_0^{(i)} - \hat{y}_0^{(i)} \right)^2 + \left(y_1^{(i)} - \hat{y}_1^{(i)} \right)^2 + \left(y_2^{(i)} - \hat{y}_2^{(i)} \right)^2 + \left(y_3^{(i)} - \hat{y}_3^{(i)} \right)^2 \right]$$



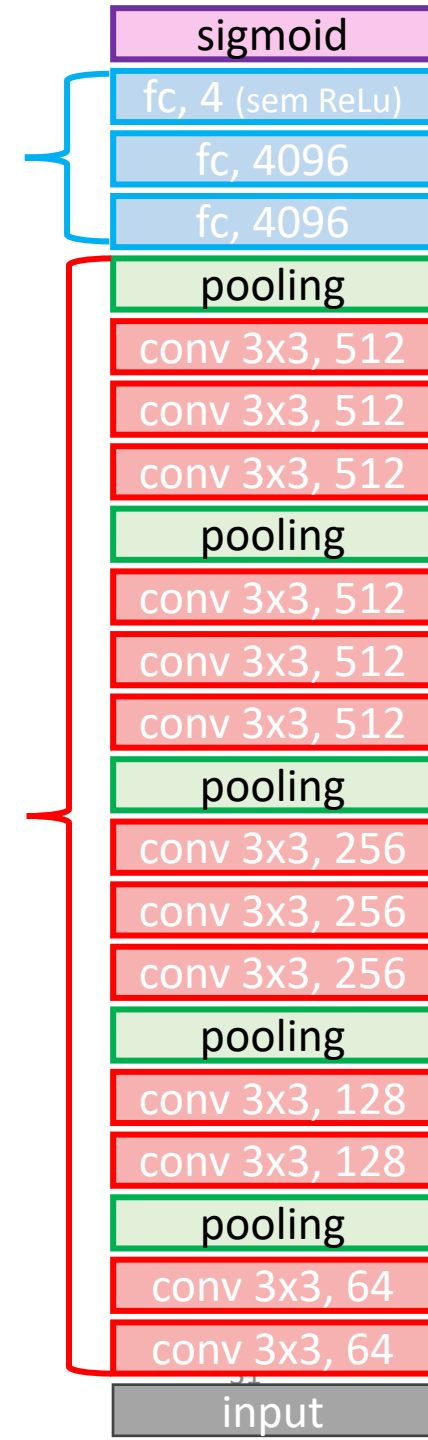
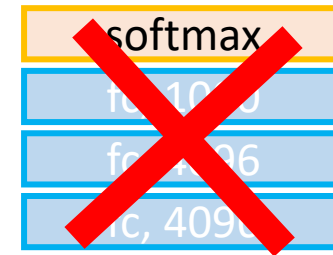
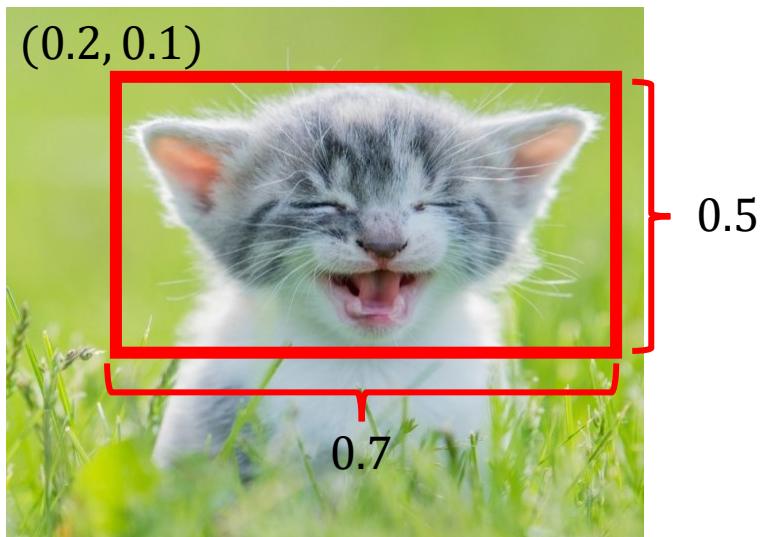
Localização

- Imagine que já treinamos a rede convolucional para classificar as fotos
 - Os parâmetros das convoluções já estão ajustados

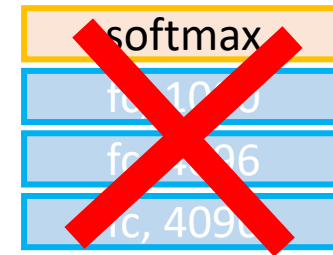


Localização

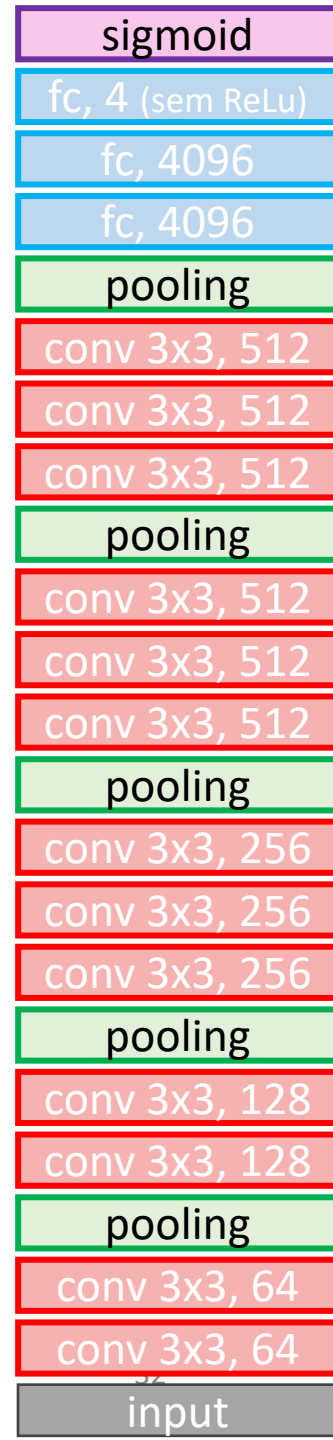
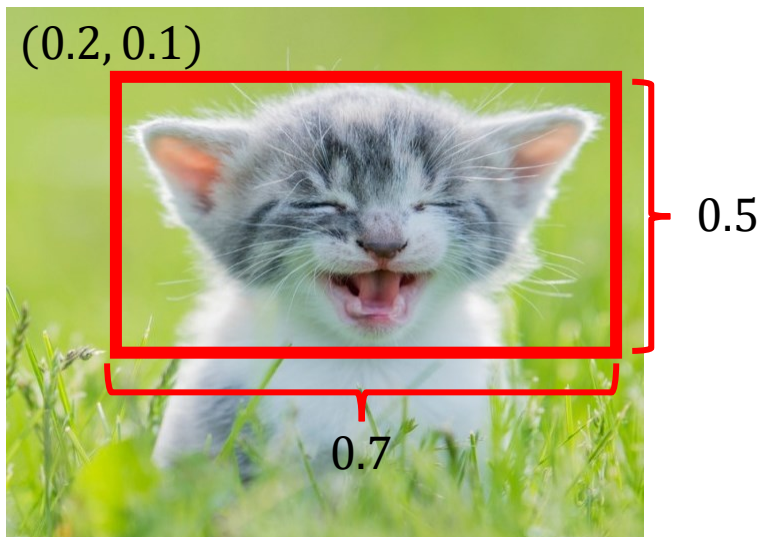
- Imagine que já treinamos a rede convolucional para classificar as fotos
 - Os parâmetros das convoluções já estão ajustados
- Vamos “congelar” os pesos da convolução e treinar só os pesos do **MLP**



Localização

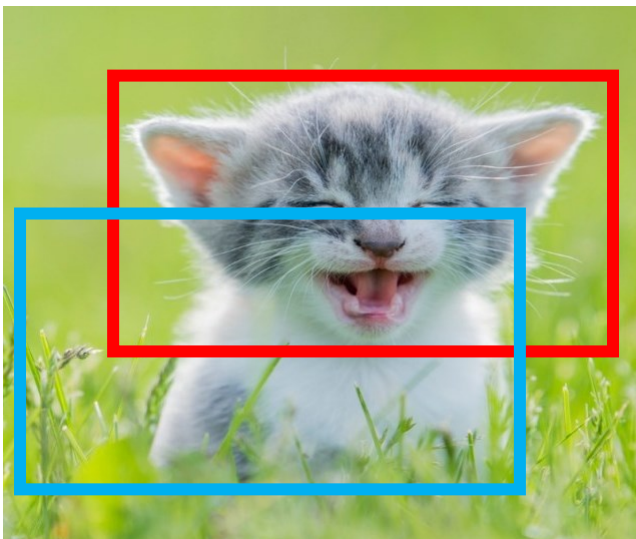


- Imagine que já treinamos a rede convolucional para classificar as fotos
 - Os parâmetros das convoluções já estão ajustados
- Vamos “congelar” os pesos da convolução e treinar só os pesos do **MLP**

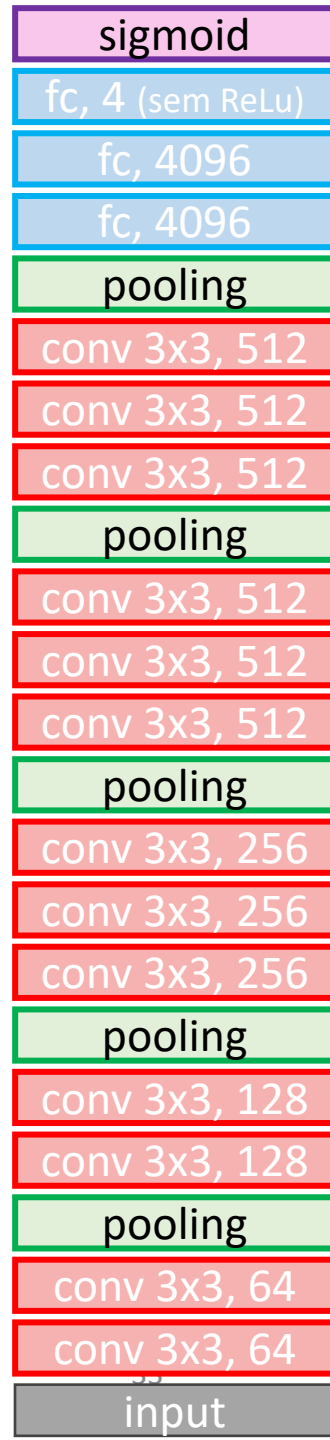


Localização

- Para avaliar a localização do objeto usamos métricas específicas
- *Intersection over Union*
- Computa a intersecção do *ground truth* com a *saída da rede* sobre a união dessas duas áreas
 - Valor ideal =1

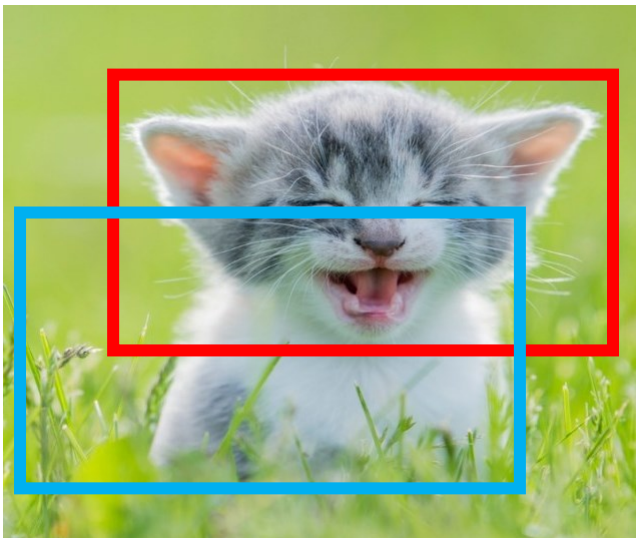


$$IoU = \frac{A \cap B}{A \cup B}$$

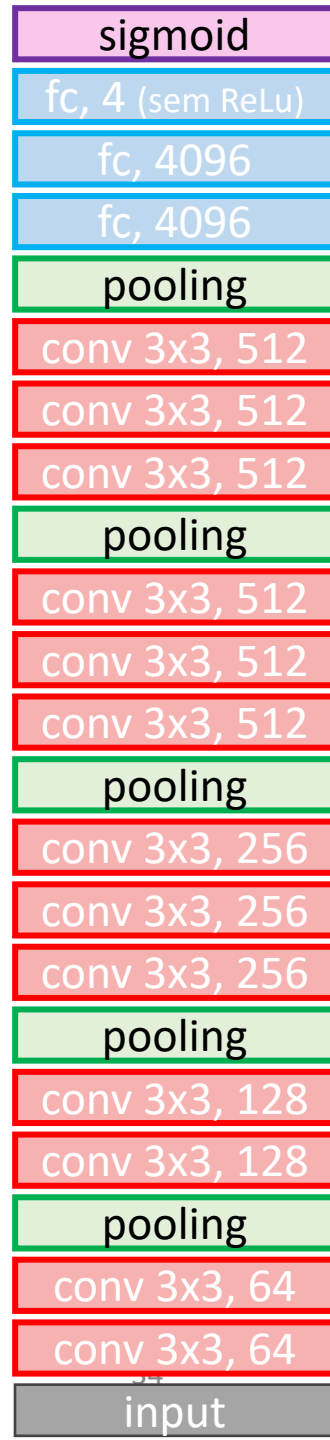


Localização

- Para avaliar a localização do objeto usamos métricas específicas
- *Dice Score*
- Computa o dobro da intersecção do *ground truth* com a *saída da rede* sobre a soma das duas áreas
 - Valor ideal =1



$$Dice = \frac{2 * (A \cap B)}{Area(A) + Area(B)}$$

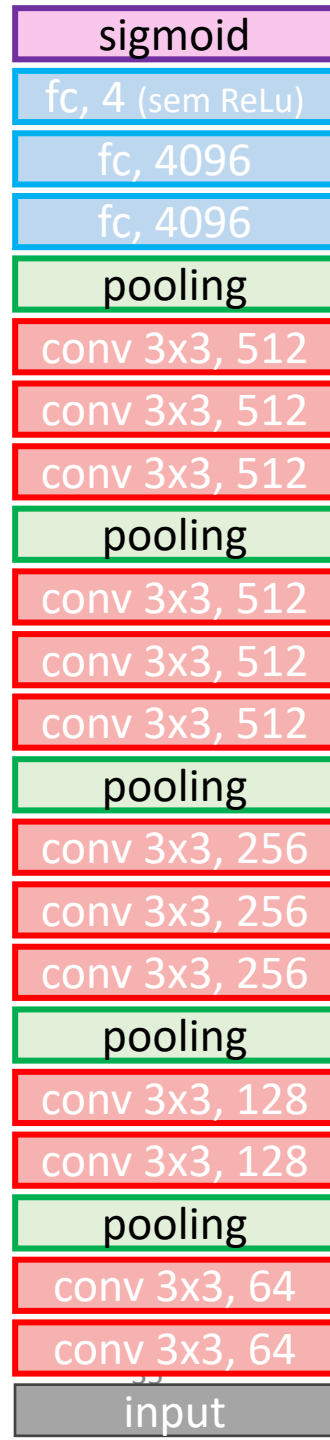
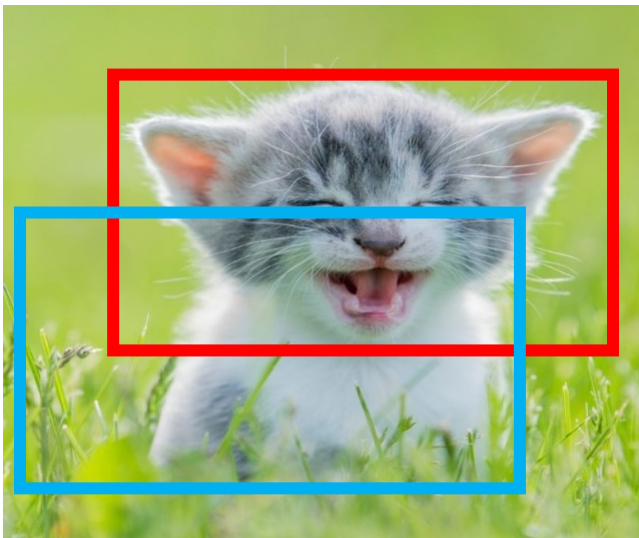


Localização

- *IoU vs Dice Score*
- Ambas métricas computam a qualidade da localização usando a sobreposição das *bounding boxes*

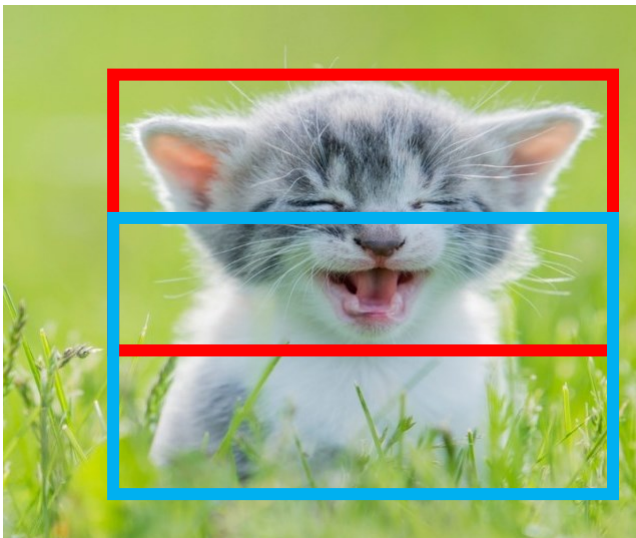
$$IoU = \frac{A \cap B}{A \cup B}$$

$$Dice = \frac{2 * (A \cap B)}{Area(A) + Area(B)}$$



Localização

- *IoU vs Dice Score*
- Ambas métricas computam a qualidade da localização usando a sobreposição das *bounding boxes*
- Suponha que:
 - $Area(A) = 10$
 - $Area(B) = 10$
 - $A \cap B = 5$



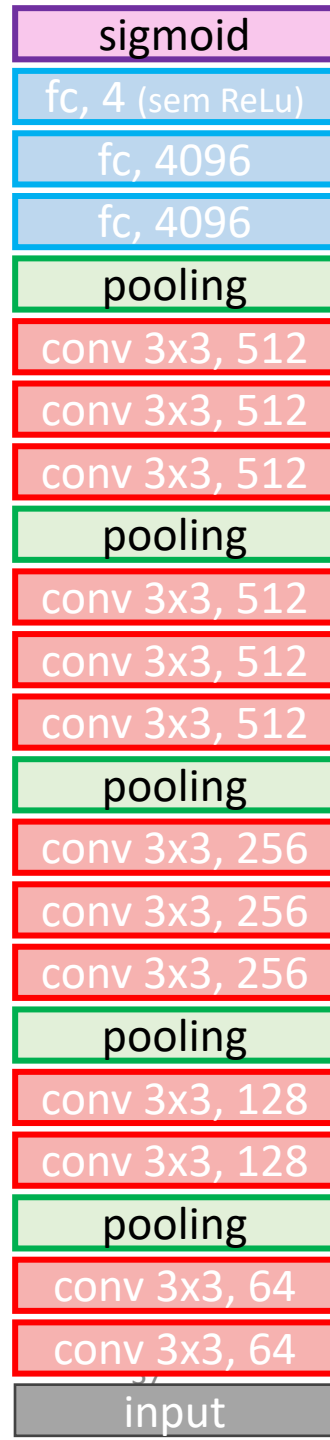
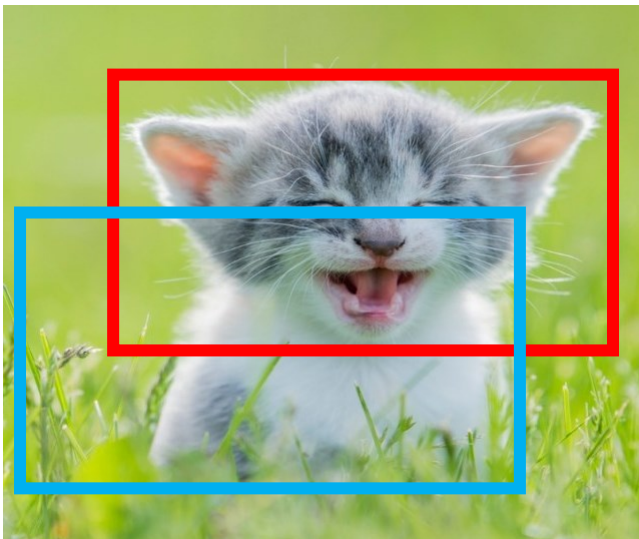
$$IoU = \frac{5}{15} = 0.333$$

Tem um efeito “quadrático” de penalização

$$Dice = \frac{2 * 5}{10 + 10} = 0.5$$

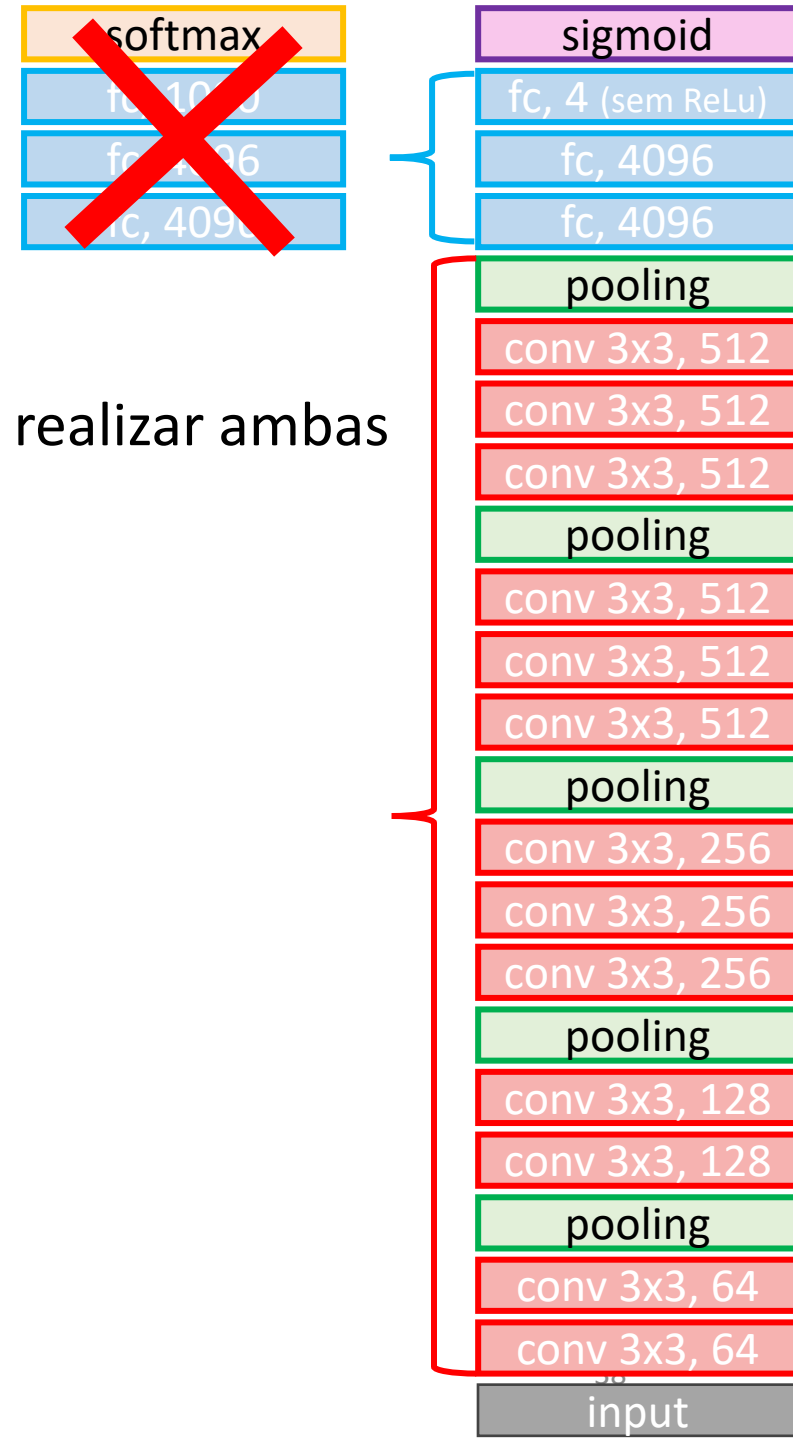
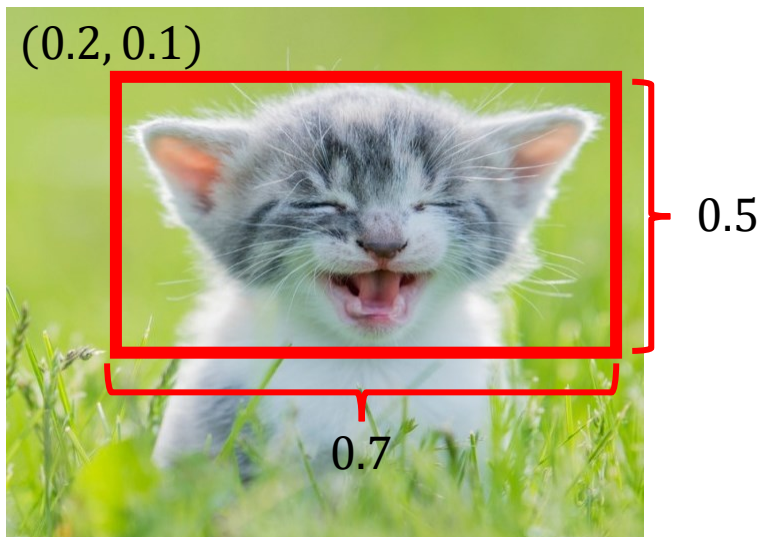
Localização - Resumindo

- *Bounding box* vira quatro saídas de regressão multivariada
- Loss MSE
- Ativação Sigmoid nas unidades de saída
- Podemos reaproveitar os pesos treinados anteriormente
 - Ou treinar toda a rede



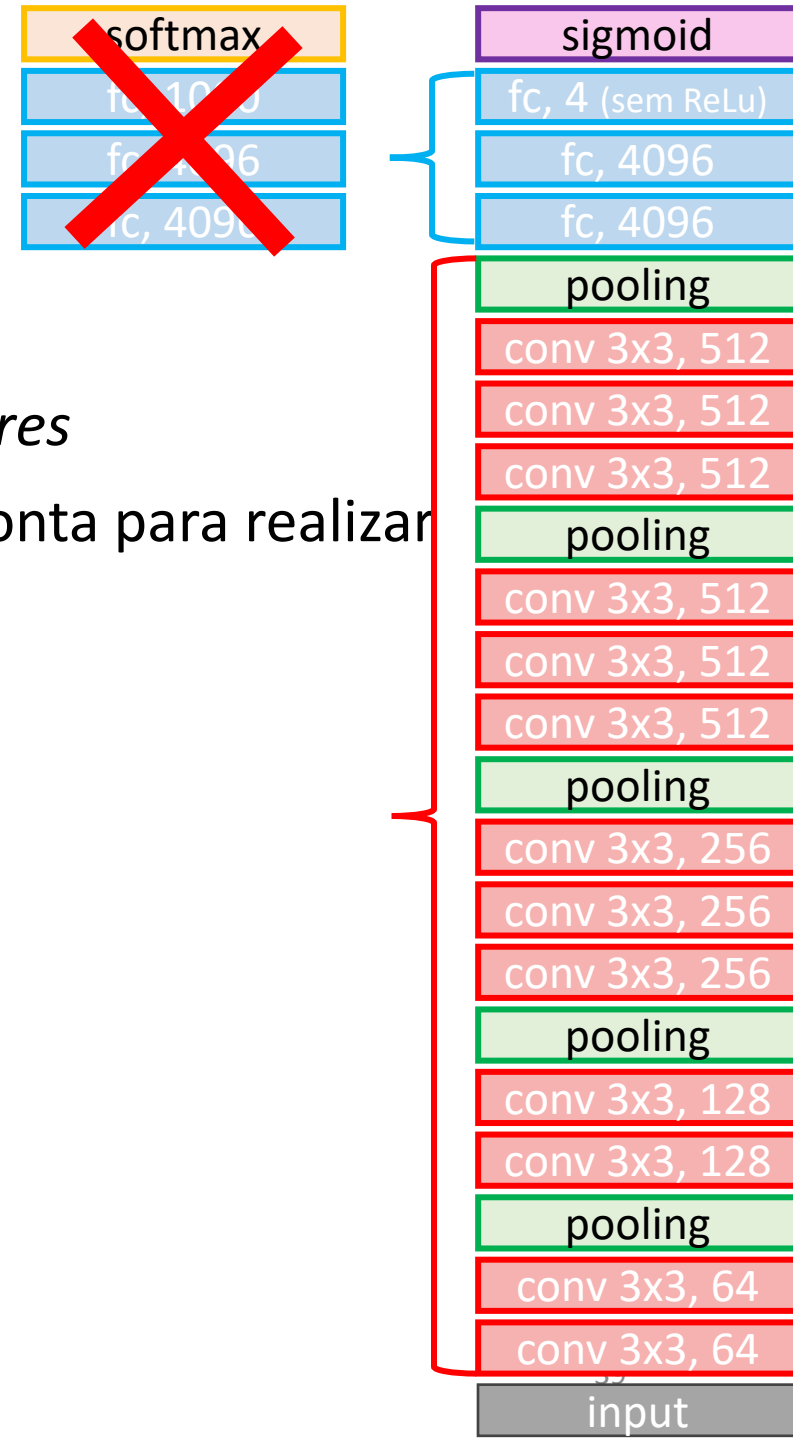
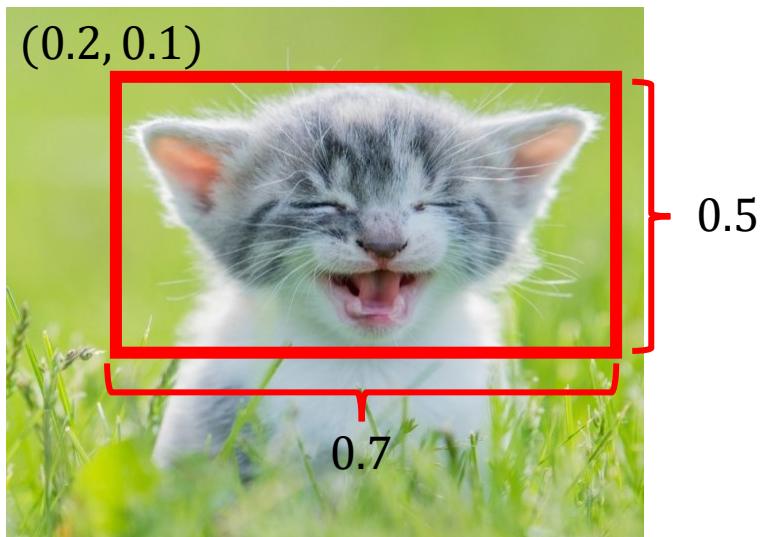
Classificação e Localização

- Vamos tentar agregar novamente nosso classificador para realizar ambas tarefas simultaneamente



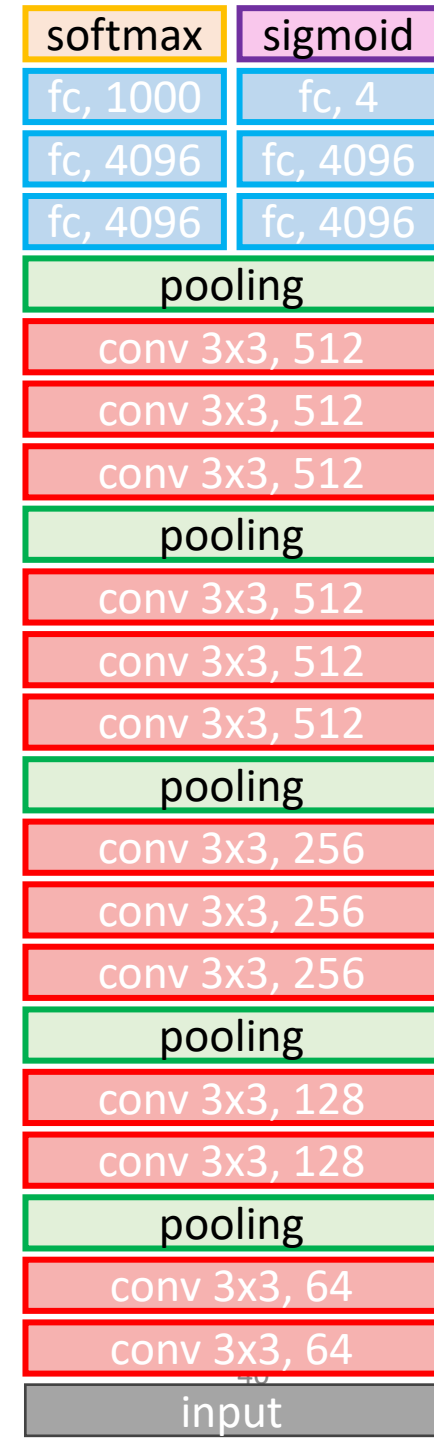
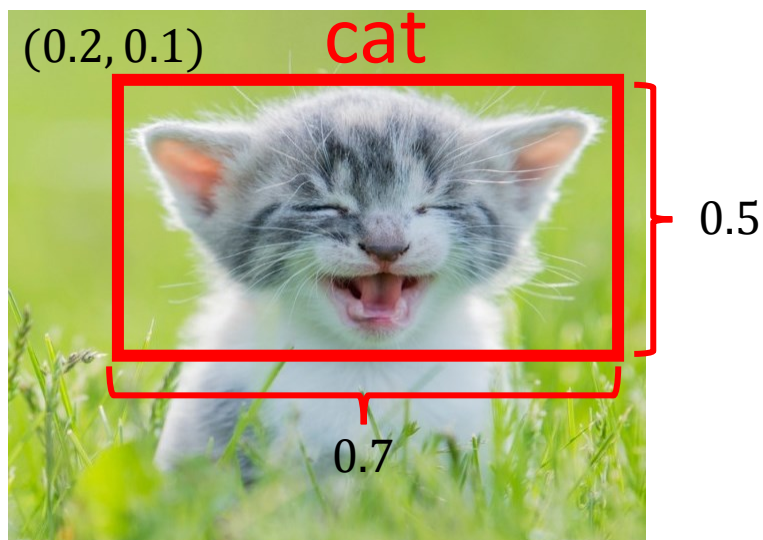
Classificação e Localização

- Toda **essa parte** da rede serve como um extrator de *features*
- Podemos adicionar **camadas totalmente conectadas** na ponta para realizar uma tarefa qualquer



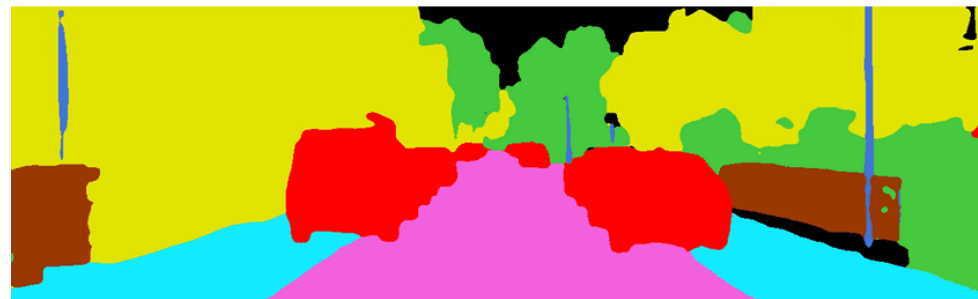
Classificação e Localização









- Repare que o **extrator de features** não mudou nesse novo treinamento
 - Portanto, nosso classificador deve seguir funcionando
- Assim podemos ter uma rede que resolve múltiplas tarefas
 - Classificação
 - Localização



Segmentação Semântica

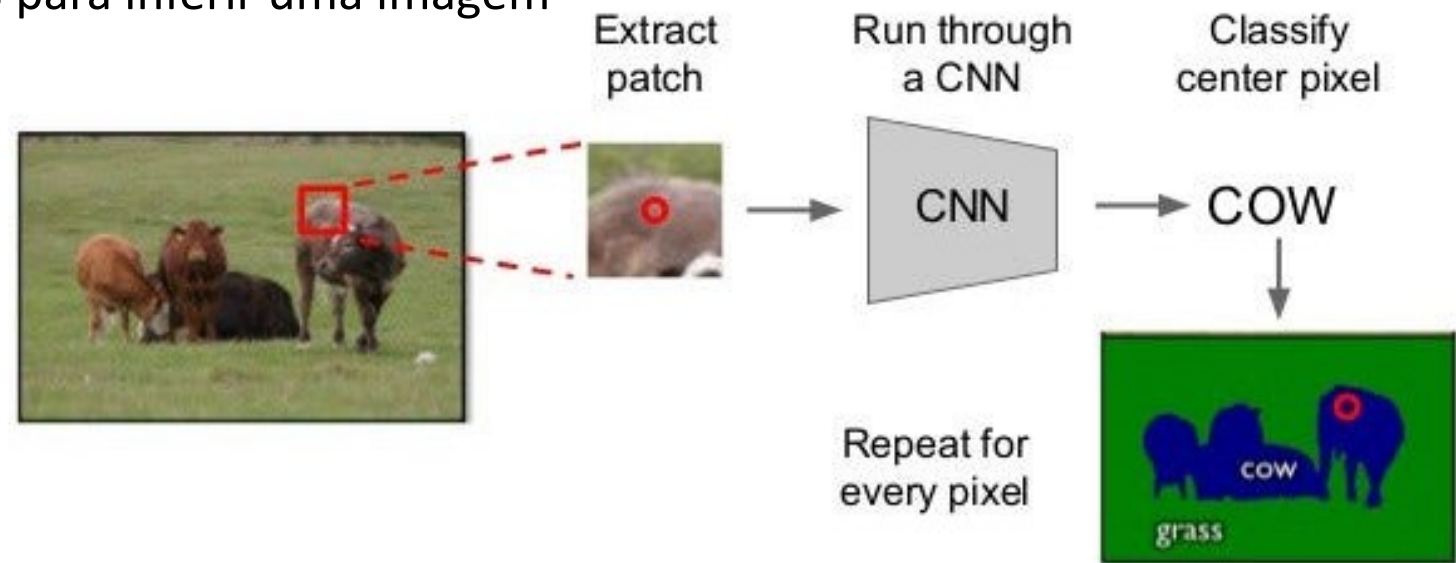
- Basicamente uma classificação em nível de pixel
 - Cada pixel da imagem recebe uma probabilidade de pertencer a uma classe



 Road	 Sidewalk	 Building	 Fence
 Pole	 Vegetation	 Vehicle	 Unlabel

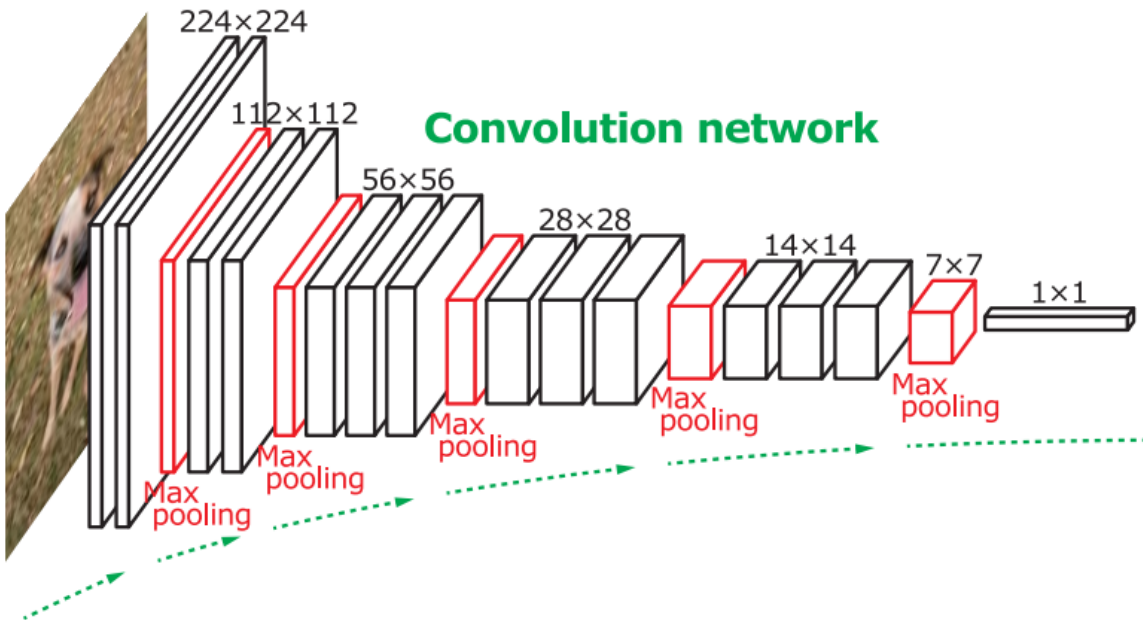
Segmentação Semântica

- Primeiras ideias
 - Separar a imagem em *patches*
 - CNN faz a classificação do pixel central do *patch*
 - Extremamente ineficiente ☹
 - Múltiplos *forwards* para inferir uma imagem



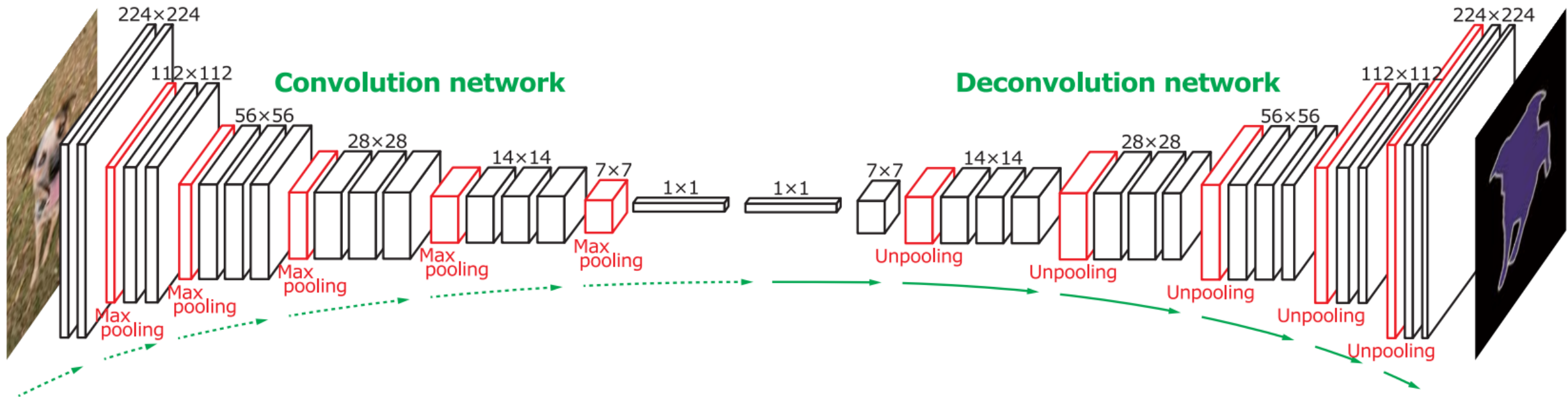
Segmentação Semântica

- Ideia: CNN deve classificar todos os pixels em um *forward*
- Diminuir a imagem com VGG-16 (*Encoder*)



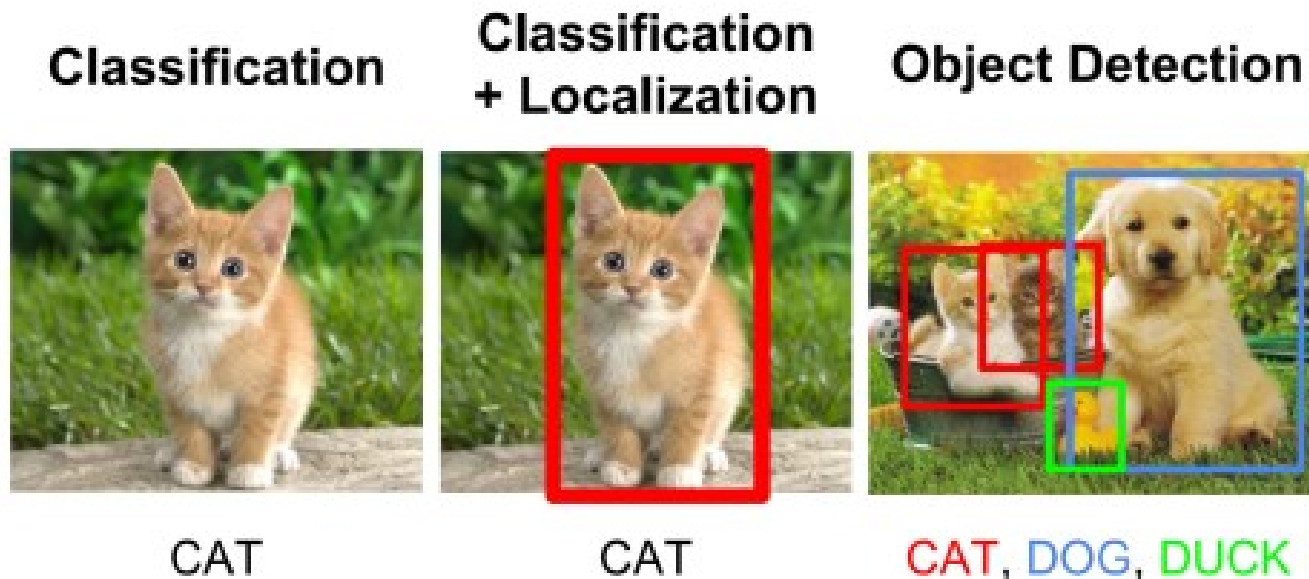
Segmentação Semântica

- Ideia: CNN deve classificar todos os pixels em um *forward*
- Diminuir a imagem com VGG-16 (*Encoder*)
- Aumentar a imagem com VGG-16 conv transposta (*Decoder*)



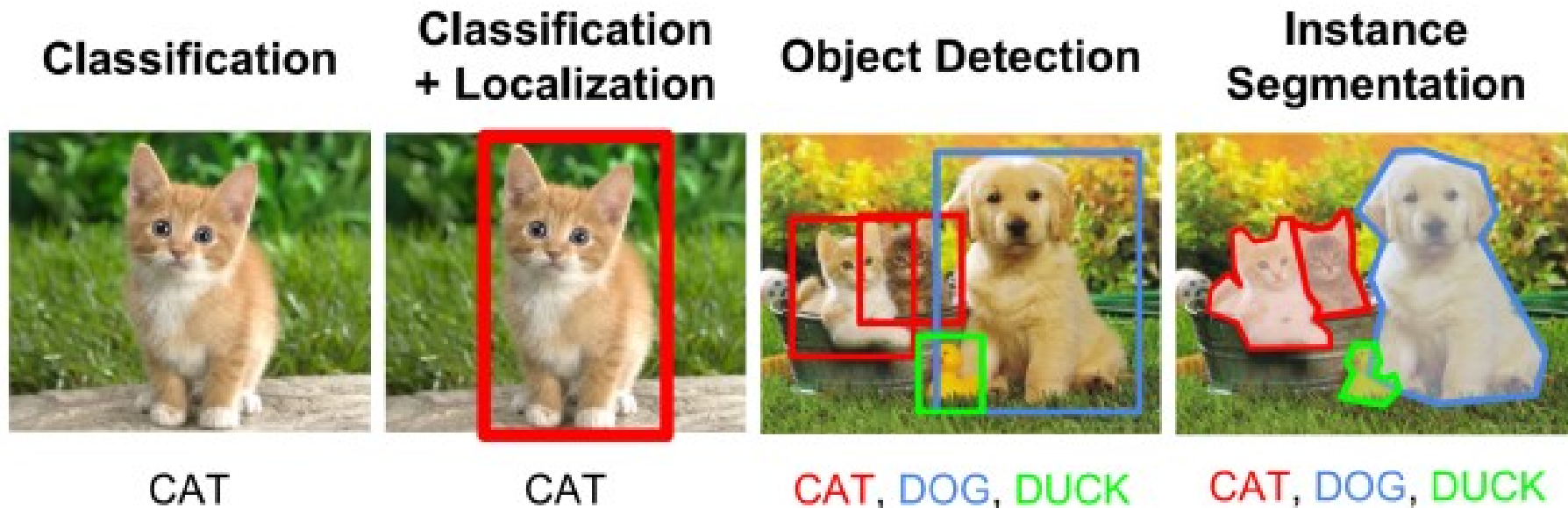
Detecção de Objetos

- Muito parecido com classificação + localização
 - Porém contempla mais de um objeto simultaneamente
 - Desafiador pois não sabemos de antemão quantos objetos podem aparecer na imagem



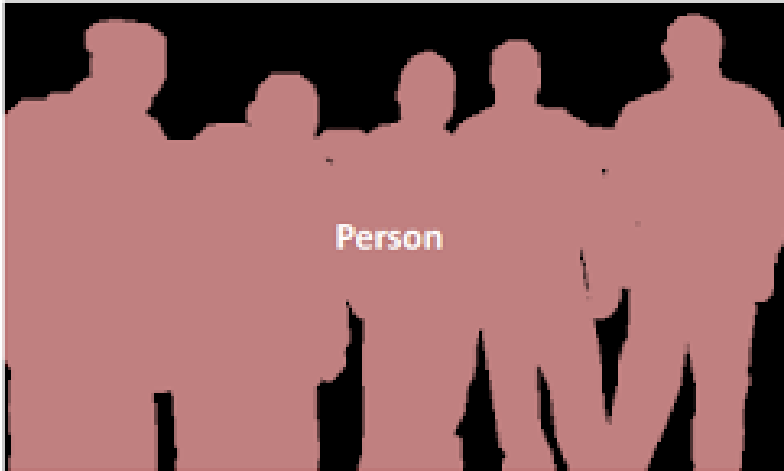
Segmentação de Instâncias

- Muito parecido com detecção de objetos
 - Porém temos um valor por pixel e não uma *bounding box*
 - Instâncias diferentes da mesma classe são diferenciadas

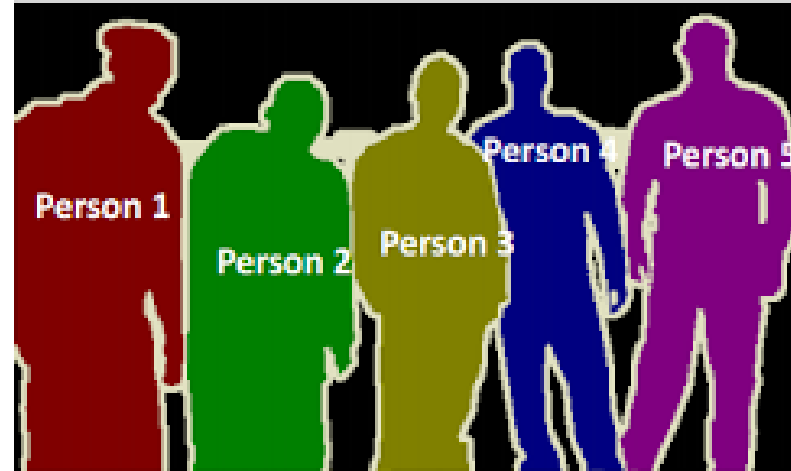


Segmentação Semântica

- Muito parecido com segmentação de instâncias, mas não diferenciamos dois objetos da mesma classe



Semantic Segmentation



Instance Segmentation

Sobre as tarefas

- Falaremos mais sobre detecção de objetos, segmentação semântica e segmentação de instâncias após falarmos de camadas residuais

Referências:

- Sugere-se ***fortemente*** a leitura de:
 - Capítulo 10 de Understanding Deep Learning
 - <https://udlbook.github.io/udlbook/>
- Os artigos abaixo são fáceis de ler e fornecem a visão original dos autores sobre as ideias apresentadas nessa aula
 - SIMONYAN, Karen; ZISSERMAN, Andrew. Very deep convolutional networks for large-scale image recognition. **arXiv preprint arXiv:1409.1556**, 2014.
 - SERMANET, Pierre et al. Overfeat: Integrated recognition, localization and detection using convolutional networks. **arXiv preprint arXiv:1312.6229**, 2013.
 - NOH, Hyeonwoo; HONG, Seunghoon; HAN, Bohyung. Learning deconvolution network for semantic segmentation. In: **Proceedings of the IEEE international conference on computer vision**. 2015. p. 1520-1528.