DEEP LEARNING BRASIL
SUMMER SCHOOL

Arquiteturas avançadas de Deep Learning para dispositivos embarcados.

Msc. Lucas Assis e Eng. Alexandre Cadore

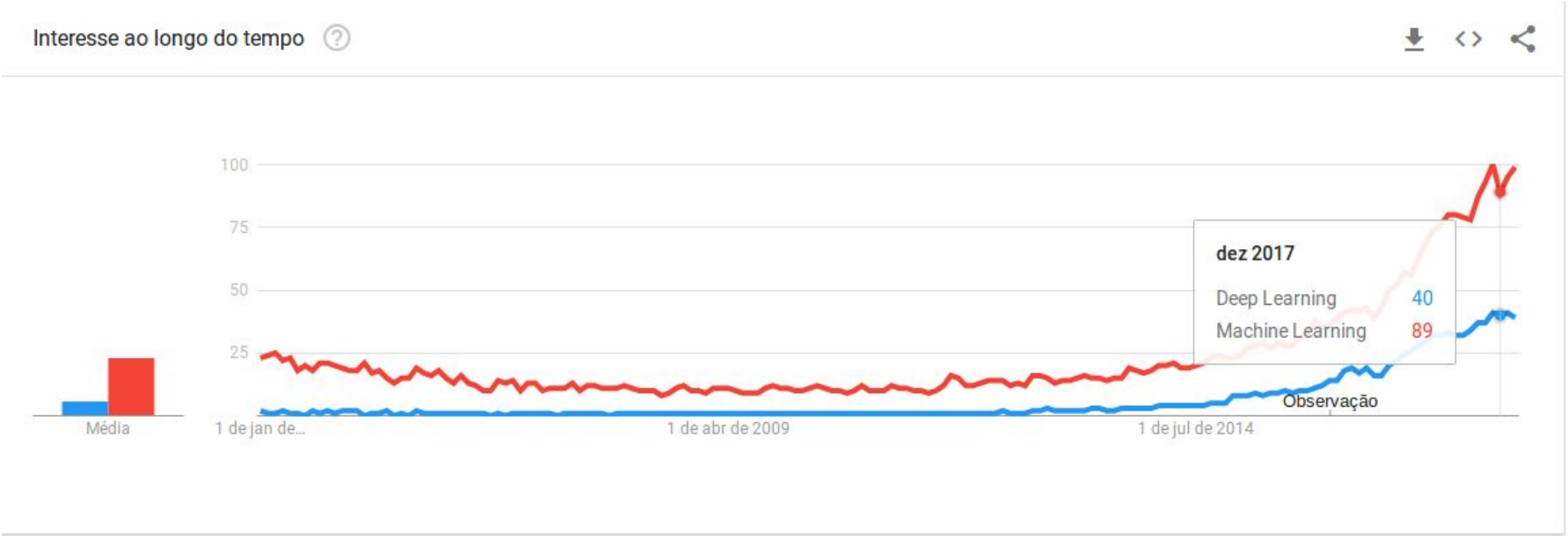
DATA H
Artificial Intelligence

Lucas da Silva Assis: Engenheiro Eletricista pela EMC - UFG. Durante a graduação co-fundou o Núcleo de Robótica Pequi Mecânico - UFG e tornou-se coordenador da equipe de futebol de robôs (categoria IEEE Very Small Size Soccer). Mestre em Ciência da Computação pelo INF - UFG e Doutorando em Ciência da Computação pela Universidade Federal de Goiás. Possui experiência nas áreas de Robótica, Eletrônica Embarcada, Internet das Coisas, Inteligência Computacional, Computação Evolutiva e Aprendizado de Máquina. Atualmente está no mercado de soluções baseadas em Inteligência Artificial através da Data H e parcerias com a RYD Engenharia e Yalla Soluções.

Alexandre Cadore Tondolo: Bacharel em Engenharia de Computação - UFG, com graduação sanduíche na Universidade do Porto (Portugal). Foi coordenador da equipe IEEE - Open do Núcleo de Robótica Pequi Mecânico - UFG. Possui experiência nas áreas de Robótica, Eletrônica Embarcada, Internet das Coisas, Análise de dados, Inteligência Computacional, Computação Evolutiva e Aprendizado de Máquina.
Atualmente está no mercado de Análise de Dados pela Webradar e no de soluções baseadas em Inteligência Artificial através da Data H e parcerias com a RYD Engenharia e Yalla Soluções.

Motivação

- A Explosão do “Deep Learning” - Imagem, Voz e Texto.



Motivação

- Um problema no sistema de produção, no entanto, é o tamanho dos modelos:

  A maioria das ConvNets populares (VGG, ResNet, Inception, etc.) ocupam facilmente várias centenas de MB com seus parâmetros !

Motivação

Motivação

- Embora esses modelos possam ser hospedados para consulta, é mais atrativo comprimir um modelo para um tamanho pequeno o suficiente, de modo que os usuários possam baixá-lo em seus dispositivos para uso off-line.

Bases de Comparação

- VGG16
  - 138M Parâmetros ~ 528MB
  - 16 Camadas
  - Filtros 3x3
  - top-5 accuracy de 92.3 % na ImageNet.

- AlexNet
  - 60M Parâmetros ~ 285MB
  - 8 Camadas
  - Filtros 5x5 e 3x3
  - top-5 accuracy de 80.6% na ImageNet.

Algumas Arquiteturas Compactas :

- SqueezeNet (Nov/2016), cujos autores afirmam ter uma precisão similar à AlexNet com 50x menos parâmetros, resultando em modelos com tamanhos <0.5MB.

- MobileNet(Abril/2017) , precisão similar à VGG-16 com 32x menos parâmetros. Resultando em modelos <20MB

Vantagens de uma arquitetura com poucos parâmetros:

- Treinamento distribuído mais eficiente
- Menos sobrecarga ao exportar novos modelos para clientes
- Deploy em placas FPGA
- Economia de recursos como tempo, energia e dinheiro

Qual a mágica ?

- Na verdade, é um fato conhecido que as ConvNets tem muito mais conexões do que precisam para fazer seu trabalho.

- No paper “Deep Compression” de Han et al. o autor prova que o tamanho da VGG - 16 pode ser reduzido por um fator 49 somente pela poda de conexões sem importância, mantendo a acurácia da rede.

Squeezenet - Novembro 2016

- Principais pontos de diferença com uma ConvNet:
  - Maioria dos kernels reduzidos de 3x3 para 1x1 (Redução de 9x nos parâmetros)
  - Canais de entrada nos filtros 3x3 restantes são alterados para diminuir parâmetros
    i.e. NParam = (**input channels**) * (number of filters) * (3*3).
  - Ausência de camadas FC
  - Output gerada com o uso de Avg. Pooling

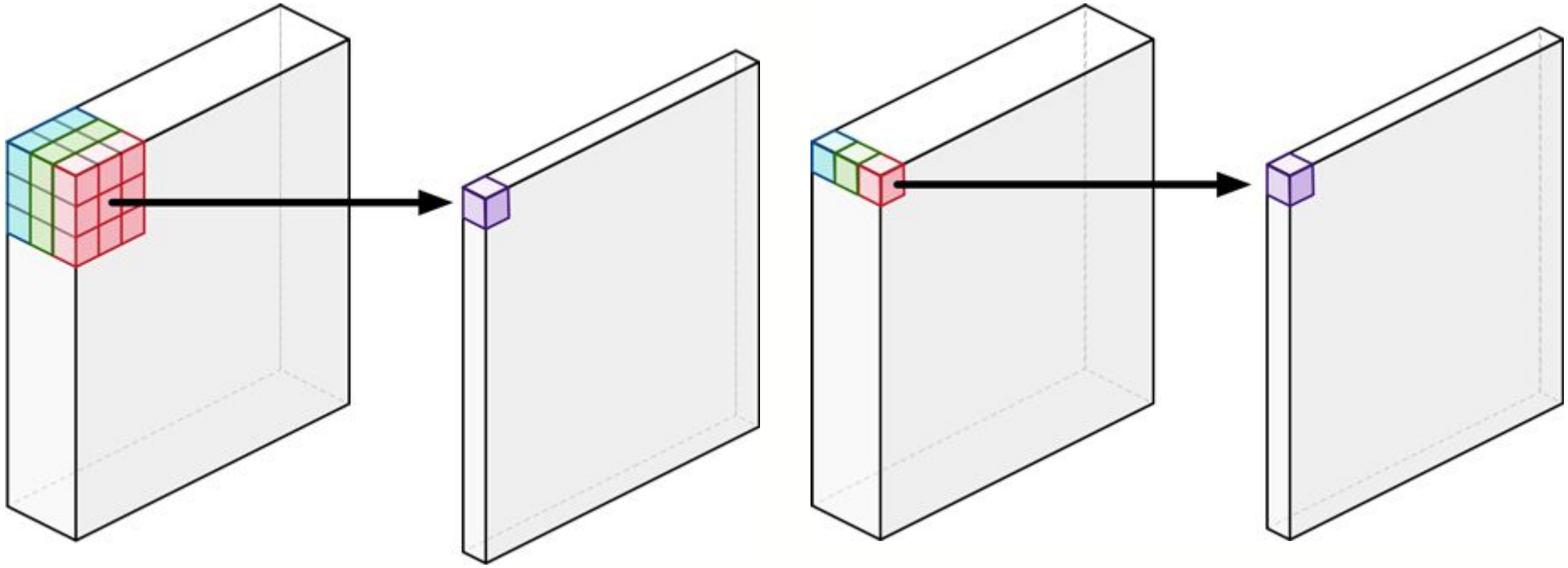
- Introdução do módulo FIRE (Squeeze + Expand)

Squeezenet

- Quê ?? Filtro de convolução 1x1 ?

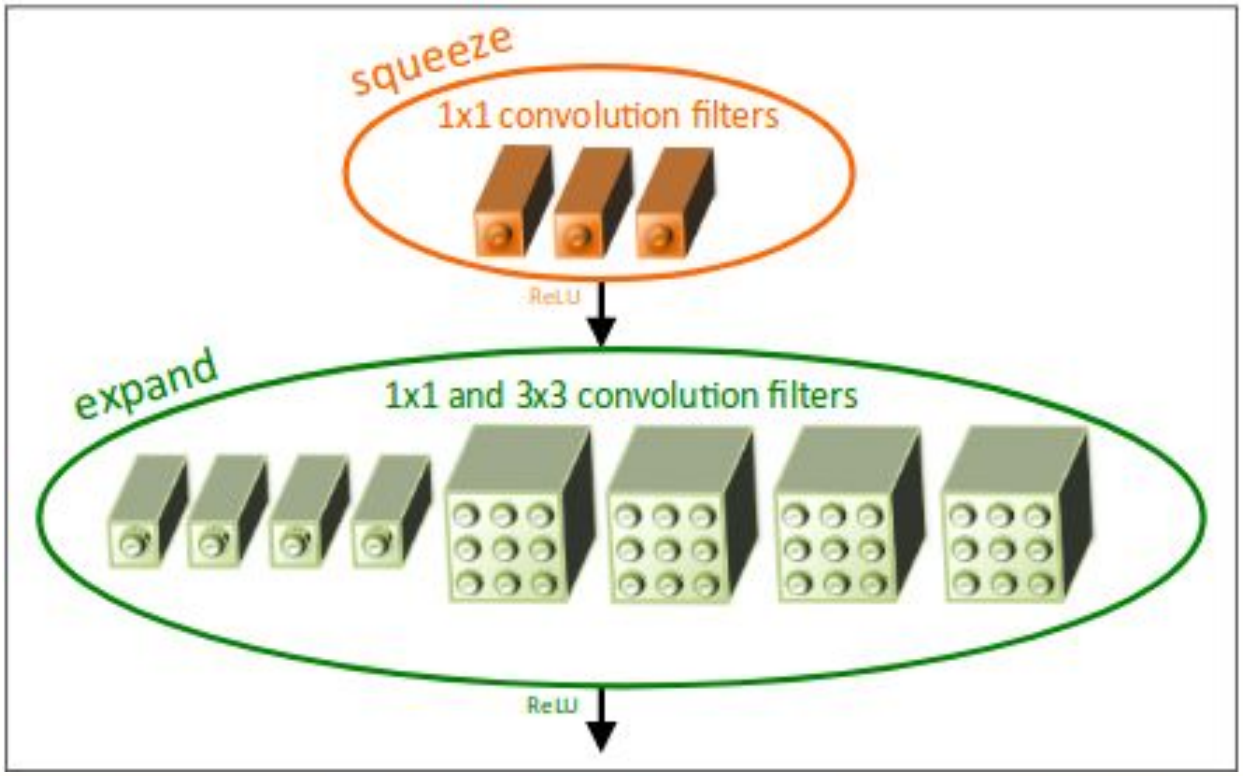
Squeezenet

- Quê ?? Filtro de convolução 1x1 ?



Squeezenet - Novembro 2016

- Módulo **Fire**



Squeezenet

Módulo Fire

```
activations = []
...
def fire(inputs, squeezeTo, expandTo):
    h = squeeze(inputs, squeezeTo)
    h = expand(h, expandTo)
    activations.append(h)
```

Squeezenet

Módulo Fire

```
def squeeze(inputs,squeezeTo):
    with tf.name_scope('squeeze'):
    inputSize = inputs.get_shape().as_list()[3]
    w =
tf.Variable(tf.truncated_normal([1,1,inputSize,squeezeTo]))
    h = tf.nn.relu(tf.nn.conv2d(inputs,w,[1,1,1,1],'SAME'))
    return h
```

Squeezenet

Módulo Fire

```
def expand(inputs,expandTo):
    with tf.name_scope('expand'):
    squeezeTo = inputs.get_shape().as_list()[3]
    w =
tf.Variable(tf.truncated_normal([1,1,squeezeTo,expandTo]))
    h1x1 = tf.nn.relu(tf.nn.conv2d(inputs,w,[1,1,1,1],'SAME'))
    w =
tf.Variable(tf.truncated_normal([3,3,squeezeTo,expandTo]))
    h3x3 = tf.nn.relu(tf.nn.conv2d(inputs,w,[1,1,1,1],'SAME'))
    h = tf.concat([h1x1,h3x3],3)
    return h
```

Squeezenet - Arquitetura Completa

3X3 CONV

MAXPOOL

FIRE X 3

MAXPOOL

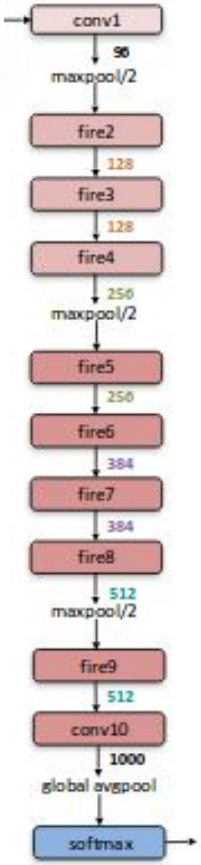
FIRE X 4

MAXPOOL

FIRE

3X3 CONV

SOFTMAX



90% DOG
9% T-REX
1% VAGABUNDO

Squeezenet - Resultados

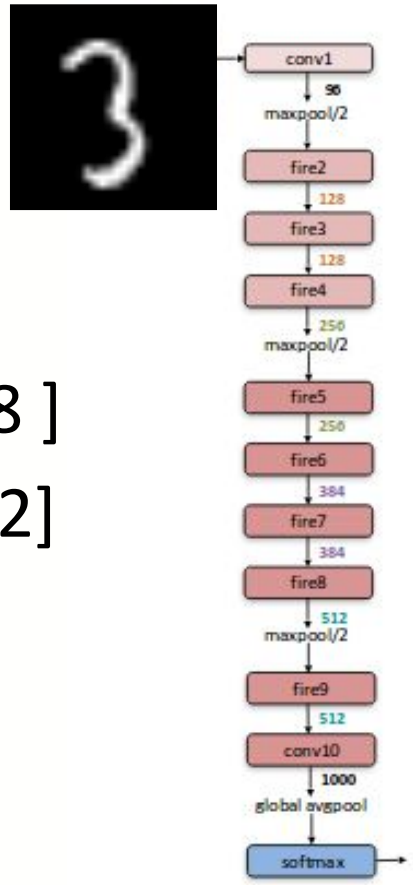
- Modelo customizado - MNIST:

  3X3 CONV - 64 Filtros

  FIRE :

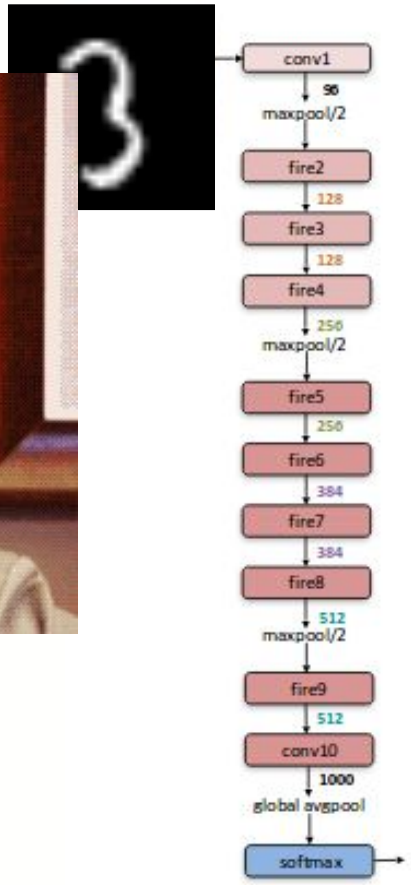
  Squeezes [2, 2, 4 , 4 , 6 , 6 , 8, 8]

  Expansions [8, 8, 16, 16,24,24,32,32]



Squeezenet - Resultados

- Mo
  3X3
  FIRE
    S
    E



94.22% Acc

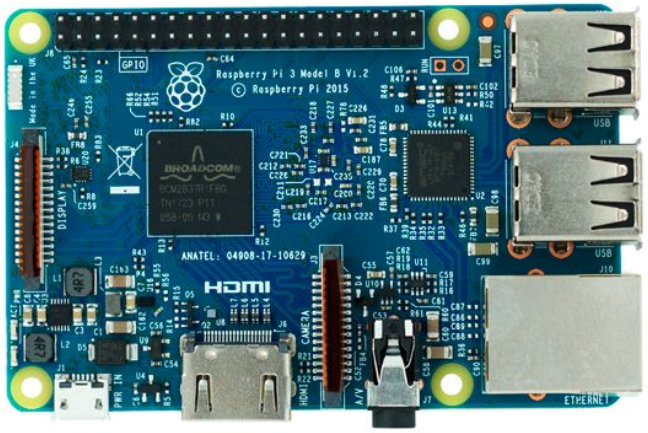
~ 57.6KB = "SÓ" 9472 PARÂMETROS

Raspberry Pi

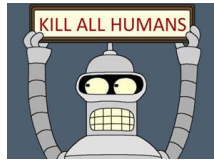
Especificações:

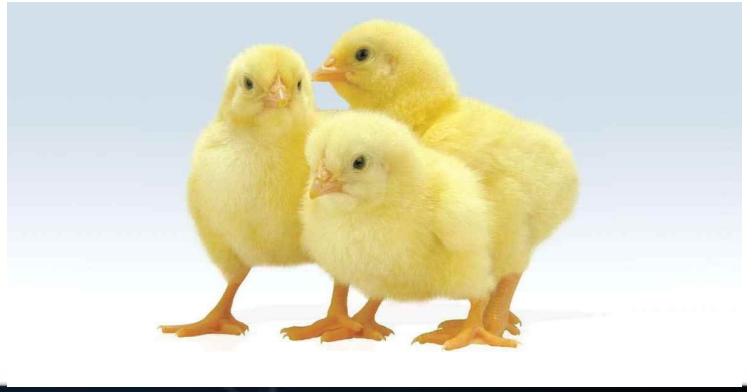
– Raspberry Pi 3 Model B Anatel
– Processador Broadcom BCM2837 64bit ARMv8 Cortex-A53 Quad-Core
– Clock 1.2 GHz
– Memória RAM: 1GB

```
pi@raspberrypi:~/Squeeze $ python3 Test_MNIST.py
Extracting MNIST\data/train-images-idx3-ubyte.gz
Extracting MNIST\data/train-labels-idx1-ubyte.gz
Extracting MNIST\data/t10k-images-idx3-ubyte.gz
Extracting MNIST\data/t10k-labels-idx1-ubyte.gz
Graph Loaded
Time elapsed : 0.004124
Network Output :
[[  0.00000000e+00   1.00000000e+00   0.00000000e+00   0.00000000e+00
    0.00000000e+00   0.00000000e+00   0.00000000e+00   0.00000000e+00
    2.80259693e-44   0.00000000e+00]]
Label from Dataset :
[ 0.  1.  0.  0.  0.  0.  0.  0.  0.  0.]
```



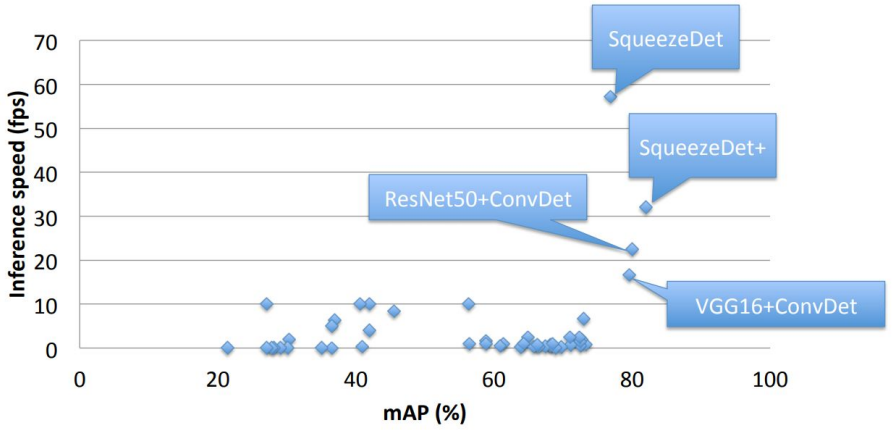
Exemplos de aplicações





Bônus : SqueezeDet - 29/11/2017

method	car			cyclist			pedestrian			mAP	model size (MB)	speed (FPS)
	E	M	H	E	M	H	E	M	H			
SubCNN [26]	90.8	**89.0**	79.3	79.5	71.1	62.7	83.3	71.3	66.4	77.0	-	0.2
MS-CNN [3]	90.0	89.0	76.1	84.1	75.5	66.1	83.9	73.7	68.3	78.5	-	2.5
PNET*	81.8	83.6	74.2	74.3	58.6	51.7	77.2	64.7	60.4	69.6	-	10
Pie*	89.4	89.2	74.2	84.6	76.3	67.6	**84.9**	**73.2**	67.6	78.6	-	0.83
FRCN+VGG16 [2]	92.9	87.9	77.3	-	-	-	-	-	-	-	485	1.7
FRCN+Alex [2]	**94.7**	84.8	68.3	-	-	-	-	-	-	-	240	2.9
SqueezeDet (ours)	90.2	84.7	73.9	82.9	75.4	72.1	77.1	68.3	65.8	76.7	**7.9**	**57.2**
SqueezeDet+ (ours)	90.4	87.1	78.9	**87.6**	**80.3**	**78.1**	81.4	71.3	**68.5**	**80.4**	26.8	32.1
VGG16 + *ConvDet* (ours)	93.5	88.1	79.2	85.2	78.4	75.2	77.9	69.1	65.1	79.1	57.4	16.6
ResNet50 + *ConvDet* (ours)	92.9	87.9	**79.4**	85.0	78.5	76.6	67.3	61.6	55.6	76.1	35.1	22.5



Obrigado