Redes Residuais

Vamos dar continuidade ao aprendizado com Keras na segunda tarefa desta semana! Voce irá aprender como construir uma rede convolucional profunda utilizando redes residuais (ResNets). Na teoria, redes muito profundas podem representar funções bastante complexas; porém, na prática, elas são dificeis de serem treinadas. Redes Residuais, introduzidas por He et al. (https://arxiv.org/pdf/1512.03385.pdf), permitem treinar redes muito profundas que anteriormente não era possível de ser realizado.

Nesta tarefa você irá:

- Implementar os blocos básicos de uma ResNet.
- Ligar estes blocos para implementar e treinar uma rede neural de última geração para classificar imagens.

Esta tarefa será feita utilizando Keras.

Vamos exeutar a célula abaixo para carregar todos os pacotes necessários.

In [2]:
```
import numpy as np
from tensorflow.keras import layers
from tensorflow.keras.layers import Input, Add, Dense, Activation,
ZeroPadding2D, BatchNormalization, Flatten, Conv2D, AveragePooling2
D, MaxPooling2D, GlobalMaxPooling2D
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.preprocessing import image
import pydot
from IPython.display import SVG
from tensorflow.keras.utils import plot_model
from resnets_utils import *
from tensorflow.keras.initializers import glorot_uniform
import scipy.misc
from matplotlib.pyplot import imshow
%matplotlib inline

import tensorflow.keras.backend as K
K.set_image_data_format('channels_last')
K.set_learning_phase(1)
```

1 - O problema de redes neurais muito profundas

Você já construiu a sua primeira rede neural convolucional. Recentemente as redes neurais ficaram mais profundas, onde o estado da arte partiu de apenas algumas camadas (como na AlexNet) para mais de 100 camadas.

O principal beneficio de uma rede neural muito profunda é que ela consegue representar funções muito complexas. Ela pode ainda aprender características em níveis diferentes de abstração, de arestas (nos primeiros níveis) até características complexas (nos níveis mais profundos). Porém, utilizar uma rede neural muito profunda nem sempre ajuda. Uma grande barreira para treiná-las são os gradientes que podem ir para zero muito rapidamente conforme nos aprofundamos na rede, fazendo com que o gradiente descendente seja muito lento. Mais especificamente, durante o gradiente descendente, conforme é feita a propagação para trás da camada de saída para as camadas anteriores as matrizes de peso são multiplicadas em cada etapa e o gradiente pode ir para zero rapidamente (ou, em casos raros, crecerem exponencialmente e explodir com valores muito altos).

Durante o treinamento você deve verificar se a magnitude (ou a norm) do gradiente para as camadas anteriores estão decrescendo para zero conforme o processo de treinamento caminha:

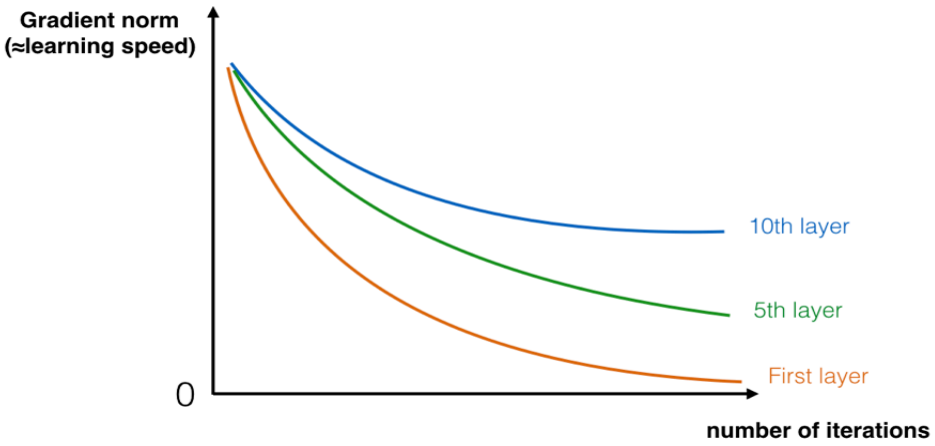


Figure 1 : **Gradiente indo a zero**
A velocidade do treinamento decresce rapidamente nas camadas iniciais durante o treinamento da rede

Uma rede residual irá nos auxiliar a resolver este problema.

2 - Construindo uma rede residual

Em ResNets, um "atalho" ou uma "conexão escapada" permite que o gradiente seja diretamente propagado para camadas menos profundas:

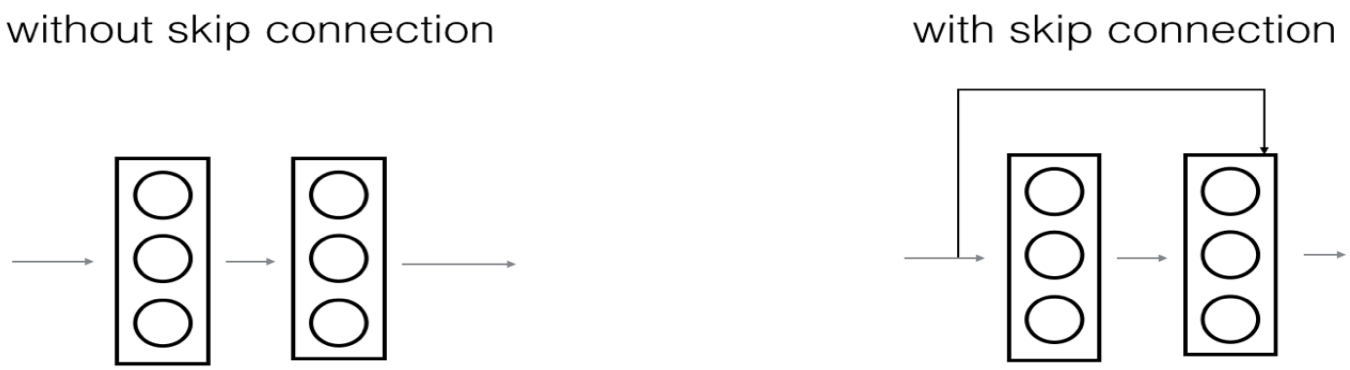


Figura 2 : Um bloco de ResNet mostrando uma **conexão escapada**

A imagem à esquerda mostra o caminho principal através da rede. A imagem à direita mostra umbloco ResNet que adiciona um atalho ao caminho principal. Ligando blocos ResNet é possível construir uma rede muito profunda.
Foi visto também em aula que os blocos ResNet tornam o aprendizado da função identidade algo bem simples. Isto quer dizer que você pode adicionar blocos ResNet sem o risco de prejudicar o desempenho da rede durante o treinamento. (Existem evidências de que a facilidade de aprender a função identidade, mais do que auxiliar nos gradientes indo para zero, é um dos fatores do ótimo desempenho das ResNets.)

Uma ResNet utiliza dois tipos de blocos, dependendo principalmente no fato das dimensões de entrada e saída serem ou não as mesmas. Você irá implementar os dois casos.

2.1 - O bloco identidade

O bloco identidade é um blocopadrão utilizado em ResNets e corresponde ao caso onde a ativação de entrada ($a^{[l]}$) possui a mesma dimensão da saída da ativação ($a^{[l+2]}$). Para ilustrar os passos do que acontece em um bloco de identidade de uma ResNet a figura abaixo mostra um diagrama alternativo:

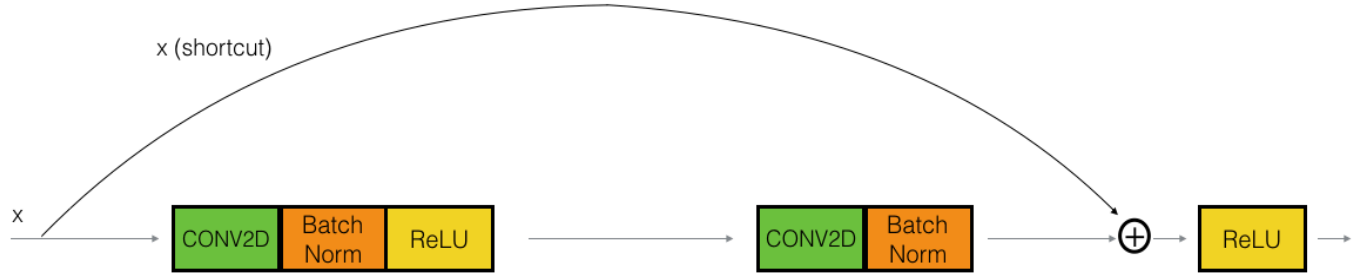


Figura 3 : **Bloco Identidade.** Conexão escapada "escapa sobre" 2 camadas.

O caminho de cima é o "atalho". O caminho de baixo éo caminho principal. Neste diagrama está sendo apresentado explicitamente os passos da CONV2D e ReLU em cada camada. Para acelerar o treinamento foi adicionado um passo de BatchNorm. Não se preocupe se você achar isto tudo muito complicado, você verá que, por exemplo, BatchNorm é apenas uma linha de código em Keras!
Neste exercício você irá implementar uma versão ainda mais poderosa do bloco de identidade, onde a conexão escapada pula 3 camadas ao invés de 2. Ele se parece com isto:

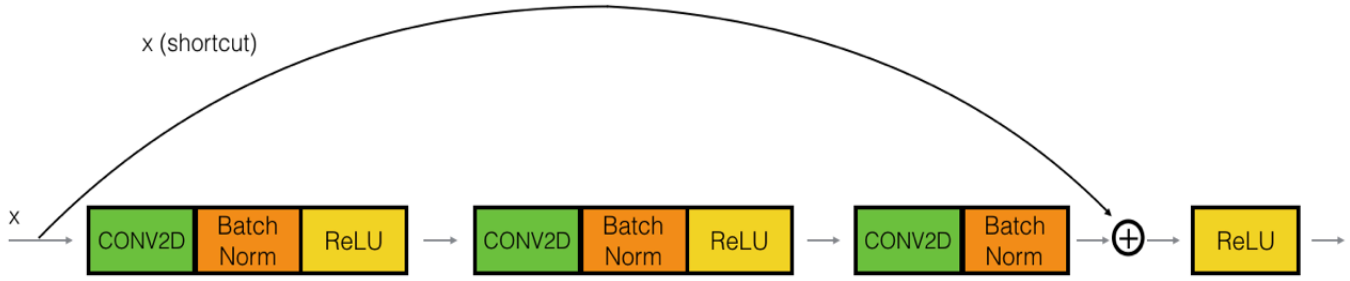


Figura 4 : **Bloco Identidade.** Conexão escapada sobre 3 camadas.

Estes são os passos individuais.
Primeira componente do caminho principal:

- O primeiro CONV2D possui F_1 filtros no formato (1,1) e com stride de (1,1). O padding utilizado é "valid" e seu nome deve ser `conv_name_base + '2a'`. Use 0 como semente para as inicializações aleatórias.
- O primeiro BatchNorm estará normalizando o eixo dos canais. Seu nome deve ser `bn_name_base + '2a'`.
- Aplique então a função de ativação ReLU. Ela não possui nome e nem hiper parâmetros.

Segunda componente do caminho principal:

- O segundo CONV2D possui F_2 filtros no formato (f,f) e com stride de (1,1). O padding utilizado é "same" e seu nome deve ser `conv_name_base + '2b'`. Use 0 como semente para as inicializações aleatórias.
- O segundo BatchNorm estará normalizando o eixo dos canais. Seu nome deve ser `bn_name_base + '2b'`.
- Aplique então a função de ativação ReLU. Ela não possui nome e nem hiper parâmetros.

Terceira componente do caminho principal:

- O terceiro CONV2D possui F_3 filtros no formato (1,1) e com stride de (1,1). O padding utilizado

é "valid" e seu nome deve ser `conv_name_base + '2c'`. Use 0 como semente para as inicializações aleatórias.
- O terceiro BatchNorm estará normalizando o eixo dos canais. Seu nome deve ser `bn_name_base + '2c'`. Note que não existe função de ativação neste componente.

Componente Final:

- O atalho e a entrada são adicionados.
- A função de ativação ReLU é aplicada a esta soma. Ela não possui nome e nem hiper parâmetros.

Exercício: Implemente o bloco de identidade da ResNet. A primeira componente do caminho principal foi implementada. Por favor leia atentamente estas instruções para entender o que você está fazendo. Você deve implementar as demais componentes.

- Para implementar a Conv2D: Veja a referência (https://keras.io/layers/convolutional/#conv2d)
- Para implementar a BatchNorm: Veja a referência (https://faroit.github.io/keras-docs/1.2.2/layers/normalization/) (axis: Integer, o eixo que deve ser normalizado (typicamente o eixo dos canais))
- Para a ativação utilize: `Activation('relu')(X)`
- Para adicionar os valores passados para frente pelo atalho: Veja referência (https://keras.io/layers/merge/#add)

In [20]:
```
# FUNÇÃO DE ATIVAÇÃO: identity_block

def identity_block(X, f, filters, stage, block):
    """
    Implementa o bloco de identidade definido na figura 3

    Argumentos:
    X -- tensor de entrada no formato (m, n_H_prev, n_W_prev, n_C_p
rev)
    f -- inteiro, especifica o formato do meio da janela do CONV pa
ra o caminho principal.
    filters -- lista python de inteiros, definindo o número de filt
ros nas camadas CONV do caminho principal
    stage -- inteiro, utilizado para dar nome as camadas, dependend
o de sua posição na rede.
    block -- string/caracter, usado para dar nome as camadas, depen
dendo de sua posição na rede.

    Retorna:
    X -- saída do bloco identidade, um tensor no formato (n_H, n_W,
n_C)
    """

    # definindo os nomes base
    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'

    # Recupera os Filtros
```

```
    F1, F2, F3 = filters

    # Salva o valor da entrada. Ela será necessária mais tarde para
adição no caminho principal.
    X_shortcut = X

    # Primeiro componente do caminho principal
    X = Conv2D(filters = F1, kernel_size = (1, 1), strides = (1,1),
padding = 'valid', name = conv_name_base + '2a', kernel_initializer
= glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2a')(X)
    X = Activation('relu')(X)

    ### INICIE SEU CÓDIGO AQUI ###

    # Segundo componente do caminho principal (≈3 linhas)
    X = Conv2D(filters = F2, kernel_size = (f, f), strides =(1,1),p
adding = 'same', name = conv_name_base + '2b', kernel_initializer =
glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2b')(X)
    X = Activation('relu')(X)

    # Terceiro componente do caminho principal (≈2 linhas)
    X = Conv2D(filters = F3, kernel_size =(1, 1), strides = (1,1),
padding = 'valid', name = conv_name_base + '2c', kernel_initializer
= glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2c')(X)

    # Componente final: Adiciona o valor do atalho ao caminho princ
ipal, aplica a função de ativação RELU (2 linhas)
    X = Add()([X, X_shortcut])
    X = Activation('relu')(X)

    ### TÉRMINO DO CÓDIGO AQUI ###

    return X
```

In [21]:

```
tf.reset_default_graph()

with tf.Session() as test:
    np.random.seed(1)
    A_prev = tf.placeholder("float", [3, 4, 4, 6])
    X = np.random.randn(3, 4, 4, 6)
    A = identity_block(A_prev, f = 2, filters = [2, 4, 6], stage =
1, block = 'a')
    test.run(tf.global_variables_initializer())
    out = test.run([A], feed_dict={A_prev: X, K.learning_phase(): 0
})
    print("saída = " + str(out[0][1][1][0]))
```

```
saída = [ 0.94823   -0.         1.1610144  2.747859  -0.         1
.36677  ]
```

Saída esperada:

saída	[0.94822985 0. 1.16101444 2.747859 0. 1.36677003]

2.2 - O bloco de convolução

Foi implementado o bloco identidade da ResNet. Em seguida, o bloco convolucional da ResNet é o próximo que iremos implementar. Você pode utilizar este tipo de bloco quando as dimensões da entrada e da saída são diferentes. A diferença, com relação ao bloco de identidade é que existe uma camada CONV2D no atalho:

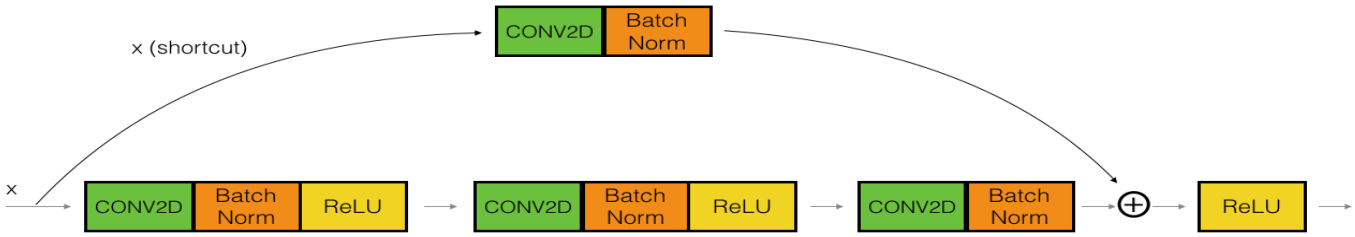


Figura 4 : **Bloco Convolucional**

A camada CONV2D no atalho é utilizada para redimensionar a entrada x para uma dimensão diferente e compatibilizar com a adição no final do atalho com o caminho principal. (Este bloco tem função semelhante à matriz W_s discutida em aula) Por exemplo, para reduzir a dimensão da altura e largura na ativação por um fator de 2, pode-se utilizar uma convolução de 1x1 com stride igual a 2. A camada CONV2D no atalho não utiliza função de ativação não-linear. Sua função básica é aplicar uma função linear aprendida que reduz a dimensão da entrada e compatibiliza com a dimensão a ser somada no caminho principal.
Os detalhes do bloco de convolução são os seguintes:

Primeiro componente do caminho principal:

- O primeiro CONV2D possui F_1 filtros no formato (1,1) e com stride de (s,s). O padding utilizado é "valid" e seu nome deve ser `conv_name_base + '2a'`.
- O primeiro BatchNorm normaliza o eixo dos canais. Seu nome deve ser `bn_name_base + '2a'`.
- Aplique a função de ativação ReLU. Está função não possui nome e nem hiper parâmetros.

Segunda componente do caminho principal:

- O segundo CONV2D possui F_2 filtros no formato (f,f) e uma stride de (1,1). O padding utilizado é "same" e seu nome deve ser `conv_name_base + '2b'`.
- O segundo BatchNorm normaliza o eixo dos canais. Seu nome deve ser `bn_name_base + '2b'`.
- Aplique a função de ativação ReLu. Está função não possui nome e nem hiper parâmetros.

Terceira componente do caminho principal:

- O terceiro CONV2D possui F_3 filtros no formato (1,1) e uma stride de (1,1). O padding utilizado é "valid" e seu nome deve ser `conv_name_base + '2c'`.
- O terceiro BatchNorm normaliza o eixo dos canais. Seu nome deve ser `bn_name_base + '2c'`. Note que não existe uma função de ativaçào ReLU nesta componente.

Atalho:

- O CONV2D tem F_3 filtros no formato (1,1) e uma stride de (s,s). O padding utilizado é "valid" e seu nome deve ser `conv_name_base + '1'`.
- O BatchNorm normaliza o eixo dos canais. Seu nome deve ser `bn_name_base + '1'`.

Componente Final:

- Os valores do atalho e do caminho principal são adicionados.
- A função de ativação ReLU é aplicada ao resultado da soma. Esta etapa não possui nome e nem hiper parâmetros.

Exercício: Implemente o bloco convolucional. A primeira componente do caminho principal já foi implementada para você, implemnte as demais componentes deste bloco. Como feito antes, utilize 0 como a semente para os incializadores aleatórios, para garantir consistência nos resultados.

- Dica Conv (https://keras.io/layers/convolutional/#conv2d)
- Dica BatchNorm (https://keras.io/layers/normalization/#batchnormalization) (axis: Integer, o eixo que deve ser normalizado (tipicamente o eixo de características))
- Para a ativação utilize: `Activation('relu')(X)`
- Dica Adição (https://keras.io/layers/merge/#add)

In [34]:
```
# FUNÇÃO DE AVALIAÇÃO: convolutional_block

def convolutional_block(X, f, filters, stage, block, s = 2):
    """
    Implementação do bloco convolucional como definido na Figura 4

    Argumentos:
    X -- tensor de entrada no formato (m, n_H_prev, n_W_prev, n_C_p
rev)
    f -- inteiro, especificando o formato do meio da janela do CONV
para o caminho principal.
    filters -- uma lista do python de inteiros, definindo o número
de filtros na camada CONV do caminho principal.
    stage -- inteiro, usado para dar nomes as camadas, dependendo d
e sua posição na rede.
    block -- string/caracter, usado para dar nome às camadas, depen
dendo de sua posição na rede.
    s -- Inteiro, especifica o valor do stride a ser utilizado.

    Retorna:
    X -- saída do bloco convolucional, tensor no formato (n_H, n_W,
n_C)
```

```
    """

    # define o nome base
    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'

    # Recupera os filtros
    F1, F2, F3 = filters

    # Salva o valor da entrada
    X_shortcut = X


    ##### CAMINHO PRINCIPAL #####
    # Primeira componente do caminho principal
    X = Conv2D(F1, (1, 1), strides =(s,s), padding = 'valid', name
= conv_name_base + '2a',kernel_initializer = glorot_uniform(seed=0)
)(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2a')(X)
    X = Activation('relu')(X)

    ### INICIE O SEU CÓDIGO AQUI ###

    # Segunda componente do caminho principal (≈3 linhas)
    X = Conv2D(F2, (f, f), strides = (1,1), padding ='same', name =
conv_name_base + '2b', kernel_initializer = glorot_uniform(seed=0))
(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2b')(X)
    X = Activation('relu')(X)

    # Terceira componente do caminho principal (≈2 linhas)
    X = Conv2D(F3, (1, 1), strides = (1,1),  padding = 'valid', nam
e = conv_name_base + '2c', kernel_initializer = glorot_uniform(seed
=0))(X)
    X = BatchNormalization(axis = 3, name = bn_name_base + '2c')(X)

    ##### ATALHO #### (≈2 linhas)
    X_shortcut = Conv2D(F3, (1, 1), strides = (s,s),  padding = 'va
lid', name = conv_name_base + '1', kernel_initializer = glorot_unif
orm(seed=0))(X_shortcut)
    X_shortcut = BatchNormalization(axis = 3, name = bn_name_base +
'1')(X_shortcut)

    # Componente final: Adiciona os valores do atalho ao caminhopri
ncipal e passa pela função de ativação ReLU (≈2 linhas)
    X = Add()([X, X_shortcut])
    X = Activation('relu')(X)

    ### TÉRMINO DO CÓDIGO ###

    return X
```

In [35]:
```
tf.reset_default_graph()

with tf.Session() as test:
    np.random.seed(1)
    A_prev = tf.placeholder("float", [3, 4, 4, 6])
    X = np.random.randn(3, 4, 4, 6)
    A = convolutional_block(A_prev, f = 2, filters = [2, 4, 6], sta
ge = 1, block = 'a')
    test.run(tf.global_variables_initializer())
    out = test.run([A], feed_dict={A_prev: X, K.learning_phase(): 0
})
    print("saída = " + str(out[0][1][1][0]))
```

```
saída = [ 0.09018461  1.2348979   0.46822017  0.03671762 -0.
0.65516603]
```

Saída esperada:

saída	[0.09018463 1.23489773 0.46822017 0.0367176 0. 0.65516603]

3 - Construindo seu primeiro modelo ResNet (50 camadas)

Você já tem os blocos necessários para construir a sua primeira rede muito profunda do tipo ResNet. A Figura a seguir descreve em detalhes a arquitetura desta rede neural. "ID BLOCK" no diagrama significa um "Bloco Identidade," e "ID BLOCK x3" significa que você deve combinar 3 blocos identidade juntos.

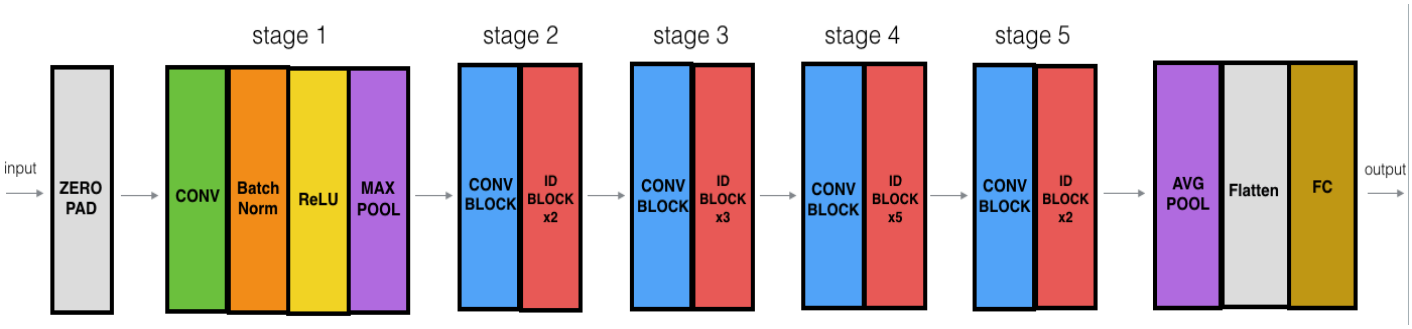


Figura 5 : **ResNet-50 model**

Os detalhes do modelo ResNet-50 são:

- Zero-padding adiciona zeros à entrada no formato (3,3)
- Estágio 1:
  - A Convolução 2D possui 64 filtros no formato (7,7) e utiliza um stride de (2,2). Seu nome é "conv1".
  - BatchNorm é aplicado ao eixo dos canais da entrada.
  - MaxPooling utiliza uma janela (3,3) e um stride de (2,2).
- Estágio 2:

    - O bloco convolucional utiliza três conjuntos de filtros de tamanho [64,64,256], "f" é 3, "s" é 1 e o bloco é "a".
    - O 2 bloco de identidade utiliza três conjuntos de filtros de tamanho [64,64,256], "f" é 3 e os blocos são "b" e "c".
- Estágio 3:
    - O bloco convolucional utiliza três conjuntos de filtros de tamanho [128,128,512], "f" é 3, "s" é 2 e o bloco é o "a".
    - Os 3 blocos de identidade utilizam três conjuntos de filtros de tamanho [128,128,512], "f" é 3 e os blocos são "b", "c" e "d".
- Estágio 4:
    - O bloco convolucional utiliza três conjuntos de filtros de tamanho [256, 256, 1024], "f" é 3, "s" é 2 e o bloco é o "a".
    - Os 5 blocos de identidade utilizam três conjuntos de filtros de tamanho [256, 256, 1024], "f" é 3 e os blocos são "b", "c", "d", "e" e "f".
- Estágio 5:
    - O bloco convolucional utiliza três conjuntos de filtros de tamanho [512, 512, 2048], "f" é 3, "s" é 2 e o bloco é o "a".
    - Os 2 blocos de identidade utilizam três conjuntos de filtros de tamanho [512, 512, 2048], "f" é 3 e os blocos são "b" e "c".
- O Average Pooling 2D utiliza uma janela no formato (2,2) e seu nome é "avg_pool".
- A vetorização não possui nome ou hiper parâmetros.
- A camada totalmente conectada (Densa) reduz sua entrada para o número de classes utilizando uma ativação do tipo softmax. Seu nome deve ser `'fc' + str(classes)`.

Exercício: Implemente a ResNet com 50 camadas descrita na figura acima. Os estágios 1 e 2 já estão implementados, implemente o restante. (A sintaxe para implementar os estágios de 3 a 5 deve ser similares aos do 2o estágio.) Tenha certeza de seguir as convenções de nomes dados no texto acima.

Voce irá precisar desta função:

- Average pooling veja referência (https://keras.io/layers/pooling/#averagepooling2d)

Aqui estão algumas outras funções que serão usadas no código abaixo:

- Conv2D: Veja referência (https://keras.io/layers/convolutional/#conv2d)
- BatchNorm: Veja referência (https://keras.io/layers/normalization/#batchnormalization) (axis: Inteiro, o eixo onde deve ocorrer a normalização (tipicamente o eixo de características))
- Zero padding: Veja referência (https://keras.io/layers/convolutional/#zeropadding2d)
- Max pooling: Veja referência (https://keras.io/layers/pooling/#maxpooling2d)
- Camada Fully conected: Veja referência (https://keras.io/layers/core/#dense)
- Addition: Veja referência (https://keras.io/layers/merge/#add)

In [54]:
```
# FUNÇÃO DE AVALIAÇÃO: ResNet50

def ResNet50(input_shape = (64, 64, 3), classes = 6):
    """
    Implementação da popular ResNet50 com a seginte arquitetura:
    CONV2D -> BATCHNORM -> RELU -> MAXPOOL -> CONVBLOCK -> IDBLOCK*
```

```
2 -> CONVBLOCK -> IDBLOCK*3
    -> CONVBLOCK -> IDBLOCK*5 -> CONVBLOCK -> IDBLOCK*2 -> AVGPOOL
-> TOPLAYER

    Argumentos:
    input_shape -- formato das imagens da base de dados.
    classes -- inteiro, número de classes.

    Retorna:
    model -- uma instância de modelo do Keras
    """

    # Define a entrada como um tensor com formato input_shape
    X_input = Input(input_shape)


    # Zero-Padding
    X = ZeroPadding2D((3, 3))(X_input)

    # Estágio 1
    X = Conv2D(64, (7, 7), strides = (2, 2), name = 'conv1', kernel
_initializer = glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis = 3, name = 'bn_conv1')(X)
    X = Activation('relu')(X)
    X = MaxPooling2D((3, 3), strides=(2, 2))(X)

    # Estágio 2
    X = convolutional_block(X, f = 3, filters = [64, 64, 256], stag
e = 2, block='a', s = 1)
    X = identity_block(X, 3, [64, 64, 256], stage=2, block='b')
    X = identity_block(X, 3, [64, 64, 256], stage=2, block='c')

    ### INICIO DO CÓDIGO AQUI ###

    # Estágio 3 (≈4 linhas)
    X = convolutional_block(X, f = 3, filters = [128,128,512], stag
e = 3, block='a', s = 2)
    X = identity_block(X, 3, [128,128,512], stage=3, block='b')
    X = identity_block(X, 3, [128,128,512], stage=3, block='c')
    X = identity_block(X, 3, [128,128,512], stage=3, block='d')

    # Estágio 4 (≈6 linhas)
    X = convolutional_block(X, f = 3, filters = [256, 256, 1024], s
tage = 4, block='a', s = 2)
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='b')
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='c')
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='d')
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='e')
    X = identity_block(X, 3, [256, 256, 1024], stage=4, block='f')

    # Estágio 5 (≈3 linhas)
    X = convolutional_block(X, f = 3, filters = [512, 512, 2048], s
tage = 5, block='a', s = 2)
    X = identity_block(X, 3, [512, 512, 2048], stage=5, block='b')
```

```
    X = identity_block(X, 3, [512, 512, 2048], stage=5, block='c')

    # AVGPOOL (≈1 linha). Use "X = AveragePooling2D(...)(X)"
    X = AveragePooling2D(pool_size=(2, 2), name ="avg_pool")(X)

    ### TÉRMINO DO CÓDIGO ###

    # Camada de saída
    X = Flatten()(X)
    X = Dense(classes, activation='softmax', name='fc' + str(classe
s), kernel_initializer = glorot_uniform(seed=0))(X)


    # Cria modelo
    model = Model(inputs = X_input, outputs = X, name='ResNet50')

    return model
```

Execute a célula abaixo para construir o grafo do modelo. Se sua implementação não estiver correta você perceberá verificando a precisão quando executar o `model.fit(...)` abaixo.

In [55]:
```
model = ResNet50(input_shape = (64, 64, 3), classes = 6)
```

Como visto no tutorial do Keras, antes de treinar o modelo é preciso configurar oprocesso de aprendizado compilando o modelo.

In [56]:
```
model.compile(optimizer='adam', loss='categorical_crossentropy', me
trics=['accuracy'])
```

O modelo está pronto para ser treinado. A única coisa que você precisa é uma base de dados.

Vamos carregar novamente a base de dados SIGNS.

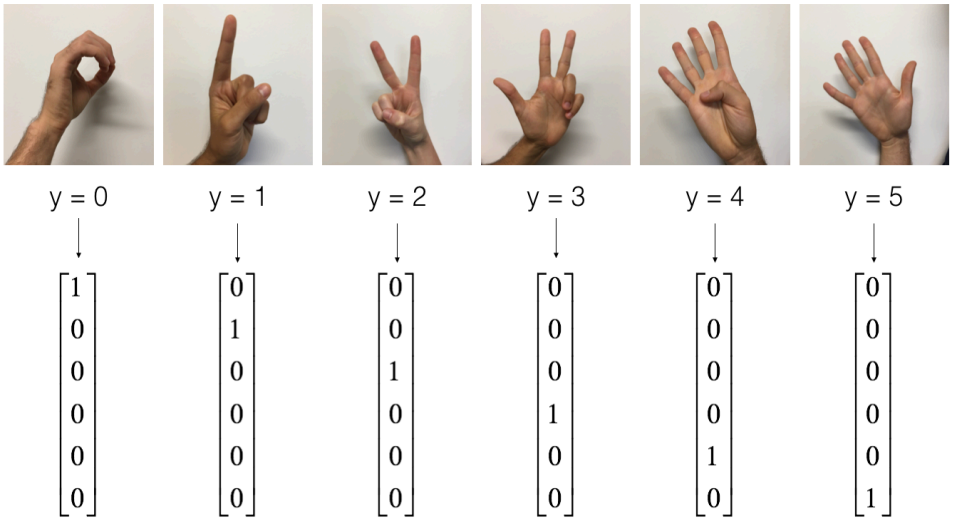


Figura 6 : **Base de dados SIGNS**

In [57]:
```
X_train_orig, Y_train_orig, X_test_orig, Y_test_orig, classes = loa
d_dataset()

# Normaliza os vetores das imagens
X_train = X_train_orig/255.
X_test = X_test_orig/255.

# Converte os rótulos de treinamento e teste em uma matriz do tipo
one hot.
Y_train = convert_to_one_hot(Y_train_orig, 6).T
Y_test = convert_to_one_hot(Y_test_orig, 6).T

print ("número de exemplos de treinamento = " + str(X_train.shape[0
]))
print ("número de exemplos de teste = " + str(X_test.shape[0]))
print ("Formato do X_train: " + str(X_train.shape))
print ("Formato do Y_train: " + str(Y_train.shape))
print ("Formato do X_test: " + str(X_test.shape))
print ("Formato do Y_test: " + str(Y_test.shape))
```

```
número de exemplos de treinamento = 1080
número de exemplos de teste = 120
Formato do X_train: (1080, 64, 64, 3)
Formato do Y_train: (1080, 6)
Formato do X_test: (120, 64, 64, 3)
Formato do Y_test: (120, 6)
```

Execute a célula abaixo por 5 épocas com um tamanho de batch de 32. Em CPU isto deve levar em torno de 5 minutos por época.

In [58]:
```
model.fit(X_train, Y_train, epochs = 5, batch_size = 32)
```

```
Epoch 1/5
1080/1080 [==============================] - 180s 167ms/step - los
s: 2.2991 - acc: 0.4444
Epoch 2/5
1080/1080 [==============================] - 87s 81ms/step - loss:
1.4249 - acc: 0.6046
Epoch 3/5
1080/1080 [==============================] - 97s 90ms/step - loss:
0.8542 - acc: 0.7083
Epoch 4/5
1080/1080 [==============================] - 85s 79ms/step - loss:
0.3303 - acc: 0.9028
Epoch 5/5
1080/1080 [==============================] - 82s 76ms/step - loss:
0.2695 - acc: 0.9046
```

Out[58]:
```
<tensorflow.python.keras.callbacks.History at 0xbdf53c860>
```

Saída esperada:

** Época 1/5**	perda: entre 1 e 5, precisão: entre 0.2 e 0.5, embora seus resultados possam ser um pouco diferentes.
** Época 2/5**	perda: entre 1 e 5, precisão: entre 0.2 e 0.5, você deve ver a perda decrescer e a precisão aumentar.

Vamos ver como o seu modelo se comporta (mesmo treinado com apenas 5 épocas) no conjunto de teste.

In [59]:
```
preds = model.evaluate(X_test, Y_test)
print ("Perda = " + str(preds[0]))
print ("Precisão no teste = " + str(preds[1]))
```

```
120/120 [==============================] - 8s 70ms/step
Perda = 4.133584372202555
Precisão no teste = 0.16666666666666666
```

Saída esperada:

Precisão do teste	entre 0.16 e 0.25

Para o propósito desta tarefa foi pedido para treinar o modelo em apenas 5 épocas. Você pode perceber que o desempenho não é muito bom. Após salvar o seu trabalho, caso deseje, rode o treinamento por mais épocas e compare os resultados. O desempenho melhora bem após 20 épocas, mas isto deve levar mais de 1 hora em CPU.

Foi treinada uma rede igual a que você desenvolveu utilizando GPU sobre a base de dados SIGNS. Você pode carregar esta rede executando a célula abaixo e testando esta rede no conjunto de testes.

In [61]:
```
model = load_model('ResNet50.h5')
```

```
---------------------------------------------------------------------
---------
KeyError                                  Traceback (most recent c
all last)
<ipython-input-61-65db21b7dfed> in <module>()
----> 1 model = load_model('ResNet50.h5')

/anaconda3/lib/python3.6/site-packages/tensorflow/python/keras/eng
ine/saving.py in load_model(filepath, custom_objects, compile)
    249       metrics = convert_custom_objects(training_config['me
trics'])
    250       weighted_metrics = convert_custom_objects(
--> 251           training_config['weighted_metrics'])
    252       sample_weight_mode = training_config['sample_weight_
mode']
    253       loss_weights = training_config['loss_weights']

KeyError: 'weighted_metrics'
```

In [62]:
```
preds = model.evaluate(X_test, Y_test)
print ("Perda = " + str(preds[0]))
print ("Precisão no Teste = " + str(preds[1]))
```

```
120/120 [==============================] - 12s 101ms/step
Perda = 4.133584372202555
Precisão no Teste = 0.16666666666666666
```

ResNet50 é um modelo poderoso para classificação de imagens quando treinado por um número adequado de épocas. Espero que você possa utilizar o que você aprendeu e aplicar em algum caso de classificação e obter uma precisão da ordem do estado da arte.

Parabéns, você concluiu esta tarefa! Você implementou um sistema de classificação de alto nível com desempenho da ordem do estado da arte.

4 - Teste na sua própria imagem (Opcional)

Se você desejar, você pode tirar uma foto de sua mão e ver o resultado do modelo com esta foto. Para fazer isto:

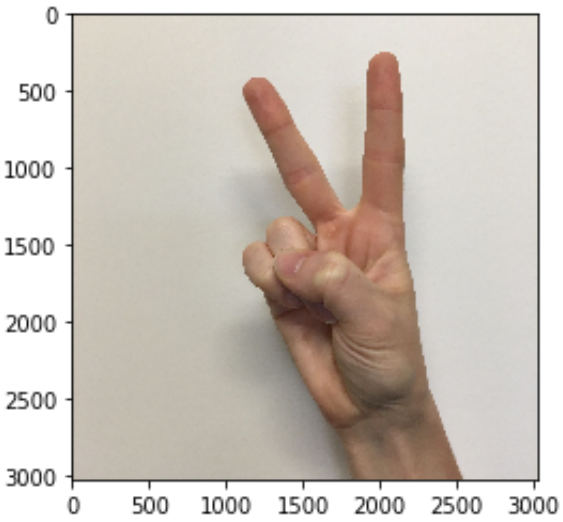
1. Clique em "File" na barra superior deste notebook, e clique em "Open " para ir para o diretório do notebook.
2. Adicione sua imagem ao diretório imagens do notebook.
3. Escreva o nome da sua imagem na célula abaixo.
4. Execute o código da célula abaixo e veja se o algoritmo acerta!

In [65]:
```
img_path = 'images/my_image.jpg'
img = image.load_img(img_path, target_size=(64, 64))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
#x = preprocess_input(x)
print('Formato da imagem de entrada:', x.shape)
my_image = scipy.misc.imread(img_path)
imshow(my_image)
print("vetor para prever a classe [p(0), p(1), p(2), p(3), p(4), p(5)] = ")
print(model.predict(x))
```

```
Formato da imagem de entrada: (1, 64, 64, 3)

/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:7: DeprecationWarning: `imread` is deprecated!
`imread` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imread`` instead.
  import sys

vetor para prever a classe [p(0), p(1), p(2), p(3), p(4), p(5)] = 
[[1. 0. 0. 0. 0. 0.]]
```



Voce pode ainda imprimir o resumo do seu modelo executando a célula abaixo.

In [66]:
```
model.summary()
```

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_4 (InputLayer)            (None, 64, 64, 3)    0
__________________________________________________________________________________________________
zero_padding2d_3 (ZeroPadding2D (None, 70, 70, 3)    0           input_4[0][0]
__________________________________________________________________________________________________
conv1 (Conv2D)                  (None, 32, 32, 64)   9472        zero_padding2d_3[0][0]
__________________________________________________________________________________________________
bn_conv1 (BatchNormalization)   (None, 32, 32, 64)   256         conv1[0][0]
__________________________________________________________________________________________________
activation_140 (Activation)     (None, 32, 32, 64)   0           bn_conv1[0][0]
__________________________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D)  (None, 15, 15, 64)   0           activation_140[0][0]
__________________________________________________________________________________________________
res2a_branch2a (Conv2D)         (None, 15, 15, 64)   4160        max_pooling2d_3[0][0]
__________________________________________________________________________________________________
bn2a_branch2a (BatchNormalizati (None, 15, 15, 64)   256         res2a_branch2a[0][0]
__________________________________________________________________________________________________
activation_141 (Activation)     (None, 15, 15, 64)   0           bn2a_branch2a[0][0]
__________________________________________________________________________________________________
res2a_branch2b (Conv2D)         (None, 15, 15, 64)   36928       activation_141[0][0]
__________________________________________________________________________________________________
bn2a_branch2b (BatchNormalizati (None, 15, 15, 64)   256         res2a_branch2b[0][0]
__________________________________________________________________________________________________
```

```
__________________________________________________________________________________________________
activation_142 (Activation)     (None, 15, 15, 64)   0           b
n2a_branch2b[0][0]
__________________________________________________________________________________________________
res2a_branch2c (Conv2D)         (None, 15, 15, 256)  16640       a
ctivation_142[0][0]
__________________________________________________________________________________________________
res2a_branch1 (Conv2D)          (None, 15, 15, 256)  16640       m
ax_pooling2d_3[0][0]
__________________________________________________________________________________________________
bn2a_branch2c (BatchNormalizati (None, 15, 15, 256)  1024        r
es2a_branch2c[0][0]
__________________________________________________________________________________________________
bn2a_branch1 (BatchNormalizatio (None, 15, 15, 256)  1024        r
es2a_branch1[0][0]
__________________________________________________________________________________________________
add_45 (Add)                    (None, 15, 15, 256)  0           b
n2a_branch2c[0][0]

bn2a_branch1[0][0]
__________________________________________________________________________________________________
activation_143 (Activation)     (None, 15, 15, 256)  0           a
dd_45[0][0]
__________________________________________________________________________________________________
res2b_branch2a (Conv2D)         (None, 15, 15, 64)   16448       a
ctivation_143[0][0]
__________________________________________________________________________________________________
bn2b_branch2a (BatchNormalizati (None, 15, 15, 64)   256         r
es2b_branch2a[0][0]
__________________________________________________________________________________________________
activation_144 (Activation)     (None, 15, 15, 64)   0           b
n2b_branch2a[0][0]
__________________________________________________________________________________________________
res2b_branch2b (Conv2D)         (None, 15, 15, 64)   36928       a
ctivation_144[0][0]
__________________________________________________________________________________________________
bn2b_branch2b (BatchNormalizati (None, 15, 15, 64)   256         r
es2b_branch2b[0][0]
__________________________________________________________________________________________________
activation_145 (Activation)     (None, 15, 15, 64)   0           b
n2b_branch2b[0][0]
```

```
__________________________________________________________________________________________________
res2b_branch2c (Conv2D)         (None, 15, 15, 256)  16640       a
ctivation_145[0][0]
__________________________________________________________________________________________________
bn2b_branch2c (BatchNormalizati (None, 15, 15, 256)  1024        r
es2b_branch2c[0][0]
__________________________________________________________________________________________________
add_46 (Add)                    (None, 15, 15, 256)  0           b
n2b_branch2c[0][0]

activation_143[0][0]
__________________________________________________________________________________________________
activation_146 (Activation)     (None, 15, 15, 256)  0           a
dd_46[0][0]
__________________________________________________________________________________________________
res2c_branch2a (Conv2D)         (None, 15, 15, 64)   16448       a
ctivation_146[0][0]
__________________________________________________________________________________________________
bn2c_branch2a (BatchNormalizati (None, 15, 15, 64)   256         r
es2c_branch2a[0][0]
__________________________________________________________________________________________________
activation_147 (Activation)     (None, 15, 15, 64)   0           b
n2c_branch2a[0][0]
__________________________________________________________________________________________________
res2c_branch2b (Conv2D)         (None, 15, 15, 64)   36928       a
ctivation_147[0][0]
__________________________________________________________________________________________________
bn2c_branch2b (BatchNormalizati (None, 15, 15, 64)   256         r
es2c_branch2b[0][0]
__________________________________________________________________________________________________
activation_148 (Activation)     (None, 15, 15, 64)   0           b
n2c_branch2b[0][0]
__________________________________________________________________________________________________
res2c_branch2c (Conv2D)         (None, 15, 15, 256)  16640       a
ctivation_148[0][0]
__________________________________________________________________________________________________
bn2c_branch2c (BatchNormalizati (None, 15, 15, 256)  1024        r
es2c_branch2c[0][0]
__________________________________________________________________________________________________
add_47 (Add)                    (None, 15, 15, 256)  0           b
n2c_branch2c[0][0]
```

```
activation_146[0][0]
__________________________________________________________________________________________________
____________________________________
activation_149 (Activation)     (None, 15, 15, 256)  0           a
dd_47[0][0]
__________________________________________________________________________________________________
____________________________________
res3a_branch2a (Conv2D)         (None, 8, 8, 128)    32896       a
ctivation_149[0][0]
__________________________________________________________________________________________________
____________________________________
bn3a_branch2a (BatchNormalizati (None, 8, 8, 128)    512         r
es3a_branch2a[0][0]
__________________________________________________________________________________________________
____________________________________
activation_150 (Activation)     (None, 8, 8, 128)    0           b
n3a_branch2a[0][0]
__________________________________________________________________________________________________
____________________________________
res3a_branch2b (Conv2D)         (None, 8, 8, 128)    147584      a
ctivation_150[0][0]
__________________________________________________________________________________________________
____________________________________
bn3a_branch2b (BatchNormalizati (None, 8, 8, 128)    512         r
es3a_branch2b[0][0]
__________________________________________________________________________________________________
____________________________________
activation_151 (Activation)     (None, 8, 8, 128)    0           b
n3a_branch2b[0][0]
__________________________________________________________________________________________________
____________________________________
res3a_branch2c (Conv2D)         (None, 8, 8, 512)    66048       a
ctivation_151[0][0]
__________________________________________________________________________________________________
____________________________________
res3a_branch1 (Conv2D)          (None, 8, 8, 512)    131584      a
ctivation_149[0][0]
__________________________________________________________________________________________________
____________________________________
bn3a_branch2c (BatchNormalizati (None, 8, 8, 512)    2048        r
es3a_branch2c[0][0]
__________________________________________________________________________________________________
____________________________________
bn3a_branch1 (BatchNormalizatio (None, 8, 8, 512)    2048        r
es3a_branch1[0][0]
__________________________________________________________________________________________________
____________________________________
add_48 (Add)                    (None, 8, 8, 512)    0           b
n3a_branch2c[0][0]

bn3a_branch1[0][0]
__________________________________________________________________________________________________
```

```
__________________________________________________________________________________________________
activation_152 (Activation)     (None, 8, 8, 512)    0           a
dd_48[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
res3b_branch2a (Conv2D)         (None, 8, 8, 128)    65664       a
ctivation_152[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
bn3b_branch2a (BatchNormalizati (None, 8, 8, 128)    512         r
es3b_branch2a[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
activation_153 (Activation)     (None, 8, 8, 128)    0           b
n3b_branch2a[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
res3b_branch2b (Conv2D)         (None, 8, 8, 128)    147584      a
ctivation_153[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
bn3b_branch2b (BatchNormalizati (None, 8, 8, 128)    512         r
es3b_branch2b[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
activation_154 (Activation)     (None, 8, 8, 128)    0           b
n3b_branch2b[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
res3b_branch2c (Conv2D)         (None, 8, 8, 512)    66048       a
ctivation_154[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
bn3b_branch2c (BatchNormalizati (None, 8, 8, 512)    2048        r
es3b_branch2c[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
add_49 (Add)                    (None, 8, 8, 512)    0           b
n3b_branch2c[0][0]

activation_152[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
activation_155 (Activation)     (None, 8, 8, 512)    0           a
dd_49[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
res3c_branch2a (Conv2D)         (None, 8, 8, 128)    65664       a
ctivation_155[0][0]
__________________________________________________________________________________________________
__________________________________________________________________________________________________
bn3c_branch2a (BatchNormalizati (None, 8, 8, 128)    512         r
es3c_branch2a[0][0]
__________________________________________________________________________________________________
```

activation_156 (Activation) n3c_branch2a[0][0]	(None, 8, 8, 128)	0	b
res3c_branch2b (Conv2D) ctivation_156[0][0]	(None, 8, 8, 128)	147584	a
bn3c_branch2b (BatchNormalizati es3c_branch2b[0][0]	(None, 8, 8, 128)	512	r
activation_157 (Activation) n3c_branch2b[0][0]	(None, 8, 8, 128)	0	b
res3c_branch2c (Conv2D) ctivation_157[0][0]	(None, 8, 8, 512)	66048	a
bn3c_branch2c (BatchNormalizati es3c_branch2c[0][0]	(None, 8, 8, 512)	2048	r
add_50 (Add) n3c_branch2c[0][0] activation_155[0][0]	(None, 8, 8, 512)	0	b
activation_158 (Activation) dd_50[0][0]	(None, 8, 8, 512)	0	a
res3d_branch2a (Conv2D) ctivation_158[0][0]	(None, 8, 8, 128)	65664	a
bn3d_branch2a (BatchNormalizati es3d_branch2a[0][0]	(None, 8, 8, 128)	512	r
activation_159 (Activation) n3d_branch2a[0][0]	(None, 8, 8, 128)	0	b
res3d_branch2b (Conv2D) ctivation_159[0][0]	(None, 8, 8, 128)	147584	a
bn3d_branch2b (BatchNormalizati es3d_branch2b[0][0]	(None, 8, 8, 128)	512	r

activation_160 (Activation) n3d_branch2b[0][0]	(None, 8, 8, 128)	0	b
res3d_branch2c (Conv2D) ctivation_160[0][0]	(None, 8, 8, 512)	66048	a
bn3d_branch2c (BatchNormalizati es3d_branch2c[0][0]	(None, 8, 8, 512)	2048	r
add_51 (Add) n3d_branch2c[0][0] activation_158[0][0]	(None, 8, 8, 512)	0	b
activation_161 (Activation) dd_51[0][0]	(None, 8, 8, 512)	0	a
res4a_branch2a (Conv2D) ctivation_161[0][0]	(None, 4, 4, 256)	131328	a
bn4a_branch2a (BatchNormalizati es4a_branch2a[0][0]	(None, 4, 4, 256)	1024	r
activation_162 (Activation) n4a_branch2a[0][0]	(None, 4, 4, 256)	0	b
res4a_branch2b (Conv2D) ctivation_162[0][0]	(None, 4, 4, 256)	590080	a
bn4a_branch2b (BatchNormalizati es4a_branch2b[0][0]	(None, 4, 4, 256)	1024	r
activation_163 (Activation) n4a_branch2b[0][0]	(None, 4, 4, 256)	0	b
res4a_branch2c (Conv2D) ctivation_163[0][0]	(None, 4, 4, 1024)	263168	a
res4a_branch1 (Conv2D) ctivation_161[0][0]	(None, 4, 4, 1024)	525312	a

```
__________________________________________________________________________________________________
bn4a_branch2c (BatchNormalizati (None, 4, 4, 1024)   4096        r
es4a_branch2c[0][0]
__________________________________________________________________________________________________
bn4a_branch1 (BatchNormalizatio (None, 4, 4, 1024)   4096        r
es4a_branch1[0][0]
__________________________________________________________________________________________________
add_52 (Add)                    (None, 4, 4, 1024)   0           b
n4a_branch2c[0][0]

bn4a_branch1[0][0]
__________________________________________________________________________________________________
activation_164 (Activation)     (None, 4, 4, 1024)   0           a
dd_52[0][0]
__________________________________________________________________________________________________
res4b_branch2a (Conv2D)         (None, 4, 4, 256)    262400      a
ctivation_164[0][0]
__________________________________________________________________________________________________
bn4b_branch2a (BatchNormalizati (None, 4, 4, 256)    1024        r
es4b_branch2a[0][0]
__________________________________________________________________________________________________
activation_165 (Activation)     (None, 4, 4, 256)    0           b
n4b_branch2a[0][0]
__________________________________________________________________________________________________
res4b_branch2b (Conv2D)         (None, 4, 4, 256)    590080      a
ctivation_165[0][0]
__________________________________________________________________________________________________
bn4b_branch2b (BatchNormalizati (None, 4, 4, 256)    1024        r
es4b_branch2b[0][0]
__________________________________________________________________________________________________
activation_166 (Activation)     (None, 4, 4, 256)    0           b
n4b_branch2b[0][0]
__________________________________________________________________________________________________
res4b_branch2c (Conv2D)         (None, 4, 4, 1024)   263168      a
ctivation_166[0][0]
__________________________________________________________________________________________________
bn4b_branch2c (BatchNormalizati (None, 4, 4, 1024)   4096        r
es4b_branch2c[0][0]
__________________________________________________________________________________________________
add_53 (Add)                    (None, 4, 4, 1024)   0           b
```

```
n4b_branch2c[0][0]

activation_164[0][0]
__________________________________________________________________________
__________________________________________
activation_167 (Activation)     (None, 4, 4, 1024)   0           a
dd_53[0][0]
__________________________________________________________________________
__________________________________________
res4c_branch2a (Conv2D)         (None, 4, 4, 256)    262400      a
ctivation_167[0][0]
__________________________________________________________________________
__________________________________________
bn4c_branch2a (BatchNormalizati (None, 4, 4, 256)    1024        r
es4c_branch2a[0][0]
__________________________________________________________________________
__________________________________________
activation_168 (Activation)     (None, 4, 4, 256)    0           b
n4c_branch2a[0][0]
__________________________________________________________________________
__________________________________________
res4c_branch2b (Conv2D)         (None, 4, 4, 256)    590080      a
ctivation_168[0][0]
__________________________________________________________________________
__________________________________________
bn4c_branch2b (BatchNormalizati (None, 4, 4, 256)    1024        r
es4c_branch2b[0][0]
__________________________________________________________________________
__________________________________________
activation_169 (Activation)     (None, 4, 4, 256)    0           b
n4c_branch2b[0][0]
__________________________________________________________________________
__________________________________________
res4c_branch2c (Conv2D)         (None, 4, 4, 1024)   263168      a
ctivation_169[0][0]
__________________________________________________________________________
__________________________________________
bn4c_branch2c (BatchNormalizati (None, 4, 4, 1024)   4096        r
es4c_branch2c[0][0]
__________________________________________________________________________
__________________________________________
add_54 (Add)                    (None, 4, 4, 1024)   0           b
n4c_branch2c[0][0]

activation_167[0][0]
__________________________________________________________________________
__________________________________________
activation_170 (Activation)     (None, 4, 4, 1024)   0           a
dd_54[0][0]
__________________________________________________________________________
__________________________________________
res4d_branch2a (Conv2D)         (None, 4, 4, 256)    262400      a
ctivation_170[0][0]
__________________________________________________________________________
```

```
____________________________________
bn4d_branch2a (BatchNormalizati (None, 4, 4, 256)    1024        r
es4d_branch2a[0][0]
____________________________________
____________________________________
activation_171 (Activation)     (None, 4, 4, 256)    0           b
n4d_branch2a[0][0]
____________________________________
____________________________________
res4d_branch2b (Conv2D)         (None, 4, 4, 256)    590080      a
ctivation_171[0][0]
____________________________________
____________________________________
bn4d_branch2b (BatchNormalizati (None, 4, 4, 256)    1024        r
es4d_branch2b[0][0]
____________________________________
____________________________________
activation_172 (Activation)     (None, 4, 4, 256)    0           b
n4d_branch2b[0][0]
____________________________________
____________________________________
res4d_branch2c (Conv2D)         (None, 4, 4, 1024)   263168      a
ctivation_172[0][0]
____________________________________
____________________________________
bn4d_branch2c (BatchNormalizati (None, 4, 4, 1024)   4096        r
es4d_branch2c[0][0]
____________________________________
____________________________________
add_55 (Add)                    (None, 4, 4, 1024)   0           b
n4d_branch2c[0][0]

activation_170[0][0]
____________________________________
____________________________________
activation_173 (Activation)     (None, 4, 4, 1024)   0           a
dd_55[0][0]
____________________________________
____________________________________
res4e_branch2a (Conv2D)         (None, 4, 4, 256)    262400      a
ctivation_173[0][0]
____________________________________
____________________________________
bn4e_branch2a (BatchNormalizati (None, 4, 4, 256)    1024        r
es4e_branch2a[0][0]
____________________________________
____________________________________
activation_174 (Activation)     (None, 4, 4, 256)    0           b
n4e_branch2a[0][0]
____________________________________
____________________________________
res4e_branch2b (Conv2D)         (None, 4, 4, 256)    590080      a
ctivation_174[0][0]
```

```
__________________________________________________________________________________________________
bn4e_branch2b (BatchNormalizati (None, 4, 4, 256)    1024        r
es4e_branch2b[0][0]
__________________________________________________________________________________________________
activation_175 (Activation)     (None, 4, 4, 256)    0           b
n4e_branch2b[0][0]
__________________________________________________________________________________________________
res4e_branch2c (Conv2D)         (None, 4, 4, 1024)   263168      a
ctivation_175[0][0]
__________________________________________________________________________________________________
bn4e_branch2c (BatchNormalizati (None, 4, 4, 1024)   4096        r
es4e_branch2c[0][0]
__________________________________________________________________________________________________
add_56 (Add)                    (None, 4, 4, 1024)   0           b
n4e_branch2c[0][0]

activation_173[0][0]
__________________________________________________________________________________________________
activation_176 (Activation)     (None, 4, 4, 1024)   0           a
dd_56[0][0]
__________________________________________________________________________________________________
res4f_branch2a (Conv2D)         (None, 4, 4, 256)    262400      a
ctivation_176[0][0]
__________________________________________________________________________________________________
bn4f_branch2a (BatchNormalizati (None, 4, 4, 256)    1024        r
es4f_branch2a[0][0]
__________________________________________________________________________________________________
activation_177 (Activation)     (None, 4, 4, 256)    0           b
n4f_branch2a[0][0]
__________________________________________________________________________________________________
res4f_branch2b (Conv2D)         (None, 4, 4, 256)    590080      a
ctivation_177[0][0]
__________________________________________________________________________________________________
bn4f_branch2b (BatchNormalizati (None, 4, 4, 256)    1024        r
es4f_branch2b[0][0]
__________________________________________________________________________________________________
activation_178 (Activation)     (None, 4, 4, 256)    0           b
n4f_branch2b[0][0]
__________________________________________________________________________________________________
res4f_branch2c (Conv2D)         (None, 4, 4, 1024)   263168      a
ctivation_178[0][0]
```

```
__________________________________________________________________________________________________
bn4f_branch2c (BatchNormalizati (None, 4, 4, 1024)   4096        r
es4f_branch2c[0][0]
__________________________________________________________________________________________________
add_57 (Add)                    (None, 4, 4, 1024)   0           b
n4f_branch2c[0][0]

activation_176[0][0]
__________________________________________________________________________________________________
activation_179 (Activation)     (None, 4, 4, 1024)   0           a
dd_57[0][0]
__________________________________________________________________________________________________
res5a_branch2a (Conv2D)         (None, 2, 2, 512)    524800      a
ctivation_179[0][0]
__________________________________________________________________________________________________
bn5a_branch2a (BatchNormalizati (None, 2, 2, 512)    2048        r
es5a_branch2a[0][0]
__________________________________________________________________________________________________
activation_180 (Activation)     (None, 2, 2, 512)    0           b
n5a_branch2a[0][0]
__________________________________________________________________________________________________
res5a_branch2b (Conv2D)         (None, 2, 2, 512)    2359808     a
ctivation_180[0][0]
__________________________________________________________________________________________________
bn5a_branch2b (BatchNormalizati (None, 2, 2, 512)    2048        r
es5a_branch2b[0][0]
__________________________________________________________________________________________________
activation_181 (Activation)     (None, 2, 2, 512)    0           b
n5a_branch2b[0][0]
__________________________________________________________________________________________________
res5a_branch2c (Conv2D)         (None, 2, 2, 2048)   1050624     a
ctivation_181[0][0]
__________________________________________________________________________________________________
res5a_branch1 (Conv2D)          (None, 2, 2, 2048)   2099200     a
ctivation_179[0][0]
__________________________________________________________________________________________________
bn5a_branch2c (BatchNormalizati (None, 2, 2, 2048)   8192        r
es5a_branch2c[0][0]
__________________________________________________________________________________________________
bn5a_branch1 (BatchNormalizatio (None, 2, 2, 2048)   8192        r
```

```
es5a_branch1[0][0]
__________________________________________________________________________________________________
____________________________________
add_58 (Add)                        (None, 2, 2, 2048)   0           b
n5a_branch2c[0][0]

bn5a_branch1[0][0]
__________________________________________________________________________________________________
____________________________________
activation_182 (Activation)         (None, 2, 2, 2048)   0           a
dd_58[0][0]
__________________________________________________________________________________________________
____________________________________
res5b_branch2a (Conv2D)             (None, 2, 2, 512)    1049088     a
ctivation_182[0][0]
__________________________________________________________________________________________________
____________________________________
bn5b_branch2a (BatchNormalizati (None, 2, 2, 512)    2048        r
es5b_branch2a[0][0]
__________________________________________________________________________________________________
____________________________________
activation_183 (Activation)         (None, 2, 2, 512)    0           b
n5b_branch2a[0][0]
__________________________________________________________________________________________________
____________________________________
res5b_branch2b (Conv2D)             (None, 2, 2, 512)    2359808     a
ctivation_183[0][0]
__________________________________________________________________________________________________
____________________________________
bn5b_branch2b (BatchNormalizati (None, 2, 2, 512)    2048        r
es5b_branch2b[0][0]
__________________________________________________________________________________________________
____________________________________
activation_184 (Activation)         (None, 2, 2, 512)    0           b
n5b_branch2b[0][0]
__________________________________________________________________________________________________
____________________________________
res5b_branch2c (Conv2D)             (None, 2, 2, 2048)   1050624     a
ctivation_184[0][0]
__________________________________________________________________________________________________
____________________________________
bn5b_branch2c (BatchNormalizati (None, 2, 2, 2048)   8192        r
es5b_branch2c[0][0]
__________________________________________________________________________________________________
____________________________________
add_59 (Add)                        (None, 2, 2, 2048)   0           b
n5b_branch2c[0][0]

activation_182[0][0]
__________________________________________________________________________________________________
____________________________________
activation_185 (Activation)         (None, 2, 2, 2048)   0           a
dd_59[0][0]
__________________________________________________________________________________________________
```

```
____________________________________
res5c_branch2a (Conv2D)          (None, 2, 2, 512)    1049088     a
ctivation_185[0][0]
____________________________________________________________________
____________________________________
bn5c_branch2a (BatchNormalizati (None, 2, 2, 512)    2048        r
es5c_branch2a[0][0]
____________________________________________________________________
____________________________________
activation_186 (Activation)      (None, 2, 2, 512)    0           b
n5c_branch2a[0][0]
____________________________________________________________________
____________________________________
res5c_branch2b (Conv2D)          (None, 2, 2, 512)    2359808     a
ctivation_186[0][0]
____________________________________________________________________
____________________________________
bn5c_branch2b (BatchNormalizati (None, 2, 2, 512)    2048        r
es5c_branch2b[0][0]
____________________________________________________________________
____________________________________
activation_187 (Activation)      (None, 2, 2, 512)    0           b
n5c_branch2b[0][0]
____________________________________________________________________
____________________________________
res5c_branch2c (Conv2D)          (None, 2, 2, 2048)   1050624     a
ctivation_187[0][0]
____________________________________________________________________
____________________________________
bn5c_branch2c (BatchNormalizati (None, 2, 2, 2048)   8192        r
es5c_branch2c[0][0]
____________________________________________________________________
____________________________________
add_60 (Add)                     (None, 2, 2, 2048)   0           b
n5c_branch2c[0][0]

activation_185[0][0]
____________________________________________________________________
____________________________________
activation_188 (Activation)      (None, 2, 2, 2048)   0           a
dd_60[0][0]
____________________________________________________________________
____________________________________
avg_pool (AveragePooling2D)      (None, 1, 1, 2048)   0           a
ctivation_188[0][0]
____________________________________________________________________
____________________________________
flatten_2 (Flatten)              (None, 2048)         0           a
vg_pool[0][0]
____________________________________________________________________
____________________________________
fc6 (Dense)                      (None, 6)            12294       f
latten_2[0][0]
```

```
==================================================================
================================
Total params: 23,600,006
Trainable params: 23,546,886
Non-trainable params: 53,120
__________________________________________________________________
________________________________
```

O que você deve se lembrar:

- Redes normais muito profundas não funcionam na prática porque elas são dificeis de serem treinadas pois os gradientes costumam ir para zero.
- Os atalhos ajudam a reduzir o problema dos gradientes indo para zero. Eles também auxiliam a ResNet a aprender o bloco identidade.
- Existem dois tipos principais de blocos: o bloco identidade e o bloco convolucional.
- Redes residuais muito profundas são construidas agrupando estes blocos.

Referências

Este notebook apresentou o algoritmo da ResNet desenvolvido por He et al. (2015). A implementação feita aqui teve inspiração também na estrutura dada no repositório do github de Francois Chollet:

- Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun - Deep Residual Learning for Image Recognition (2015) (https://arxiv.org/abs/1512.03385)
- Francois Chollet's repositório github: https://github.com/fchollet/deep-learning-models/blob/master/resnet50.py (https://github.com/fchollet/deep-learning-models/blob/master/resnet50.py)