



Aprendizaje Automático Profundo (Deep Learning)

Dr. Facundo Quiroga - Dr. Franco Ronchetti



Arquitectura Inception V1

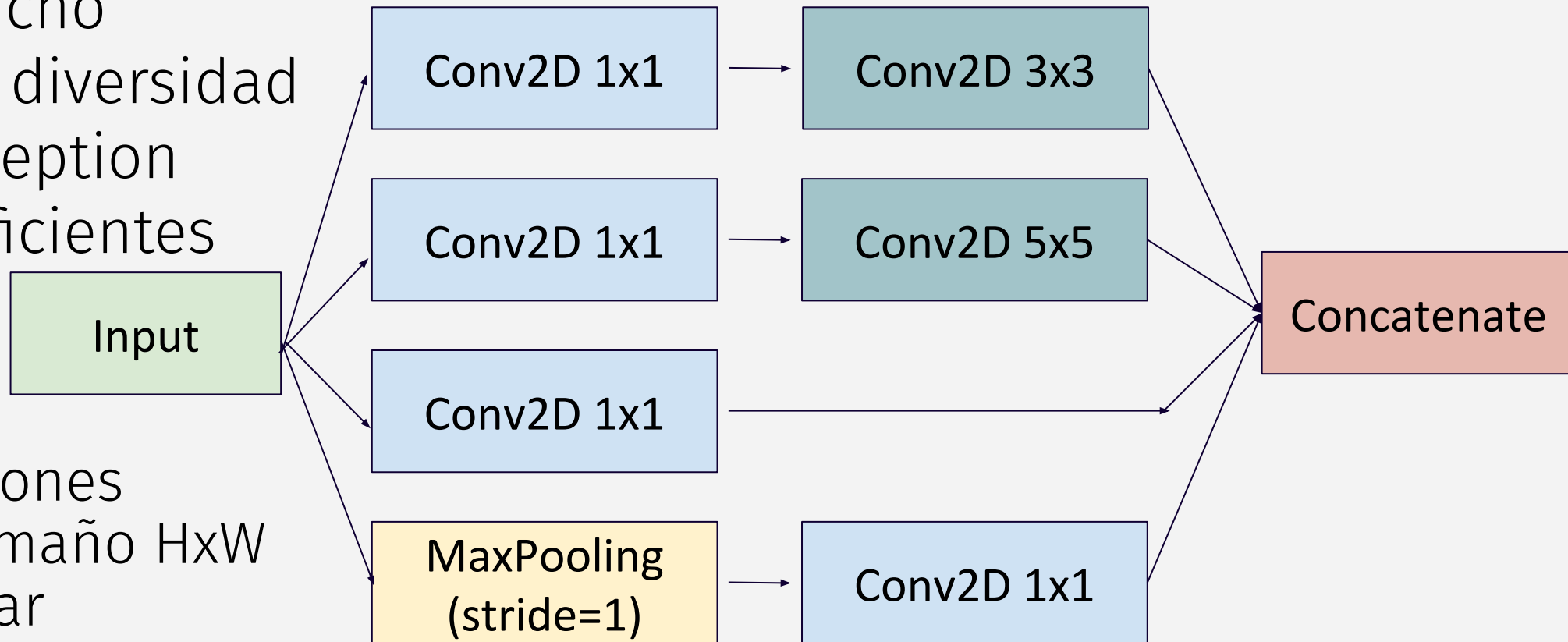
InceptionV1 ([notebook](#), [paper](#))

- Ganador de ILSVRC 2014 (ImageNet)
- También llamado GoogleNet
- Introdujo bloques Inception

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Bloques Inception

- Hasta ahora
 - ¿Mayor complejidad? => Mayor profundidad
- Otra forma
 - Mayor ancho
 - Mayor diversidad
- Bloques Inception
 - Conv + eficientes
- Método
 - Distintas convoluciones
 - Mismo tamaño HxW
 - Concatenar dimensión C



Bloques Inception

```
def inception(x, F3x3_1x1, F3x3,  
              F5x5_1x1, F5x5, F1x1, Fmp_1x1):
```

```
    c3x3_1x1 = Conv2D(F3x3_1x1, (1, 1))(x)  
    c3x3      = Conv2D(F3x3, (3, 3))(c3x3_1x1)
```

```
    c5x5_1x1 = Conv2D(F5x5_1x1, (1, 1))(x)  
    c5x5      = Conv2D(F5x5, (5, 5))(c5x5_1x1)
```

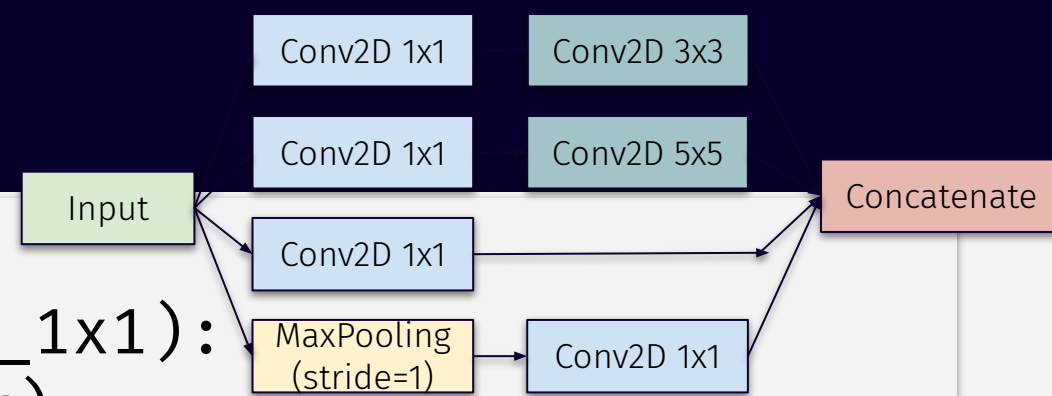
```
    c1x1      = Conv2D(F1x1, (1, 1))(x)
```

```
    mp         = MaxPooling2D((3, 3), strides=(1, 1))(x)
```

```
    mp_1x1     = Conv2D(Fmp_1x1, (1, 1))(mp)
```

```
    result     = Concatenate(axis=3)([c1x1, c3x3, c5x5, mp_1x1])
```

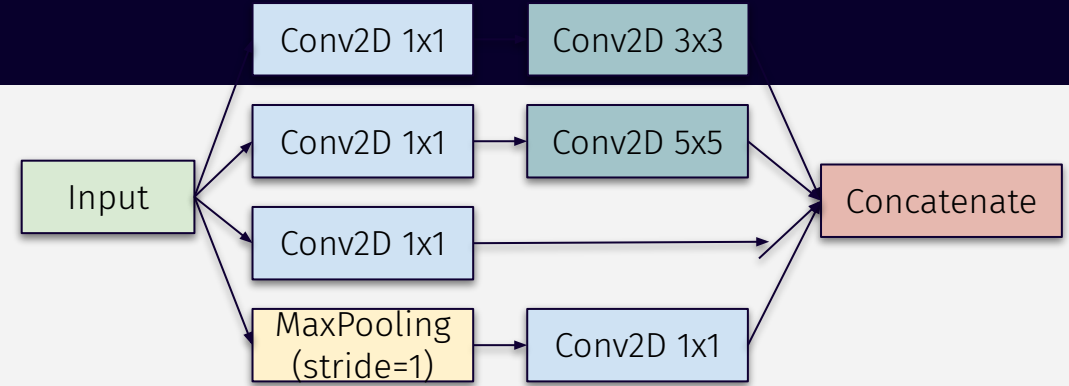
```
    return result
```



- Falta: padding="same"
 - No cambiar HxW
- Falta: activation="relu"

Bloques Inception

```
x = Input(shape=(32,32,64))
layer = inception(x,5,10,20,30,40,50)
model = Model(inputs=[x],outputs=[layer])
print(model.summary())
```



Layer (type)	Output Shape	Param #	Connected to
input (InputLayer)	(None, 32, 32, 64)	0	
block1_3x3_1x1 (Conv2D)	(None, 32, 32, 5)	325	input[0][0]
block1_3x3 (Conv2D)	(None, 32, 32, 10)	460	block1_3x3_1x1[0][0]
block1_5x5_1x1 (Conv2D)	(None, 32, 32, 20)	1300	input[0][0]
block1_5x5 (Conv2D)	(None, 32, 32, 30)	15030	block1_5x5_1x1[0][0]
block1_1x1 (Conv2D)	(None, 32, 32, 40)	2600	input[0][0]
block1_mp (MaxPooling2D)	(None, 32, 32, 64)	0	input[0][0]
block1_mp_1x1 (Conv2D)	(None, 32, 32, 50)	3250	block1_mp[0][0]
block1_concat (Concatenate)	(None, 32, 32, 160)	0	block1_1x1[0][0] block1_3x3[0][0] block1_5x5[0][0] block1_mp_1x1[0][0]

Total params: **22,965**

- 160 feature maps
- 23 000 parámetros

Comparación con Conv 3x3 o 5x5 ([comparación](#))

- Comparemos con una convolucional 3x3 o 5x5 común

```
x = Input(shape=(32,32,64))  
layer = Conv2D(160, (3,3)) (x)  
model = Model(inputs=[x], outputs=[layer])  
print(model.summary())
```

- 160 feature maps
- 5x5: 256.000 param
- 3x3: 92.000 param
- Inception: 23000 param

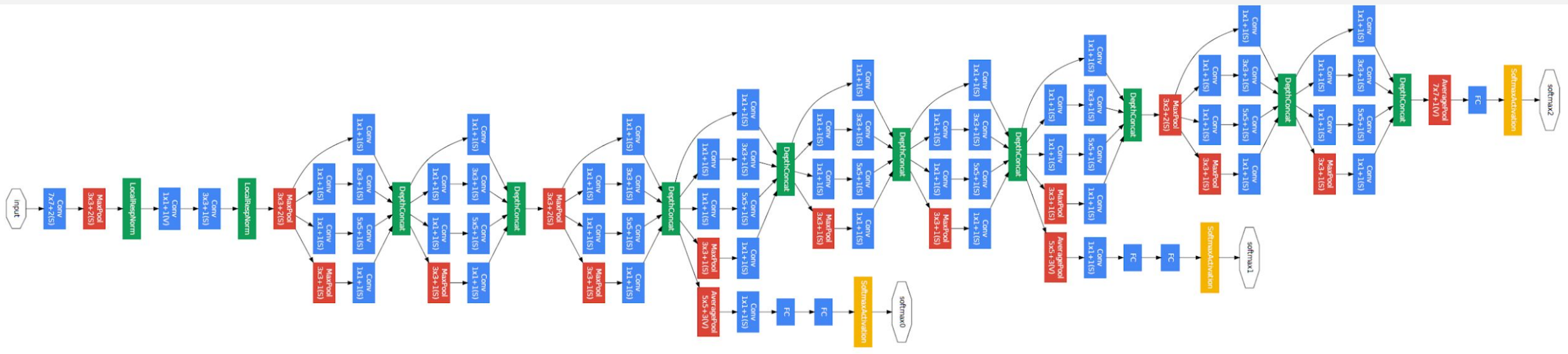
```
x = Input(shape=(32,32,64))  
layer = Conv2D(160, (5,5)) (x)
```

Layer (type)	Output Shape	Param #
input (InputLayer)	(None, 32, 32, 64)	0
conv (Conv2D)	(None, 28, 28, 160)	92320
160		
Total params:		92320

Layer (type)	Output Shape	Param #
input (InputLayer)	(None, 32, 32, 64)	0
conv (Conv2D)	(None, 28, 28, 160)	256160
Total params:		256,160

InceptionV1 ([notebook](#), [paper](#))

- Diagrama de arquitectura
 - Ignorar las primeras dos salidas amarillas



Inception ([notebook](#), [paper](#))

- Implementación en Keras

- Inception v1

- Muchas partes irregulares

- Funcionan bien pero no tienen justificación

- Veremos una versión simplificada

```
def create_googlenet(weights_path=None):
    # creates GoogLeNet a.k.a. Inception v1 (Szegedy, 2015)
    input = Input(shape=(3, 224, 224))

    input_pad = ZeroPadding2D(padding=(3, 3))(input)
    conv1_7x7_s2 = Conv2D(64, (7, 7), strides=(2, 2), padding='valid', activation='relu', name='conv1_7x7_s2')
    conv1_zero_pad = ZeroPadding2D(padding=(1, 1))(conv1_7x7_s2)
    pool1_helper = PoolHelper()(conv1_zero_pad)
    pool1_3x3_s2 = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='valid', name='pool1_3x3_s2')
    pool1_norm1 = LRN(name='pool1/norm1')(pool1_3x3_s2)

    conv2_3x3_reduce = Conv2D(64, (1, 1), padding='same', activation='relu', name='conv2_3x3_reduce')
    conv2_3x3 = Conv2D(192, (3, 3), padding='same', activation='relu', name='conv2_3x3', kernel_regularizer=keras.regularizers.l2(0.0002))
    conv2_norm2 = LRN(name='conv2/norm2')(conv2_3x3)
    conv2_zero_pad = ZeroPadding2D(padding=(1, 1))(conv2_norm2)
    pool2_helper = PoolHelper()(conv2_zero_pad)
    pool2_3x3_s2 = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='valid', name='pool2_3x3_s2')

    inception_3a_1x1 = Conv2D(64, (1, 1), padding='same', activation='relu', name='inception_3a_1x1')
    inception_3a_3x3_reduce = Conv2D(96, (1, 1), padding='same', activation='relu', name='inception_3a_3x3_reduce')
    inception_3a_3x3_pad = ZeroPadding2D(padding=(1, 1))(inception_3a_3x3_reduce)
    inception_3a_3x3 = Conv2D(128, (3, 3), padding='valid', activation='relu', name='inception_3a_3x3')
```

```
pool5_7x7_s1 = AveragePooling2D(pool_size=(7, 7), strides=(1, 1), name='pool5_7x7_s1')(inception_3a_output)
loss3_flat = Flatten()(pool5_7x7_s1)
pool5_drop_7x7_s1 = Dropout(rate=0.4)(loss3_flat)
loss3_classifier = Dense(1000, name='loss3/classifier', kernel_regularizer=keras.regularizers.l2(0.0002))
loss3_classifier_act = Activation('softmax', name='prob')(loss3_classifier)
```

```
googlenet = Model(inputs=input, outputs=[loss1_classifier_act, loss2_classifier_act, loss3_classifier_act])
```

```
inception_3b_pool = MaxPooling2D(pool_size=(3, 3), strides=(1, 1), padding='same', name='inception_3b_pool')
inception_3b_pool_proj = Conv2D(64, (1, 1), padding='same', activation='relu', name='inception_3b_pool_proj')
inception_3b_output = Concatenate(axis=1, name='inception_3b/output')([inception_3b_1x1, inception_3b_pool_proj])

inception_3b_output_zero_pad = ZeroPadding2D(padding=(1, 1))(inception_3b_output)
pool3_helper = PoolHelper()(inception_3b_output_zero_pad)
pool3_3x3_s2 = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='valid', name='pool3_3x3_s2')
```

```
inception_4a_1x1 = Conv2D(192, (1, 1), padding='same', activation='relu', name='inception_4a_1x1')
inception_4a_3x3_reduce = Conv2D(96, (1, 1), padding='same', activation='relu', name='inception_4a_3x3_reduce')
inception_4a_3x3_pad = ZeroPadding2D(padding=(1, 1))(inception_4a_3x3_reduce)
inception_4a_3x3 = Conv2D(208, (3, 3), padding='valid', activation='relu', name='inception_4a_3x3')
inception_4a_5x5_reduce = Conv2D(16, (1, 1), padding='same', activation='relu', name='inception_4a_5x5_reduce')
inception_4a_5x5_pad = ZeroPadding2D(padding=(2, 2))(inception_4a_5x5_reduce)
inception_4a_5x5 = Conv2D(48, (5, 5), padding='valid', activation='relu', name='inception_4a_5x5')
inception_4a_pool = MaxPooling2D(pool_size=(3, 3), strides=(1, 1), padding='same', name='inception_4a_pool')
inception_4a_pool_proj = Conv2D(64, (1, 1), padding='same', activation='relu', name='inception_4a_pool_proj')
inception_4a_output = Concatenate(axis=1, name='inception_4a/output')([inception_4a_1x1,
```

```
inception_4b_1x1 = Conv2D(160, (1, 1), padding='same', activation='relu', name='inception_4b_1x1')
inception_4b_3x3_reduce = Conv2D(112, (1, 1), padding='same', activation='relu', name='inception_4b_3x3_reduce')
inception_4b_3x3_pad = ZeroPadding2D(padding=(1, 1))(inception_4b_3x3_reduce)
inception_4b_3x3 = Conv2D(224, (3, 3), padding='valid', activation='relu', name='inception_4b_3x3')
inception_4b_5x5_reduce = Conv2D(24, (1, 1), padding='same', activation='relu', name='inception_4b_5x5_reduce')
inception_4b_5x5_pad = ZeroPadding2D(padding=(2, 2))(inception_4b_5x5_reduce)
inception_4b_5x5 = Conv2D(64, (5, 5), padding='valid', activation='relu', name='inception_4b_5x5')
inception_4b_pool = MaxPooling2D(pool_size=(3, 3), strides=(1, 1), padding='same', name='inception_4b_pool')
inception_4b_pool_proj = Conv2D(64, (1, 1), padding='same', activation='relu', name='inception_4b_pool_proj')
inception_4b_output = Concatenate(axis=1, name='inception_4b/output')([inception_4b_1x1,
```

```
inception_4c_1x1 = Conv2D(128, (1, 1), padding='same', activation='relu', name='inception_4c_1x1')
inception_4c_3x3_reduce = Conv2D(128, (1, 1), padding='same', activation='relu', name='inception_4c_3x3_reduce')
inception_4c_3x3_pad = ZeroPadding2D(padding=(1, 1))(inception_4c_3x3_reduce)
inception_4c_3x3 = Conv2D(256, (3, 3), padding='valid', activation='relu', name='inception_4c_3x3')
inception_4c_5x5_reduce = Conv2D(24, (1, 1), padding='same', activation='relu', name='inception_4c_5x5_reduce')
inception_4c_5x5_pad = ZeroPadding2D(padding=(2, 2))(inception_4c_5x5_reduce)
inception_4c_5x5 = Conv2D(64, (5, 5), padding='valid', activation='relu', name='inception_4c_5x5')
inception_4c_pool = MaxPooling2D(pool_size=(3, 3), strides=(1, 1), padding='same', name='inception_4c_pool')
inception_4c_pool_proj = Conv2D(64, (1, 1), padding='same', activation='relu', name='inception_4c_pool_proj')
inception_4c_output = Concatenate(axis=1, name='inception_4c/output')([inception_4c_1x1,
```

```
inception_4d_1x1 = Conv2D(112, (1, 1), padding='same', activation='relu', name='inception_4d_1x1')
inception_4d_3x3_reduce = Conv2D(144, (1, 1), padding='same', activation='relu', name='inception_4d_3x3_reduce')
inception_4d_3x3_pad = ZeroPadding2D(padding=(1, 1))(inception_4d_3x3_reduce)
inception_4d_3x3 = Conv2D(288, (3, 3), padding='valid', activation='relu', name='inception_4d_3x3')
inception_4d_5x5_reduce = Conv2D(32, (1, 1), padding='same', activation='relu', name='inception_4d_5x5_reduce')
inception_4d_5x5_pad = ZeroPadding2D(padding=(2, 2))(inception_4d_5x5_reduce)
inception_4d_5x5 = Conv2D(64, (5, 5), padding='valid', activation='relu', name='inception_4d_5x5')
inception_4d_pool = MaxPooling2D(pool_size=(3, 3), strides=(1, 1), padding='same', name='inception_4d_pool')
inception_4d_pool_proj = Conv2D(64, (1, 1), padding='same', activation='relu', name='inception_4d_pool_proj')
inception_4d_output = Concatenate(axis=1, name='inception_4d/output')([inception_4d_1x1,
```

```
inception_4e_output_zero_pad = ZeroPadding2D(padding=(1, 1))(inception_4e_output)
pool4_helper = PoolHelper()(inception_4e_output_zero_pad)
pool4_3x3_s2 = MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='valid', name='pool4_3x3_s2')
```

```
inception_5a_1x1 = Conv2D(256, (1, 1), padding='same', activation='relu', name='inception_5a_1x1')
inception_5a_3x3_reduce = Conv2D(160, (1, 1), padding='same', activation='relu', name='inception_5a_3x3_reduce')
inception_5a_3x3_pad = ZeroPadding2D(padding=(1, 1))(inception_5a_3x3_reduce)
inception_5a_3x3 = Conv2D(320, (3, 3), padding='valid', activation='relu', name='inception_5a_3x3')
inception_5a_5x5_reduce = Conv2D(32, (1, 1), padding='same', activation='relu', name='inception_5a_5x5_reduce')
inception_5a_5x5_pad = ZeroPadding2D(padding=(2, 2))(inception_5a_5x5_reduce)
inception_5a_5x5 = Conv2D(128, (5, 5), padding='valid', activation='relu', name='inception_5a_5x5')
inception_5a_pool = MaxPooling2D(pool_size=(3, 3), strides=(1, 1), padding='same', name='inception_5a_pool')
inception_5a_pool_proj = Conv2D(128, (1, 1), padding='same', activation='relu', name='inception_5a_pool_proj')
inception_5a_output = Concatenate(axis=1, name='inception_5a/output')([inception_5a_1x1,
```

```
inception_5b_1x1 = Conv2D(384, (1, 1), padding='same', activation='relu', name='inception_5b_1x1')
inception_5b_3x3_reduce = Conv2D(192, (1, 1), padding='same', activation='relu', name='inception_5b_3x3_reduce')
inception_5b_3x3_pad = ZeroPadding2D(padding=(1, 1))(inception_5b_3x3_reduce)
inception_5b_3x3 = Conv2D(384, (3, 3), padding='valid', activation='relu', name='inception_5b_3x3')
inception_5b_5x5_reduce = Conv2D(48, (1, 1), padding='same', activation='relu', name='inception_5b_5x5_reduce')
inception_5b_5x5_pad = ZeroPadding2D(padding=(2, 2))(inception_5b_5x5_reduce)
inception_5b_5x5 = Conv2D(128, (5, 5), padding='valid', activation='relu', name='inception_5b_5x5')
inception_5b_pool = MaxPooling2D(pool_size=(3, 3), strides=(1, 1), padding='same', name='inception_5b_pool')
inception_5b_pool_proj = Conv2D(128, (1, 1), padding='same', activation='relu', name='inception_5b_pool_proj')
```

Implementación simplificada ([notebook](#), [paper](#))

```
def InceptionV1(input_shape, classes):
    input = Input(shape=input_shape, name="input")
    x = Conv2D(64, (7, 7), strides=(2, 2), padding="same", activation="relu", name="c1")(input)
    x = MaxPooling2D((3, 3), strides=(2, 2), padding="same", name=f"mp1")(x)
    x = Conv2D(64, (1, 1), strides=(2, 2), padding="same", activation="relu", name=f"c2_1x1")(x)
    x = Conv2D(192, (3, 3), strides=(2, 2), padding="same", activation="relu", name=f"c2_3x3")(x)
    x = MaxPooling2D((3, 3), strides=(2, 2), padding="same", name=f"mp2")(x)
    x = bottleneck(x, 96, 128, 16, 32, 64, 32, "inception3a")
    x = bottleneck(x, 128, 128, 32, 96, 128, 64, "inception3b")
    x = MaxPooling2D((3, 3), strides=(2, 2), padding="same", name=f"mp3")(x)
    x = bottleneck(x, 96, 208, 16, 48, 192, 32, "inception4a")
    x = bottleneck(x, 112, 224, 24, 64, 160, 64, "inception4b")
    x = bottleneck(x, 128, 256, 24, 64, 128, 64, "inception4c")
    x = bottleneck(x, 144, 288, 32, 64, 112, 64, "inception4d")
    x = bottleneck(x, 160, 320, 32, 128, 256, 128, "inception4e")
    x = MaxPooling2D((3, 3), strides=(2, 2), padding="same", name=f"mp4")(x)
    x = bottleneck(x, 160, 320, 32, 128, 256, 128, "inception5a")
    x = bottleneck(x, 192, 384, 48, 128, 384, 128, "inception5b")
    x = GlobalAveragePooling2D()(x)
    x = Dense(classes, activation="softmax")(x)
```