Aprendizaje Automático Profundo (Deep Learning)

Dr. Facundo Quiroga - Dr. Franco Ronchetti

Arquitectura Inception V1

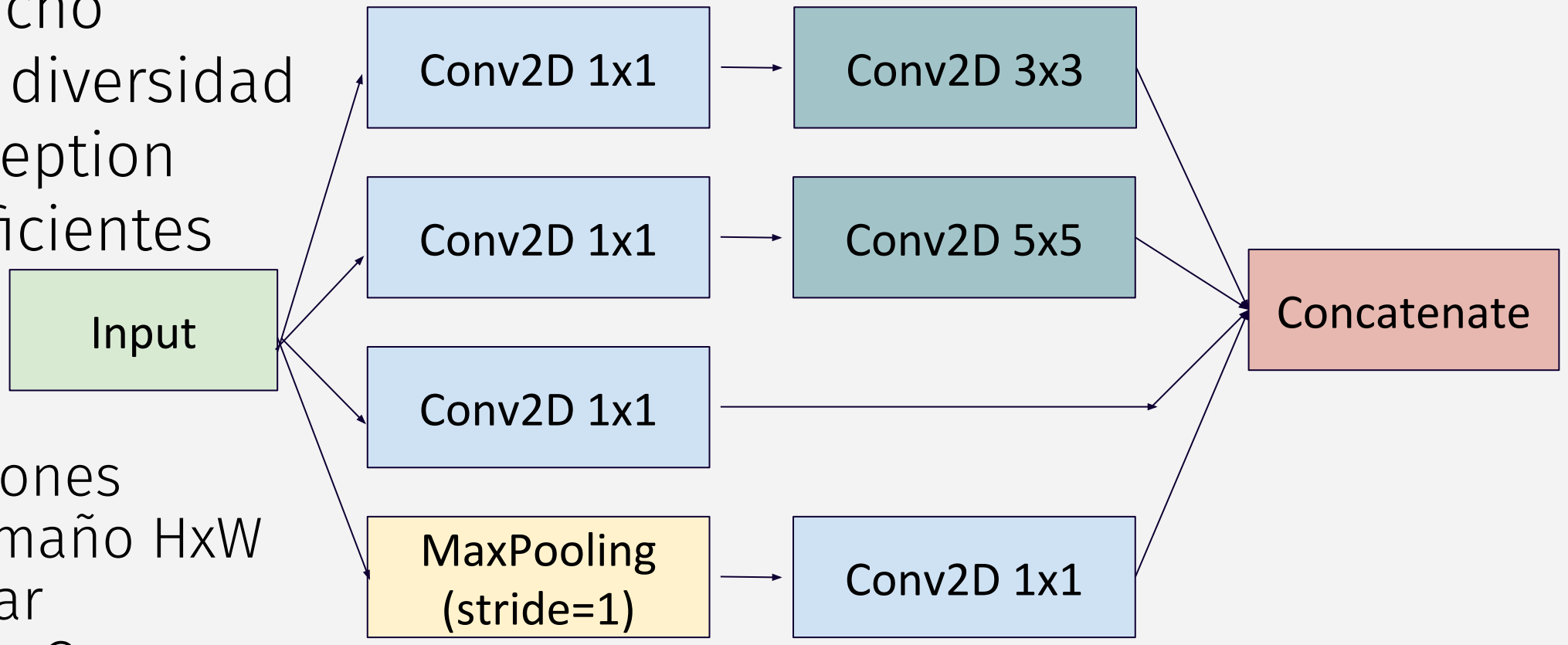
InceptionV1 (notebook, paper)

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

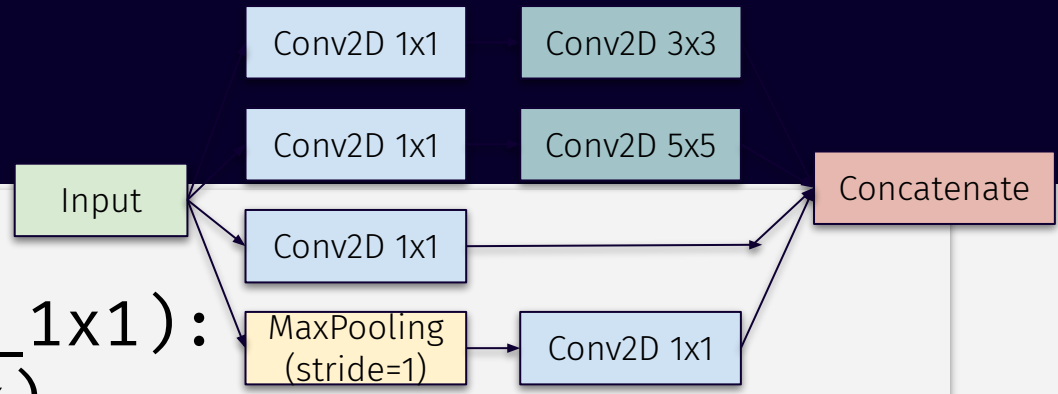
type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	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

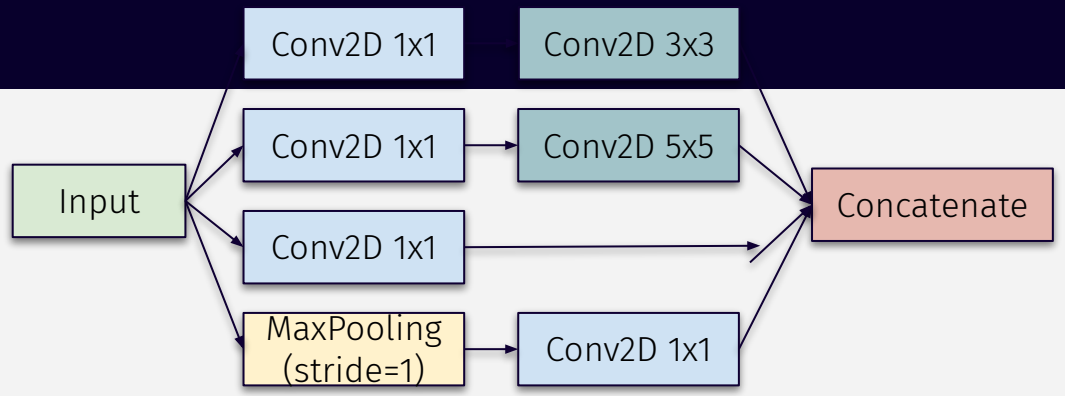



```
def inception(x,F3x3_1x1,F3x3,
                F5x5_1x1,F5x5,F1x1,Fmp_1x1):
  c3x3_1x1 = Conv2D(F3x3_1x1,(1,1))(x)
  c3x3     = Conv2D(F3x3,(3,3))(c3x3_1x1)

  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())
```

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

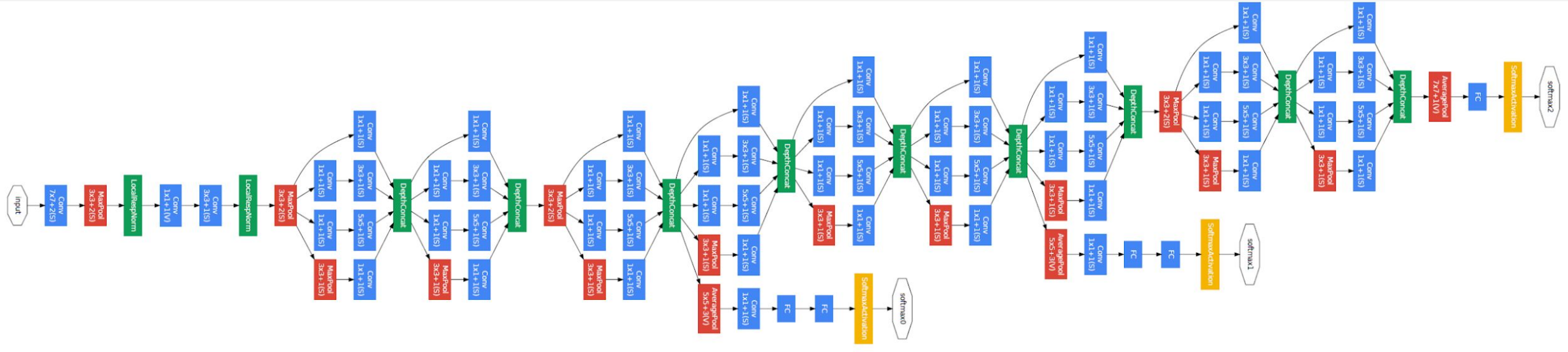
- 160 feature maps
- 5x5: 256.000 param
- 3x3: 92.000 param
- Inception: 23000 param

```
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)       256160
Total params: 256,160
```

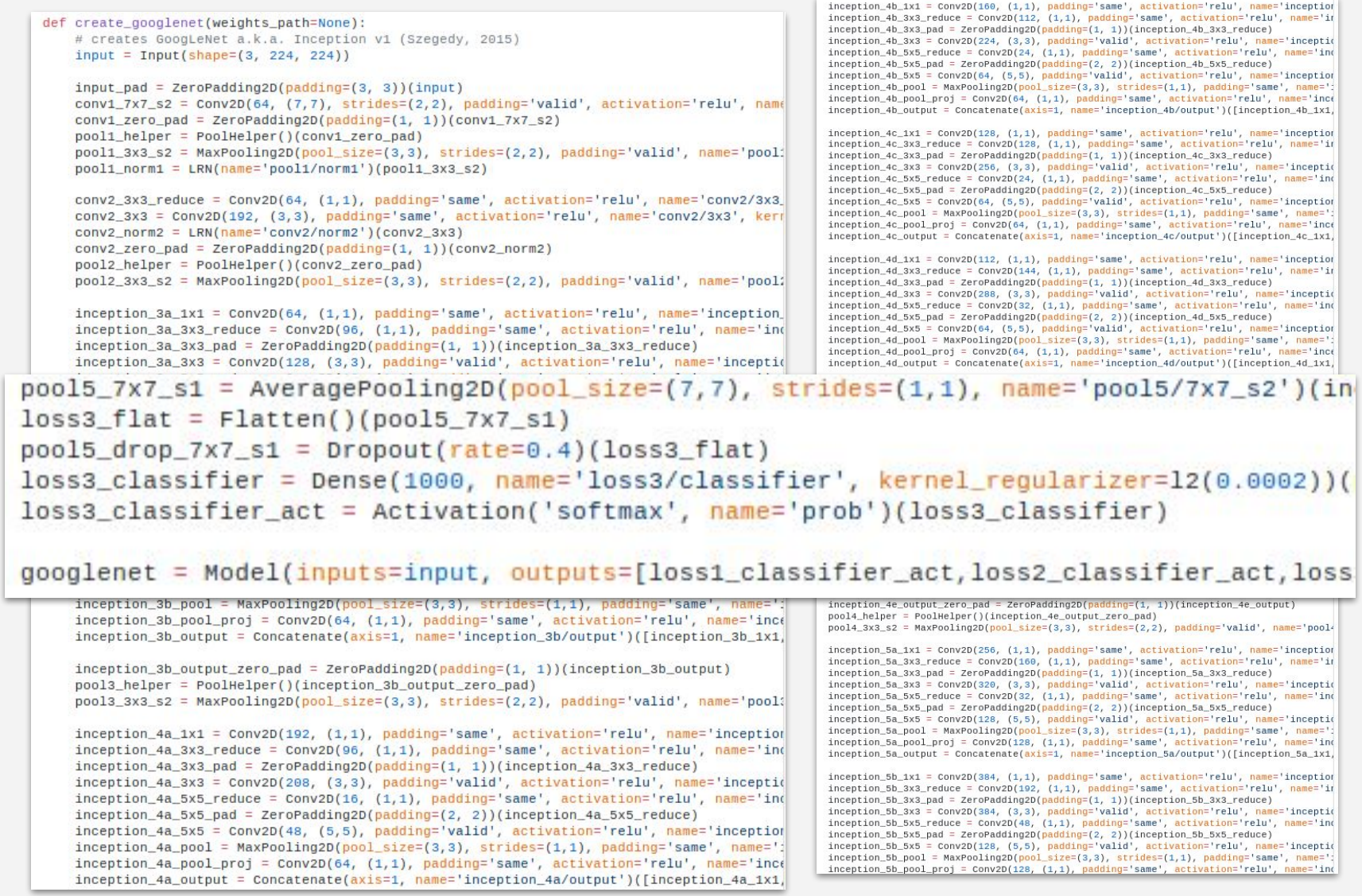
InceptionV1 (notebook, paper)

- 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



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)
```