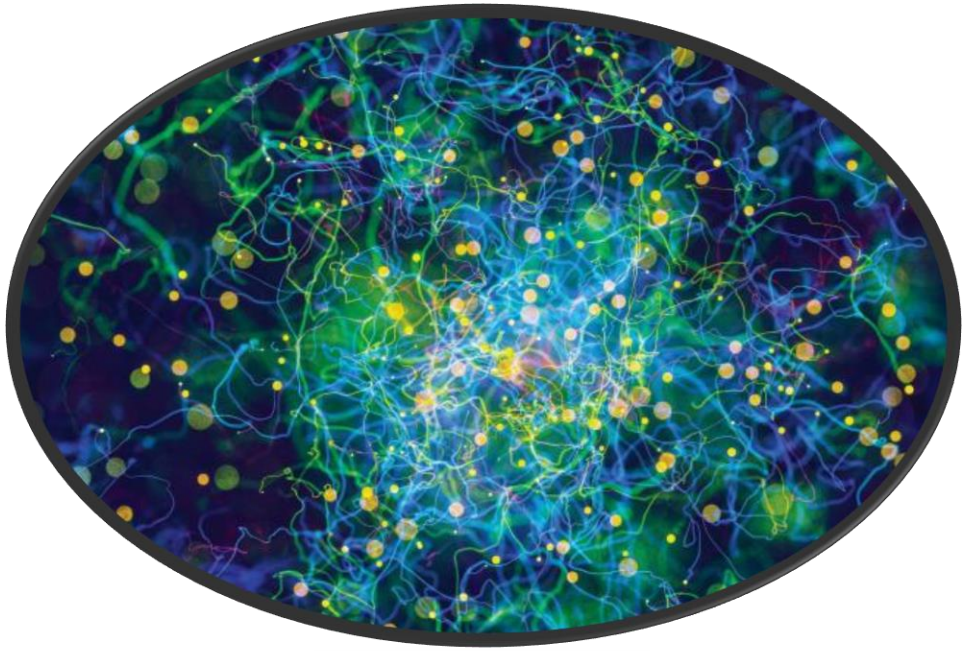
Deep Learning

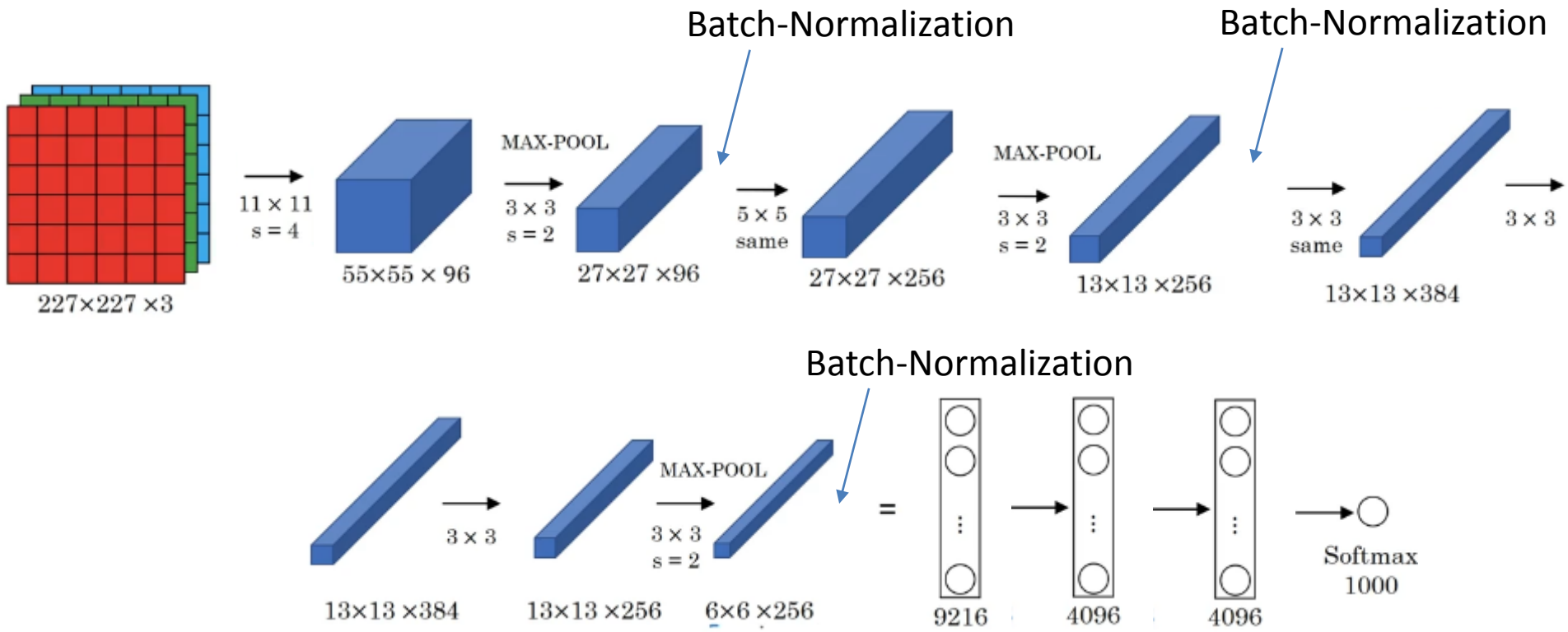


Día 3

EXPOSITOR: Ing. Giorgio Morales Luna

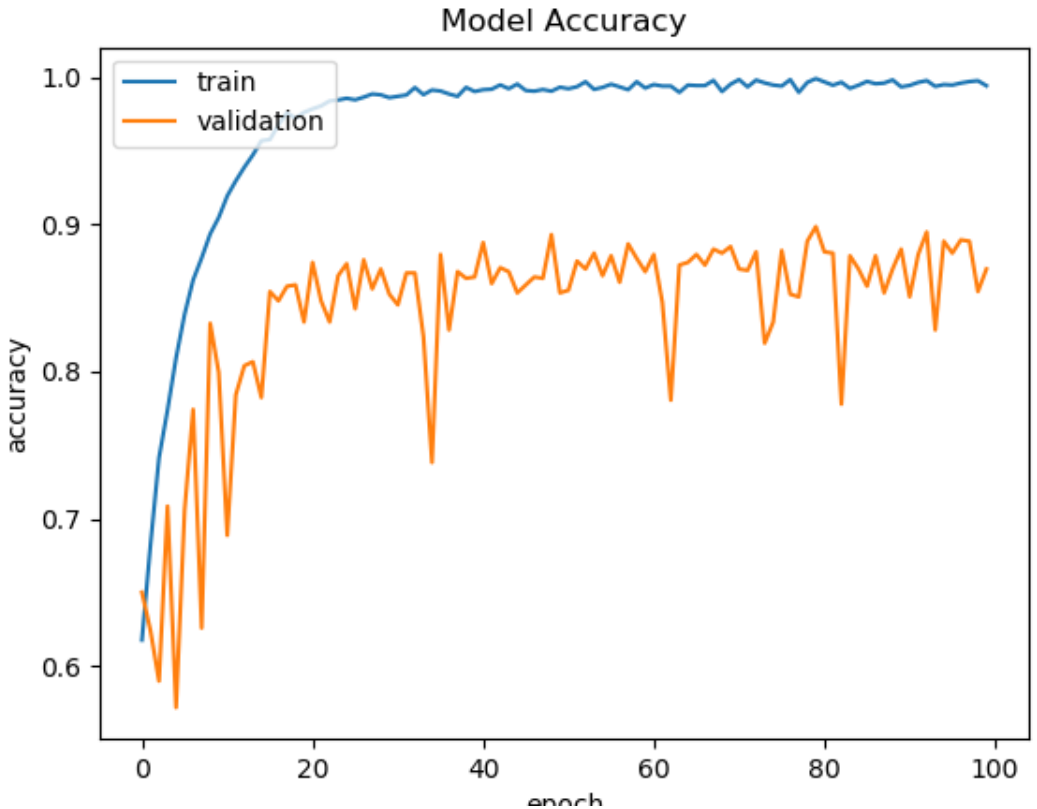
Junio 2018

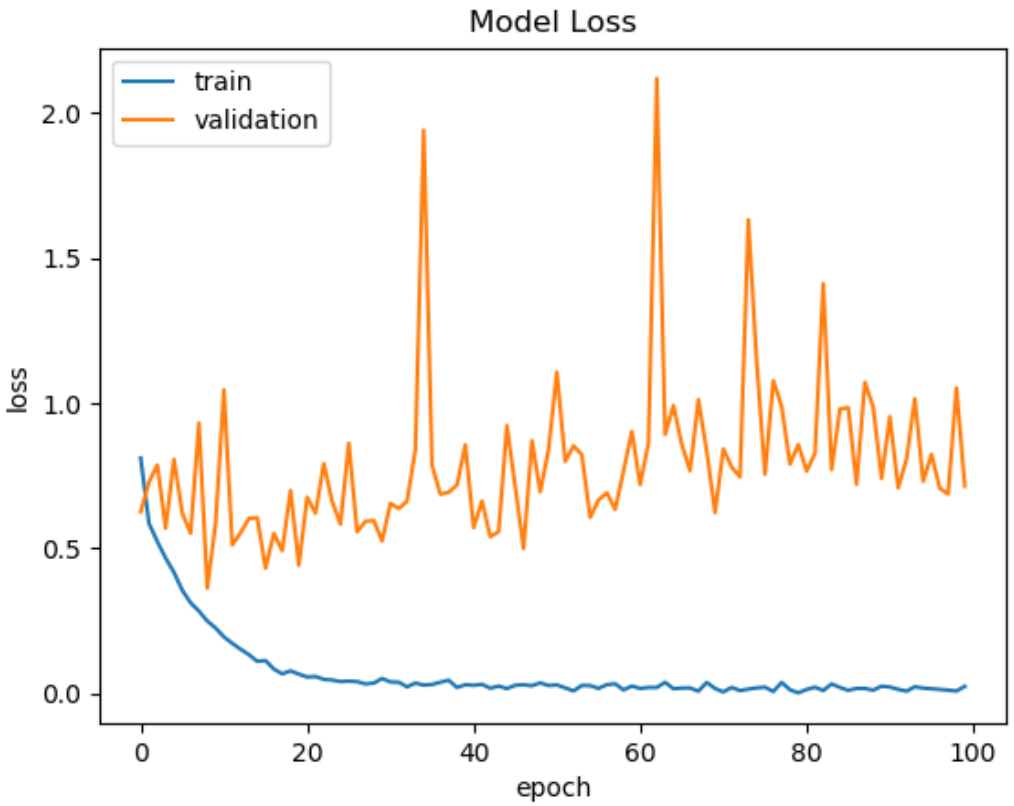
3.2.8. AlexNet from scratch



3.2.8. AlexNet from scratch

Resultado





3.2.9. VGG16 with Keras

keras.applications.vgg16.VGG16

Carga el modelo VGG16 para Keras:

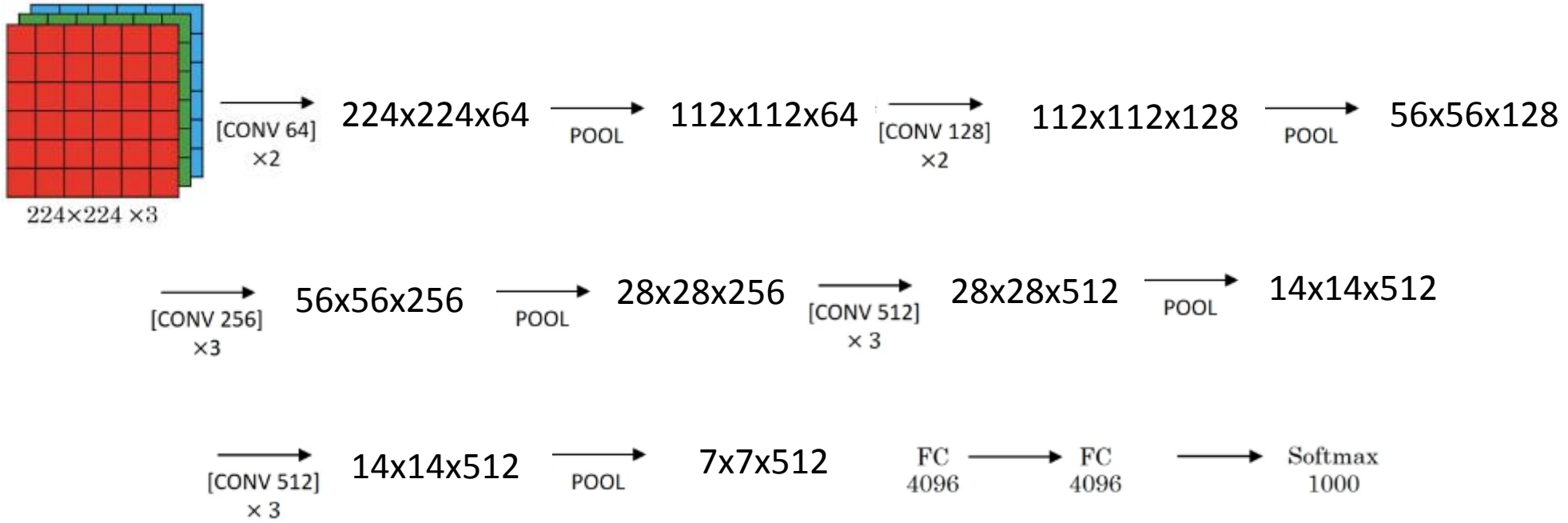
- Include_top: Si es True, incluye las 3 capas fully-connected al final de la red.
- Weights: Si es 'None', inicializa los pesos con valores aleatorios. Si es 'imagenet', carga los pesos de una red VGG16 entrenada en Imagenet.
- input_shape: (224,224,3) por defecto

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22,910,480	126
VGG16	528 MB	0.713	0.901	138,357,544	23
VGG19	549 MB	0.713	0.900	143,667,240	26
ResNet50	98 MB	0.749	0.921	25,636,712	-
ResNet101	171 MB	0.764	0.928	44,707,176	-
ResNet152	232 MB	0.766	0.931	60,419,944	-
ResNet50V2	98 MB	0.760	0.930	25,613,800	-
ResNet101V2	171 MB	0.772	0.938	44,675,560	-
ResNet152V2	232 MB	0.780	0.942	60,380,648	-
ResNeXt50	96 MB	0.777	0.938	25,097,128	-
ResNeXt101	170 MB	0.787	0.943	44,315,560	-
InceptionV3	92 MB	0.779	0.937	23,851,784	159
InceptionResNetV2	215 MB	0.803	0.953	55,873,736	572
MobileNet	16 MB	0.704	0.895	4,253,864	88
MobileNetV2	14 MB	0.713	0.901	3,538,984	88
DenseNet121	33 MB	0.750	0.923	8,062,504	121
DenseNet169	57 MB	0.762	0.932	14,307,880	169
DenseNet201	80 MB	0.773	0.936	20,242,984	201
NASNetMobile	23 MB	0.744	0.919	5,326,716	-
NASNetLarge	343 MB	0.825	0.960	88,949,818	-

The top-1 and top-5 accuracy refers to the model's performance on the ImageNet validation dataset.

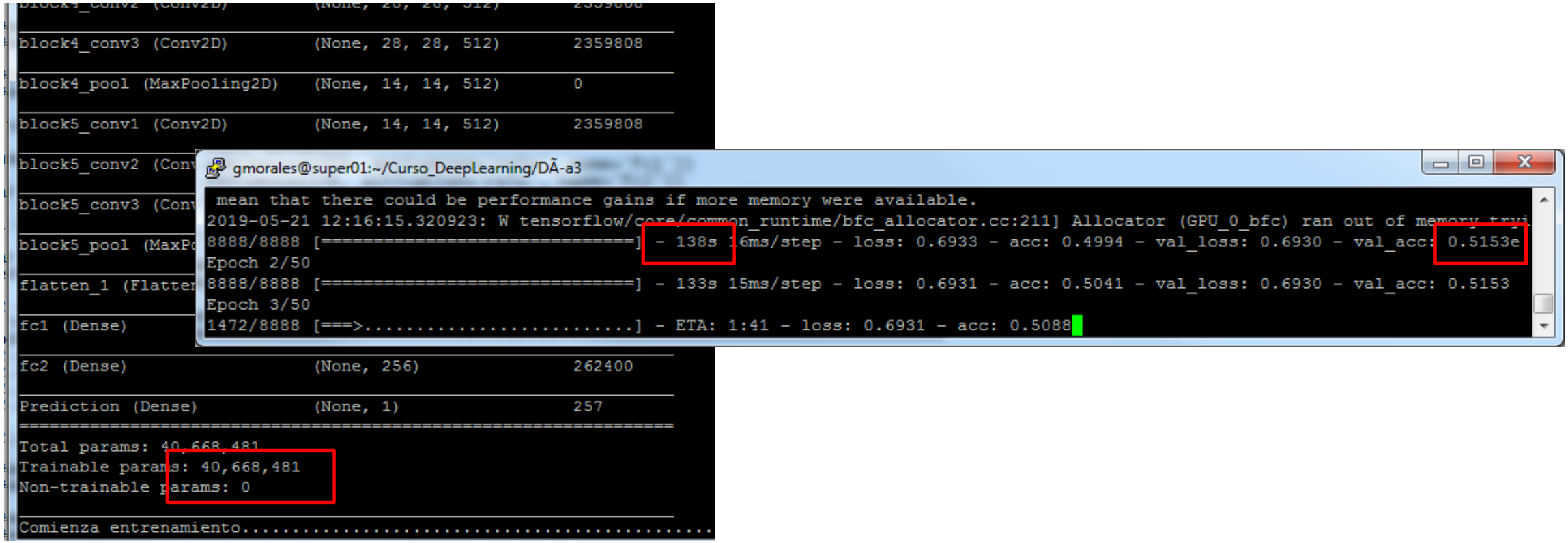
https://keras.io/applications/

3.2.9. VGG16 with Keras



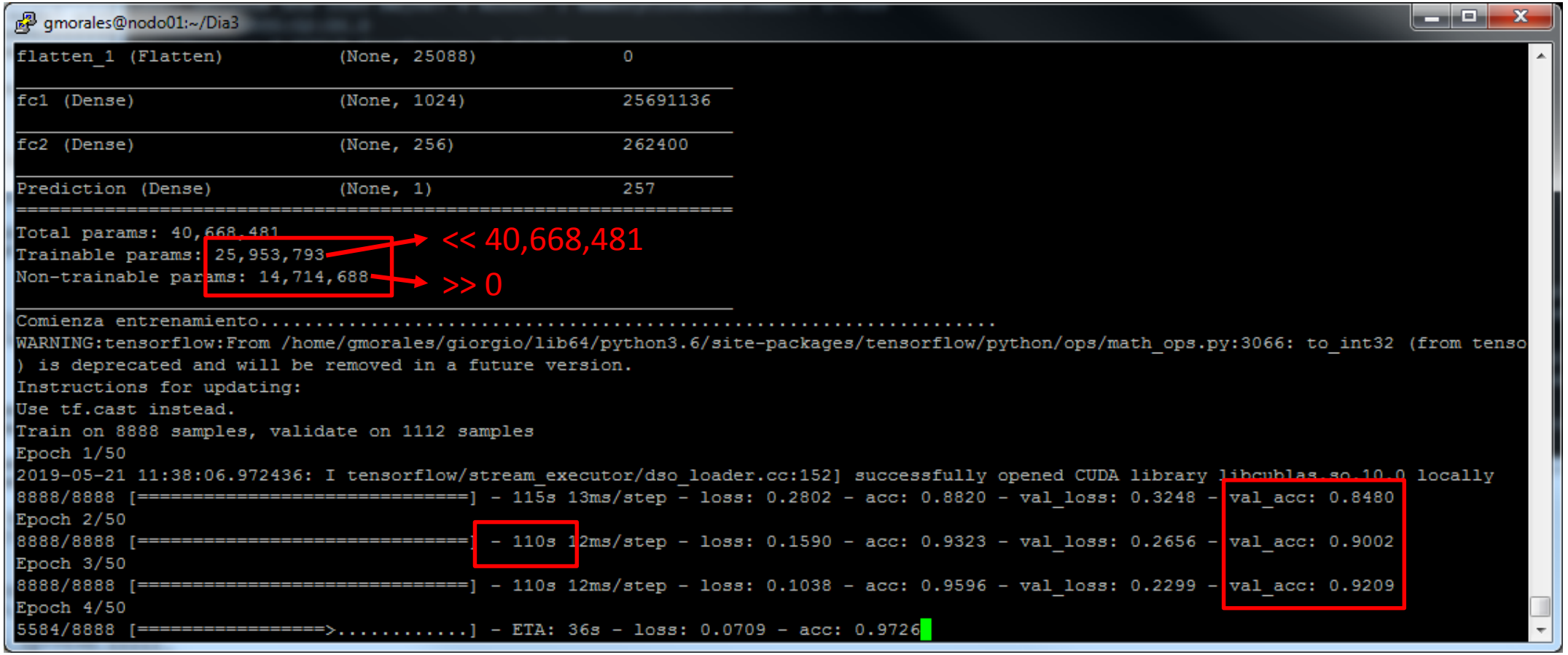
3.2.9. VGG16 with Keras

Entrenamiento VGG16 sin Transfer Learning



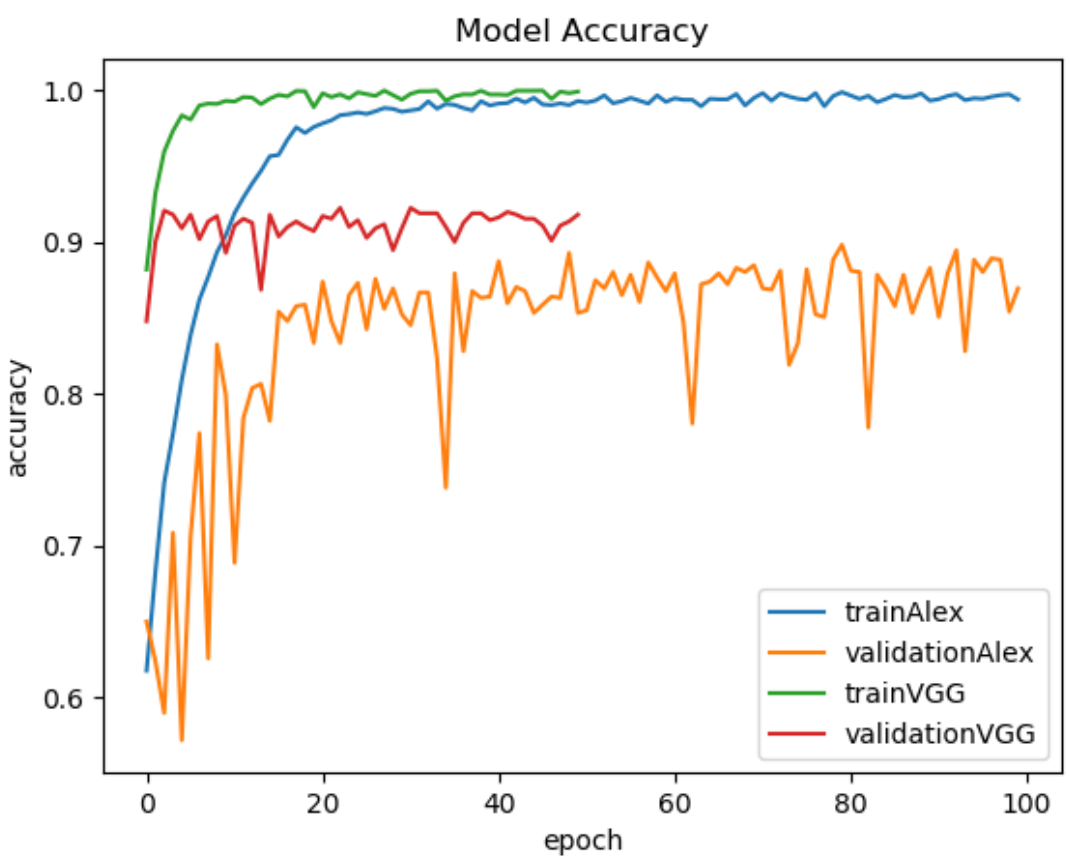
3.2.9. VGG16 with Keras

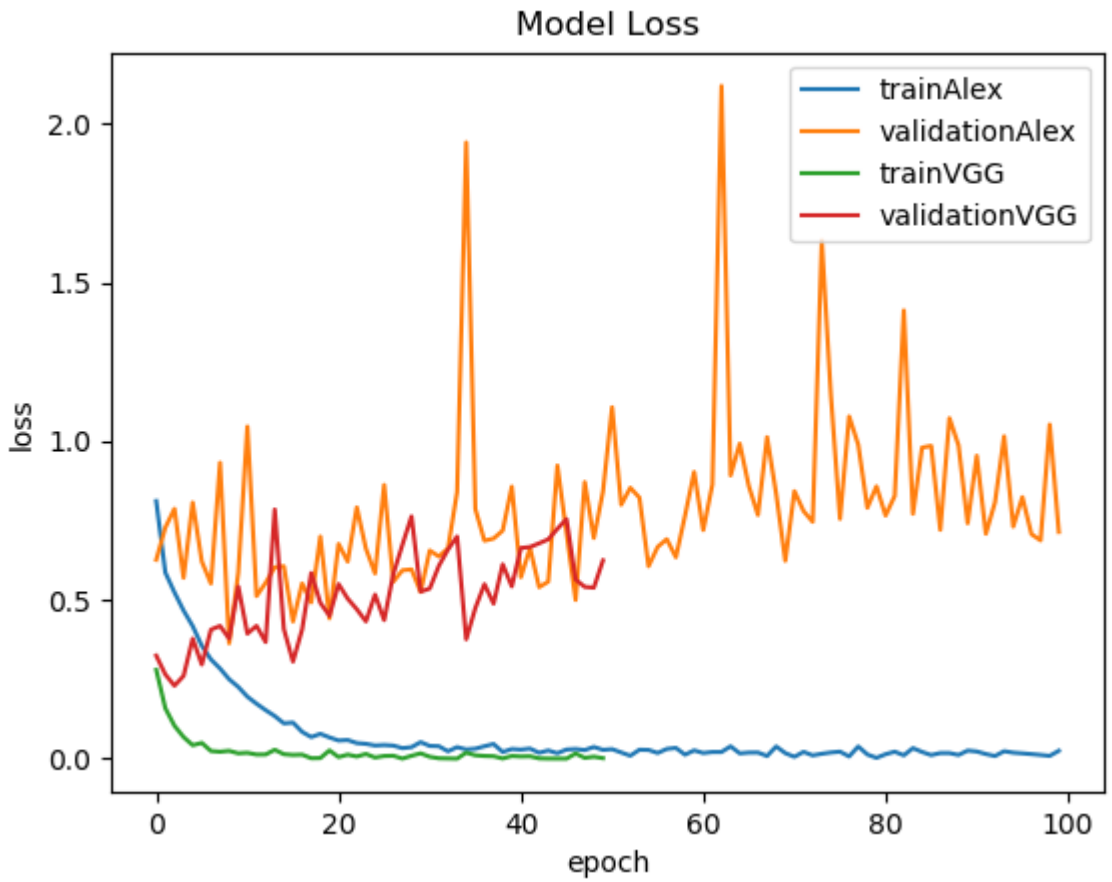
Entrenamiento VGG16 con Transfer Learning



3.2.9. VGG16 with Keras

Resultado





3.2.9. VGG16 with Keras

Resultado

3.2.10. Data Augmentation

3.2.10. Data Augmentation

Mirroring

3.2.10. Data Augmentation

Mirroring

3.2.10. Data Augmentation

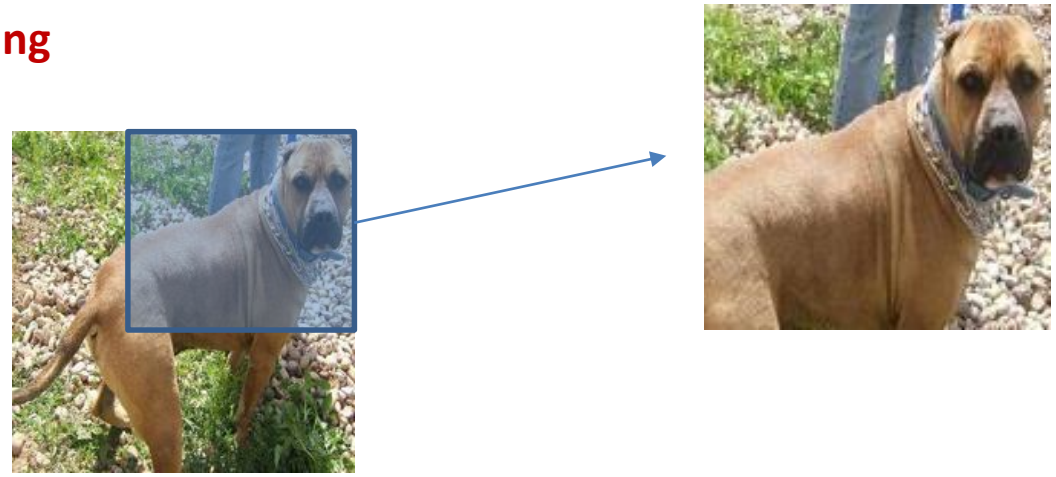
Mirroring

Random Cropping

3.2.10. Data Augmentation

Mirroring

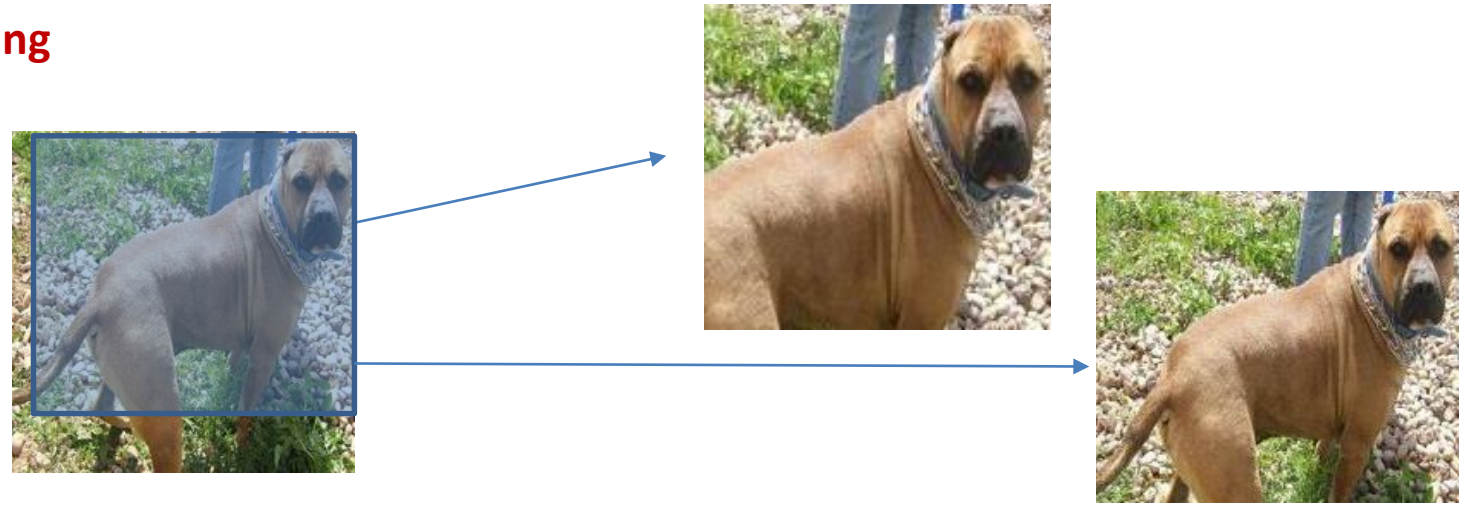
Random Cropping



3.2.10. Data Augmentation

Mirroring

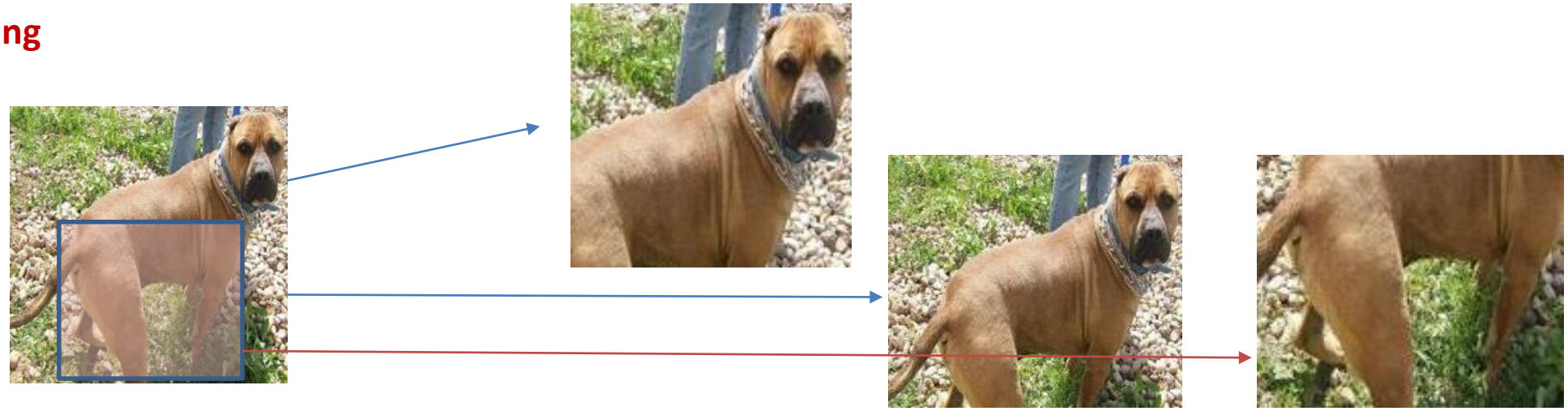
Random Cropping



3.2.10. Data Augmentation

Mirroring

Random Cropping



3.2.10. Data Augmentation

Rotación

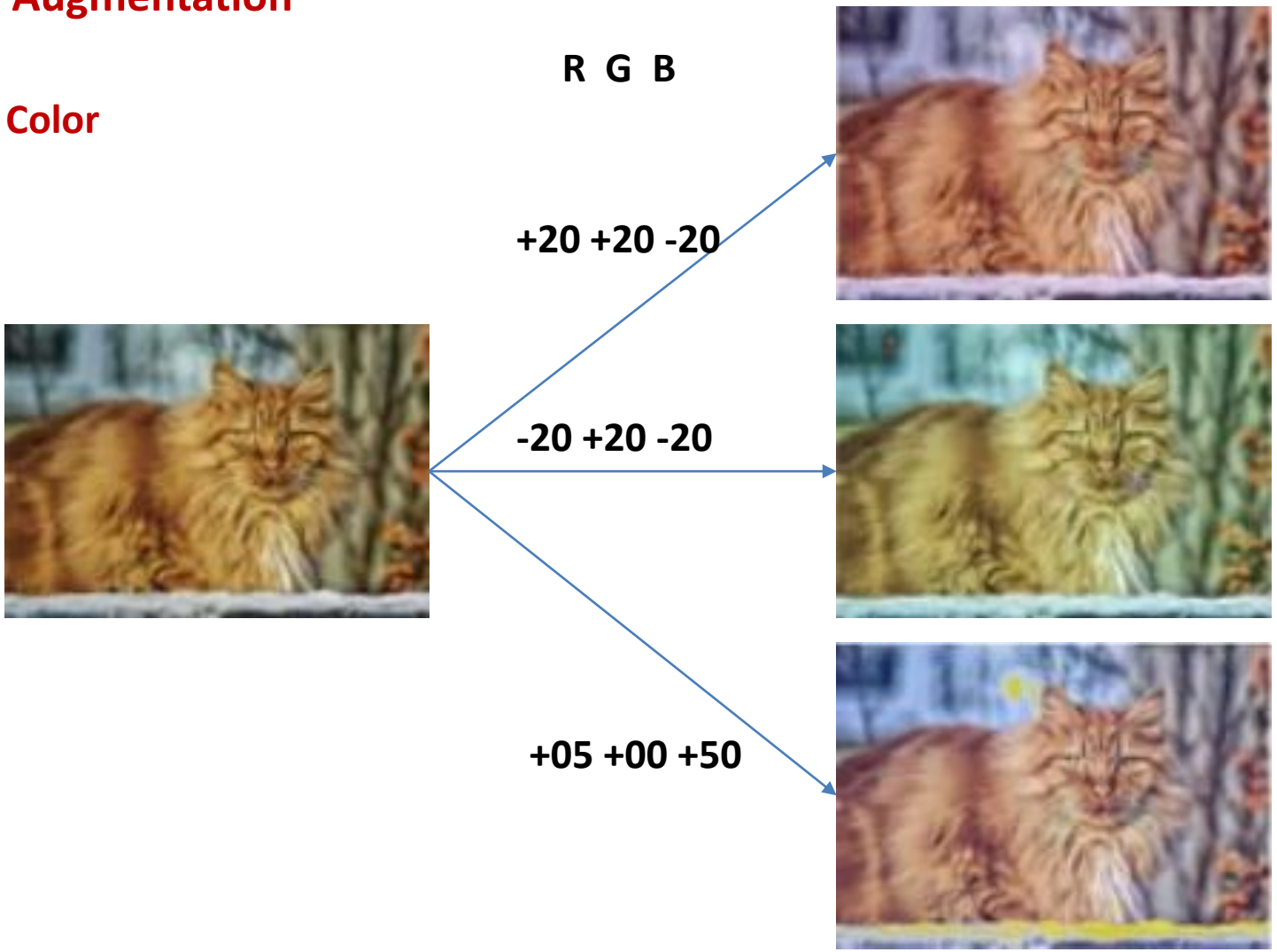
Rotación + Mirroring

3.2.10. Data Augmentation

Rotación + Cropping

3.2.10. Data Augmentation

Distorsión de Color



3.2.10. Data Augmentation

Casos Reales

3.2.10. Data Augmentation

Casos Reales

3.2.10. Data Augmentation

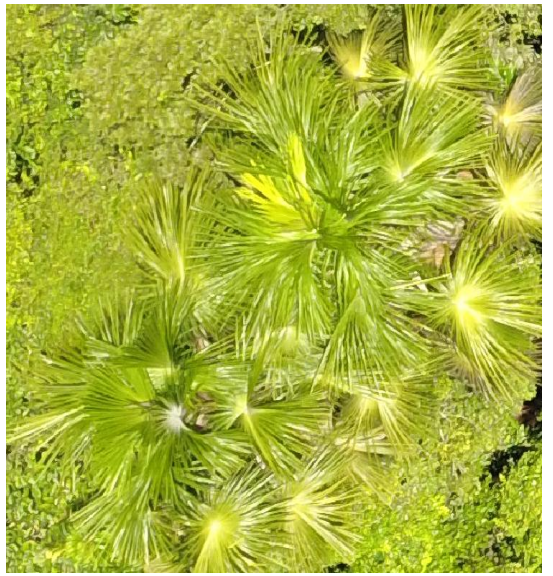
Casos Reales



3.2.10. Data Augmentation

Casos Reales

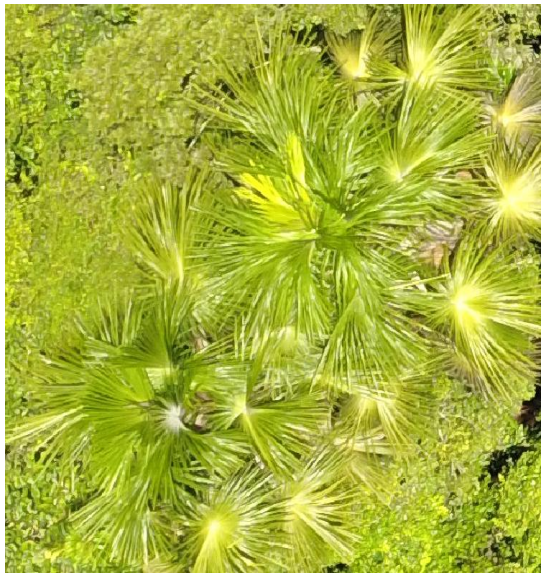


3.2.10. Data Augmentation

Casos Reales




3.2.10. Data Augmentation

¿En qué momento se aplica Data Augmentation?

- En el momento en el que se lee cada batch de entrenamiento.

3.2.10. Data Augmentation

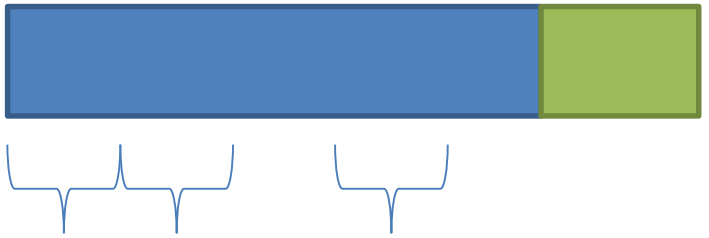
¿En qué momento se aplica Data Augmentation?

- En el momento en el que se lee cada batch de entrenamiento.

3.2.10. Data Augmentation

¿En qué momento se aplica Data Augmentation?

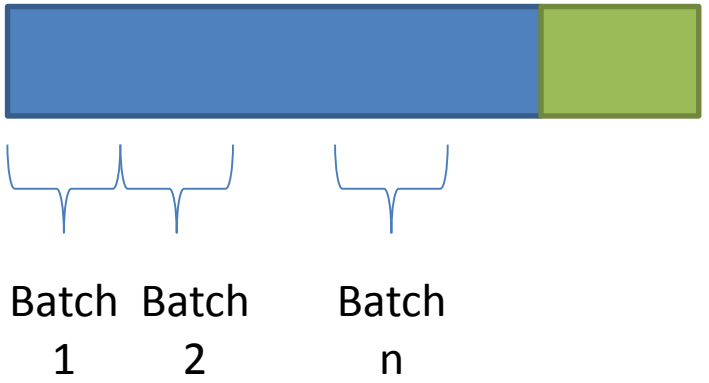
- En el momento en el que se lee cada batch de entrenamiento.



3.2.10. Data Augmentation

¿En qué momento se aplica Data Augmentation?

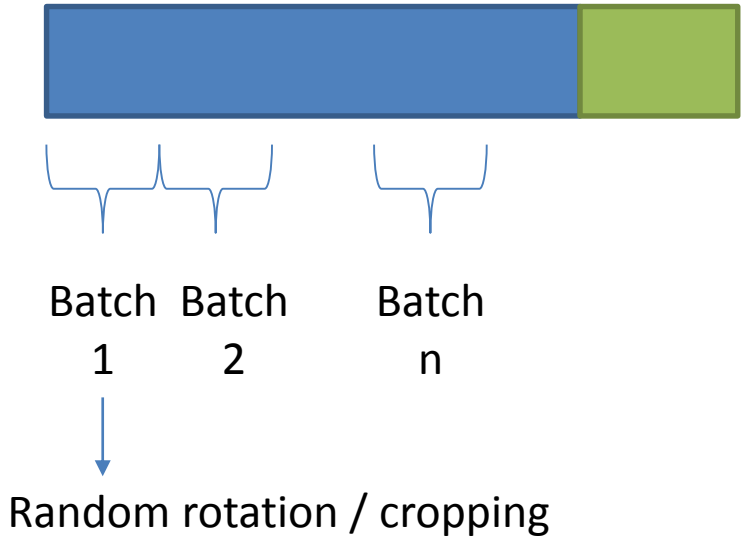
- En el momento en el que se lee cada batch de entrenamiento.



3.2.10. Data Augmentation

¿En qué momento se aplica Data Augmentation?

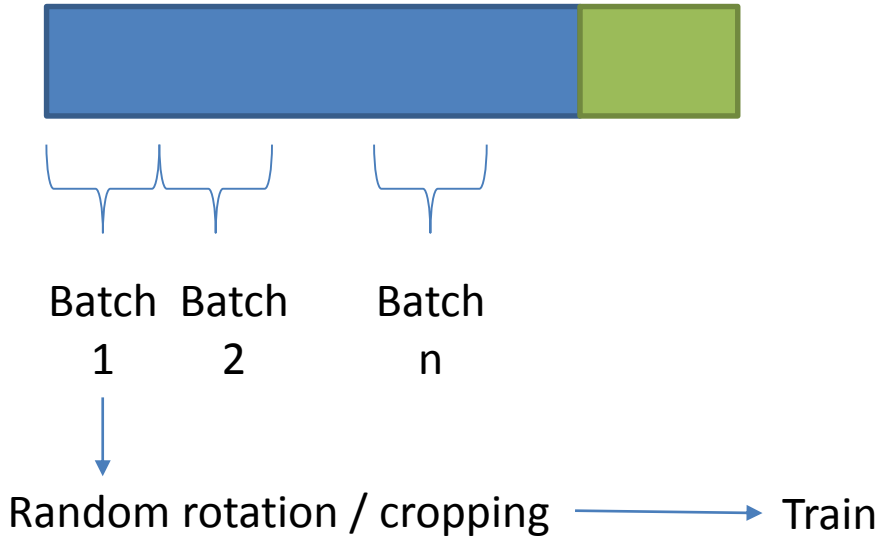
- En el momento en el que se lee cada batch de entrenamiento.



3.2.10. Data Augmentation

¿En qué momento se aplica Data Augmentation?

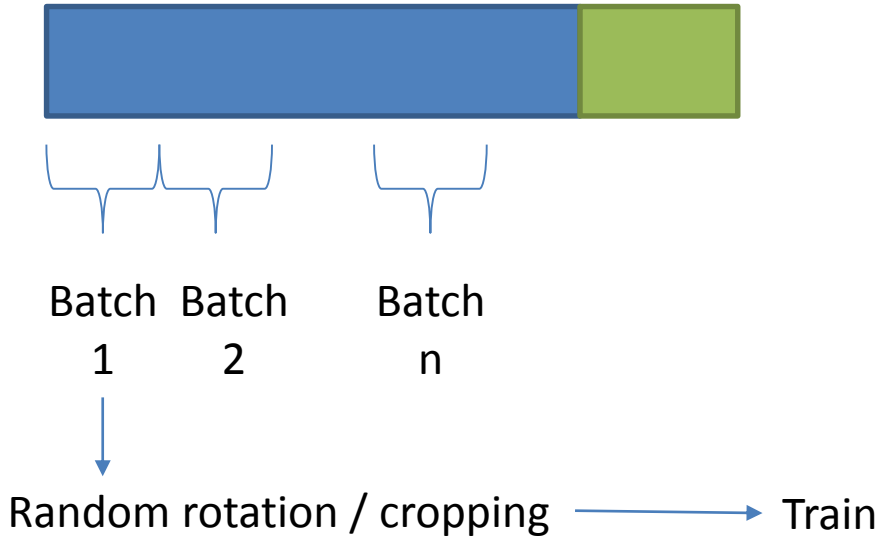
- En el momento en el que se lee cada batch de entrenamiento.



3.2.10. Data Augmentation

¿En qué momento se aplica Data Augmentation?

- En el momento en el que se lee cada batch de entrenamiento.

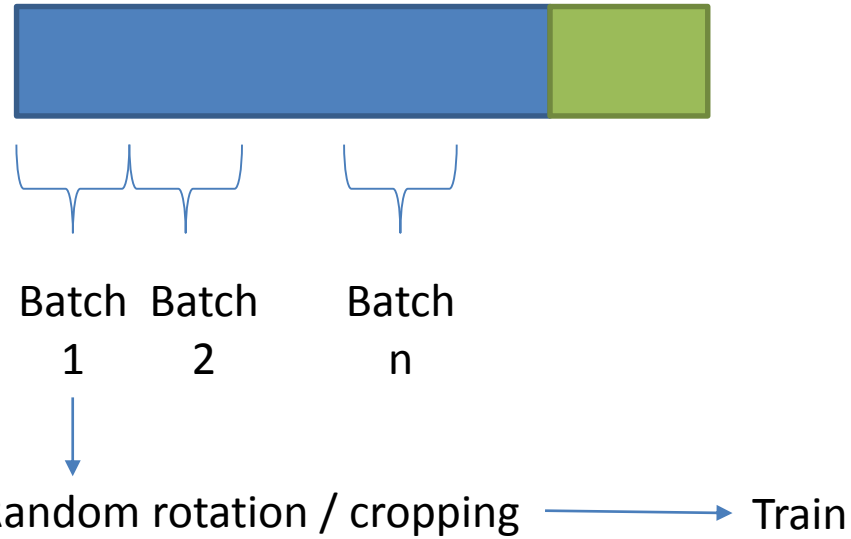


- Antes de crear los sets de entrenamiento, validación y test

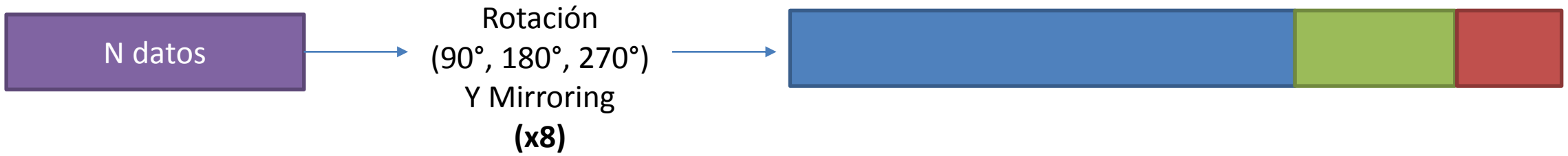
3.2.10. Data Augmentation

¿En qué momento se aplica Data Augmentation?

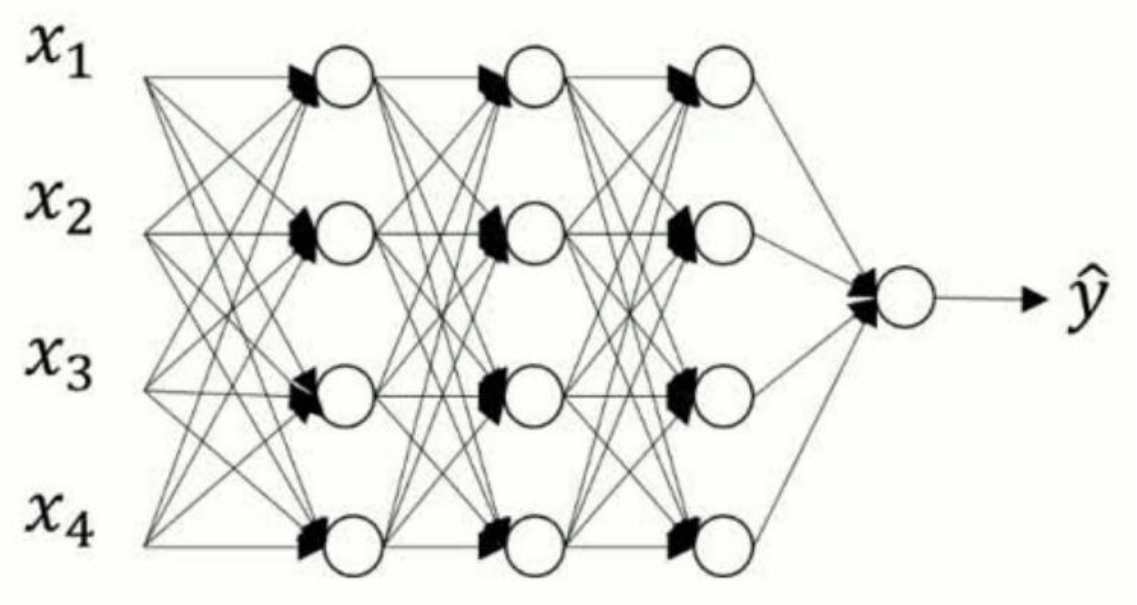
- En el momento en el que se lee cada batch de entrenamiento.

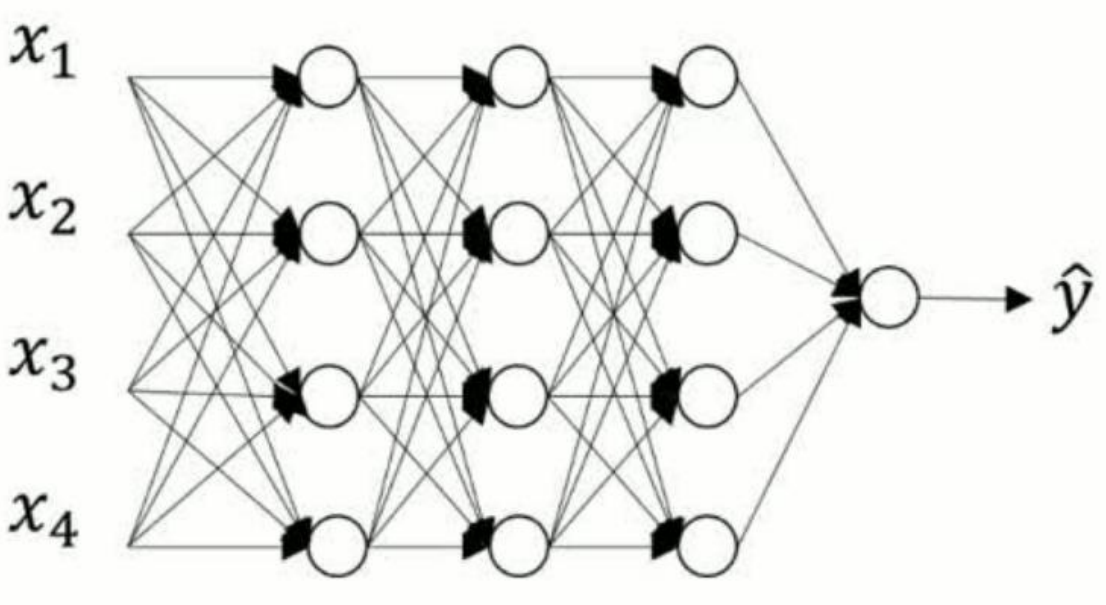


- Antes de crear los sets de entrenamiento, validación y test

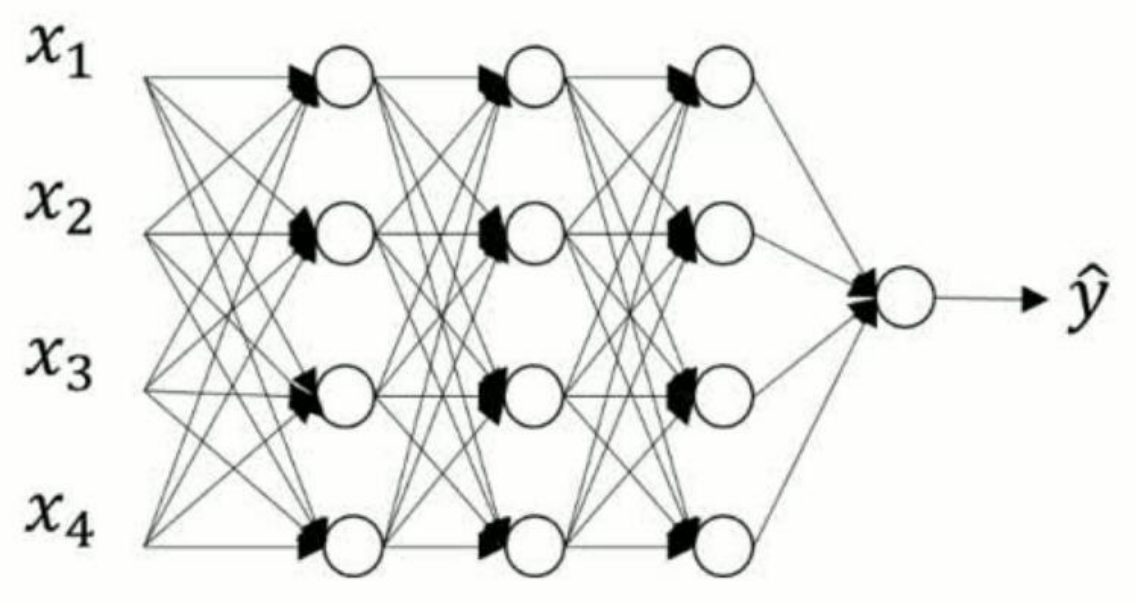


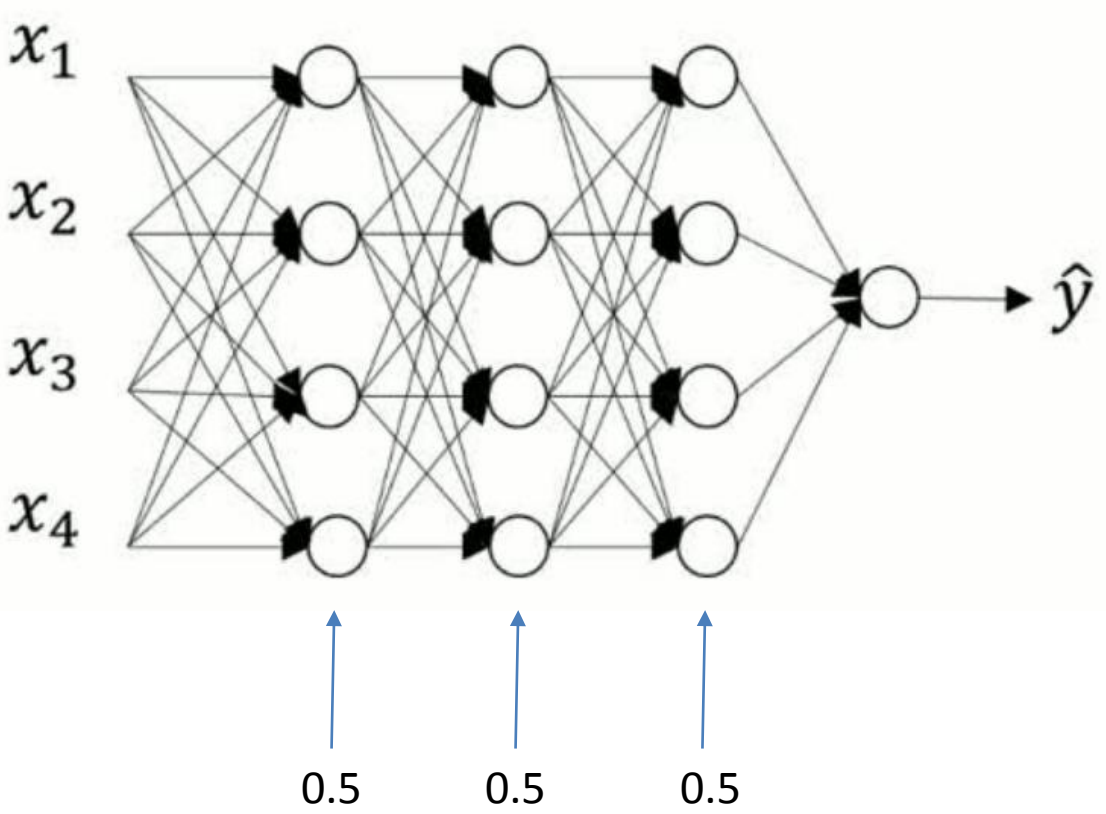
3.2.11. Dropout



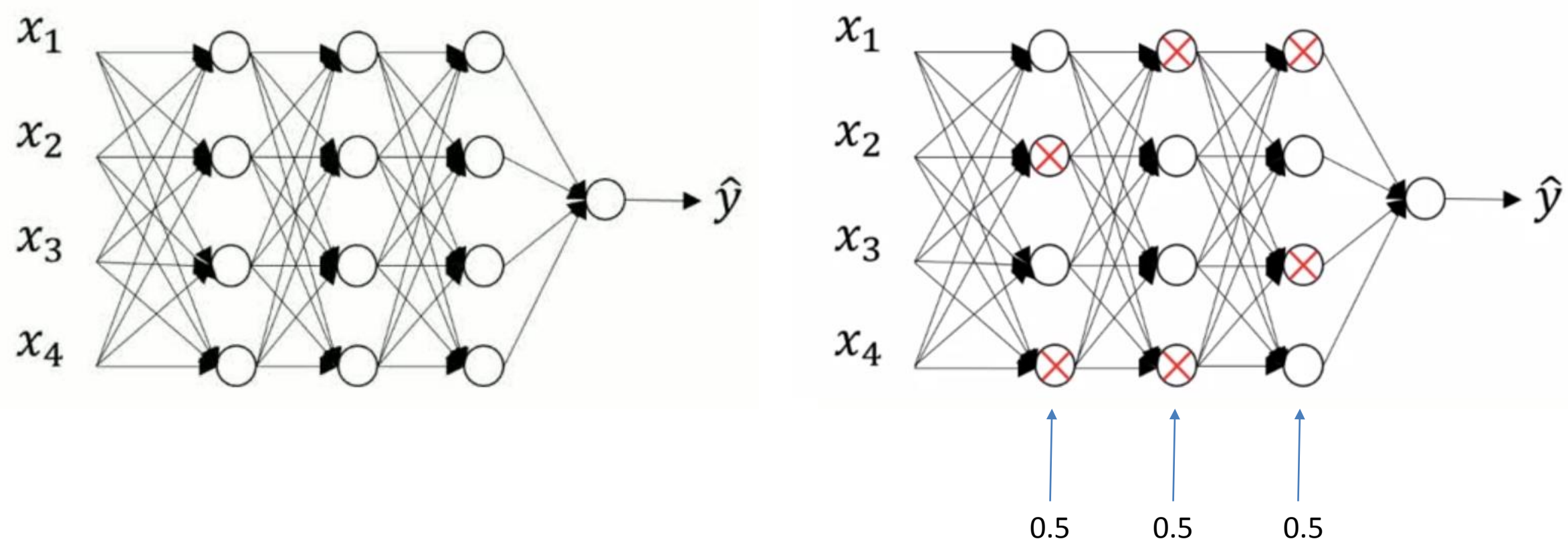


3.2.11. Dropout

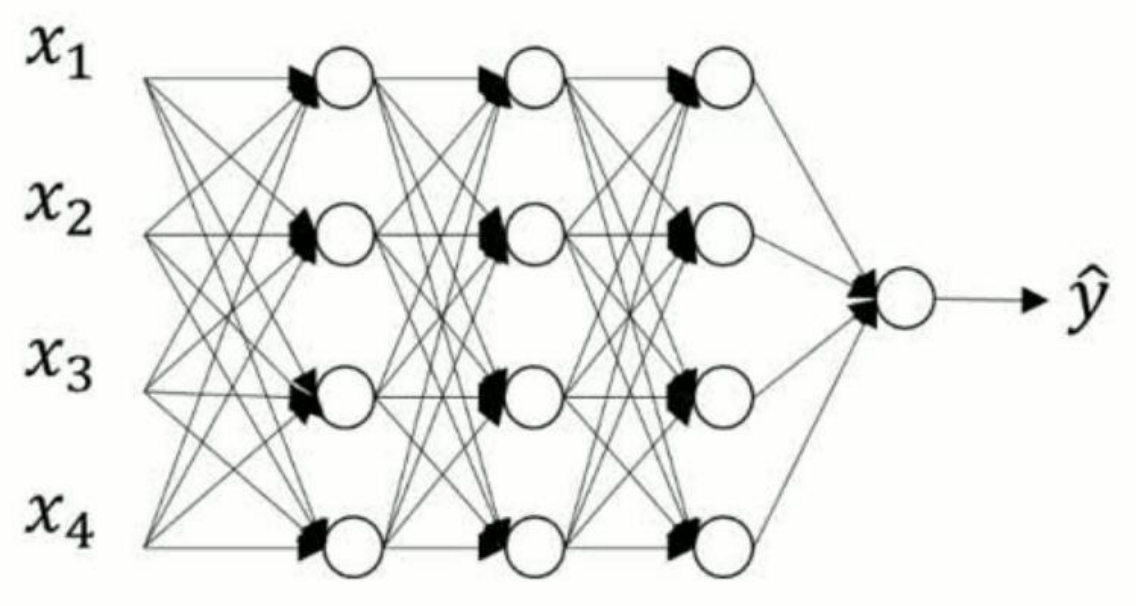


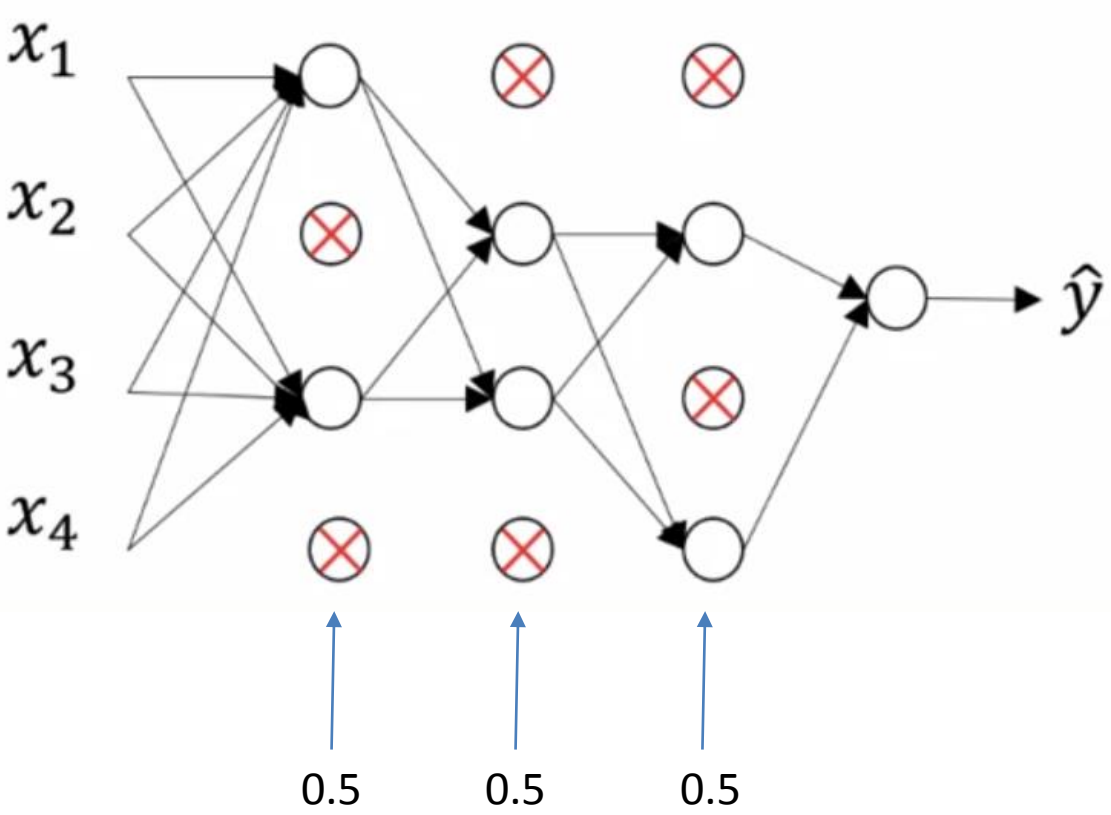


3.2.11. Dropout



3.2.11. Dropout





3.2.11. Dropout

Dropout (Srivastava et al., 2014) may be the first instance of a human curated artisanal regularization technique that entered wide scale use in machine learning. Dropout, simply described, is the concept that if you can learn how to do a task repeatedly whilst drunk, you should be able to do the task even better when sober. This insight has resulted in numerous state of the art results and a nascent field dedicated

http://smerity.com/arxiv.org/abs/1804.404/1804.404.pdf?fbclid=IwAR313y1aTjBWThO8A71EDalIOt6tlmZVyn6BhzegkXHyyGU9dUY1jPaTlnU

3.2.11. Dropout

Implementación: Inverted Dropout

Aplicamos Dropout a la tercera capa de una red neuronal con probabilidad de 0.8 de mantener cada neurona:
$l = 3, keep_{prob} = 0.8$

3.2.11. Dropout

Implementación: Inverted Dropout

Aplicamos Dropout a la tercera capa de una red neuronal con probabilidad de 0.8 de mantener cada neurona:
$l = 3, keep_{prob} = 0.8$

$$d3 = np.random.rand(a3.shape[0], a3.shape[1]) < keep\_prob$$

3.2.11. Dropout

Implementación: Inverted Dropout

Aplicamos Dropout a la tercera capa de una red neuronal con probabilidad de 0.8 de mantener cada neurona:
$l = 3, keep_{prob} = 0.8$

$$d3 = np.random.rand(a3.shape[0], a3.shape[1]) < keep\_prob$$

Crea matriz dropout

Distribución uniforme

Mismas dimensiones que a3

Binariza resultados

3.2.11. Dropout

Implementación: Inverted Dropout

Aplicamos Dropout a la tercera capa de una red neuronal con probabilidad de 0.8 de mantener cada neurona:
$l = 3, keep_{prob} = 0.8$

$$d3 = np.random.rand(a3.shape[0], a3.shape[1]) < keep\_prob$$

Crea matriz dropout

Distribución uniforme

Mismas dimensiones que a3

Binariza resultados

$$a3 = np.multiply(a3, d3)$$ #Multiplicación punto a punto

3.2.11. Dropout

Implementación: Inverted Dropout

Aplicamos Dropout a la tercera capa de una red neuronal con probabilidad de 0.8 de mantener cada neurona:
$l = 3, keep_{prob} = 0.8$

$$d3 = np.random.rand(a3.shape[0], a3.shape[1]) < keep\_prob$$

Crea matriz dropout

Distribución uniforme

Mismas dimensiones que a3

Binariza resultados

$a3 = np.multiply(a3, d3)$ #Multiplicación punto a punto

$a3 = a3\ /keep_{prob}$

3.2.11. Dropout

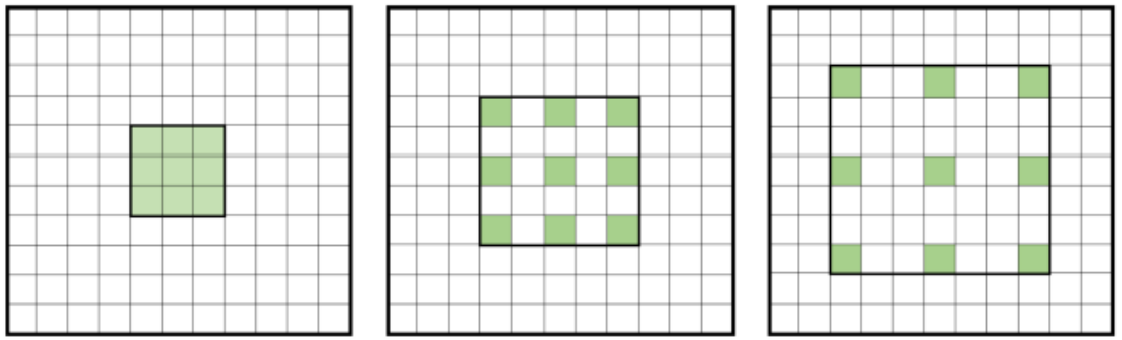
Dropout en Keras

```
from keras.layers import Input, Dense, BatchNormalization, Flatten, Conv2D, MaxPooling2D, Dropout
from keras.models import Model

def alex_net(input_shape=(227, 227, 3), classes=2):

    # Define the input as a tensor with shape input_shape
    x_input = Input(input_shape)
    # Stage 1
    x = Conv2D(96, (11, 11), strides=(4, 4), padding='valid', activation='relu', name='conv_1')(x_input)
    x = MaxPooling2D(pool_size=(3, 3), strides=(2, 2))(x)
    x = BatchNormalization(axis=3, name='bn_conv1')(x)
                    .
                    .
                    .
# Stage 4
    x = Flatten(name="flatten")(x)
    x = Dense(4096, activation='relu', name='FC_1')(x)
    x = Dropout(0.5)(x)
    x = Dense(4096, activation='relu', name='FC_2')(x)
    x = Dropout(0.5)(x)
    if classes == 2:
        x = Dense(1, activation='sigmoid', name='FC_3')(x)
    else:
        x = Dense(classes, activation='softmax', name='FC_3')(x)
    # Create model
    model = Model(input=x_input, output=x)

    return model
```

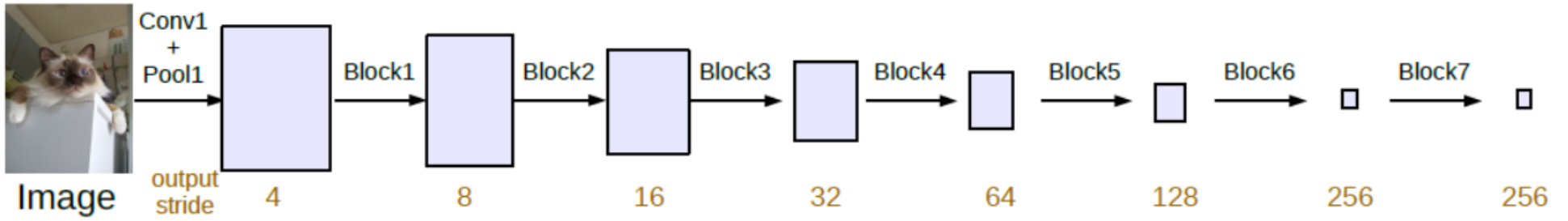
3.3. Special Convolutions

3.3.1. Atrous Convolution

Conocidaambién como “Convolución Dilatada”. Es básicamente una convolución con filtros dilatados. Permite tincrementar el campo de visión de los filtros sin necesidad de incrementar el número de parámetros o reducir el tamaño original del canal (pooling). Esto permite agregar varios filtros con distintos niveles de dilatación para un análisis multi-escala.

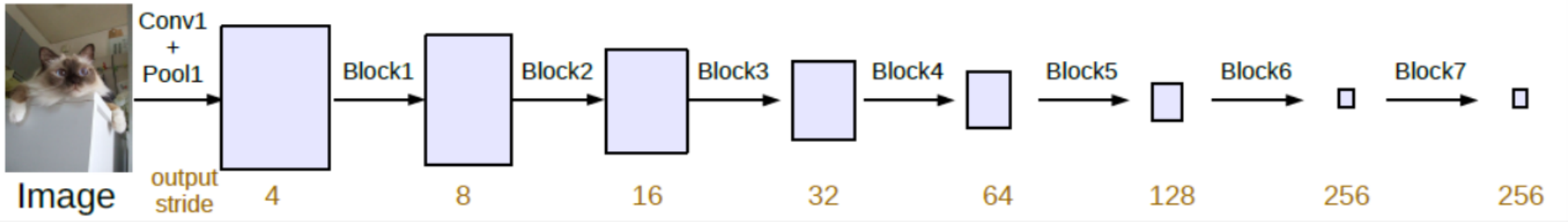


3.3.1. Atrous Convolution

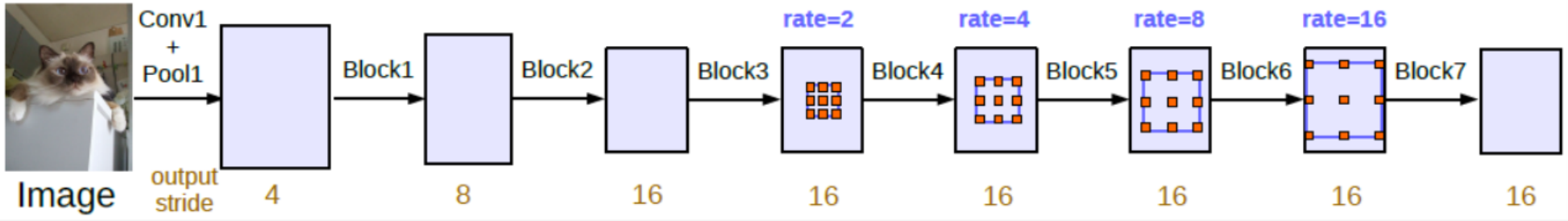


(a) Going deeper without atrous convolution.

3.3.1. Atrous Convolution

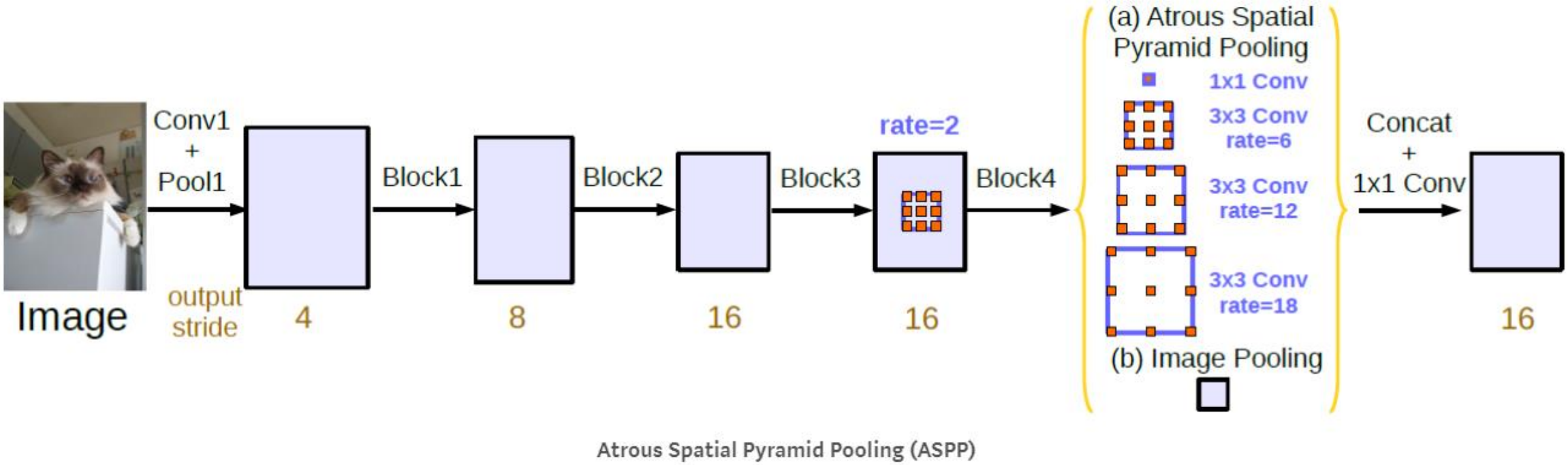


(a) Going deeper without atrous convolution.



(b) Going deeper with atrous convolution. Atrous convolution with $rate > 1$ is applied after block3 when $output\_stride = 16$.

3.3.1. Atrous Convolution



Atrous Spatial Pyramid Pooling (ASPP)

3.3.1. Atrous Convolution

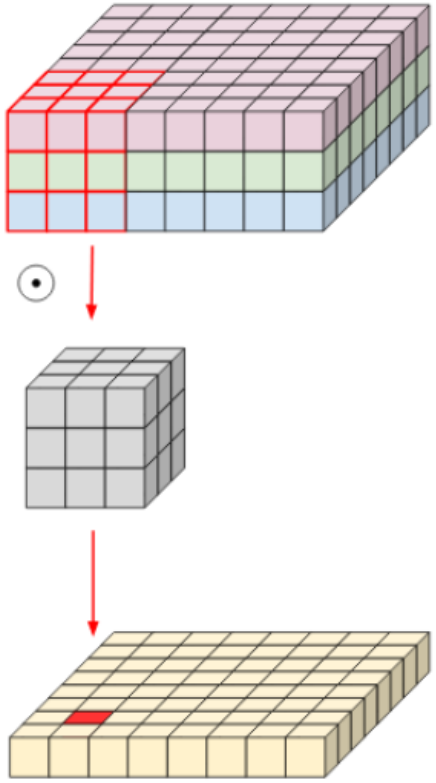
Atrous Convolution in Keras

Conv2D

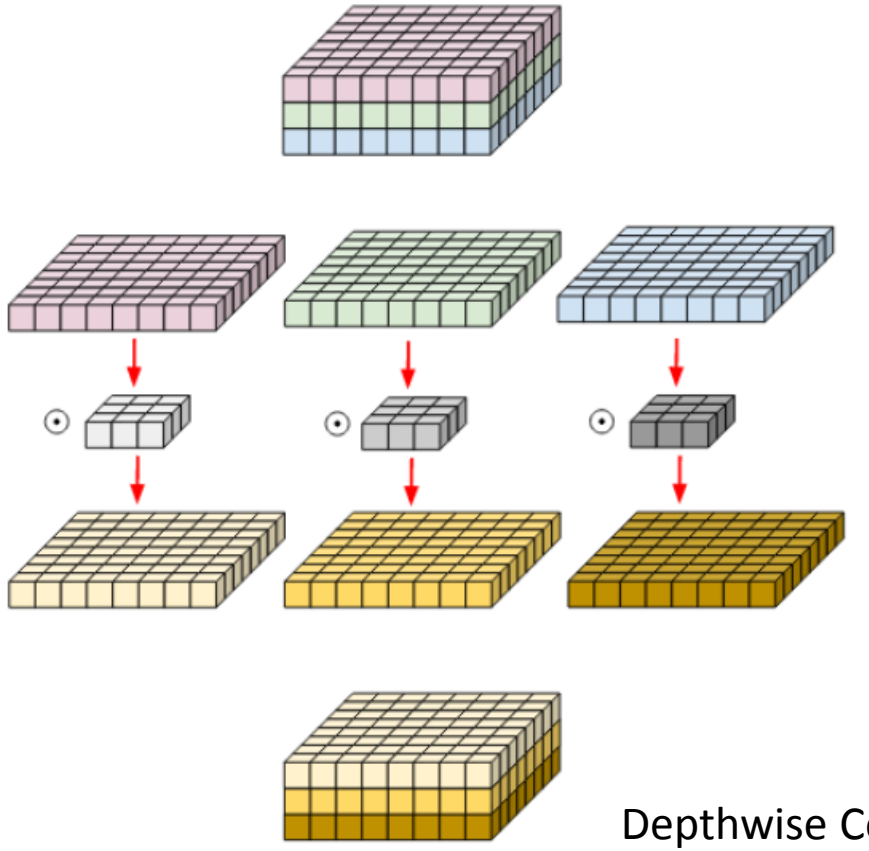
keras.layers.Conv2D(filters, kernel_size, strides=(1, 1), padding='valid', data_format=**None**, **dilation_rate**=(1, 1),)

3.3.2. Depthwise Convolution

Se aplica un filtro 2D a cada canal (“depth level”) del tensor de entrada.



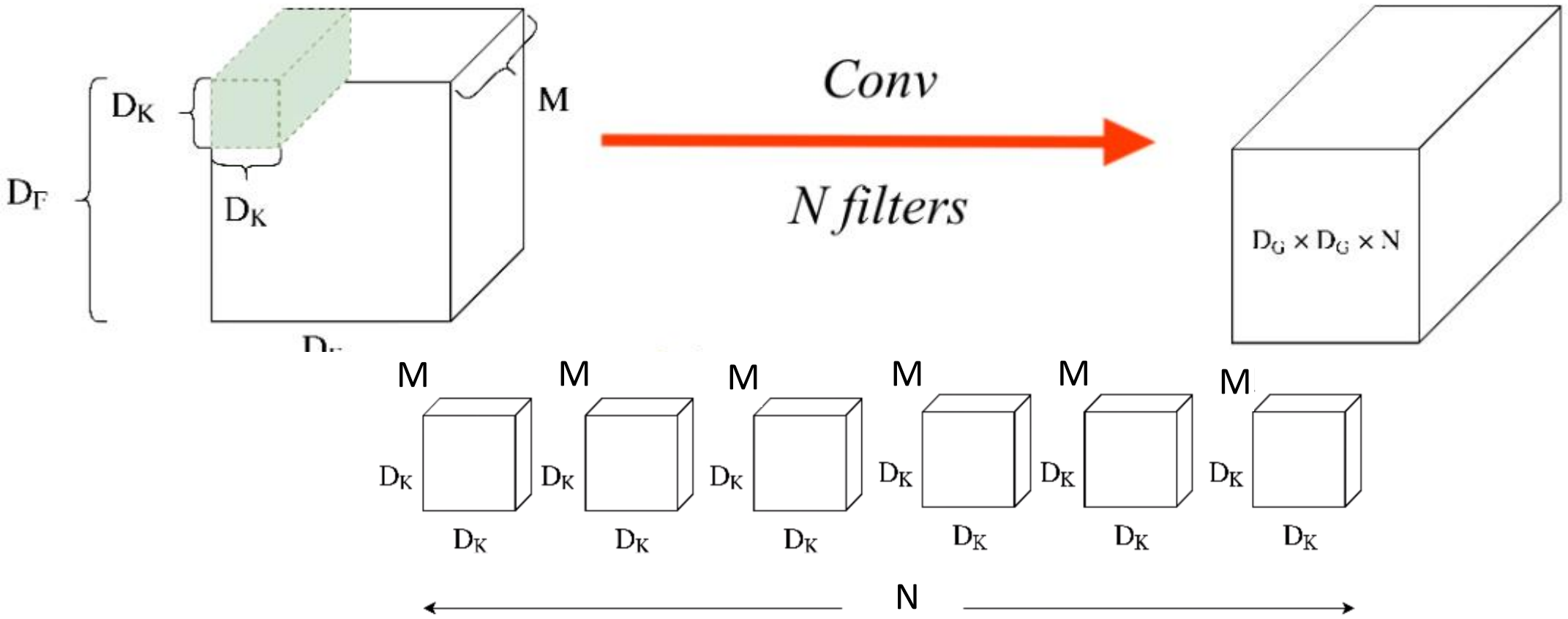
Convolución normal



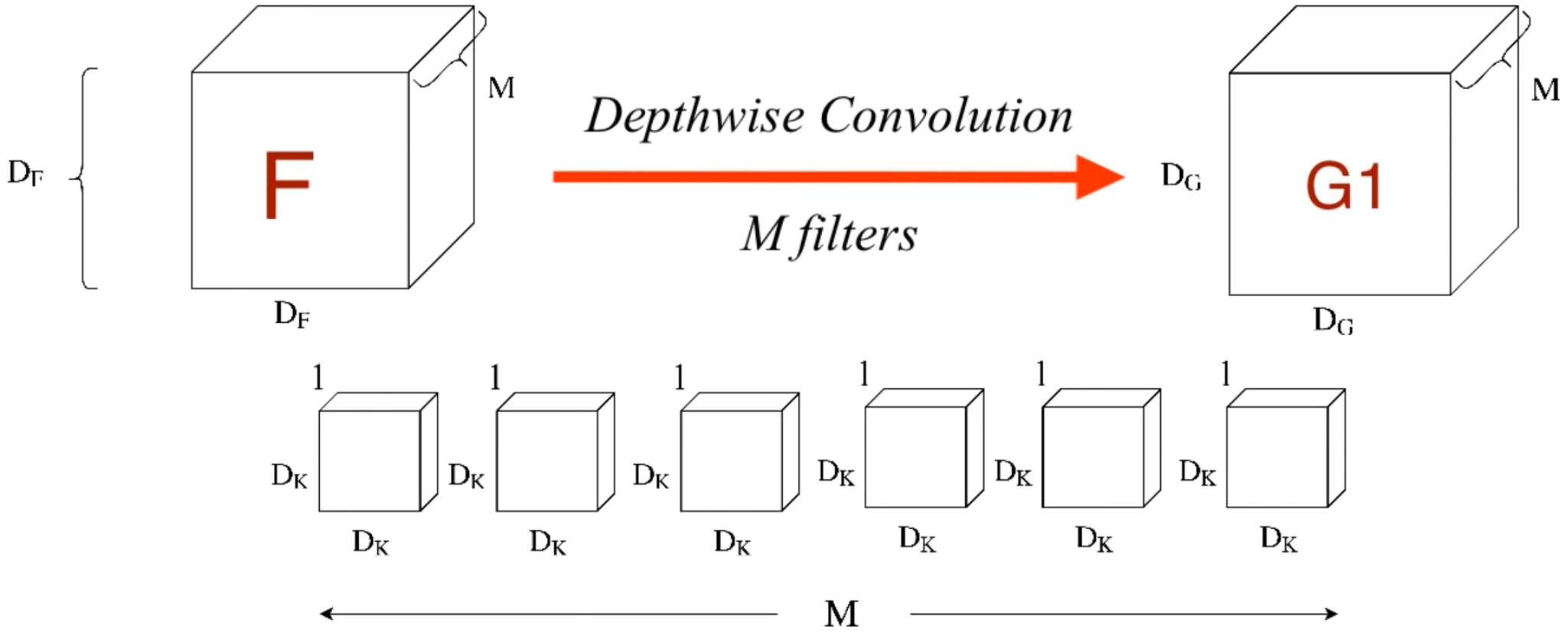
Depthwise Convolution

3.3.2. Depthwise Convolution

Convolución normal

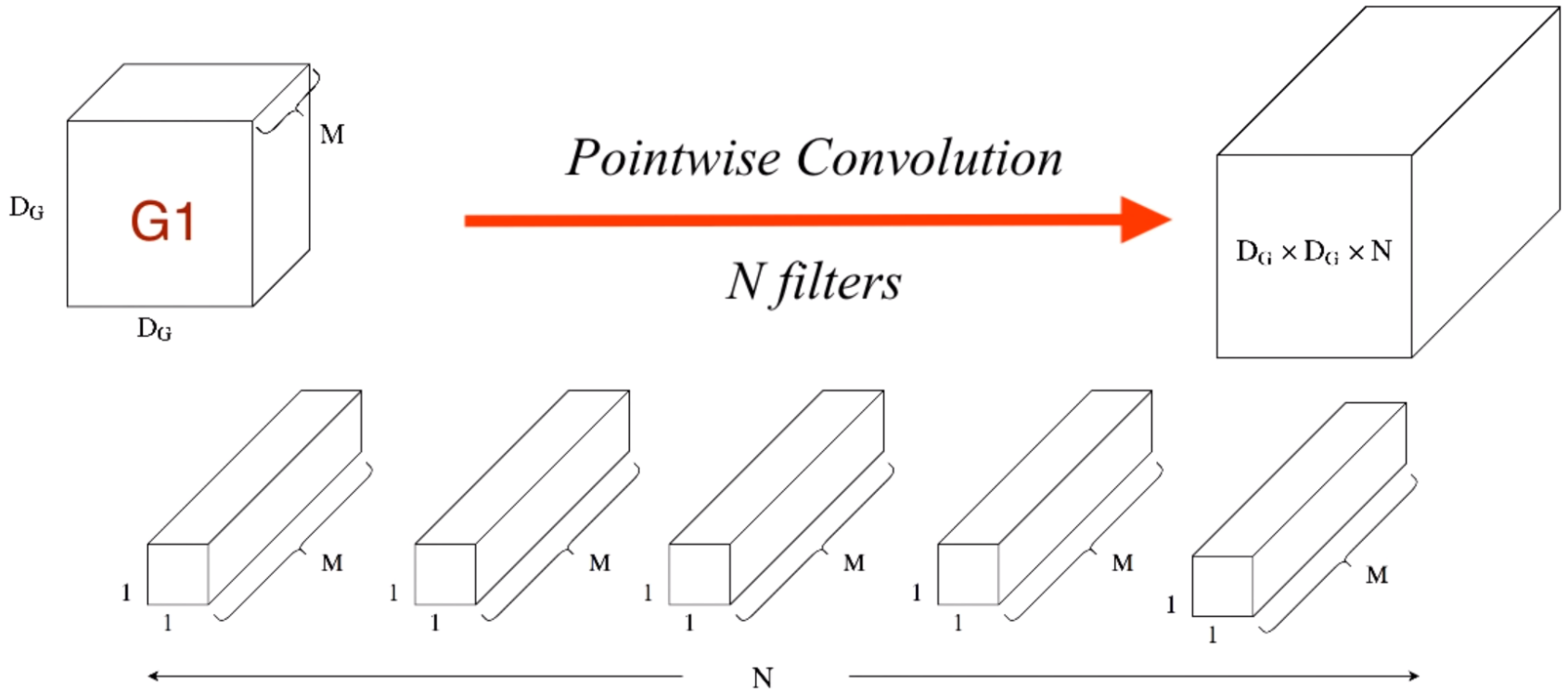


3.3.2. Depthwise Convolution



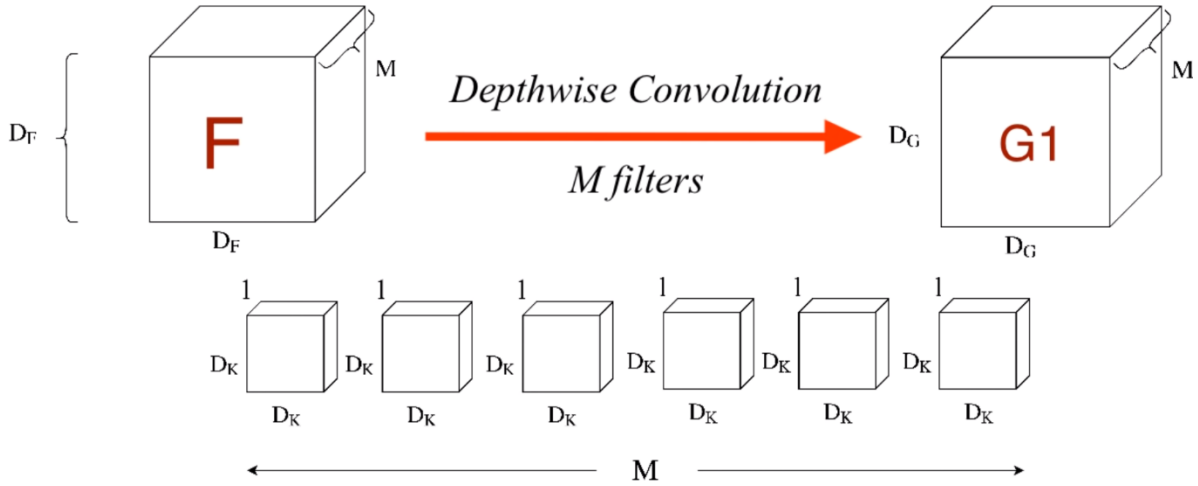
3.3.3. Separable Convolution

Consiste en una Depthwise Convolution seguida de una Poitwise Convolution (N filtros de 1x1):



3.3.3. Separable Convolution

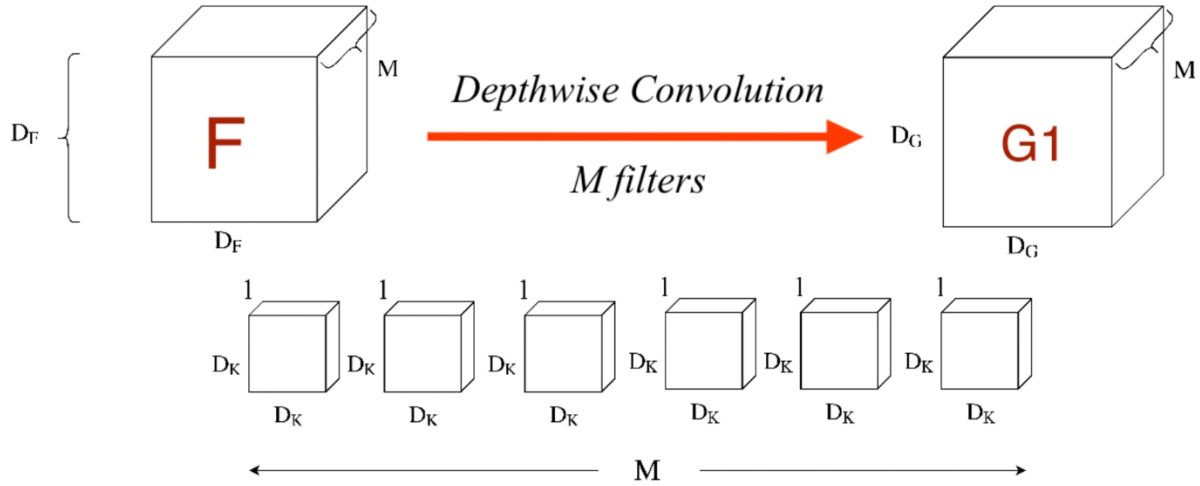
Paso 1: Depthwise. Cálculo del costo computacional



3.3.3. Separable Convolution

Paso 1: Depthwise. Cálculo del costo computacional

$$\#Op.por\ filtro = Dk \times Dk \times 1$$

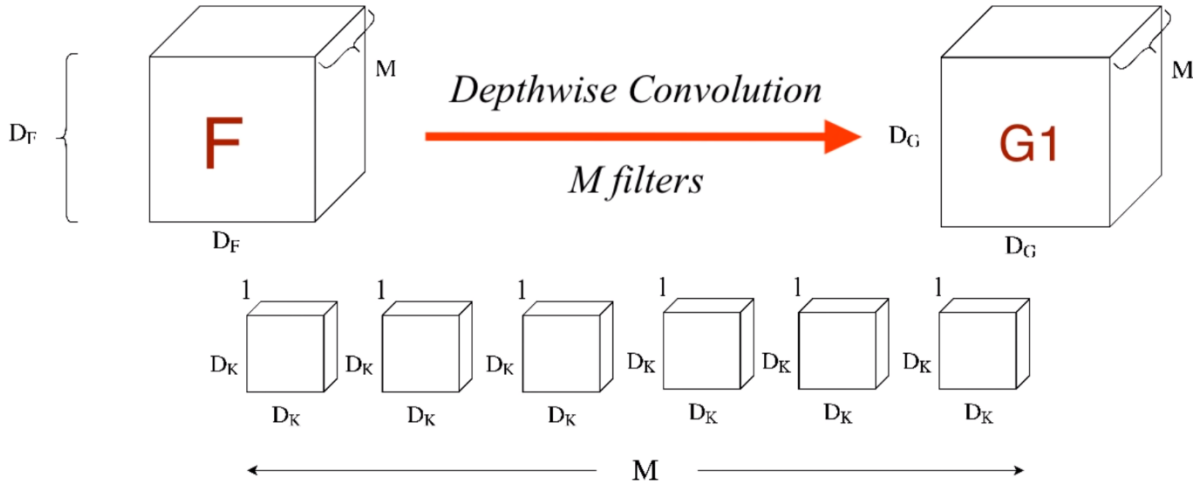


3.3.3. Separable Convolution

Paso 1: Depthwise. Cálculo del costo computacional

$$\#Op.por\ filtro = Dk \times Dk \times 1$$

$$\#Op.por\ canal = D_G^2 \times Dk^2$$



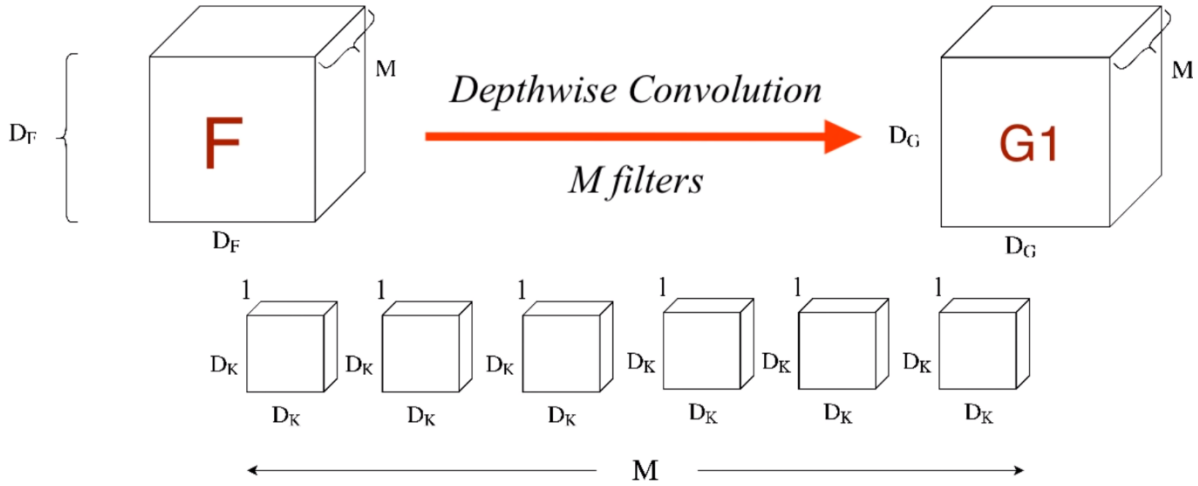
3.3.3. Separable Convolution

Paso 1: Depthwise. Cálculo del costo computacional

$$\#Op.por\ filtro = Dk \times Dk \times 1$$

$$\#Op.por\ canal = D_G^2 \times Dk^2$$

$$\#Op.total1 = M \times D_G^2 \times Dk^2$$



3.3.3. Separable Convolution

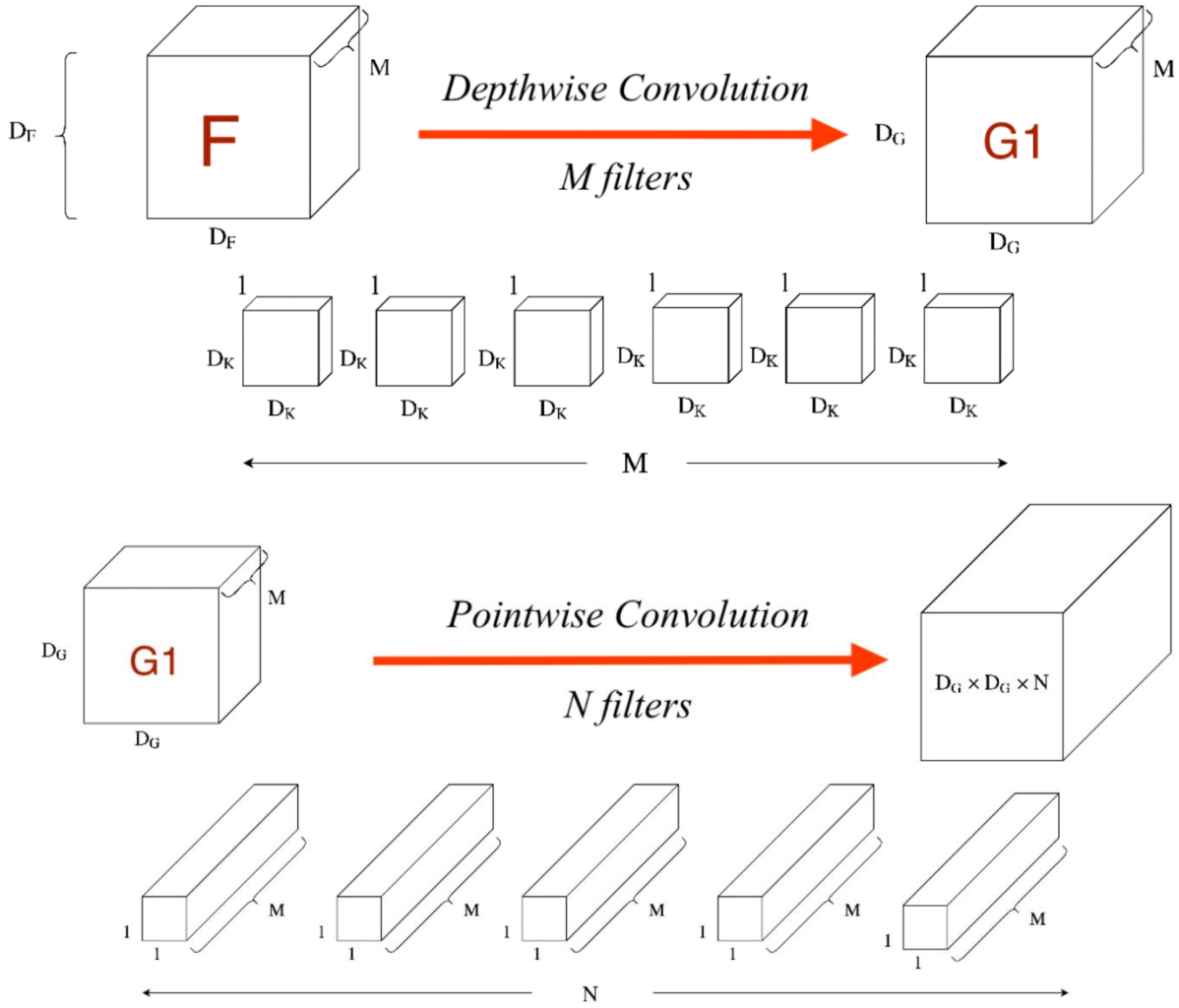
Paso 1: Depthwise. Cálculo del costo computacional

$$\#Op.por\ filtro = Dk \times Dk \times 1$$

$$\#Op.por\ canal = D_G^2 \times Dk^2$$

$$\#Op.total1 = M \times D_G^2 \times Dk^2$$

Paso 2: Pointwise.



3.3.3. Separable Convolution

Paso 1: Depthwise. Cálculo del costo computacional

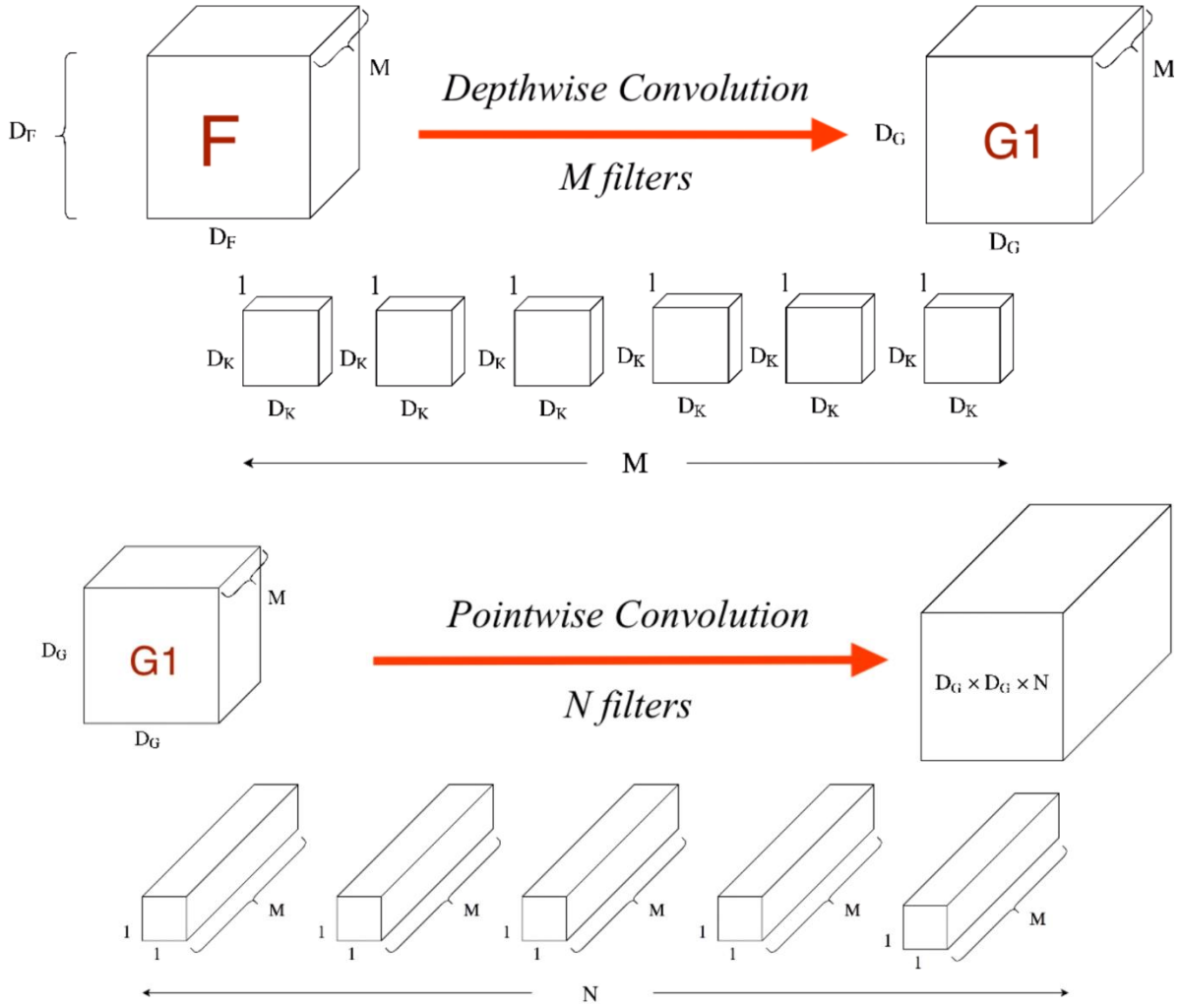
$$\#Op.por\ filtro = Dk \times Dk \times 1$$

$$\#Op.por\ canal = D_G^2 \times Dk^2$$

$$\#Op.total1 = M \times D_G^2 \times Dk^2$$

Paso 2: Pointwise.

$$\#Op.por\ filtro = 1 \times 1 \times M$$



3.3.3. Separable Convolution

Paso 1: Depthwise. Cálculo del costo computacional

$$\#Op.por\ filtro = Dk \times Dk \times 1$$

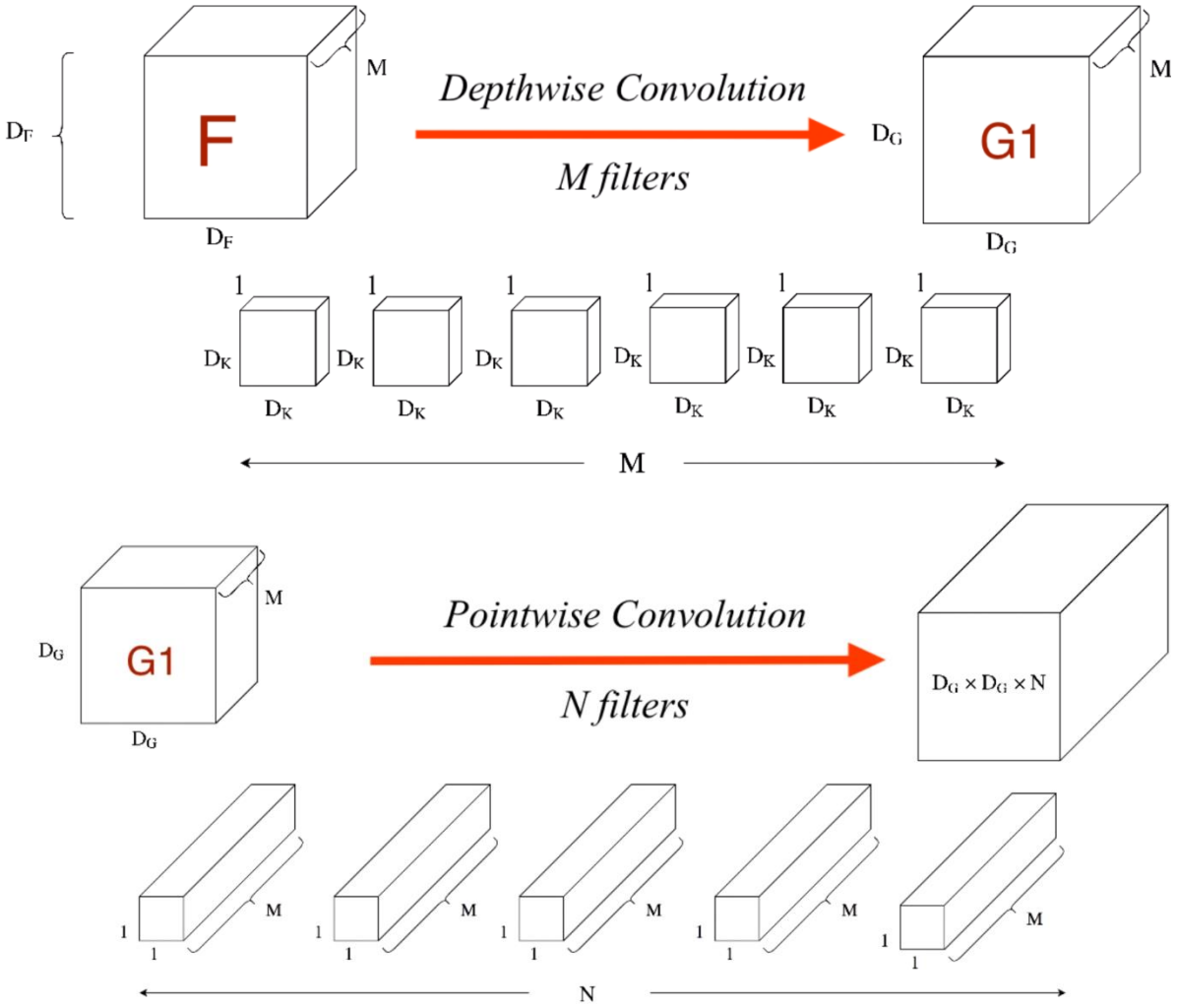
$$\#Op.por\ canal = D_G^2 \times Dk^2$$

$$\#Op.total1 = M \times D_G^2 \times Dk^2$$

Paso 2: Pointwise.

$$\#Op.por\ filtro = 1 \times 1 \times M$$

$$\#Op.por\ canal = D_G^2 \times M$$



3.3.3. Separable Convolution

Paso 1: Depthwise. Cálculo del costo computacional

$$\#Op.por\ filtro = Dk \times Dk \times 1$$

$$\#Op.por\ canal = D_G^2 \times Dk^2$$

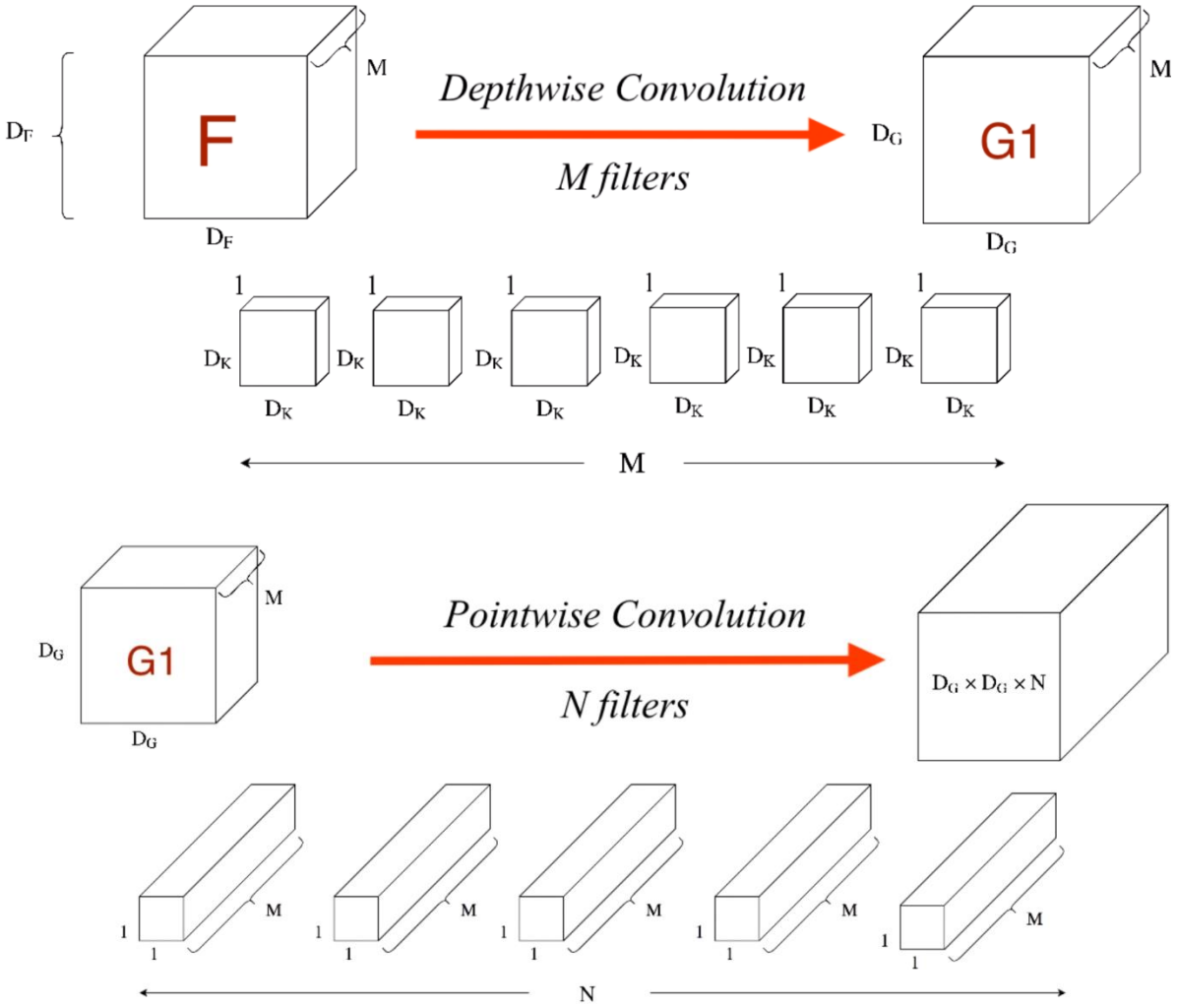
$$\#Op.total1 = M \times D_G^2 \times Dk^2$$

Paso 2: Pointwise.

$$\#Op.por\ filtro = 1 \times 1 \times M$$

$$\#Op.por\ canal = D_G^2 \times M$$

$$\#Op.total2 = N \times D_G^2 \times M$$



3.3.3. Separable Convolution

Paso 1: Depthwise. Cálculo del costo computacional

$$\#Op.por\ filtro = Dk \times Dk \times 1$$

$$\#Op.por\ canal = D_G^2 \times Dk^2$$

$$\#Op.total1 = M \times D_G^2 \times Dk^2$$

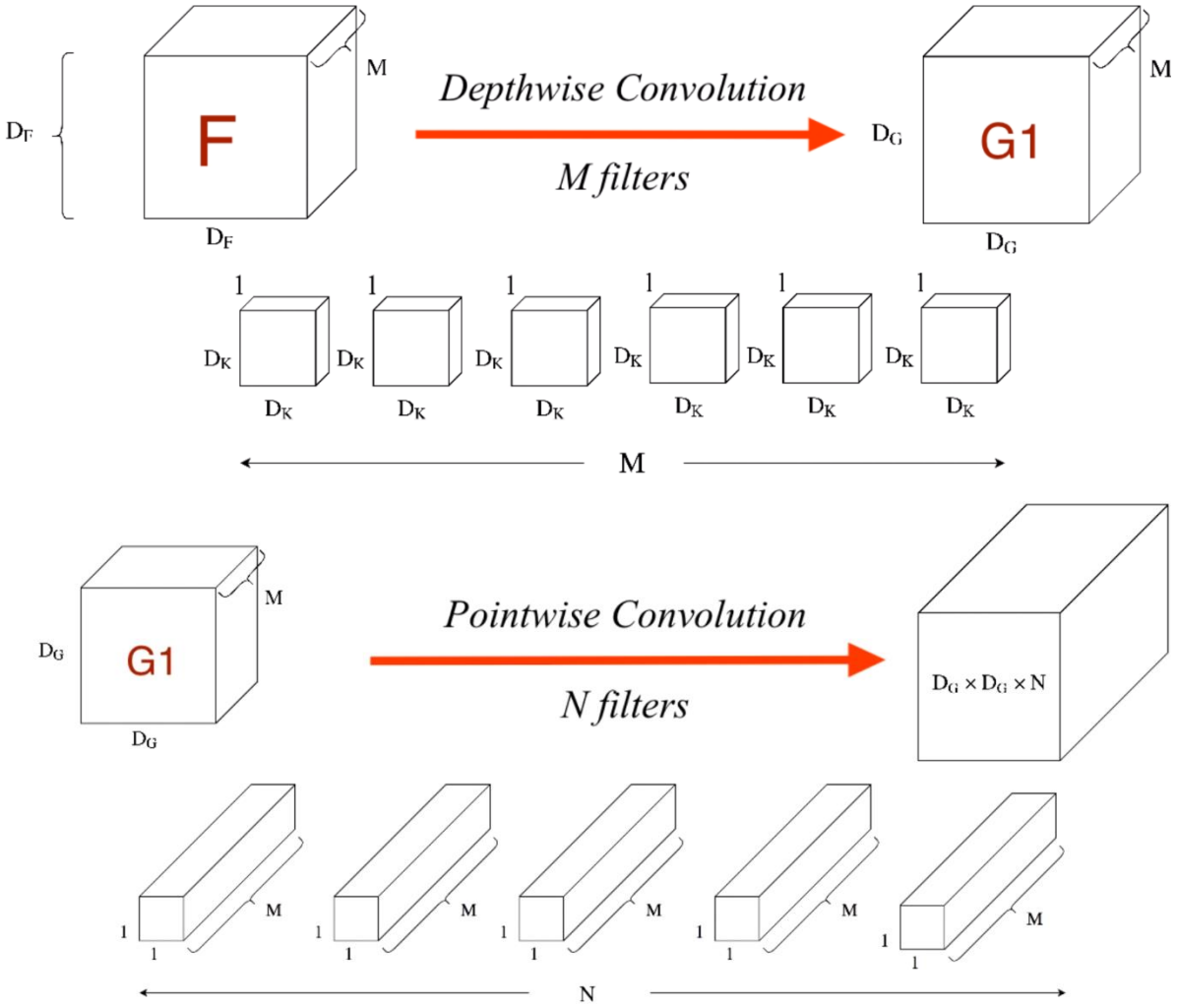
Paso 2: Pointwise.

$$\#Op.por\ filtro = 1 \times 1 \times M$$

$$\#Op.por\ canal = D_G^2 \times M$$

$$\#Op.total2 = N \times D_G^2 \times M$$

$$Total = D_G^2 \times M \times (Dk^2 + N)$$



3.3.3. Separable Convolution

Cálculo del costo computacional de una convolución normal (ver Día 2, Sección 3.2.5)

DF x DF x M

DG x DG x N

3.3.3. Separable Convolution

Cálculo del costo computacional de una convolución normal (ver Día 2, Sección 3.2.5)

DF x DF x M

DG x DG x N

#Operaciones por filtro = Dk x Dk x M

3.3.3. Separable Convolution

Cálculo del costo computacional de una convolución normal (ver Día 2, Sección 3.2.5)

DF x DF x M

DG x DG x N

#Operaciones por filtro = Dk x Dk x M

#Operaciones en total = (DG x DG x N) x (Dk x Dk x M) $= N \times M \times D_G^2 \times Dk^2)$

3.3.3. Separable Convolution

Comparación del cálculo del costo computacional de una convolución normal y una separable convolution

$$\frac{\#Op.\,Depthwise}{\#Op.\,conv.\,normal} = \frac{D_G^2 \times M \times (Dk^2 + N)}{N \times M \times D_G^2 \times Dk^2)}$$

3.3.3. Separable Convolution

Comparación del cálculo del costo computacional de una convolución normal y una separable convolution

$$\frac{\#Op.\,Depthwise}{\#Op.\,conv.\,normal} = \frac{\cancel{D_G^2 \times M} \times (Dk^2 + N)}{N \times \cancel{M \times D_G^2} \times Dk^2)}$$

3.3.3. Separable Convolution

Comparación del cálculo del costo computacional de una convolución normal y una separable convolution

$$\frac{\#Op.\,Depthwise}{\#Op.\,conv.\,normal} = \frac{\cancel{D_G^2 \times M} \times (Dk^2 + N)}{N \times \cancel{M \times D_G^2} \times Dk^2)}$$

$$\frac{\#Op.\,Depthwise}{\#Op.\,conv.\,normal} = \frac{Dk^2 + N}{Dk^2 \times N} = \frac{1}{N} + \frac{1}{Dk^2}$$

3.3.3. Separable Convolution

Comparación del cálculo del costo computacional de una convolución normal y una separable convolution

$$\frac{\#Op.\,Depthwise}{\#Op.\,conv.\,normal} = \frac{\cancel{D_G^2 \times M} \times (Dk^2 + N)}{N \times \cancel{M \times D_G^2} \times Dk^2)}$$

$$\frac{\#Op.\,Depthwise}{\#Op.\,conv.\,normal} = \frac{Dk^2 + N}{Dk^2 \times N} = \frac{1}{N} + \frac{1}{Dk^2}$$

$$N = 1024 \qquad Dk = 3$$

3.3.3. Separable Convolution

Comparación del cálculo del costo computacional de una convolución normal y una separable convolution

$$\frac{\#Op.\,Depthwise}{\#Op.\,conv.\,normal} = \frac{\cancel{D_G^2 \times M} \times (Dk^2 + N)}{N \times \cancel{M \times D_G^2} \times Dk^2)}$$

$$\frac{\#Op.\,Depthwise}{\#Op.\,conv.\,normal} = \frac{Dk^2 + N}{Dk^2 \times N} = \frac{1}{N} + \frac{1}{Dk^2}$$

$$N = 1024 \qquad Dk = 3$$

$$\frac{\#Op.\,Depthwise}{\#Op.\,conv.\,normal} = \frac{1}{1024} + \frac{1}{3k^2} = 0.112$$

3.3.3. Atrous Separable Convolution (ASC)

Depthwise con filtros dilatados + Pointwise Convolution

SeparableConv2D

```
keras.layers.SeparableConv2D(filters, kernel_size, strides=(1, 1), padding='valid', data_format=None,
    dilation_rate=(1, 1), depth_multiplier=1, activation=None, use_bias=True)
```

https://keras.io/layers/convolutional/

3.4. Training with Generators

ImageGenerator in Keras

- **featurewise_center**: Boolean. Normaliza a 0 el promedio de todo el dataset.
- **samplewise_center**: Boolean. Setea a 0 el promedio de cada muestra.
- **featurewise_std_normalization**: Boolean. Divide los input por la std de todo el dataset.
- **samplewise_std_normalization**: Boolean. Divide cada input por su std.
- **rotation_range**: Int. Rango de grados para rotaciones aleatorias.
- **width_shift_range / height_shift_range** : Float, Rangos (fracciones del ancho o alto totals) en los que aleatoriamente se moverán las imágenes horizontalmente o verticalmente.
- **shear_range**: Float. Rango de ángulo de corte en grados.
- **zoom_range**: Float. Rango de zoom.
- **horizontal_flip / vertical_flip**: Boolean. Voltea horizontal y verticalmente aleatoriamente.

https://keras.io/preprocessing/image/

https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

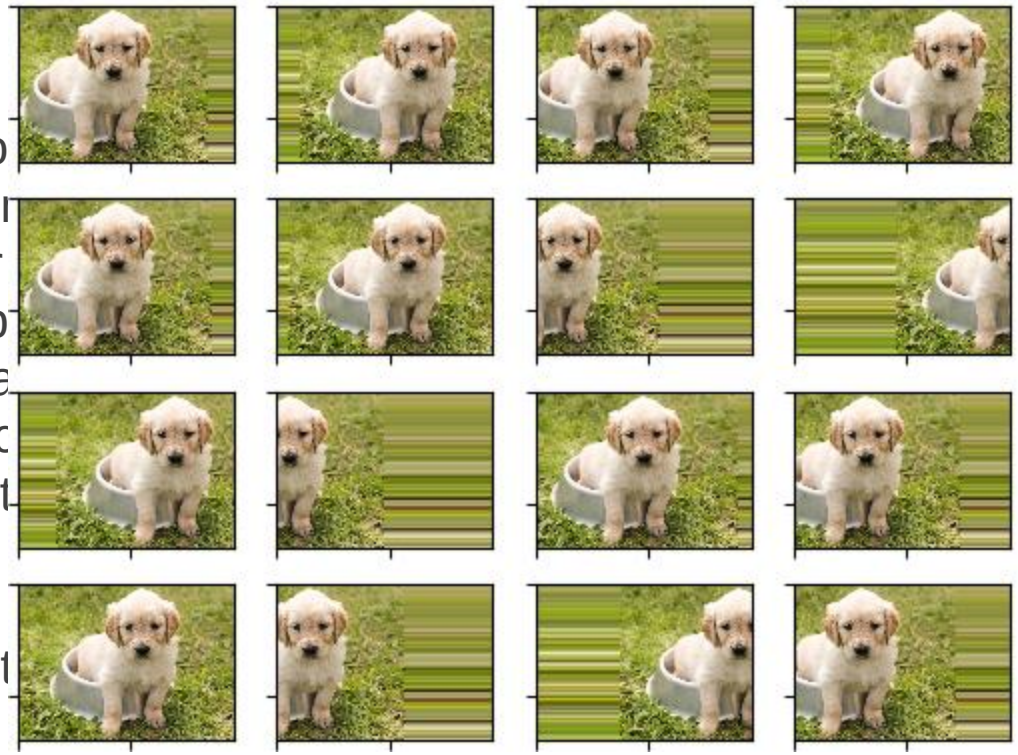
3.4. Training with Generators

ImageGenerator in Keras

- **featurewise_center**: Boolean. Normaliza a 0 el promedio de to
- **samplewise_center**: Boolean. Setea a 0 el promedio de cada i
- **featurewise_std_normalization**: Boolean. Divide los input por
- **samplewise_std_normalization**: Boolean. Divide cada input p
- **rotation_range**: Int. Rango de grados para rotaciones aleatoria
- **width_shift_range / height_shift_range** : Float, Rangos (fracc

aleatoriamente se moverán las imágenes horizontalmente o vert

- **shear_range**: Float. Rango de ángulo de corte en grados.
- **zoom_range**: Float. Rango de zoom.
- **horizontal_flip / vertical_flip**: Boolean. Voltea horizontal y vert



https://keras.io/preprocessing/image/

https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

3.4. Training with Generators

ImageGenerator in Keras

- **featurewise_center**: Boolean. Normaliza a 0 el promedio de todo el dataset.
- **samplewise_center**: Boolean. Setea a 0 el promedio de cada muestra.
- **featurewise_std_normalization**: Boolean. Divide los input por la std de todo el dataset.
- **samplewise_std_normalization**: Boolean. Divide cada input por su std.
- **rotation_range**: Int. Rango de grados para rotaciones aleatorias.
- **width_shift_range / height_shift_range** : Float, Rangos (fracciones del ancho o alto totals) en los que aleatoriamente se moverán las imágenes horizontalmente o verticalmente.
- **shear_range**: Float. Rango de ángulo de corte en grados.
- **zoom_range**: Float. Rango de zoom.
- **horizontal_flip / vertical_flip**: Boolean. Voltea horizontal y verticalmente aleatoriamente.

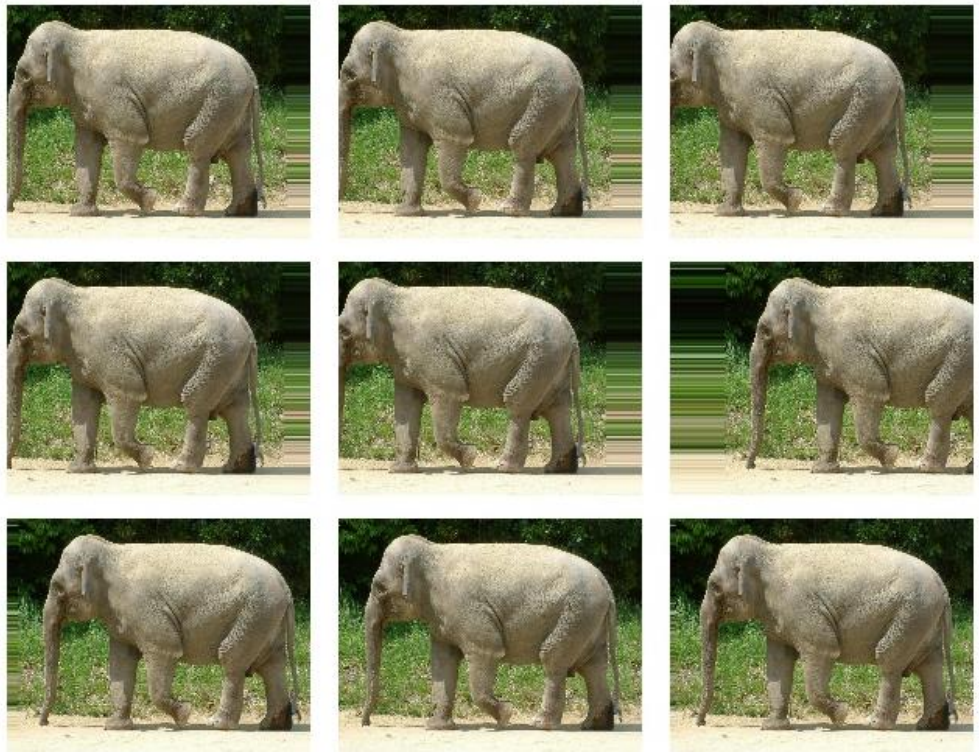
https://keras.io/preprocessing/image/

https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

3.4. Training with Generators

ImageGenerator in Keras

•**featurewise_center**: Boolean. Normaliza a 0 el promedio de toc
•**samplewise_center**: Boolean. Setea a 0 el promedio de cada n
•**featurewise_std_normalization**: Boolean. Divide los input por l
•**samplewise_std_normalization**: Boolean. Divide cada input pc
•**rotation_range**: Int. Rango de grados para rotaciones aleatorias
•**width_shift_range / height_shift_range** : Float, Rangos (fracci
aleatoriamente se moverán las imágenes horizontalmente o verti
•**shear_range**: Float. Rango de ángulo de corte en grados.
•**zoom_range**: Float. Rango de zoom.
•**horizontal_flip / vertical_flip**: Boolean. Voltea horizontal y verti



https://keras.io/preprocessing/image/

https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

3.4. Training with Generators

ImageGenerator in Keras

- **featurewise_center**: Boolean. Normaliza a 0 el promedio de todo el dataset.
- **samplewise_center**: Boolean. Setea a 0 el promedio de cada muestra.
- **featurewise_std_normalization**: Boolean. Divide los input por la std de todo el dataset.
- **samplewise_std_normalization**: Boolean. Divide cada input por su std.
- **rotation_range**: Int. Rango de grados para rotaciones aleatorias.
- **width_shift_range / height_shift_range** : Float, Rangos (fracciones del ancho o alto totals) en los que aleatoriamente se moverán las imágenes horizontalmente o verticalmente.
- **shear_range**: Float. Rango de ángulo de corte en grados.
- **zoom_range**: Float. Rango de zoom.
- **horizontal_flip / vertical_flip**: Boolean. Voltea horizontal y verticalmente aleatoriamente.

https://keras.io/preprocessing/image/

https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

3.4. Training with Generators

Flow

Toma la data “x” y sus target “y” y genera batches de data aumentada

```
datagen.flow(x_train, y_train,
batch_size=32)
```

Flow from directory

Toma como entrada la carpeta donde están los datos separados en subcarpetas (1 por clase) y genera los batches con data augmentation.

```
train_generator = datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary',
    subset='training')
```

```
data/
    train/
        dogs/
            dog001.jpg
            dog002.jpg
            ...
        cats/
            cat001.jpg
            cat002.jpg
            ...
    validation/
        dogs/
            dog001.jpg
            dog002.jpg
            ...
        cats/
            cat001.jpg
            cat002.jpg
```

https://keras.io/preprocessing/image/

3.4. Training with Generators

ImageGenerator in Keras: CatsvsDogAugmentation.py