TensorFlow

ccoss

Deck: bit.ly/CCOSS-TF
Slack: sg1.run/ccoss-slack

crispinvv@google.com

TensorFlow

CCOSS

Intro a TensorFlow 2.0
CCOSS 2019

Crispin Velez (@crispindev)
Creditos a Josh Gordon TF advocate
@random-forests

Aprendizaje profundo es aprendizaje de representación

TensorFlow

Ultimos tutoriales y guias

tensorflow.org/beta

Noticias y actualizaciones

medium.com/tensorflow

twitter.com/tensorflow

Demo

PoseNet y BodyPix

bit.ly/pose-net

bit.ly/body-pix

TensorFlow for JavaScript, S
Android, y iOS

tensorflow.org/js
tensorflow.org/swift
tensorflow.org/lite

Agenda

10:30am -> 11:30am

Ejercicios

- MNIST de moda con capas densas (y un segundo ejercicio avanzado opcional)
- CIFAR-10 con capas convolucionales
- Imágenes del mundo real con transferencia de aprendizaje

Conceptos

- Descenso gradiente, capas densas, pérdida, softmax, convolución

Juegos

- QuickDraw

Agenda

10:30am -> 12:30pm

Tutoriales y nuevos tutoriales

- Segmentación de imagen
- Sueño profundo y transferencia de estilo
- Predicción de series de tiempo

Juegos

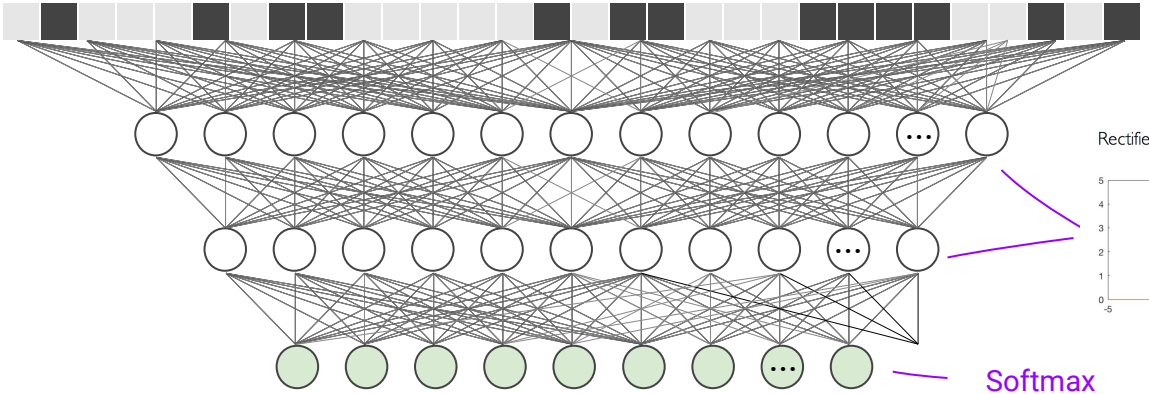
- Sketch RNN

mas...

Minimal MNIST en TF 2.0

Un modelo lineal, una red neuronal y una red neuronal profunda, luego un breve ejercicio.

https://bit.ly/2kuYgDp




```
model = Sequential()
model.add(Dense(256, activation='relu',input_shape=(784,)))
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

Modelo Lineal $$f(x) = softmax(W_1 x)$$

Red Neuronal $$f(x) = softmax(W_2(g(W_1 x)))$$

Red neuronal profunda $$f(x) = softmax(W_3(g(W_2(g(W_1 x)))))$$

```
import matplotlib.pyplot as plt

# Add a validation set
history = model.fit(x_train, y_train, validation_data=(x_test, y_test) ...)

# Get stats from the history object
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
epochs = range(len(acc))

# Plot accuracy vs epochs
plt.title('Training and validation accuracy')
plt.plot(epochs, acc, color='blue', label='Train')
plt.plot(epochs, val_acc, color='orange', label='Val')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
```

Exercicio (option 1)

https://bit.ly/2kuYgDp

tensorflow.org/beta/tutorials/keras/basic_classification

Agregar un conjunto de validación. Agregue código para trazar la pérdida frente a las epochs.

Ejercicio (option 2)

bit.ly/ijcav_adv

Respuesta:

bit.ly/ijcai_adv_answer

Sobre TensorFlow 2.0

Instalar

```
# GPU
!pip install tensorflow-gpu==2.0.0-rc1
```

```
# CPU
!pip install tensorflow==2.0.0-rc1
```

En cualquier caso, verifique su instalación (en Colab, es posible que necesite usar tiempo de ejecución -> reiniciar después de la instalación).

```
import tensorflow as tf
print(tf.__version__) # 2.0.0-rc1
```

Nightly también está disponible, pero lo mejor es seguir con un lanzamiento con nombre para la estabilidad.

TF2 es imperativo por default

```
import tensorflow as tf
print(tf.__version__) # 2.0.0-beta1

x = tf.constant(1)
y = tf.constant(2)
z = x + y

print(z) # tf.Tensor(3, shape=(), dtype=int32)
```

Puedes explorar capas

```
from tensorflow.keras.layers import Dense
layer = Dense(units=1, kernel_initializer='ones', use_bias=False)
data = tf.constant([[1.0, 2.0, 3.0]]) # Note: a batch of data
print(data) # tf.Tensor([[1. 2. 3.]], shape=(1, 3), dtype=float32)

# Call the layer on our data
result = layer(data)

print(result) # tf.Tensor([[6.]], shape=(1, 1), dtype=float32)
print(result.numpy()) # tf.Tensors have a handy .numpy() method
```

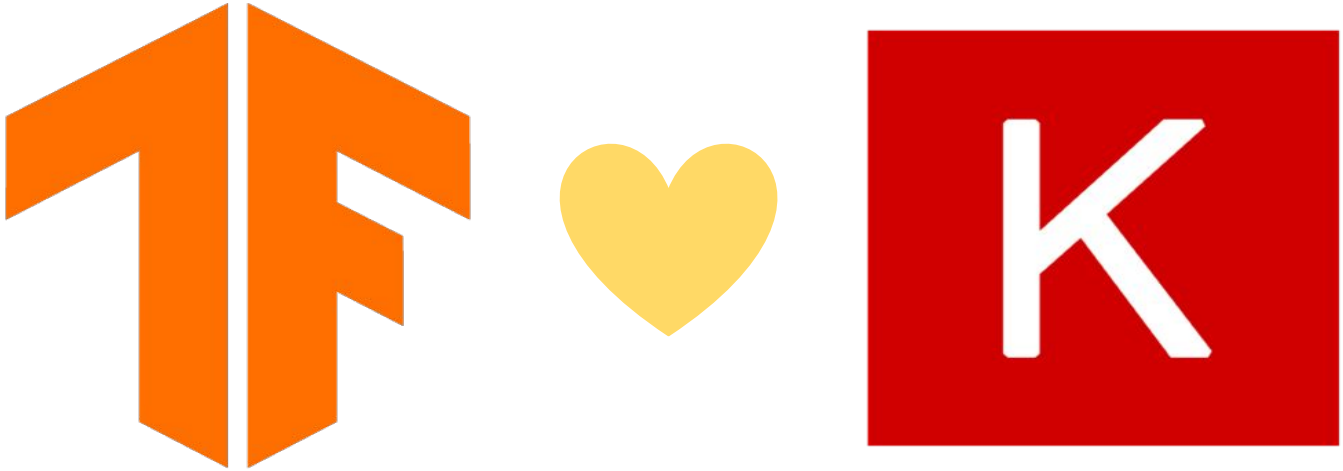
TF1: Construye un gráfico, luego ejecútalo.

```
import tensorflow as tf # 1.14.0
print(tf.__version__)

x = tf.constant(1)
y = tf.constant(2)
z = tf.add(x, y)

print(z)
```

TF1: Construye un gráfico, luego ejecútalo

```
import tensorflow as tf # 1.14.0
print(tf.__version__)

x = tf.constant(1)
y = tf.constant(2)
z = tf.add(x, y)

print(z) # Tensor("Add:0", shape=(), dtype=int32)

with tf.Session() as sess:
  print(sess.run(x)) # 3
```

Keras está integrado en TF2



Cómo importar tf.keras

Si desea usar **tf.keras** y ve el mensaje "Uso del backend TensorFlow", ha importado accidentalmente Keras (que está instalado de forma predeterminada en Colab) desde fuera de TensorFlow.

Example

```
# !pip install tensorflow==2.0.0-beta1, then

>>> from tensorflow.keras import layers # Right

>>> from keras import layers # Oops

Using TensorFlow backend. # You shouldn't see this
```

When in doubt, copy the imports from one of the tutorials on tensorflow.org/beta

Notas

Un **supersubset** de la implementación de referencia. Integrado en TensorFlow 2.0 (no es necesario instalar Keras por separado).

Documentación y ejemplos

- **Tutorials**: tensorflow.org/beta
- **Guide**: tensorflow.org/beta/guide/keras/

```
!pip install tensorflow==2.0.0-beta1
from tensorflow import keras
```

Recomiendo los ejemplos que encuentre en tensorflow.org/beta sobre otros recursos (se mantienen mejor y la mayoría de ellos se revisan cuidadosamente).

tf.keras agrega un montón de cosas, incluyendo ... subclases de modelos (construcción de modelos de estilo Chainer / PyTorch), bucles de entrenamiento personalizados usando GradientTape, una colección de estrategias de entrenamiento distribuidas, soporte para TensorFlow.js, Android, iOS, etc.

Mas notas

TF 2.0 es similar a NumPy, en:

- Soporte a GPU
- Autodiff
- Entrenamiento distribuido
- Compilacion JIT
- Un formato portátil (entrenar en Python en Mac, implementar en iOS usando Swift o en un navegador usando JavaScript)

Escriba modelos en Python, JavaScript o Swift (y corralos en cualquier lado).

API doc: tensorflow.org/versions/r2.0/api_docs/python/tf
Note: Asegurese que estan viendo la version 2.0

Tres estilos de construcción modelo

Secuencial, Funcional, Subclases

Modelos secuenciales

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

TF 1.x

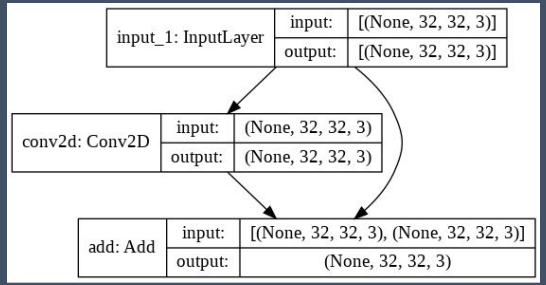
```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

TF 2.0

```
model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(512, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Modelos funcionales

```
inputs = keras.Input(shape=(32, 32, 3))

y = layers.Conv2D(3, (3, 3),activation='relu',padding='same')(inputs)
outputs = layers.add([inputs, y])
model = keras.Model(inputs, outputs)

keras.utils.plot_model(model, 'skip_connection.png', show_shapes=True)
```



Modelos subclasificados

```
class MyModel(tf.keras.Model):
  def __init__(self, num_classes=10):
    super(MyModel, self).__init__(name='my_model')
    self.dense_1 = layers.Dense(32, activation='relu')
    self.dense_2 = layers.Dense(num_classes,activation='sigmoid')

  def call(self, inputs):
    # Define your forward pass here
    x = self.dense_1(inputs)
    return self.dense_2(x)
```

Dos estilos de entrenamiento

Incorporado y personalizado

Use ‘loop’ de entrenamiento incorporado

```
model.fit(x_train, y_train, epochs=5)
```

O define tu propio

```
model = MyModel()

with tf.GradientTape() as tape:
  logits = model(images)
  loss_value = loss(logits, labels)

grads = tape.gradient(loss_value, model.trainable_variables)
optimizer.apply_gradients(zip(grads, model.trainable_variables))
```

Algunos conceptos

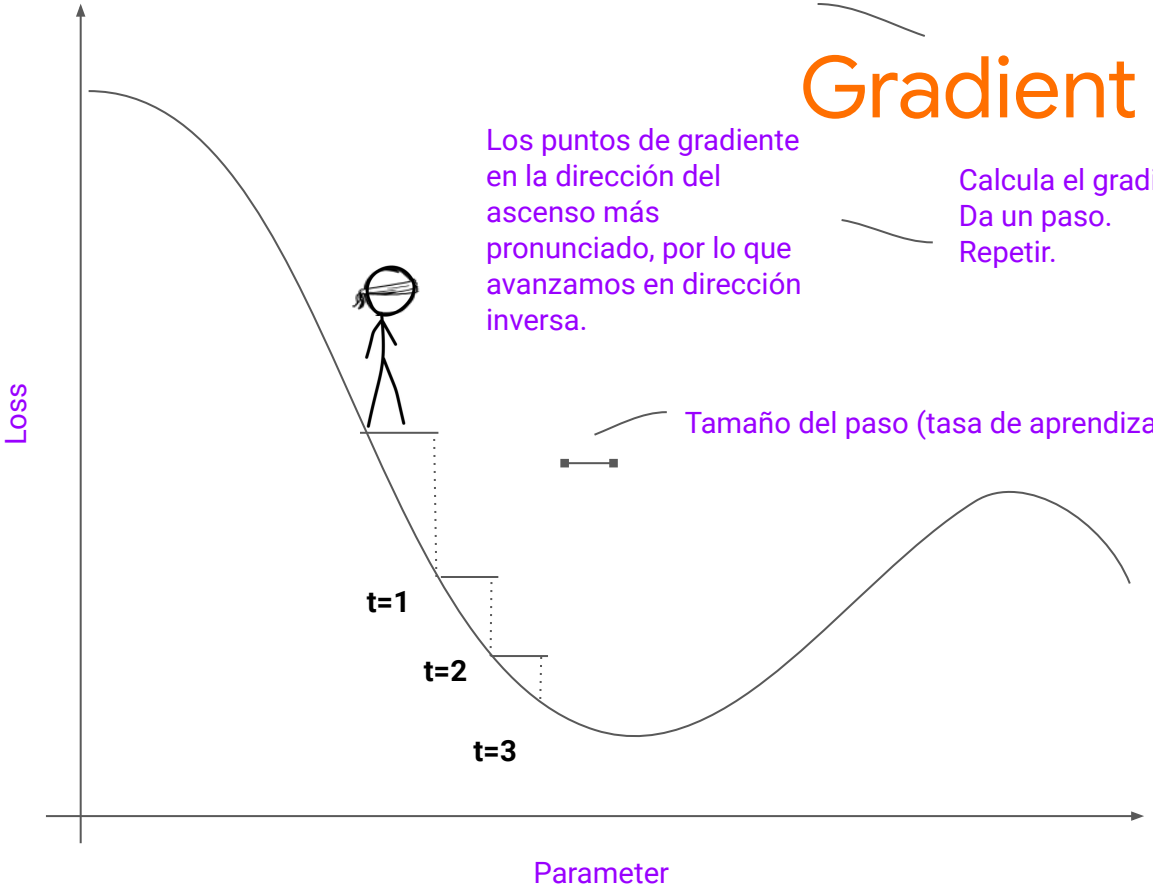
Gradient descent

Un vector de derivadas parciales.

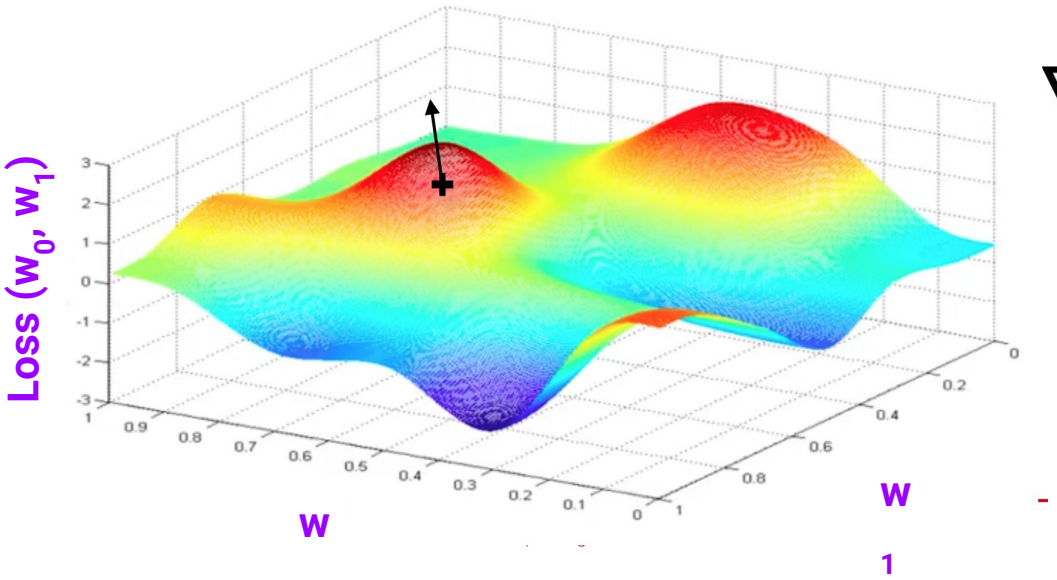
Los puntos de gradiente en la dirección del ascenso más pronunciado, por lo que avanzamos en dirección inversa.

Calcula el gradiente.
Da un paso.
Repetir.

Tamaño del paso (tasa de aprendizaje).



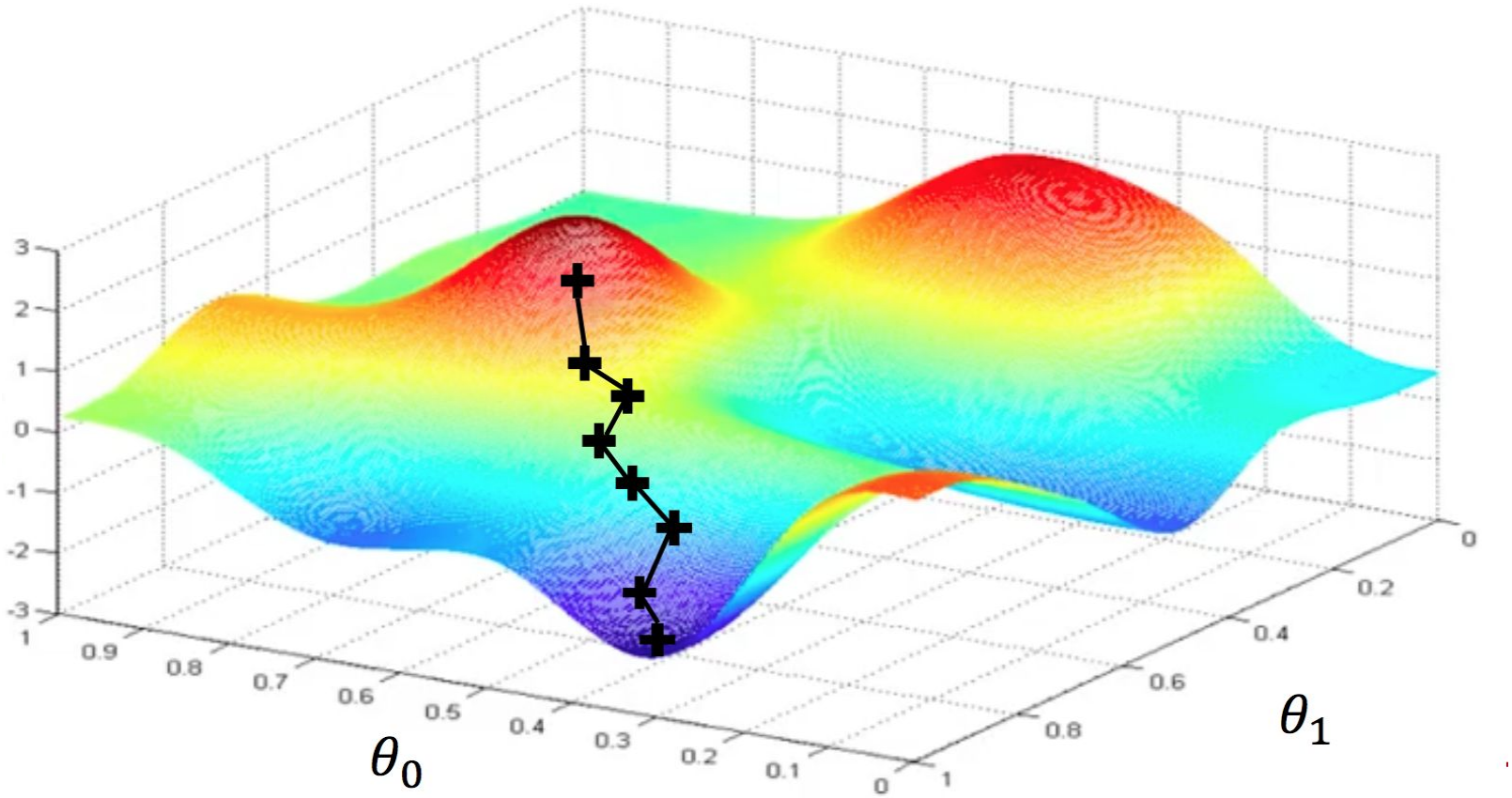
Con más de una variable



$$\nabla_w Loss = \frac{\partial Loss}{\partial w_0}, \frac{\partial Loss}{\partial w_1}$$

El gradiente es un vector de derivadas parciales (la derivada de una función w.r.t. cada variable mientras que las otras se mantienen constantes).

El gradiente apunta en la dirección del ascenso más pronunciado. Por lo general, queremos minimizar una función (como la pérdida), por lo que damos un paso en la dirección opuesta.


3
2
1
0
-1
-2
-3
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
θ_0
1
0.8
0.6
0.4
0.2
0
θ_1

Modelos de entrenamiento con descenso gradiente

Forward pass

- Regresión lineal: $y = mx + b$
- Red neuronal: $f(x) = \text{softmax}(W2(g(W1x)))$

Calculate loss

- Regresión: error al cuadrado.
- Clasificación: entropía cruzada o Matriz de confusión

Backward pass

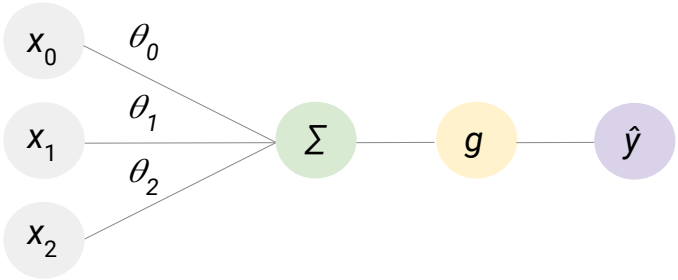
- Backprop: método eficiente para calcular gradientes
- Descenso de gradiente: empuje los parámetros un poco en la dirección opuesta

Pruébalo: regresión lineal

bit.ly/tf-ws1

Bonificación: el bucle de entrenamiento Deep Dream será similar.

A neuron



Inputs **weights** **sum** **activation** **output**

Combinación lineal de entradas y pesos.

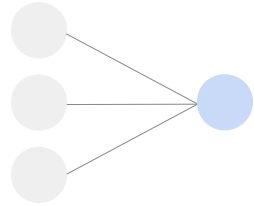
$$\hat{y} = g\left(\sum x_i \theta_i\right)$$

Puede reescribirse como un producto de punto

$$\hat{y} = g\left(x^T \theta\right)$$

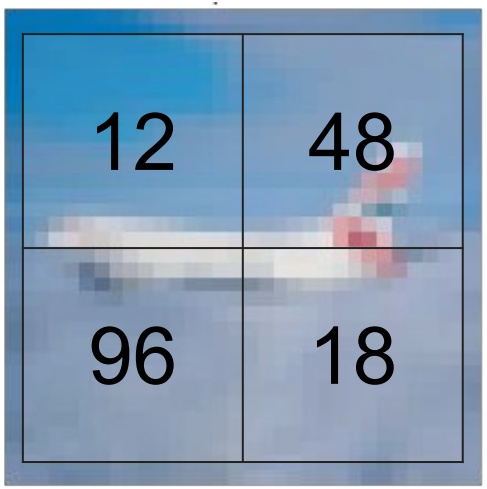
Bias not drawn (you could set x_1 to be a constant input of 1).

Una imagen y una clase.



Entradas múltiples; una salida

Interpretе como "¿con qué **fuerza** cree que esta imagen es un avión?"



1.4	0.5	0.7	1.2

12
48
96
18

+

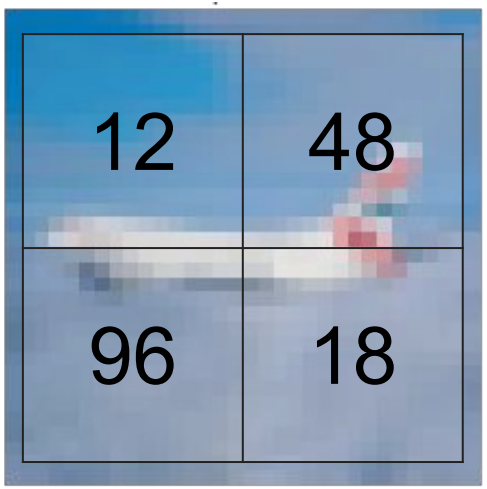
0.5

=

130.1	Plane

w	x	b	Output
Weights	Inputs	Bias	Scores

Una imagen y dos clases.



1.4	0.5	0.7	1.2
-2.0	0.1	0.2	-0.7

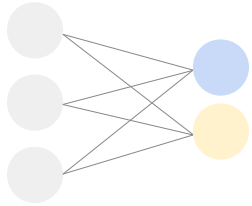
12
48
96
18

+

0.5
1.2

=

130.1	Plane
-11.4	Car



Multiple inputs; multiple outputs

W is now a matrix

W	x	b	Output
Weights	Inputs	Bias	Scores

Dos imágenes y dos clases.

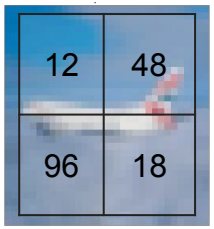


Image 1

Image 2

N x D

1.4	0.5	0.7	1.2
-2.0	0.1	0.2	-0.7
0.2	0.9	-0.2	0.5

W

Weights

D x batch_size

12	4
48	18
96	2
18	96

x

Inputs

+

N x 1

0.5
1.2
0.2

b

Bias

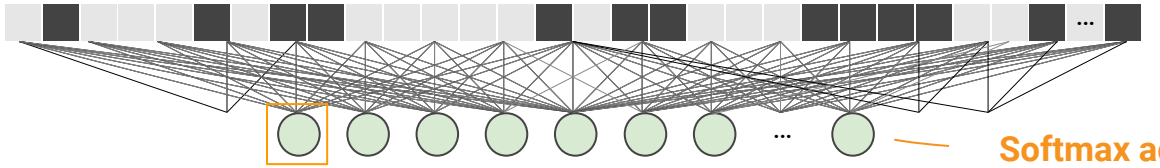
=

N x batch_size

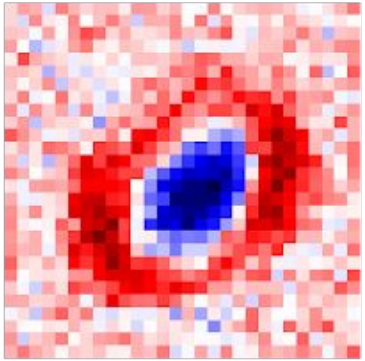
Image 1	Image 2	
130.1	131.7	Plane
-11.4	-71.7	Car
12.8	64.8	Truck

Output

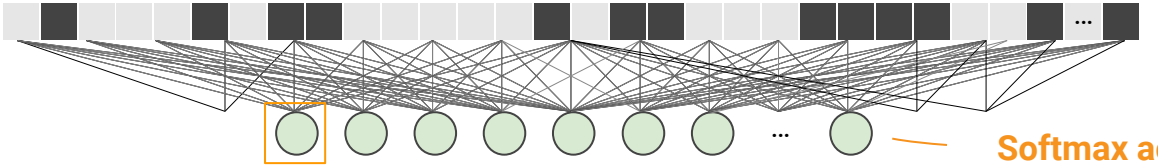
Scores



Después del entrenamiento, seleccione todos los pesos conectados a esta salida.

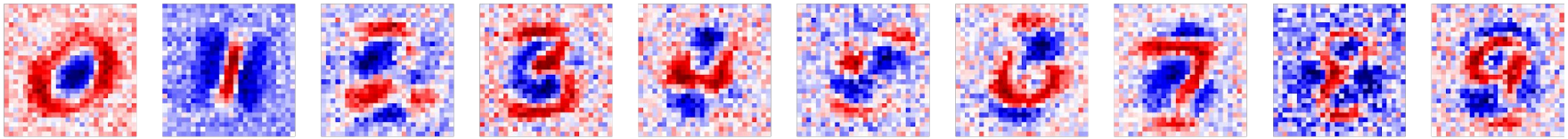


```
model.layers[0].get_weights()

# Your code here
# Select the weights for a single output
# ...

img = weights.reshape(28,28)
plt.imshow(img, cmap = plt.get_cmap('seismic'))
```



Softmax activation

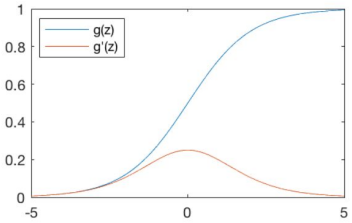
Después del entrenamiento, seleccione todos los pesos conectados a esta salida.



Una red neuronal

$$f = W_2\, g(\, W x\,)$$

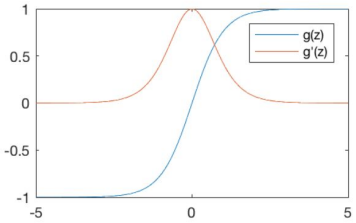
Sigmoid Function



$$g\,(z) = \frac{1}{1 + e^{-z}}$$

$$g'(z) = g(z)(1 - g(z))$$

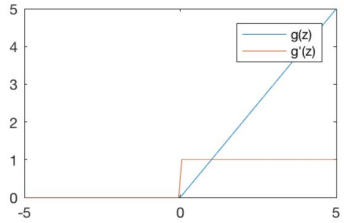
Hyperbolic Tangent



$$g\,(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

$$g'(z) = 1 - g(z)^2$$

Rectified Linear Unit (ReLU)



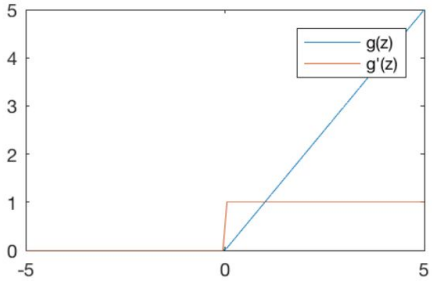
$$g\,(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

ReLU

130.1	Plane
-11.4	Car
12.8	Truck

Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

g(130.1)	Plane
g(-11.4)	Car
g(12.8)	Truck

=

?	Plane
?	Car
?	Truck

$$f = W_2 g(Wx)$$

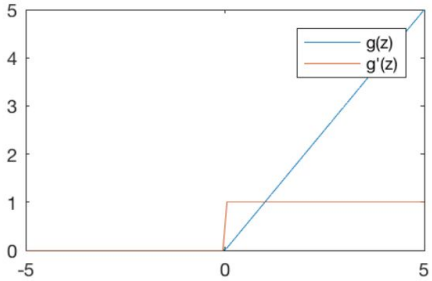
Output

Scores

Aplicado por partes

130.1	Plane
-11.4	Car
12.8	Truck

Rectified Linear Unit (ReLU)



$$g(z) = \max(0, z)$$

$$g'(z) = \begin{cases} 1, & z > 0 \\ 0, & \text{otherwise} \end{cases}$$

g(130.1)	Plane
g(-11.4)	Car
g(12.8)	Truck

=

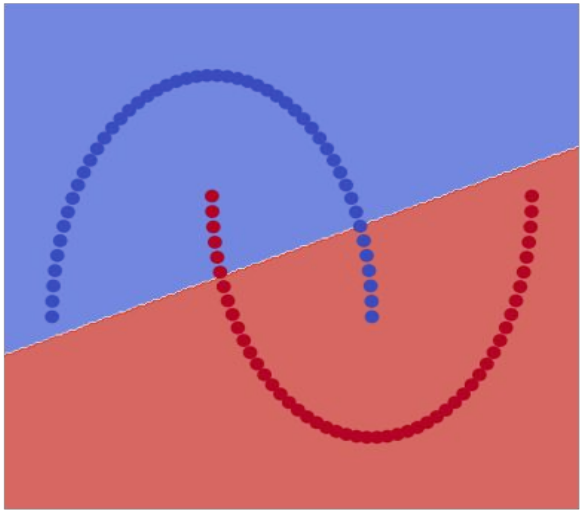
130.1	Plane
0	Car
12.8	Truck

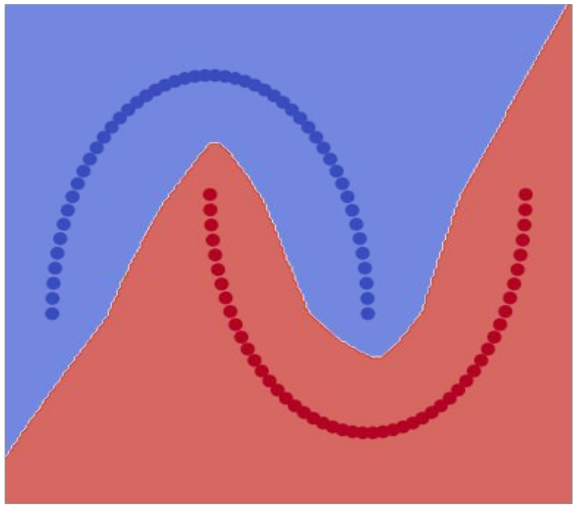
$$f = W_2 g(Wx)$$

Output

Scores

Las funciones de activación introducen no linealidades





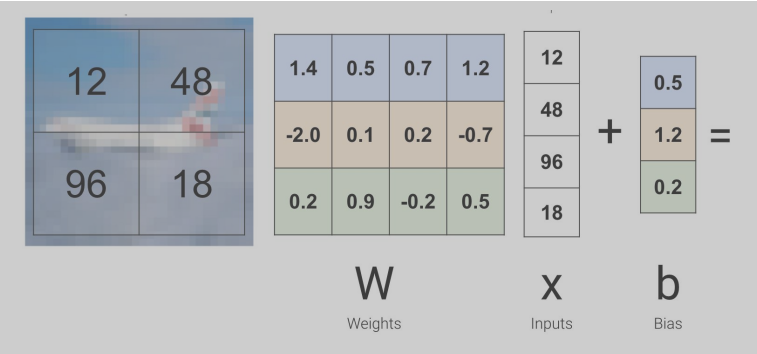
Notes
- Each of our convolutional layers used an activation as well (not shown in previous slides).
- You can make a demo of this in TensorFlow Playground by setting activation = Linear (or none)

Sin activación, muchas capas son equivalentes a una

```
# If you replace 'relu' with 'None', this model ...
model = Sequential([
  Dense(256, activation='relu', input_shape=(2,)),
  Dense(256, activation='relu'),
  Dense(256, activation='relu'),
  Dense(1, activation='sigmoid')
])
```

```
# ... has the same representation power as this one
model = Sequential([Dense(1, activation='sigmoid', input_shape=(2,))])
```

Softmax convierte puntajes en probabilidades



130.1	Plane
-11.4	Car
12.8	Truck

Scores

```
softmax([130.1, -11.4, 12.8])
>>> 0.999, 0.001, 0.001
```

Probabilities

Note: these are 'probability like' numbers (do not go to vegas and bet in this ratio).

La entropía cruzada compara dos distribuciones

Cada ejemplo tiene una etiqueta en un formato hot-one

Este es un pájaro

0	1	2	3	4	5	6	7	8	9
0	0	1	0	0	0	0	0	0	0
0.1	0.2	0.6	0.2	0.0	0.0	0.0	0.0	0.0	0.0

Probabilidades verdaderas

Probabilidades predecidas

Redondeado! La salida de Softmax siempre 0 < x < 1

Pérdida de entropía cruzada para un lote de ejemplos

Verdadera Prob (1 o 0) en nuestro caso!

$$L = -\sum \hat{y}\, ln(y_i)$$

Suma de todo los ejemplos

Prob predecida(entre 0-1)

Ejercicio

bit.ly/ijcai_1-a

Completa el cuaderno para Fashion MNIST

Exercise

bit.ly/ijcai_1-a

Completa el cuaderno para Fashion MNIST

Respuesta: bit.ly/ijcai_1-a_answers

TensorFlow

Convolution

No es un concepto de aprendizaje profundo

```
import scipy
from skimage import color, data
import matplotlib.pyplot as plt
img = data.astronaut()
img = color.rgb2gray(img)
plt.axis('off')
plt.imshow(img, cmap=plt.cm.gray)
```

Ejemplo de convolución

Alguien sabe quien es ella?

-1	-1	-1
-1	8	-1
-1	-1	-1

Notas
Intuición de detección de bordes: el producto de puntos del filtro con una región de la imagen será cero si todos los píxeles alrededor del borde tienen el mismo valor que el centro.

Ejemplo de convolución

Eileen Collins

-1	-1	-1
-1	8	-1
-1	-1	-1

Notas
Intuición de detección de bordes: el producto de puntos del filtro con una región de la imagen será cero si todos los píxeles alrededor del borde tienen el mismo valor que el centro.

Un detector de borde simple

```
kernel = np.array([[-1,-1,-1],
                   [-1,8,-1],
                   [-1,-1,-1]])
result = scipy.signal.convolve2d(img, kernel, 'same')
plt.axis('off')
plt.imshow(result, cmap=plt.cm.gray)
```

Más fácil de ver con sísmica

Eileen Collins

-1	-1	-1
-1	8	-1
-1	-1	-1

Notas
Intuición de detección de bordes: el producto de puntos del filtro con una región de la imagen será cero si todos los píxeles alrededor del borde tienen el mismo valor que el centro.

Ejemplo

2	0	1	1
0	1	0	0
0	0	1	0
0	3	0	0

Una imagen de entrada
(sin relleno)

1	0	1
0	0	0
0	1	0

Un filter
(3x3)

Imagen de salida
(después de convolucionarse
con 1 pasada)

Ejemplo

2	0	1	1
0	1	0	0
0	0	1	0
0	3	0	0

Una imagen de entrada
(sin relleno)

1	0	1
0	0	0
0	1	0

Un filter
(3x3)

3	

Imagen de salida
(después de convolucionarse
con 1 pasada)

2*1 + 0*0 + 1*1 + 0*0 + 1*0 + 0*0 + 0*0 + 0*1 + 1*0

Ejemplo

2	0	1	1
0	1	0	0
0	0	1	0
0	3	0	0

Una imagen de entrada
(sin relleno)

1	0	1
0	0	0
0	1	0

Un filter
(3x3)

3	2

Imagen de salida
(después de convolucionarse
con 1 pasada)

Ejemplo

2	0	1	1
0	1	0	0
0	0	1	0
0	3	0	0

Una imagen de entrada
(sin relleno)

1	0	1
0	0	0
0	1	0

Un filter
(3x3)

3	2
3	

Imagen de salida
(después de convolucionarse
con 1 pasada)

Ejemplo

2	0	1	1
0	1	0	0
0	0	1	0
0	3	0	0

Una imagen de entrada
(sin relleno)

1	0	1
0	0	0
0	1	0

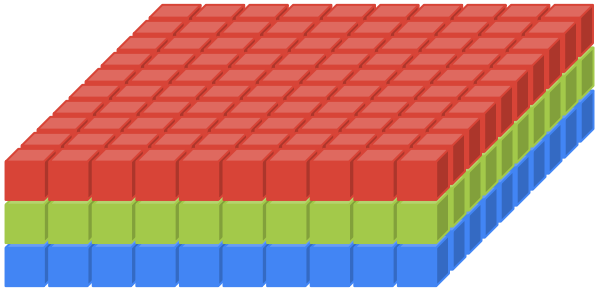
Un filter
(3x3)

3	2
3	1

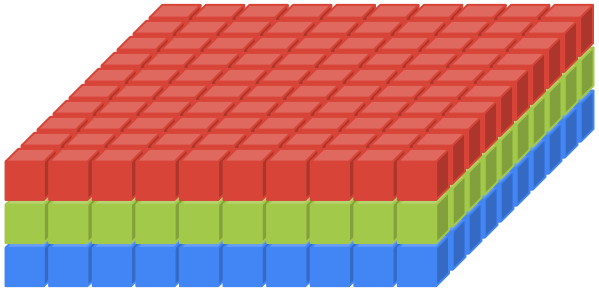
Imagen de salida
(después de convolucionarse
con 1 pasada)

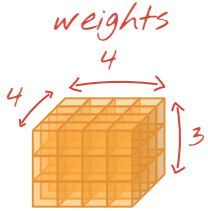
en 3d

```
model = Sequential()

model.add(Conv2D(filters=4,
                 kernel_size=(4,4),
                 input_shape=(10,10,3))
```

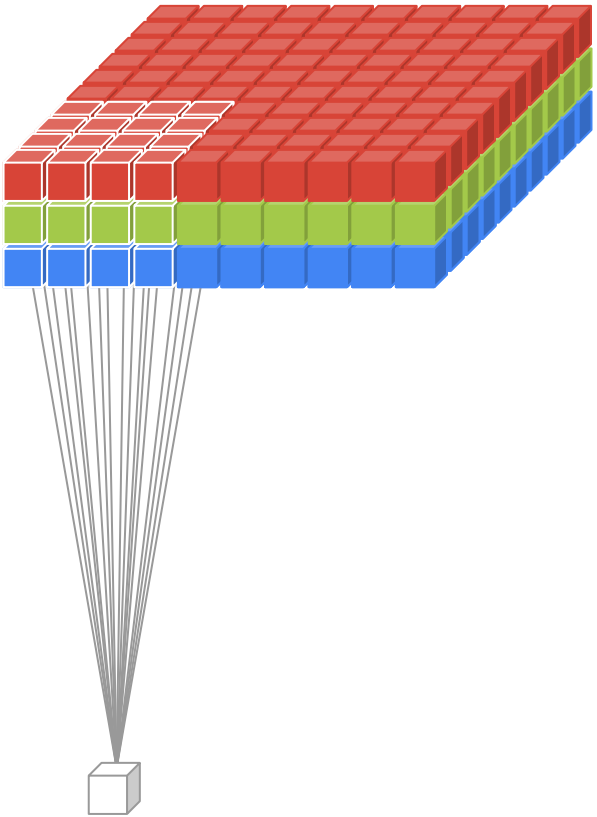


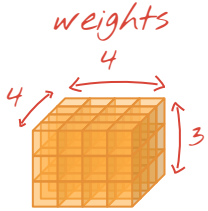
Una imagen RGB como un volumen **3d**. Cada color (o canal) es una capa.

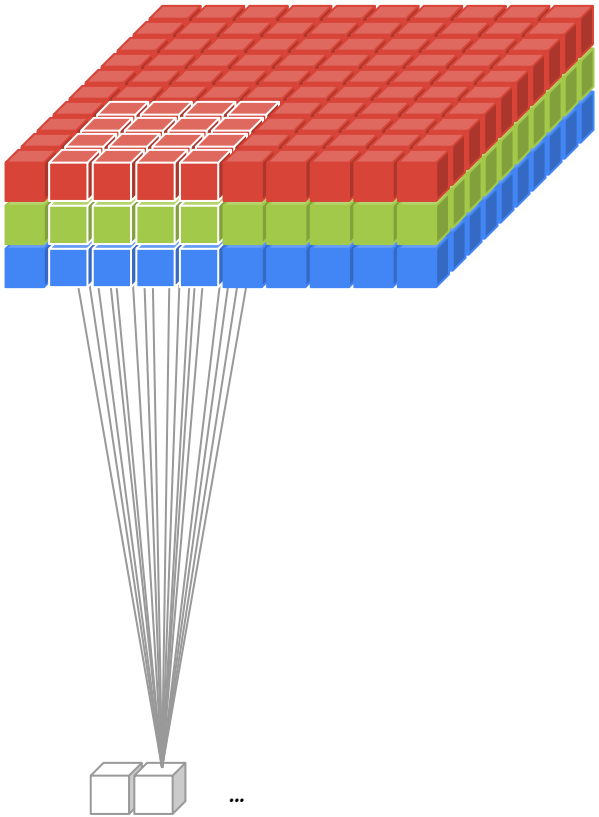


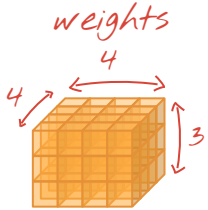


En 3d, nuestros filtros tienen ancho, alto y profundidad.

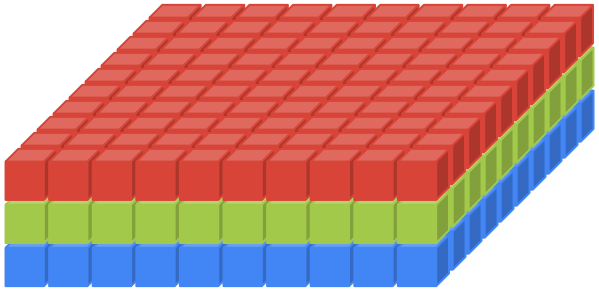


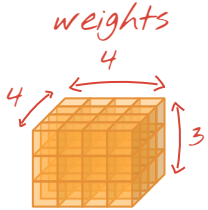

weights
4
4
3



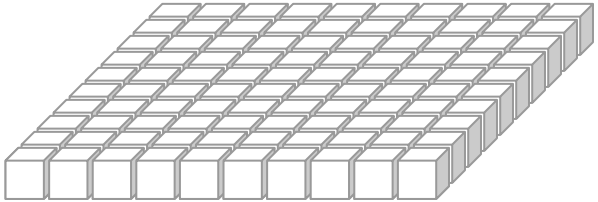


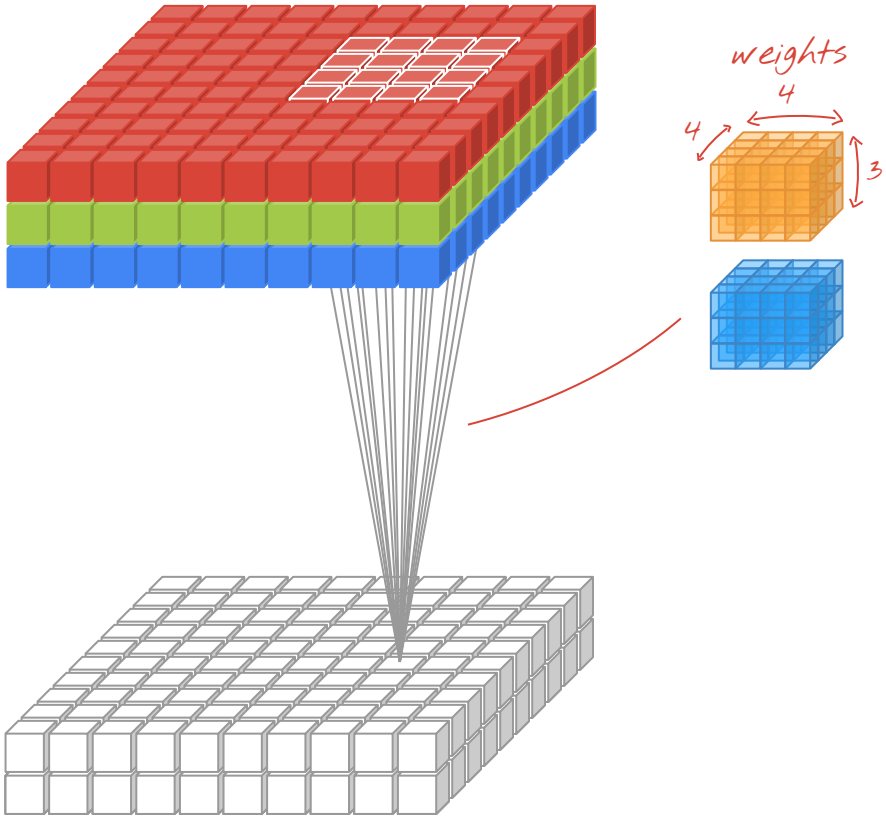
Aplicado de la misma manera que 2d (suma de peso * valor de píxel a medida que se deslizan por la imagen).





Aplicando la convolución sobre el resto de la imagen de entrada.

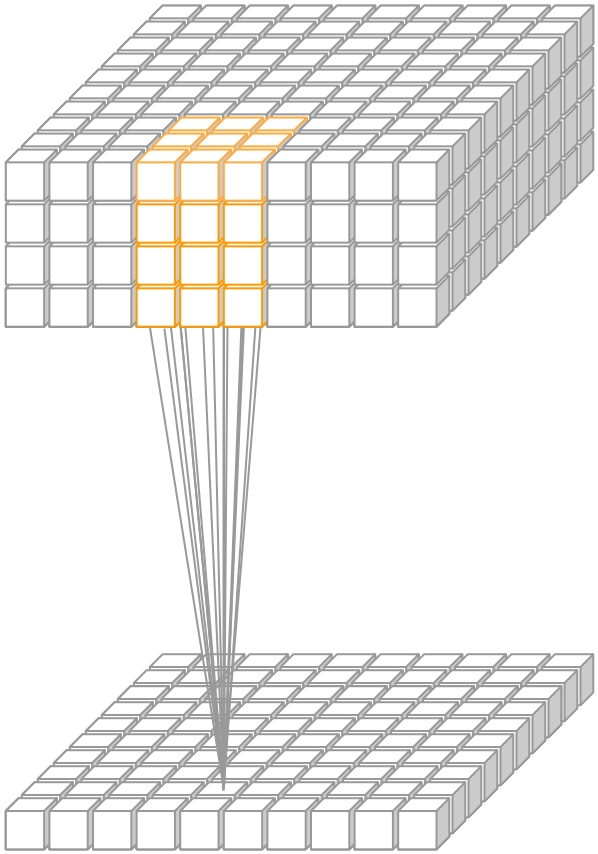


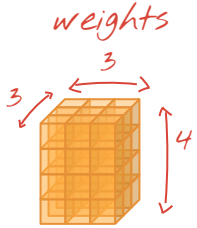


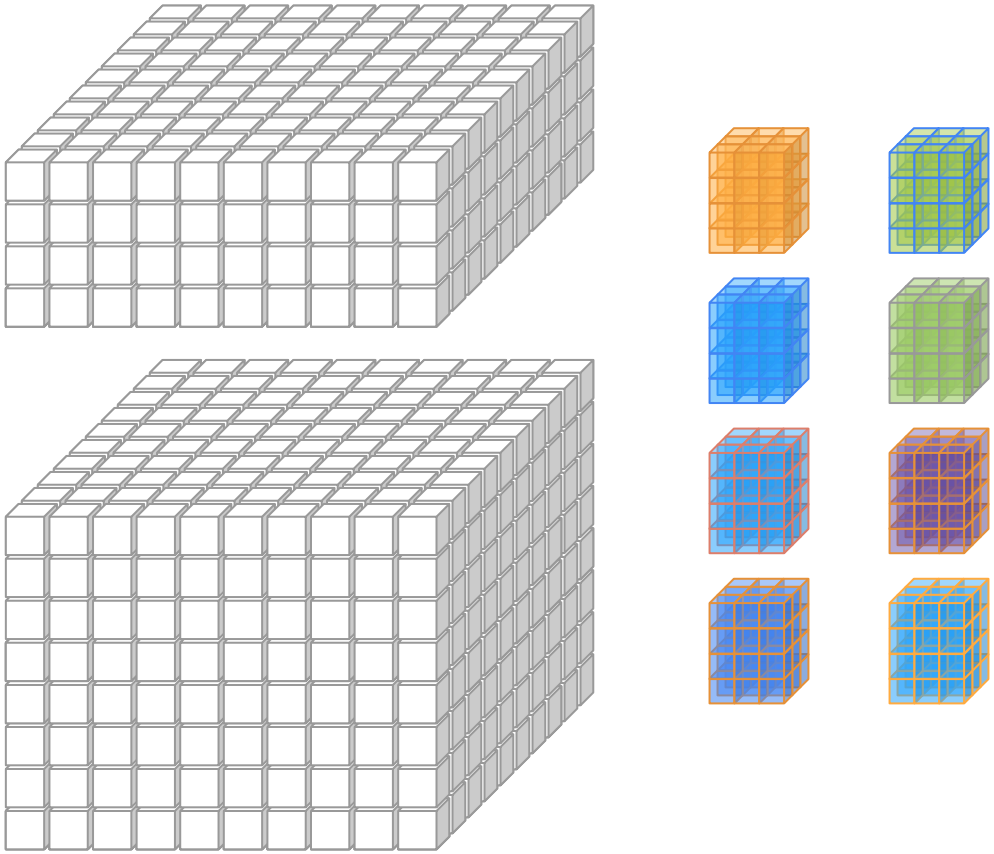
Más filtros, más canales de salida.

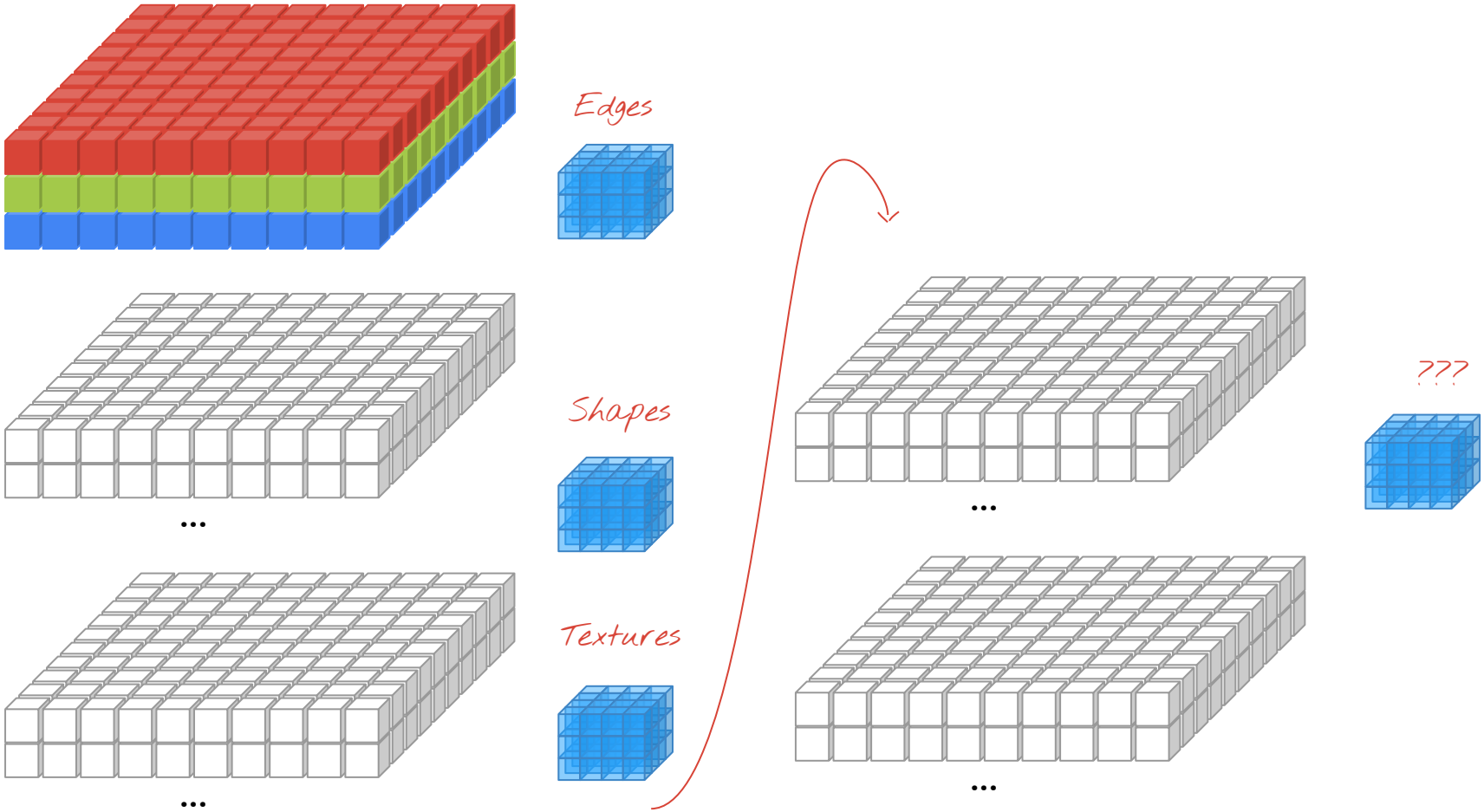
Mas profundo

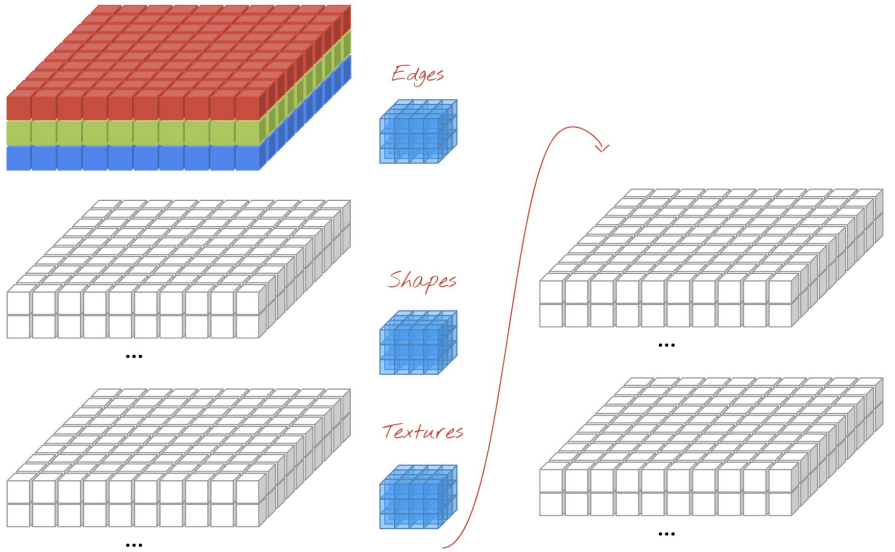
```
model = Sequential()

model.add(Conv2D(filters=4,
                 kernel_size=(4,4),
                 input_shape=(10,10,3))

model.add(Conv2D(filters=8,
                 kernel_size=(3,3))
```

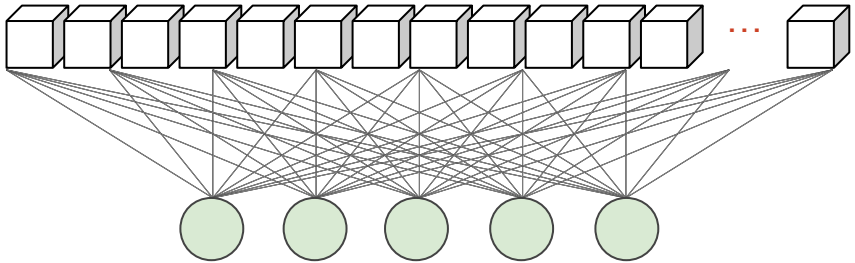



weights
3
3
4




Edges
Shapes
Textures
???


Edges
Shapes
Textures
...
...
...
...


...

Ejercicio

bit.ly/ijcai_1_b

Escriba un CNN desde cero para CIFAR-10.

Respuestas: siguiente diapositiva.

Ref: tensorflow.org/beta/tutorials/images/intro_to_cnns

Ejercicio

bit.ly/ijcai_1b

Escriba un CNN desde cero para CIFAR-10.

Respuesta: bit.ly/ijcai_1_b_answers

Juego 1

¿Te gustaría ser voluntario?

quickdraw.withgoogle.com

Ejemplo: transferencia de aprendizaje

bit.ly/ijcai_2

Aprendizaje de transferencia utilizando una MobileNet pre-entrenada y una capa densa.

Ref: tensorflow.org/beta/tutorials/images/transfer_learning

Ref: tensorflow.org/beta/tutorials/images/hub_with_keras

Ejemplo: transferencia de aprendizaje

bit.ly/ijcai_2

Transfiera el aprendizaje utilizando una MobileNet preentrenada y una capa densa.

Rspuesta: bit.ly/ijcai_2_answers

Deep Dream

Nuevo Tutorial

bit.ly/dream-wip

Segmentación de imagen

Tutorial Reciente

bit.ly/im-seg

Time series forecasting

Recent tutorial

Game 2

Who would like to volunteer?

magenta.tensorflow.org/assets/sketch_rnn_demo/index.html

CycleGAN

Recent tutorial

TensorFlow

Under the hood

Hagamos esto más rápido

```
lstm_cell = tf.keras.layers.LSTMCell(10)


def fn(input, state):
  return lstm_cell(input, state)

input = tf.zeros([10, 10]); state = [tf.zeros([10, 10])] * 2
lstm_cell(input, state); fn(input, state) # warm up

# benchmark
timeit.timeit(lambda: lstm_cell(input, state), number=10) # 0.03
```

Hagamos esto más rápido

```
lstm_cell = tf.keras.layers.LSTMCell(10)

@tf.function
def fn(input, state):
  return lstm_cell(input, state)

input = tf.zeros([10, 10]); state = [tf.zeros([10, 10])] * 2
lstm_cell(input, state); fn(input, state) # warm up

# benchmark
timeit.timeit(lambda: lstm_cell(input, state), number=10) # 0.03
timeit.timeit(lambda: fn(input, state), number=10) # 0.004
```

AutoGraph hace esto posible

```
@tf.function
def f(x):
  while tf.reduce_sum(x) > 1:
    x = tf.tanh(x)
  return x

# you never need to run this (unless curious)
print(tf.autograph.to_code(f))
```

Código generado

```
def tf__f(x):
  def loop_test(x_1):
    with ag__.function_scope('loop_test'):
      return ag__.gt(tf.reduce_sum(x_1), 1)
  def loop_body(x_1):
    with ag__.function_scope('loop_body'):
      with ag__.utils.control_dependency_on_returns(tf.print(x_1)):
        tf_1, x = ag__.utils.alias_tensors(tf, x_1)
        x = tf_1.tanh(x)
        return x,
  x = ag__.while_stmt(loop_test, loop_body, (x,), (tf,))
  return x
```

A lo grande: tf.distribute.Strategy

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Dense(64, input_shape=[10]),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')])

model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

A lo grande: Multi-GPU

```
strategy = tf.distribute.MirroredStrategy()

with strategy.scope():
  model = tf.keras.models.Sequential([
      tf.keras.layers.Dense(64, input_shape=[10]),
      tf.keras.layers.Dense(64, activation='relu'),
      tf.keras.layers.Dense(10, activation='softmax')])
  model.compile(optimizer='adam', loss='categorical_crossentropy',
                metrics=['accuracy'])
```

Aprendiendo más

Últimos tutoriales y guías

- tensorflow.org/beta

Libros

- Hands-on ML with Scikit-Learn, Keras and TensorFlow **(2nd edition)**
- Deep Learning with Python

Detalles

- deeplearningbook.org