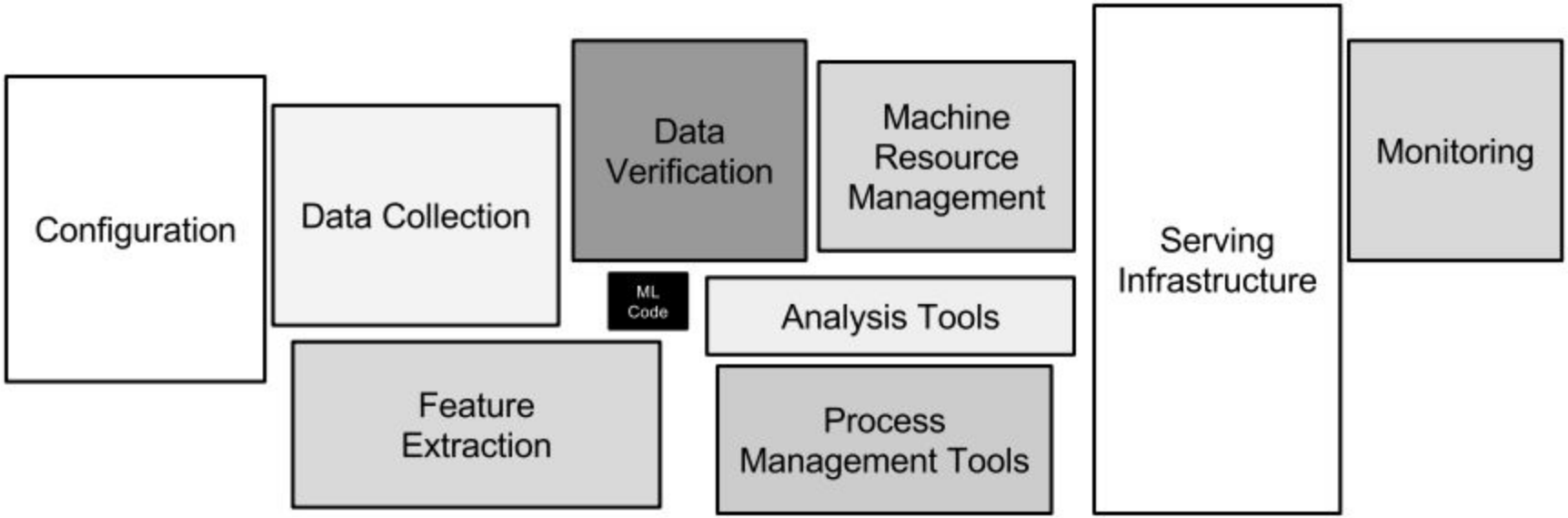
End to End Machine Learning

Cómo llevar tus proyectos de Machine Learning a Producción

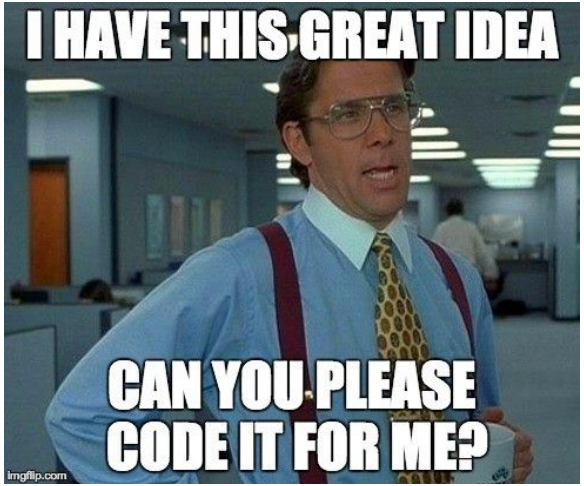
osanseviero@gmail.com

Learners

https://github.com/TheHackerLlama/ml_en_produccion

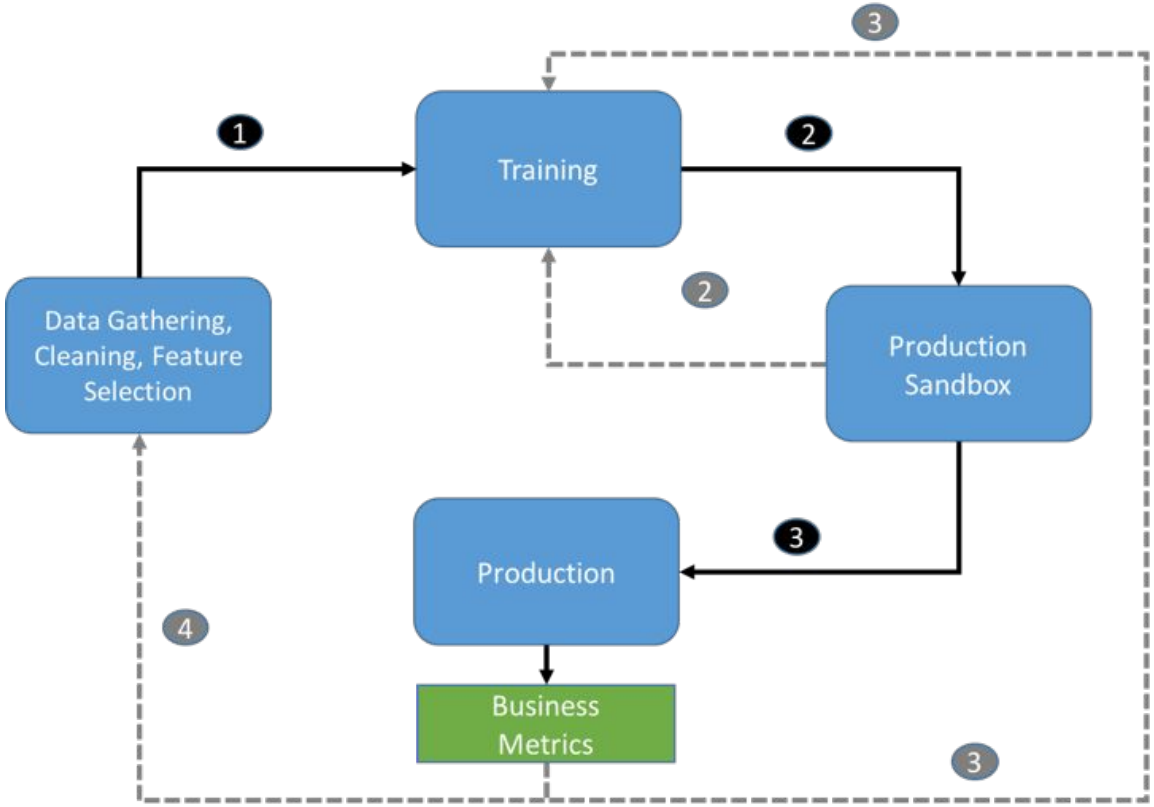

Configuration
Data Collection
Data Verification
Machine Resource Management
Monitoring
Serving Infrastructure
ML Code
Analysis Tools
Feature Extraction
Process Management Tools


dont worry about it if you don't
understand


I HAVE THIS GREAT IDEA
CAN YOU PLEASE
CODE IT FOR ME?
imgflip.com


Hotdog!
Share
No Thanks
Not hotdog!
Share
No Thanks

Iteración # 1


3
1
Training
2
Data Gathering, Cleaning, Feature Selection
2
Production Sandbox
3
Production
4
Business Metrics
3

Definir bien el problema

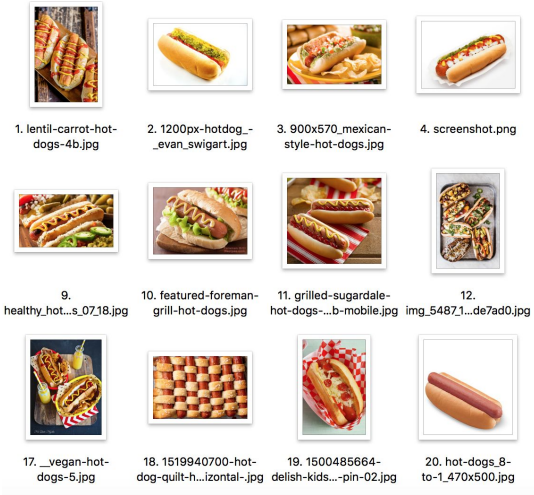
1. Es un proyecto de Aprendizaje Supervisado.
2. Es un clasificador.
3. Queremos que sea en una app y en el browser.

Idea #1. Conseguir datos haciendo scraping

Google Images Download

Python Script for 'searching' and 'downloading' hundreds of Google images to the local hard disk!

```
[osanseviero v0 $ googleimagesdownload -k hot-dog

Item no.: 1 --> Item name = hot-dog
Evaluating...
Starting Download...
Completed Image ====> 1. lentil-carrot-hot-dogs-4b.jpg
Completed Image ====> 2. 1200px-hotdog_-_evan_swigart.jpg
Completed Image ====> 3. 900x570_mexican-style-hot-dogs.jpg
Completed Image ====> 4. screenshot.png
Completed Image ====> 5. 03dba6be-540b-41f6-adea-8be7ab899ef8.jpg
URLError on an image...trying next one... Error: <urlopen error [SSL: CERTIFICAT
E_VERIFY_FAILED] certificate verify failed (_ssl.c:749)>
Completed Image ====> 6. 41qvya7aqql.jpg
Completed Image ====> 7. img_hot_dogs_52505_orig.jpg
Completed Image ====> 8. basic-hot-dogs-600x500.jpg
Completed Image ====> 9. healthy_hotdogs_07_18.jpg
Completed Image ====> 10. featured-foreman-grill-hot-dogs.jpg
```



Idea #1. Conseguir datos haciendo scraping

Google Images Download

Python Script for 'searching' and 'downloading' hundreds of Google images to the local hard disk!

```
osanseviero v0 $ googleimagesdownload -k hamburger

Item no.: 1 --> Item name = hamburger
Evaluating...
Starting Download...
Completed Image ====> 1. big_mac_hamburger.jpg
Completed Image ====> 2. hamburger-hot-dog-58add5f03df78c345bdef6ff.jpg
Completed Image ====> 3. how-to-grill-an-excellent-burger-1-2.jpg
Completed Image ====> 4. beef-hamburgers_7-2-500x375.jpg
Completed Image ====> 5. recipe_papas-favorite-wild-west-hamburger_i166_500x750.
jpg
Completed Image ====> 6. easy-homemade-parmesan-hamburger-buns-recipe.jpg
Completed Image ====> 7. all-american-hamburgers_exps_thjj17_29321_d02_03_5b.jpg
Completed Image ====> 8. hamburger_hamelet_facebook.0.png
Completed Image ====> 9. the-ultimate-hamburger.jpg
Completed Image ====> 10. american-hamburger-foreman-grill.jpg
```

Cargamos el dataset

Conseguir dataset

```
import glob
import  cv2
import numpy as np

hamburger_paths = glob.glob("/Users/osanseviero/Desktop/hot-dog-or-not/v0/downloads/hamburger/*")
hot_dog_paths = glob.glob("/Users/osanseviero/Desktop/hot-dog-or-not/v0/downloads/hot-dog/*")

print('Imágenes de hamburguesas: ', len(hamburger_paths))
print('Imágenes de hot-dogs: ', len(hot_dog_paths))
```

```
Imágenes de hamburguesas:  98
Imágenes de hot-dogs:  91
```

Algunos problemas con las imágenes

1. Son de diferente tamaño

Algunos problemas con las imágenes

1. Son de diferente tamaño

Solución: Hacemos un reshape para que todas sean iguales (64x64)

Algunos problemas con las imágenes

1. Son de diferente tamaño

Solución: Hacemos un reshape para que todas sean iguales (64x64)

2. Los modelos de ML no pueden (en general) utilizar imágenes sin procesarlas antes

Algunos problemas con las imágenes

1. Son de diferente tamaño

Solución: Hacemos un reshape para que todas sean iguales (64x64)

2. Los modelos de ML no pueden (en general) utilizar imágenes sin procesarlas antes

Solución: Hacemos un flattening.

Algunos problemas con las imágenes

Flattening y agregando label

In [63]:
```
def flatten(imagePath, label):
    x_data = np.array([])
    for i in range(len(imagePath)):
        resized = cv2.resize(cv2.imread(imagePath[i]), dsize=(64, 64), interpolation=cv2.INTER_CUBIC)

        x_data = np.append(x_data, np.append(resized, [label]))
    pixels = x_data.flatten().reshape(len(imagePath), 64*64*3+1)

    return pixels
```

In [64]:
```
hamburger_data = flatten(hamburger_paths, 0)
hot_dog_data = flatten(hot_dog_paths, 1)
```

In [67]:
```
print(hamburger_data.shape)
print(hot_dog_data.shape)
print(hamburger_data[0])
```

```
(98, 12289)
(91, 12289)
[255. 255. 255. ... 255. 255.   0.]
```

Resize

Flatten

Originalmente: 98 imágenes de diferentes tamaños y RGB (3 dimensiones)
Ahora: 98 listas de 12,289 (64*64*3 + 1 por el label)

Modelo versión 0

Modelo

```
from sklearn import svm, metrics

classifier = svm.SVC(gamma=0.001)

classifier.fit(x, label)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma=0.001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

Modelo versión 0

Métricas

```
: expected = label
  predicted = classifier.predict(x)
```

```
: print("Classification report for classifier %s:\n%s\n"
        % (classifier, metrics.classification_report(expected, predicted)))
```

```
Classification report for classifier SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape=None, degree=3, gamma=0.001, kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False):
             precision    recall  f1-score   support

        0.0       1.00      1.00      1.00        98
        1.0       1.00      1.00      1.00        91

avg / total       1.00      1.00      1.00       189
```

Lanzamos el producto

Problemas

1. Sólo sabe distinguir entre algunos hot dogs y hamburguesas.
2. Tuvimos que aplanar la imagen
3. Evaluamos el modelo con los mismos datos que utilizamos para entrenar.

Iteración # 2

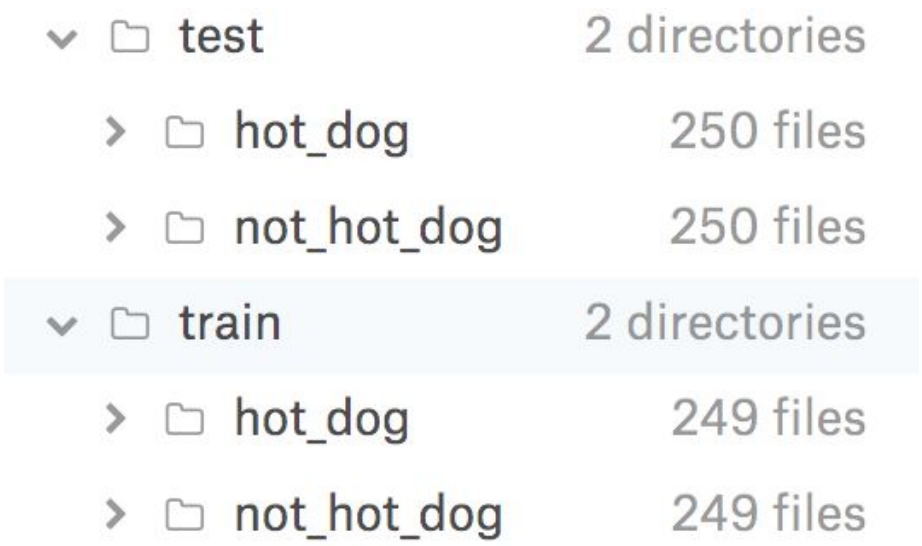
Fix #1. Conseguir mejores datos

The Food-101 Data Set

We introduce a challenging data set of 101 food categories, with 101'000 images. For each class, 250 manually reviewed test images are provided as well as 750 training images. On purpose, the training images were not cleaned, and thus still contain some amount of noise. This comes mostly in the form of intense colors and sometimes wrong labels. All images were rescaled to have a maximum side length of 512 pixels.

Download (5 GB)

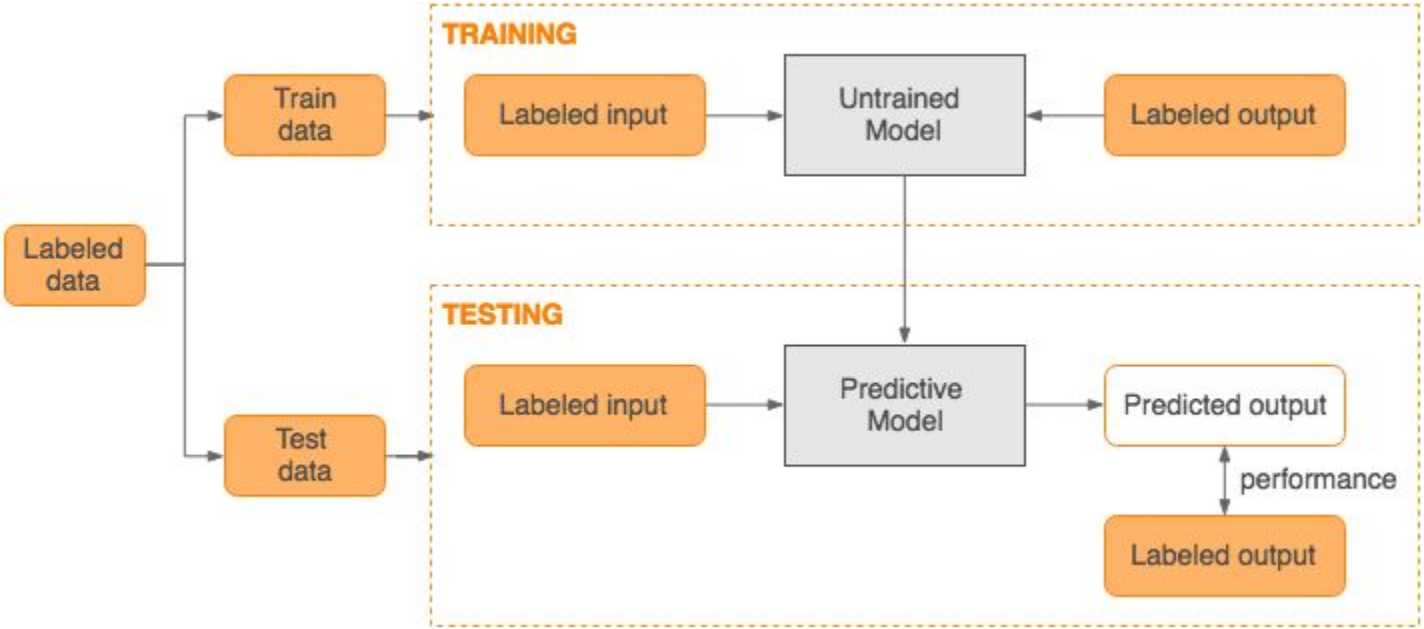
Fix #1. Conseguir mejores datos



kaggle

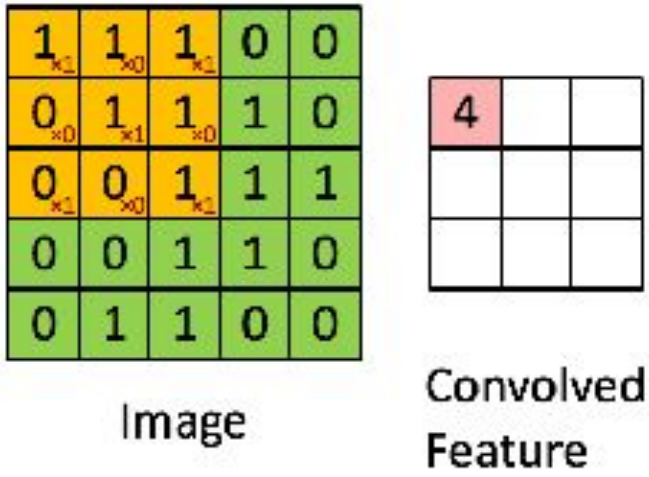
https://www.kaggle.com/dansbecker/hot-dog-not-hot-dog

Fix #2. Split de dataset



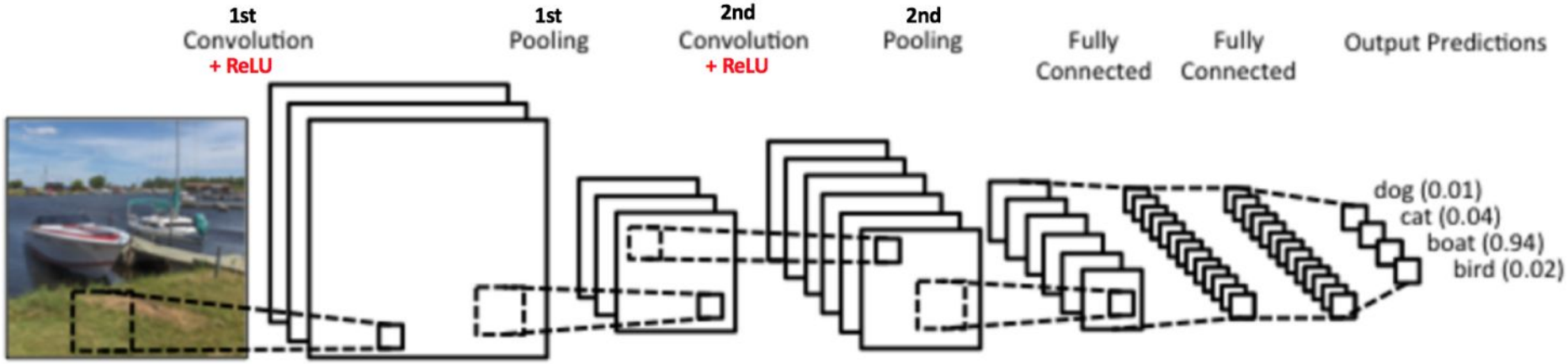
Fix #3. Desarrollar un mejor modelo

- Los algoritmos clásicos necesitan aplanar la imagen.
- Las Redes Neuronales Convolucionales (CNNs) trabajan particularmente bien con las imágenes.



Fix #3. Desarrollar un mejor modelo

- Los algoritmos clásicos necesitan aplanar la imagen.
- Las Redes Neuronales Convolucionales (CNNs) trabajan particularmente bien con las imágenes.



Fix #3. Desarrollar un mejor modelo

- Los algoritmos clásicos necesitan aplanar la imagen.
- Las Redes Neuronales Convolucionales (CNNs) trabajan particularmente bien con las imágenes.

Fix #3. Desarrollar un mejor modelo

Keras

```
model = Sequential()
model.add(Conv2D(8, kernel_size=(3,3),
            activation='relu',
            input_shape=(img_size, img_size, 3)))
Dropout(.5)
model.add(Conv2D(8, kernel_size=(3,3),
            activation='relu'))
Dropout(.5)

model.add(Conv2D(8, kernel_size=(3,3),
                 activation='relu'))
model.add(Conv2D(8, kernel_size=(3,3),
                 activation='relu'))
model.add(Conv2D(8, kernel_size=(3,3),
                 activation='relu'))
model.add(Conv2D(8, kernel_size=(3,3),
                 activation='relu'))

model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
```

Layer (type)	Output Shape	Param #
conv2d_13 (Conv2D)	(None, 222, 222, 8)	224
conv2d_14 (Conv2D)	(None, 220, 220, 8)	584
conv2d_15 (Conv2D)	(None, 218, 218, 8)	584
conv2d_16 (Conv2D)	(None, 216, 216, 8)	584
conv2d_17 (Conv2D)	(None, 214, 214, 8)	584
conv2d_18 (Conv2D)	(None, 212, 212, 8)	584
flatten_3 (Flatten)	(None, 359552)	0
dense_5 (Dense)	(None, 128)	46022784
dense_6 (Dense)	(None, 2)	258

Total params: 46,026,186
Trainable params: 46,026,186
Non-trainable params: 0

Empezamos a entrenar...

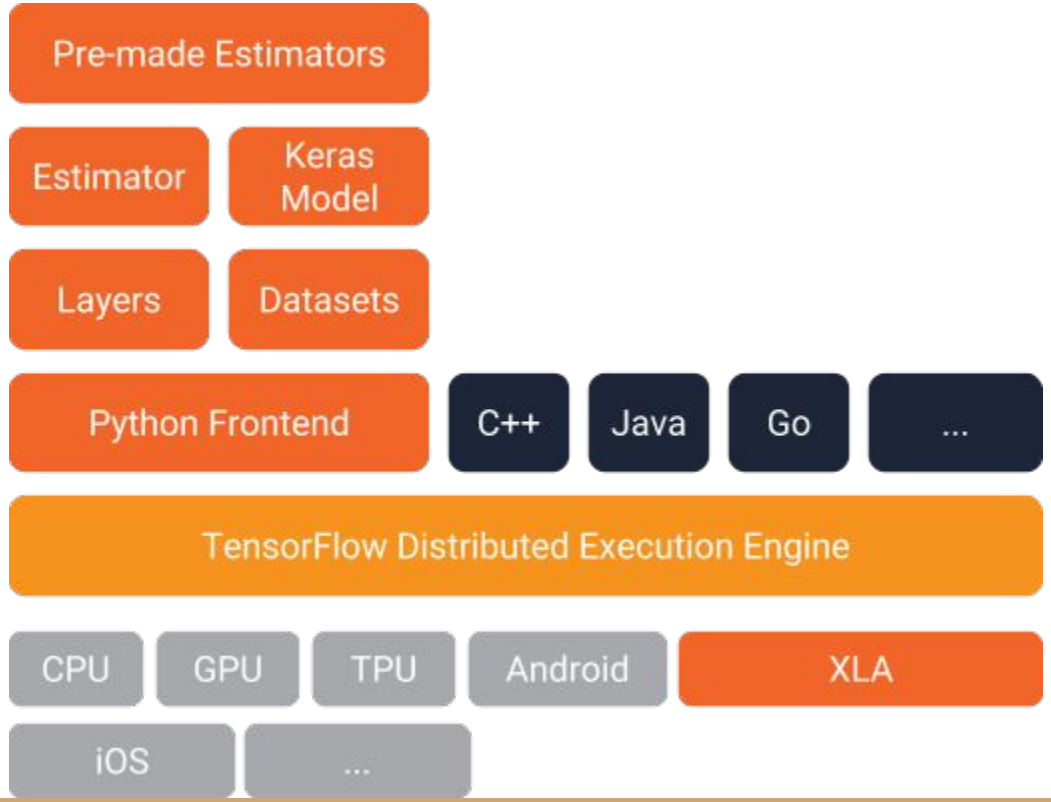
```
Epoch 1/5
498/498 [==============================] - 111s 224ms/step - loss: 0.7820 - acc: 0.4679
Epoch 2/5
498/498 [==============================] - 81s 163ms/step - loss: 0.6939 - acc: 0.5000
Epoch 3/5
498/498 [==============================] - 88s 177ms/step - loss: 0.6907 - acc: 0.6044
Epoch 4/5
498/498 [==============================] - 91s 183ms/step - loss: 0.6715 - acc: 0.5863
Epoch 5/5
192/498 [==========>...................] - ETA: 49s - loss: 0.5625 - acc: 0.7708
```

Empezamos a entrenar y...muy lento

```
Epoch 1/5
498/498 [==============
Epoch 2/5
498/498 [==============
Epoch 3/5
498/498 [==============
Epoch 4/5
498/498 [==============
Epoch 5/5
192/498 [==========>....
```


```
s: 0.7820 - acc: 0.4679
: 0.6939 - acc: 0.5000
: 0.6907 - acc: 0.6044
: 0.6715 - acc: 0.5863
25 - acc: 0.7708
```

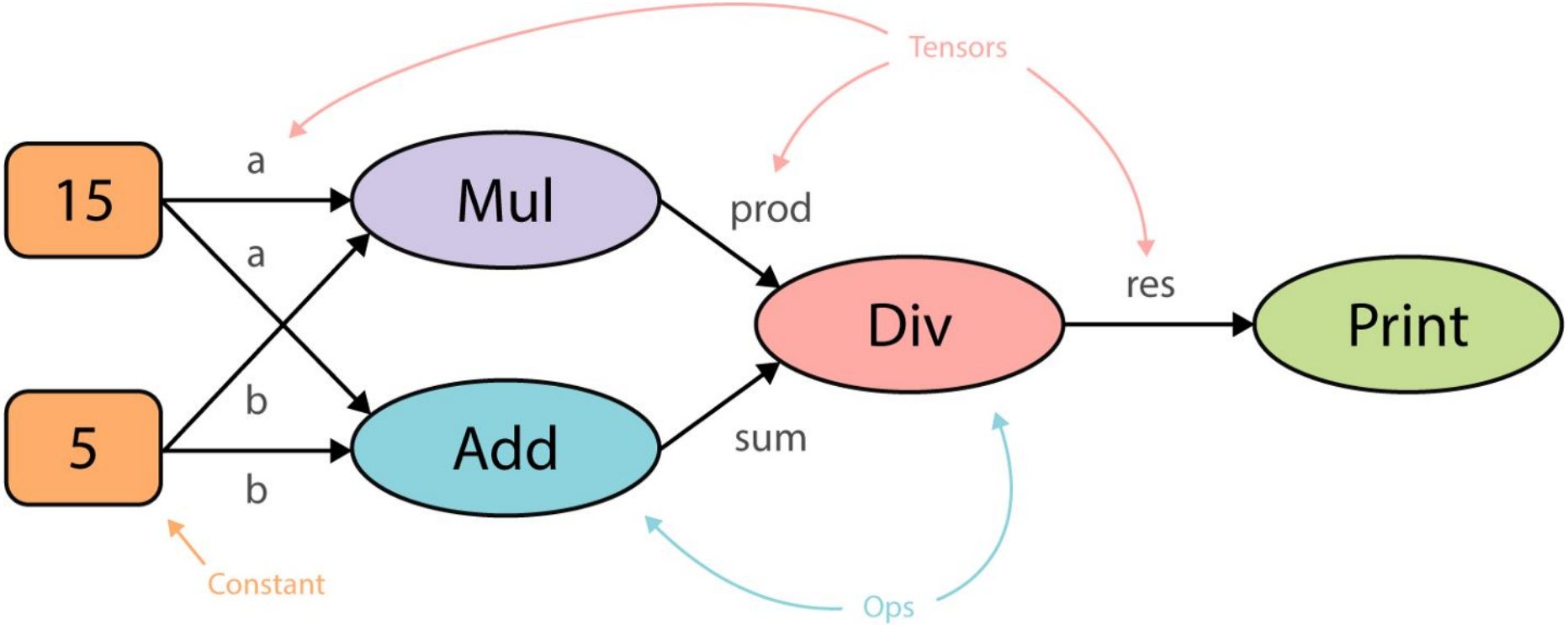
TensorFlow


Pre-made Estimators
Estimator
Keras Model
Layers
Datasets
Python Frontend
C++
Java
Go
...
TensorFlow Distributed Execution Engine
CPU
GPU
TPU
Android
XLA
iOS
...

Paralelización

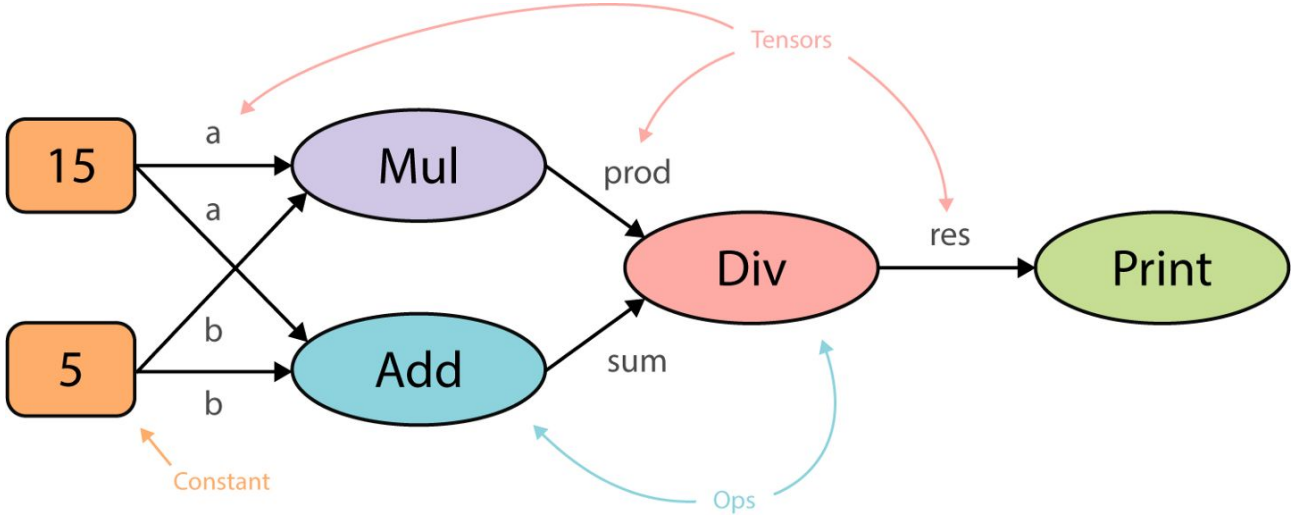
```
1 a = 15
2 b = 5
3 prod = a * b
4 sum = a + b
5 res = prod / sum
6 print(res)
```

Paralelización



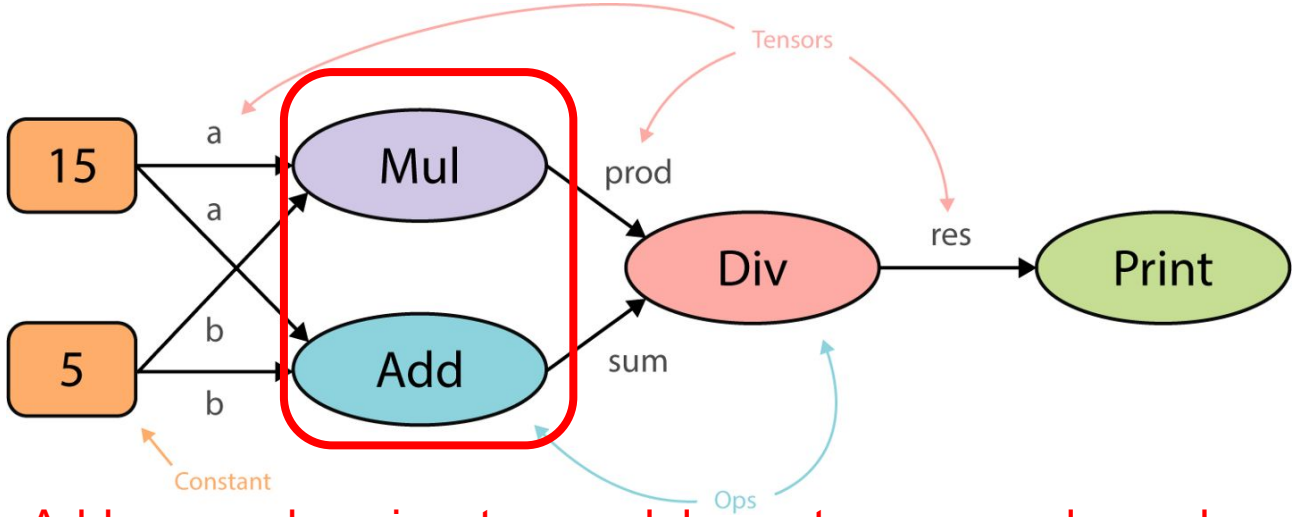
Ventajas de grafos

- Manera abstracta de describir un programa y sus computaciones.
- **Especifica dependencias entre operaciones.**
  - Es decir, cómo fluyen los datos a través de estas.



Ventajas de grafos

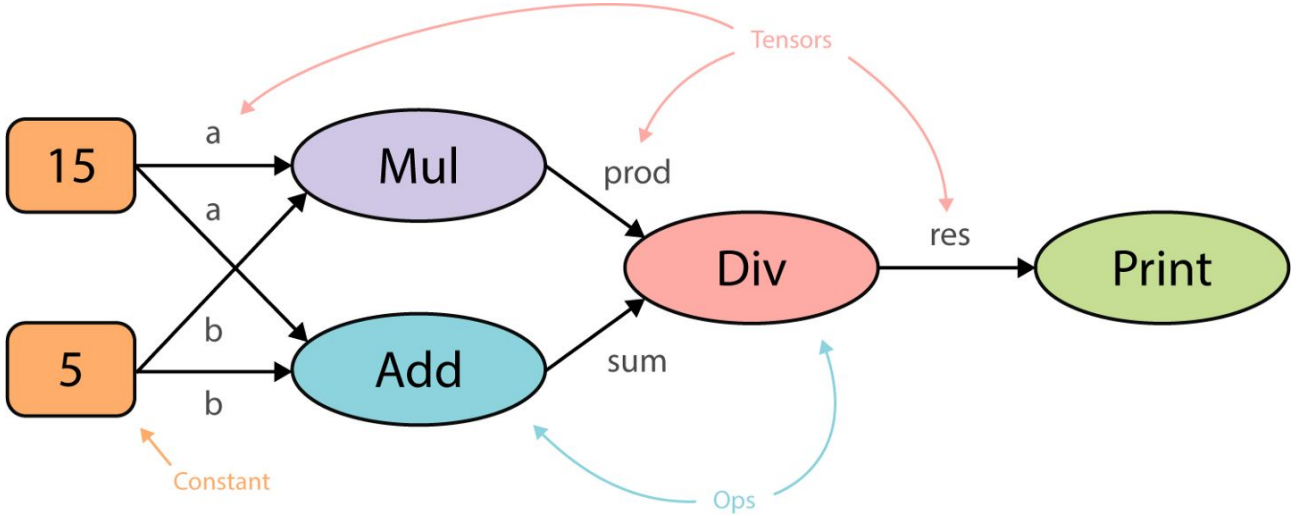
- Manera abstracta de describir un programa y sus computaciones.
- Especifica dependencias entre operaciones.
  - Es decir, cómo fluyen los datos a través de estas.



Mul y Add se pueden ejecutar paralelamente pues no dependen entre ellas.

Ventajas de grafos

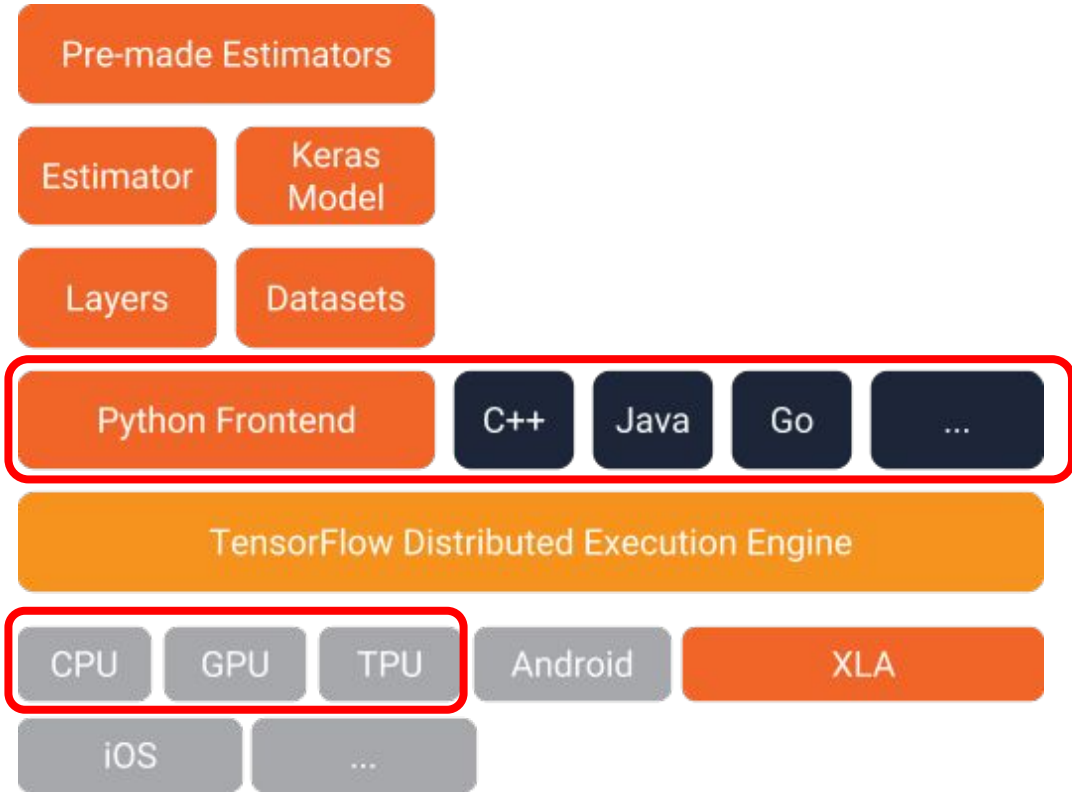
- La topología del grafo nos permite ejecutarlas de manera eficiente
  - Usando varios GPUs
  - Distribuir en muchas máquinas



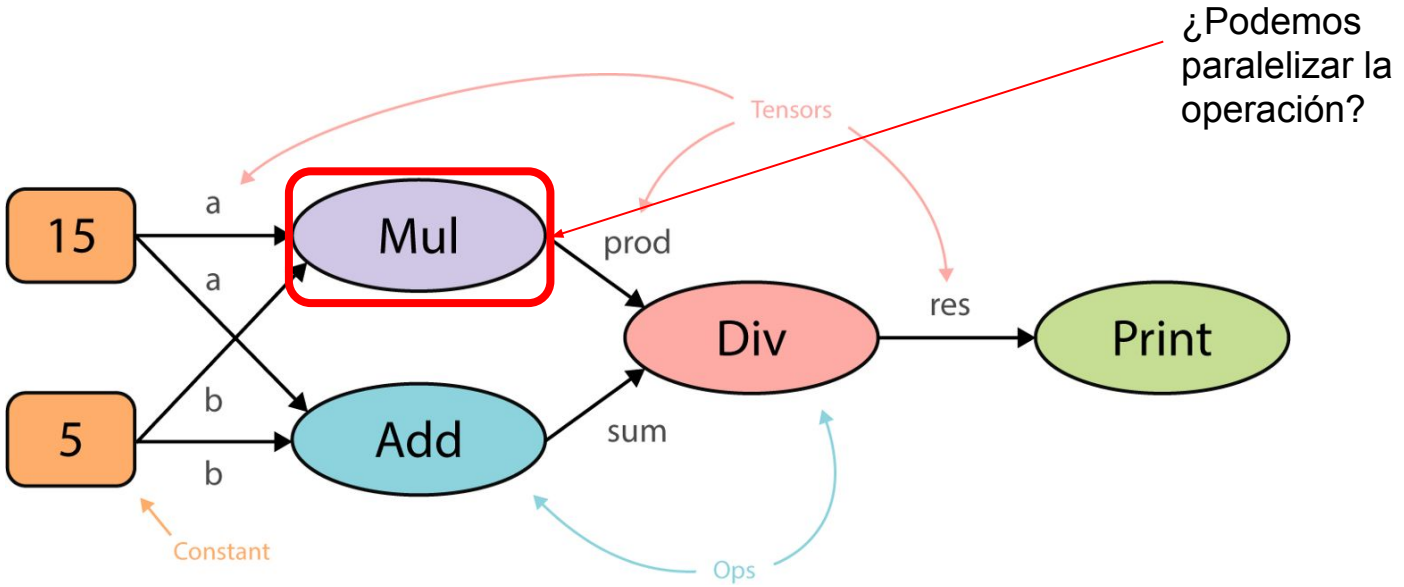
Ventajas de grafos

- Portabilidad
  - Grafos son una representación que no depende del lenguaje

- Podemos construir el grafo en Python, guardar el modelo y restaurarlo en C++ si queremos eficiencia.



¿Cómo funcionan las operaciones en GPU?



¿Cómo funcionan las operaciones en GPU?

- El CPU y el GPU tienen 2 implementaciones: C++ y CUDA
- Ejemplo: tf.transpose

```
x = tf.constant([[1, 2, 3], [4, 5, 6]])
tf.transpose(x)  # [[1, 4]
                 #  [2, 5]
                 #  [3, 6]]
```

¿Cómo funcionan las operaciones en GPU?

- El CPU y el GPU tienen 2 implementaciones: C++ y CUDA
- Ejemplo: tf.transpose

```
REGISTER_OP("Transpose")
    .Input("x: T")
    .Input("perm: Tperm")
    .Output("y: T")
    .Attr("T: type")
    .Attr("Tperm: {int32, int64} = DT_INT32")
    .SetShapeFn(TransposeShapeFn);
```

array_ops.cc

¿Cómo funcionan las operaciones en GPU?

- El CPU y el GPU tienen 2 implementaciones: C++ y CUDA
- Ejemplo: tf.transpose

```
class TransposeGpuOp : public TransposeOp { ... }
class TransposeCpuOp : public TransposeOp { ... }
```

transpose_op.h

¿Cómo funcionan las operaciones en GPU?

- El CPU y el GPU tienen 2 implementaciones: C++ y CUDA
- Ejemplo: tf.transpose

```cpp
template <typename T, bool conjugate>
void TransposeSimple(const CPUDevice& device, const Tensor& in,
                     const gtl::ArraySlice<int32> perm, Tensor* out) {
  const int ndims = in.dims();
  gtl::InlinedVector<int64, 8> in_strides = ComputeStride<int64>(in.shape());
  gtl::InlinedVector<int64, 8> out_strides = ComputeStride<int64>(out->shape());
  const T* p = reinterpret_cast<const T*>(in.tensor_data().data());
  T* q = reinterpret_cast<T*>(const_cast<char*>((out->tensor_data().data())));
  auto transpose_fn = [=, &in_strides, &out_strides, &perm](int64 begin,
                                                            int64 end) {
    for (int64 o_idx = begin; o_idx < end; ++o_idx) {
      int64 i_idx = 0;
      int64 t = o_idx;
      for (int i = 0; i < ndims; ++i) {
        const int64 ratio = t / out_strides[i];
        t -= ratio * out_strides[i];
        i_idx += ratio * in_strides[perm[i]];
      }
      if (conjugate) {
        q[o_idx] = Eigen::numext::conj(p[i_idx]);
      } else {
        q[o_idx] = p[i_idx];
      }
    }
  };
  double cycles_per_element =
      (conjugate ? 1 : 0) + ndims * (Eigen::TensorOpCost::DivCost<int64>() +
                                     2 * Eigen::TensorOpCost::MulCost<int64>() +
                                     2 * Eigen::TensorOpCost::AddCost<int64>());
  Eigen::TensorOpCost cost(/*bytes_loaded=*/sizeof(T),
                           /*bytes_stored=*/sizeof(T), cycles_per_element);
  device.parallelFor(in.NumElements(), cost, std::move(transpose_fn));
}
```

```cpp
template <typename T, bool conjugate>
__global__ void TransposeKernel(int nthreads, const T* src, const int32* buf,
                                const int32 ndims, T* dst) {
  const int32* in_strides = buf;
  const int32* out_strides = buf + ndims;
  const int32* perm = buf + ndims * 2;
  CUDA_1D_KERNEL_LOOP(o_idx, nthreads) {
    int32 i_idx = 0;
    int32 t = o_idx;
    for (int32 i = 0; i < ndims; ++i) {
      const int32 ratio = t / out_strides[i];
      t -= ratio * out_strides[i];
      i_idx += ratio * in_strides[perm[i]];
    }
    if (conjugate) {
      dst[o_idx] = Eigen::numext::conj(ldg(src + i_idx));
    } else {
      dst[o_idx] = ldg(src + i_idx);
    }
  }
}
```

¿Y…hay GPU?

```
import tensorflow as tf

print("Hay GPU disponible: "),
print(tf.test.is_gpu_available())
```

```
Hay GPU disponible:
False
```

¿Y…hay GPU?

```
import tensorflow as tf

print("Hay GPU disponible: "),
print(tf.test.is_gpu_available())
```

```
Hay GPU disponible:
False
```

	Costo de hardware	**Google Cloud (por hora)**	**Amazon Web Services (por hora)** **Precios bajos pagando 3 años**
NVIDIA K80	$5000-$7000	$0.45	$0.425-$0.900
NVIDIA P100	$6000-$9500	$1.46	NA
NVIDIA V100	$9000-$11000	$2.48	$1.05-$3.06
TPU	NA	$1.35-$4.50	NA

Google Colaboratory

Notebook settings

Runtime type

Python 3

Hardware accelerator

GPU

Omit code cell output when saving this notebook

CANCEL SAVE

GPU compartido entre varias máquinas virtuales.

```
[3]  import tensorflow as tf

     print("Hay GPU disponible: "),
     print(tf.test.is_gpu_available())
```

```
Hay GPU disponible:
True
```

Entrenamiento con GPU

```
Epoch 1/5
498/498 [==============================] - 6s 13ms/step - loss: 0.9946 - acc: 0.5141
Epoch 2/5
498/498 [==============================] - 4s 7ms/step - loss: 0.6958 - acc: 0.5040
Epoch 3/5
498/498 [==============================] - 4s 7ms/step - loss: 0.6904 - acc: 0.5703
Epoch 4/5
498/498 [==============================] - 4s 7ms/step - loss: 0.6733 - acc: 0.6446
Epoch 5/5
498/498 [==============================] - 4s 7ms/step - loss: 0.5421 - acc: 0.7510
<tensorflow.python.keras.callbacks.History at 0x7f44d34d4048>
```

	Loss	**Accuracy**
Training set	0.5421	0.751
Testing set	0.733	0.56

Problemas

1. El resultado no es particularmente bueno en prueba.
2. No estamos muy seguros de lanzar el producto.

Iteración # 3

- Vamos a intentar mejorar el modelo

Iteración # 3

- Vamos a intentar mejorar el modelo

Layer (type)	Output Shape	Param #
conv2d_36 (Conv2D)	(None, 222, 222, 8)	224
conv2d_37 (Conv2D)	(None, 220, 220, 8)	584
conv2d_38 (Conv2D)	(None, 218, 218, 8)	584
conv2d_39 (Conv2D)	(None, 216, 216, 8)	584
conv2d_40 (Conv2D)	(None, 214, 214, 8)	584
conv2d_41 (Conv2D)	(None, 212, 212, 8)	584
conv2d_42 (Conv2D)	(None, 210, 210, 8)	584
flatten_6 (Flatten)	(None, 352800)	0
dense_12 (Dense)	(None, 500)	176400500
dense_13 (Dense)	(None, 128)	64128
dense_14 (Dense)	(None, 2)	258

Total params: 176,468,614
Trainable params: 176,468,614
Non-trainable params: 0

Iteración # 3

- Vamos a intentar mejorar el modelo

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_36 (Conv2D)           (None, 222, 222, 8)       224
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 220, 220, 8)       584
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 218, 218, 8)       584
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 216, 216, 8)       584
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 214, 214, 8)       584
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 212, 212, 8)       584
_________________________________________________________________
conv2d_42 (Conv2D)           (None, 210, 210, 8)       584
_________________________________________________________________
flatten_6 (Flatten)          (None, 352800)            0
_________________________________________________________________
dense_12 (Dense)             (None, 500)               176400500
_________________________________________________________________
dense_13 (Dense)             (None, 128)               64128
_________________________________________________________________
dense_14 (Dense)             (None, 2)                 258
=================================================================
Total params: 176,468,614
Trainable params: 176,468,614
Non-trainable params: 0
_________________________________________________________________
```

	Loss	**Accuracy**
Training set	0.1587	0.9458
Testing set	1.69	0.52

Iteración # 3

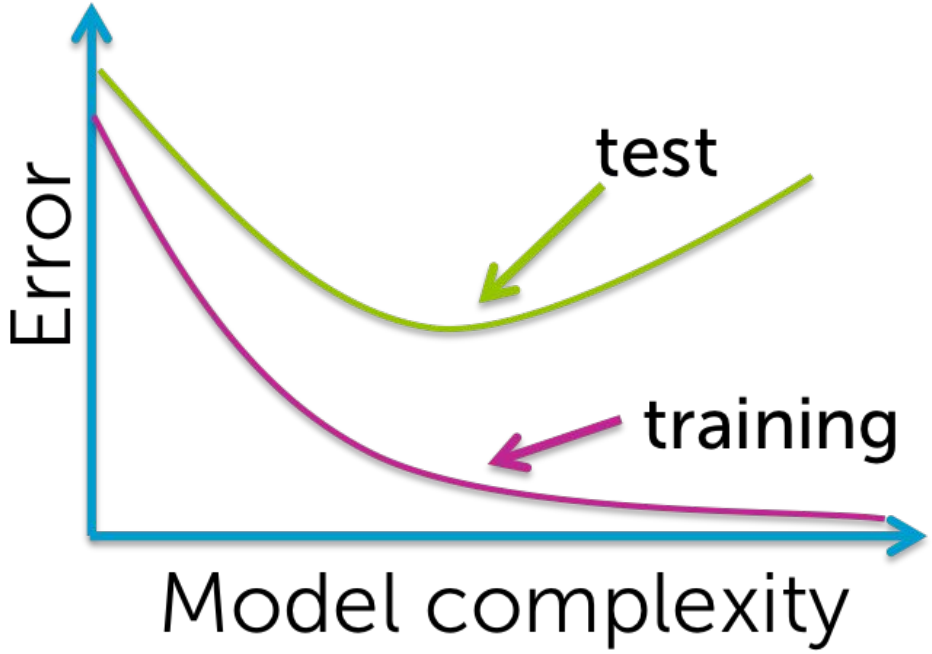
- Vamos a intentar mejorar el modelo

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_36 (Conv2D)           (None, 222, 222, 8)       224
_________________________________________________________________
conv2d_37 (Conv2D)           (None, 220, 220, 8)       584
_________________________________________________________________
conv2d_38 (Conv2D)           (None, 218, 218, 8)       584
_________________________________________________________________
conv2d_39 (Conv2D)           (None, 216, 216, 8)       584
_________________________________________________________________
conv2d_40 (Conv2D)           (None, 214, 214, 8)       584
_________________________________________________________________
conv2d_41 (Conv2D)           (None, 212, 212, 8)       584
_________________________________________________________________
conv2d_42 (Conv2D)           (None, 210, 210, 8)       584
_________________________________________________________________
flatten_6 (Flatten)          (None, 352800)            0
_________________________________________________________________
dense_12 (Dense)             (None, 500)               176400500
_________________________________________________________________
dense_13 (Dense)             (None, 128)               64128
_________________________________________________________________
dense_14 (Dense)             (None, 2)                 258
=================================================================
Total params: 176,468,614
Trainable params: 176,468,614
Non-trainable params: 0
_________________________________________________________________
```

Overfitting!!

	Loss	**Accuracy**
Training set	0.1587	0.9458
Testing set	1.69	0.52

Overfitting

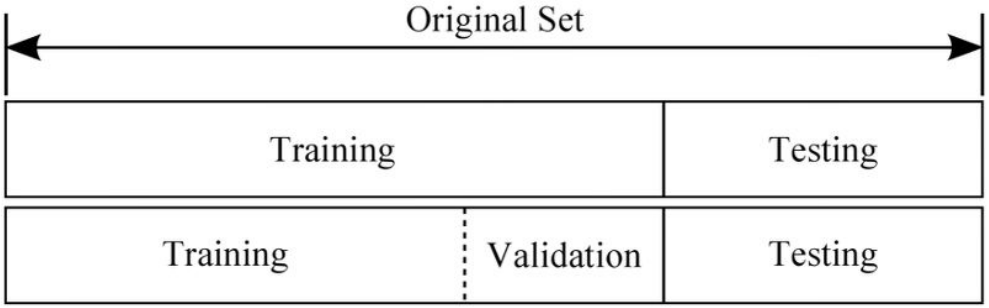


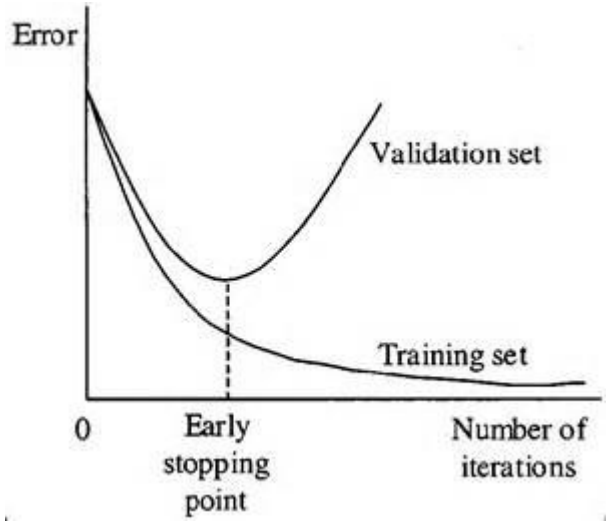
Overfitting

- Podemos cambiar hiperparámetros y arquitectura del modelo para mejorar métricas
- Problema:
  - Si vamos creando diferentes arquitecturas para conseguir buenas métricas, vamos a hacer un modelo bueno para el testing set. ¡Esto también es overfitting!
- Solución:
  - Utilizar un tercer set que se use para la selección de modelo.
  - Utilizar early stopping.

Overfitting

- Solución:
  - Utilizar un tercer set que se use para la selección de modelo.
  - Utilizar early stopping o checkpointing.





Overfitting

- Solución:
  - **Utilizar un tercer set que se use para la selección de modelo.**
  - Utilizar early stopping o checkpointing.

```
model.fit(input_x,
          input_y,
          batch_size=24,
          epochs=5,
          validation_split=0.2)
```

```
Train on 398 samples, validate on 100 samples
Epoch 1/5
398/398 [==============================] - 7s 18ms/step - loss: 0.0825 - acc: 0.9824 - val_loss: 0.0705 - val_acc: 0.9900
Epoch 2/5
398/398 [==============================] - 6s 15ms/step - loss: 0.0359 - acc: 0.9925 - val_loss: 0.0505 - val_acc: 1.0000
Epoch 3/5
398/398 [==============================] - 6s 15ms/step - loss: 0.0232 - acc: 0.9975 - val_loss: 0.0454 - val_acc: 0.9900
Epoch 4/5
398/398 [==============================] - 6s 15ms/step - loss: 0.0118 - acc: 0.9975 - val_loss: 0.0305 - val_acc: 0.9900
Epoch 5/5
398/398 [==============================] - 6s 15ms/step - loss: 6.2014e-04 - acc: 1.0000 - val_loss: 0.0358 - val_acc: 0.9800
```

Overfitting

- Solución:
  - Utilizar un tercer set que se use para la selección de modelo.
  - **Utilizar early stopping** o checkpointing.

```
from keras.callbacks import EarlyStopping
earlystop = EarlyStopping(monitor='val_acc', min_delta=0.0001, patience=5, \
                          verbose=1, mode='auto')
callbacks_list = [earlystop]
```

```
model.fit(input_x,
          input_y,
          batch_size=24,
          epochs=30,
          callbacks=callbacks_list,
          validation_split=0.2)
```

Overfitting

- Solución:
  - Utilizar un tercer set que se use para la selección de modelo.
  - Utilizar early stopping o **checkpointing.**

```
from keras.callbacks import ModelCheckpoint
checkpoint = ModelCheckpoint('saved_model', monitor='val_acc', verbose=1,
                             save_best_only=True, mode='max')
callbacks_list = [checkpoint]
```

```
Epoch 1/30
384/398 [===========================>..] - ETA: 0s - loss: 0.7887 - acc: 0.5260
Epoch 00001: val_acc improved from -inf to 0.51000, saving model to saved_model
398/398 [==============================] - 8s 19ms/step - loss: 0.7853 - acc: 0.5302 - val_loss: 0.6926 - val_acc: 0.5100
Epoch 2/30
384/398 [===========================>..] - ETA: 0s - loss: 0.6806 - acc: 0.5885
Epoch 00002: val_acc improved from 0.51000 to 0.52000, saving model to saved_model
398/398 [==============================] - 6s 14ms/step - loss: 0.6812 - acc: 0.5804 - val_loss: 0.6912 - val_acc: 0.5200
Epoch 3/30
384/398 [===========================>..] - ETA: 0s - loss: 0.6352 - acc: 0.7057
Epoch 00003: val_acc did not improve from 0.52000
398/398 [==============================] - 3s 8ms/step - loss: 0.6354 - acc: 0.6985 - val_loss: 0.6890 - val_acc: 0.5200
Epoch 4/30
384/398 [===========================>..] - ETA: 0s - loss: 0.5520 - acc: 0.8073
Epoch 00004: val_acc improved from 0.52000 to 0.56000, saving model to saved_model
```

¿Cuál es el mejor modelo?

- Hyperparameter Tuning

- Automatic model creation

Hyperparameter

- Hyperparameters:
  - Optimization: batch size, learning rate, optimizer
  - Model structure: dropout, neurons per layer, number of neurons

Grid Search

```
param_grid = [
  {'C': [1, 10, 100, 1000], 'kernel': ['linear']},
  {'C': [1, 10, 100, 1000], 'gamma': [0.001, 0.0001], 'kernel': ['rbf']},
 ]
```

Hyperparameter

- Hyperparameters:
  - Optimization: batch size, learning rate, optimizer
  - Model structure: dropout, neurons per layer, number of neurons

Random Search

```
{'C': scipy.stats.expon(scale=100), 'gamma': scipy.stats.expon(scale=.1),
  'kernel': ['rbf'], 'class_weight':['balanced', None]}
```

- CMA-ES

Hyperparameter

- Hyperparameters:
  - Optimization: batch size, learning rate, optimizer
  - Model structure: dropout, neurons per layer, number of neurons

Servicio distribuido en la nube (Google Cloud)

https://cloud.google.com/ml-engine/docs/tensorflow/using-hyperparameter-tuning

```
hyperparams = {
    'goal': 'MAXIMIZE',
    'hyperparameterMetricTag': 'metric1',
    'maxTrials': 30,
    'maxParallelTrials': 1,
    'enableTrialEarlyStopping': True,
    'params': []}

hyperparams['params'].append({
    'parameterName':'hidden1',
    'type':'INTEGER',
    'minValue': 40,
    'maxValue': 400,
    'scaleType': 'UNIT_LINEAR_SCALE'})
```

Creación de modelos automática (AutoML)

- Neural Architecture Search (NAS)
  - Hay bloques básicos
  - Se usa una RNN para construir una arquitectura

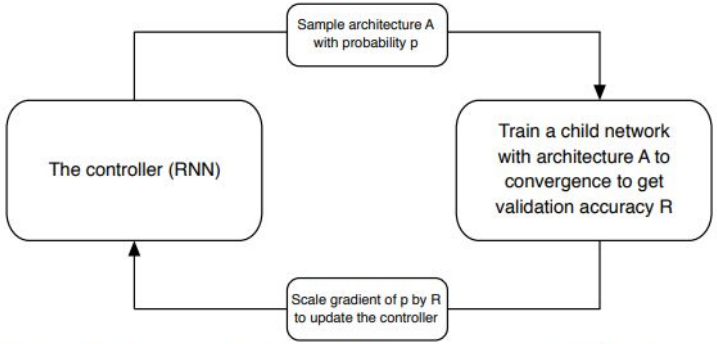
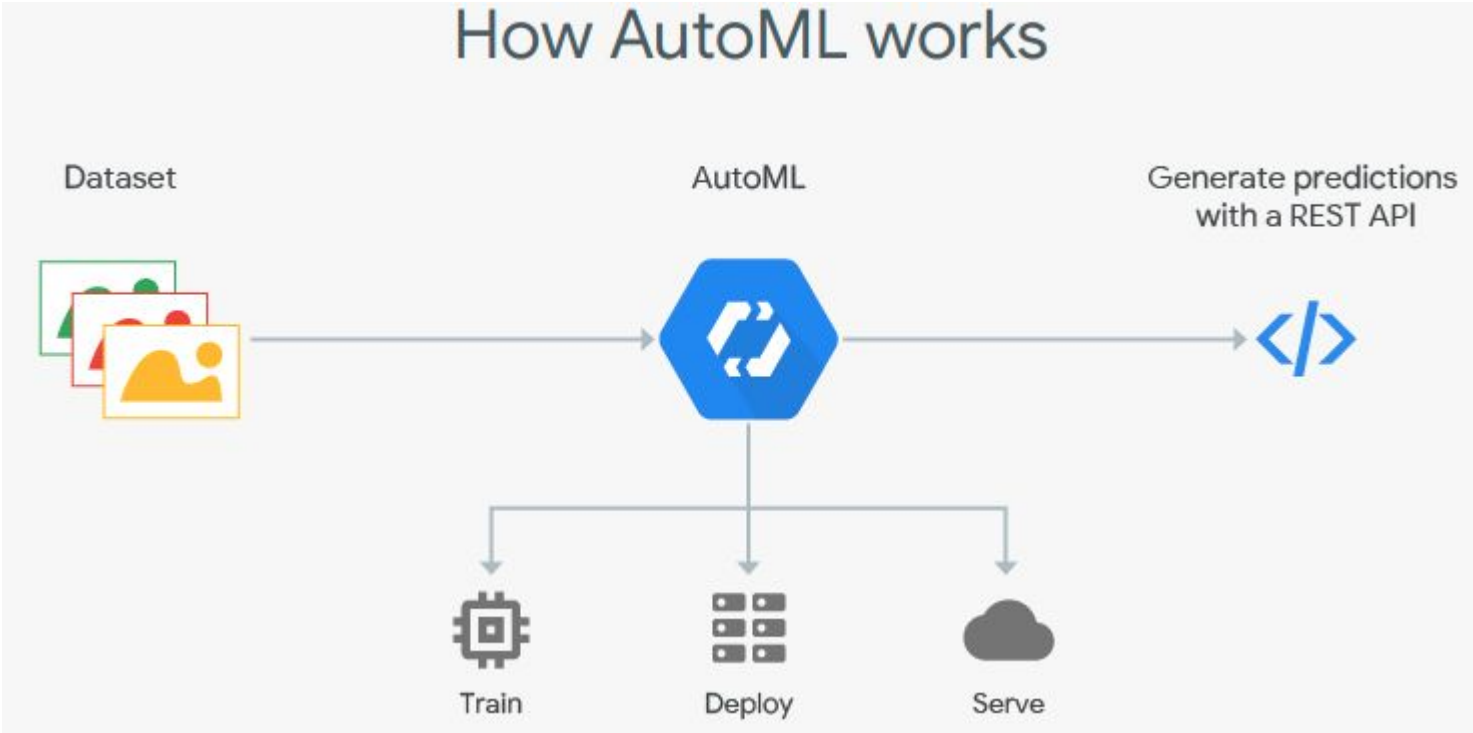


Figure 1. Overview of Neural Architecture Search [71]. A controller RNN predicts architecture A from a search space with probability p. A child network with architecture A is trained to convergence achieving accuracy R. Scale the gradients of p by R to update the RNN controller.

Paper:

https://arxiv.org/pdf/1707.07012.p

Creación de modelos automática (AutoML)



AUTO KERAS

AutoKeras

Instalar: **pip install autokeras**

```
import autokeras as ak

clf = ak.ImageClassifier(verbose=True)
clf.fit(input_x, input_y, time_limit=12 * 60)
y = clf.evaluate(output_x, output_y)
print(y)
```

AutoKeras

```
+---------------------------------------------+
|               Training model 0              |
+---------------------------------------------+

No loss decrease after 5 epochs.


Saving model.
+-----------------------------------------------------------------------+
|        Model ID        |          Loss          |      Metric Value      |
+-----------------------------------------------------------------------+
|           0            |   0.7001973628997803   |   0.4292682926829269   |
+-----------------------------------------------------------------------+


+---------------------------------------------+
|               Training model 1              |
+---------------------------------------------+
Epoch-1, Current Metric - 0:   0%|                                        | 0/4 [00:00<?, ? batch/s
Current model size is too big. Discontinuing training this model to search for other models.
```

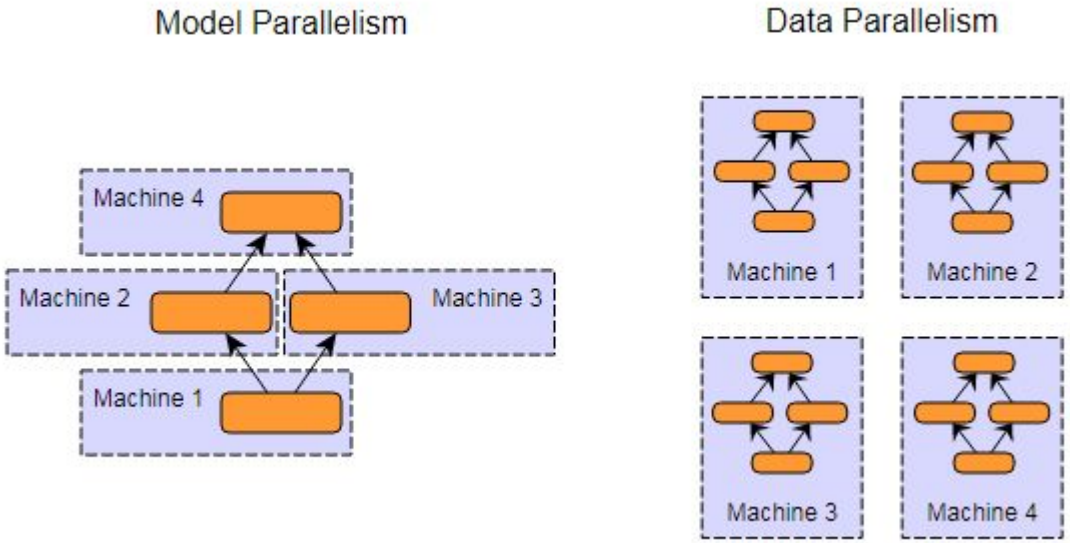
AutoKeras

66% de accuracy

```
+---------------------------------------------+
|               Training model 0              |
+---------------------------------------------+

No loss decrease after 5 epochs.


Saving model.
+-----------------------------------------------------------------------+
|        Model ID        |          Loss          |      Metric Value      |
+-----------------------------------------------------------------------+
|           0            |   0.7001973628997803   |   0.4292682926829269   |
+-----------------------------------------------------------------------+


+---------------------------------------------+
|               Training model 1              |
+---------------------------------------------+
Epoch-1, Current Metric - 0:   0%|                                        | 0/4 [00:00<?, ? batch/s
Current model size is too big. Discontinuing training this model to search for other models.
```

Distribuido

- Hemos hablado de paralelización de operaciones.



TensorFlow Distribuido

- Cluster: conjunto de tareas (tasks) que participan en la ejecución de un grafo.
- Cada task está asociado a un server.
- El server tiene
  - Master: permite crear sesiones
  - Worker: ejecuta operaciones del grafo

TensorFlow Distribuido

1. Crear tf.train.ClusterSpec

```
cluster = tf.train.ClusterSpec({"worker": ["worker0.example.com:2222",
                                           "worker1.example.com:2222",
                                           "worker2.example.com:2222"],
                                "ps": ["ps0.example.com:2222",
                                       "ps1.example.com:2222"]})
```

2. Crear tf.train.Server

```
server = tf.train.Server(cluster, job_name="local", task_index=0)
```

TensorFlow Distribuido

3. Modificar el programa de entrenamiento

4. Correr trainer

```
python trainer.py \
   --ps_hosts=ps0.example.com:2222,ps1.example.com:2222 \
   --worker_hosts=worker0.example.com:2222,worker1.example.com:2222 \
   --job_name=ps --task_index=0
On ps1.example.com:
python trainer.py \
   --ps_hosts=ps0.example.com:2222,ps1.example.com:2222 \
   --worker_hosts=worker0.example.com:2222,worker1.example.com:2222 \
   --job_name=ps --task_index=1
```

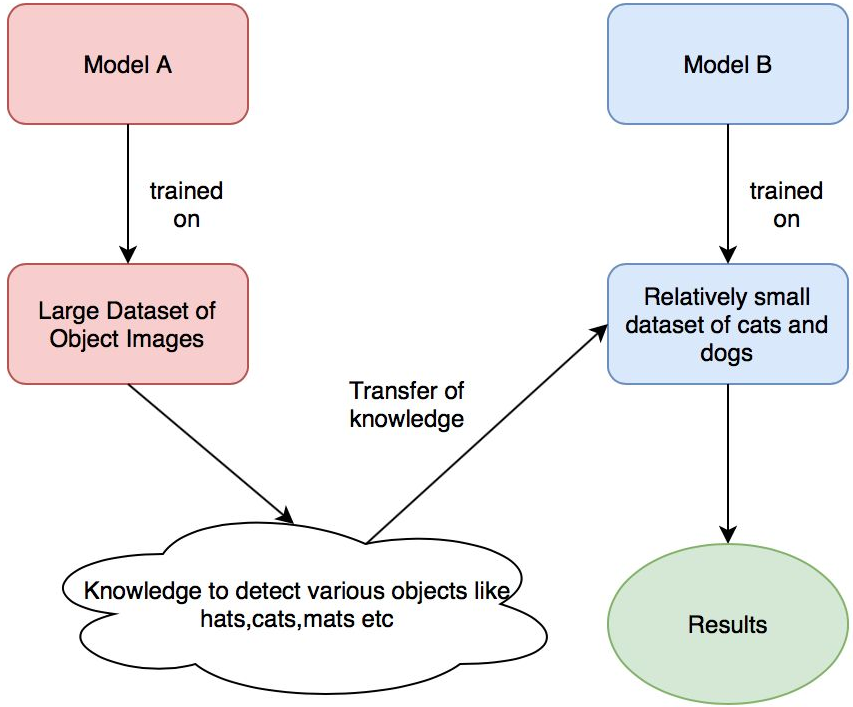
Problemas

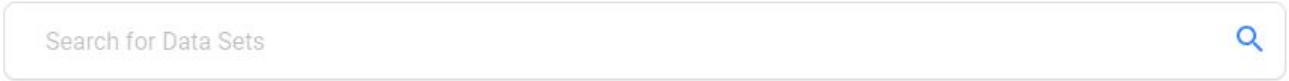
- Keep it simple, stupid (KISS).
- Poca transparencia en el modelo.

Iteración # 4

- Vamos a buscar un pipeline más simple, transparente y eficiente
  - Transfer Learning
  - Métricas

Transfer Learning





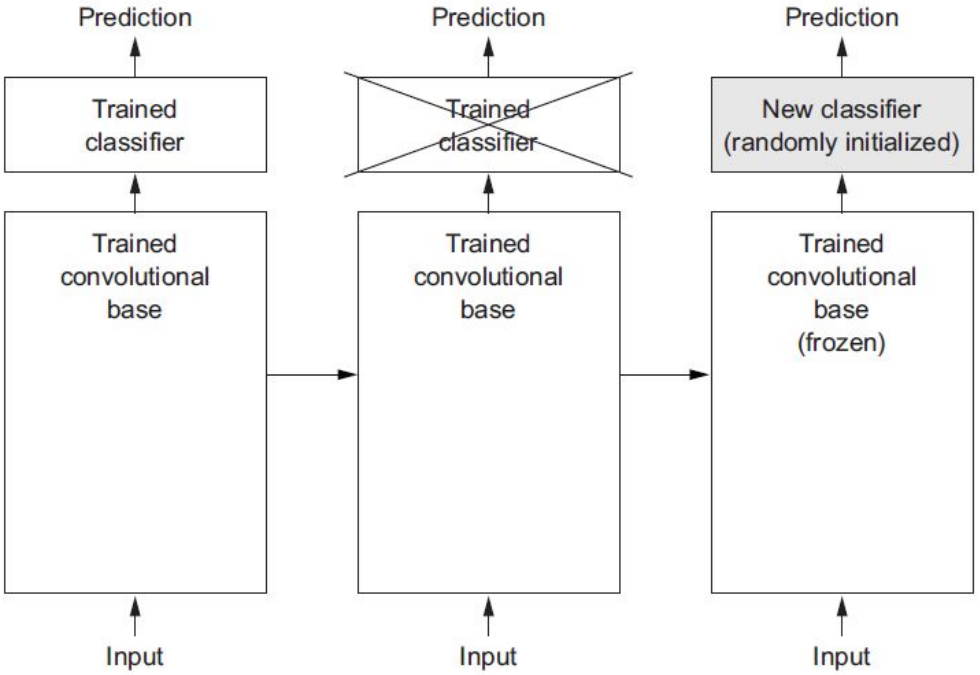
Try boston education data or weather site:noaa.gov

ImageNet Dataset

IMAGENET

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., ... & Fei-Fei, L. (2015). Imagenet large scale visual recognition challenge. *arXiv preprint arXiv:1409.0575*. [web]

Transfer Learning



ImageNet

- 1000 clases
- 14,000,000 de imágenes
- Diferentes modelos ya existen

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	99 MB	0.749	0.921	25,636,712	168
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	-

Cargando InceptionResNetV2

```
from tensorflow.python.keras.applications import InceptionResNetV2

conv_base = InceptionResNetV2(weights='imagenet', include_top=False)
conv_base.summary()
```

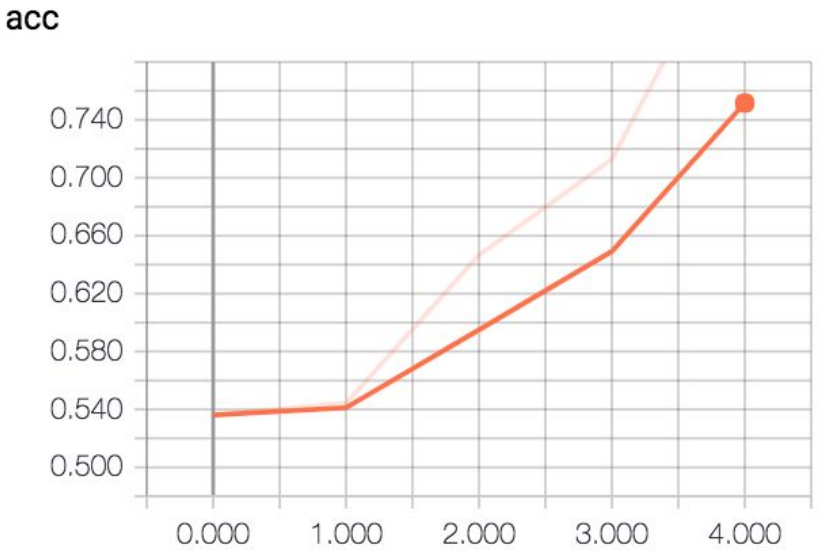
```
activation_504 (Activation)     (None, None, None, 2 0           batch_normalization_406[0][0]
__________________________________________________________________________________________________
block8_10_mixed (Concatenate)   (None, None, None, 4 0           activation_501[0][0]
                                                                 activation_504[0][0]
__________________________________________________________________________________________________
block8_10_conv (Conv2D)         (None, None, None, 2 933920      block8_10_mixed[0][0]
__________________________________________________________________________________________________
block8_10 (Lambda)              (None, None, None, 2 0           block8_9_ac[0][0]
                                                                 block8_10_conv[0][0]
__________________________________________________________________________________________________
conv_7b (Conv2D)                (None, None, None, 1 3194880     block8_10[0][0]
__________________________________________________________________________________________________
conv_7b_bn (BatchNormalization) (None, None, None, 1 4608        conv_7b[0][0]
__________________________________________________________________________________________________
conv_7b_ac (Activation)         (None, None, None, 1 0           conv_7b_bn[0][0]
==================================================================================================
Total params: 54,336,736
Trainable params: 54,276,192
Non-trainable params: 60,544
__________________________________________________________________________________________________
```

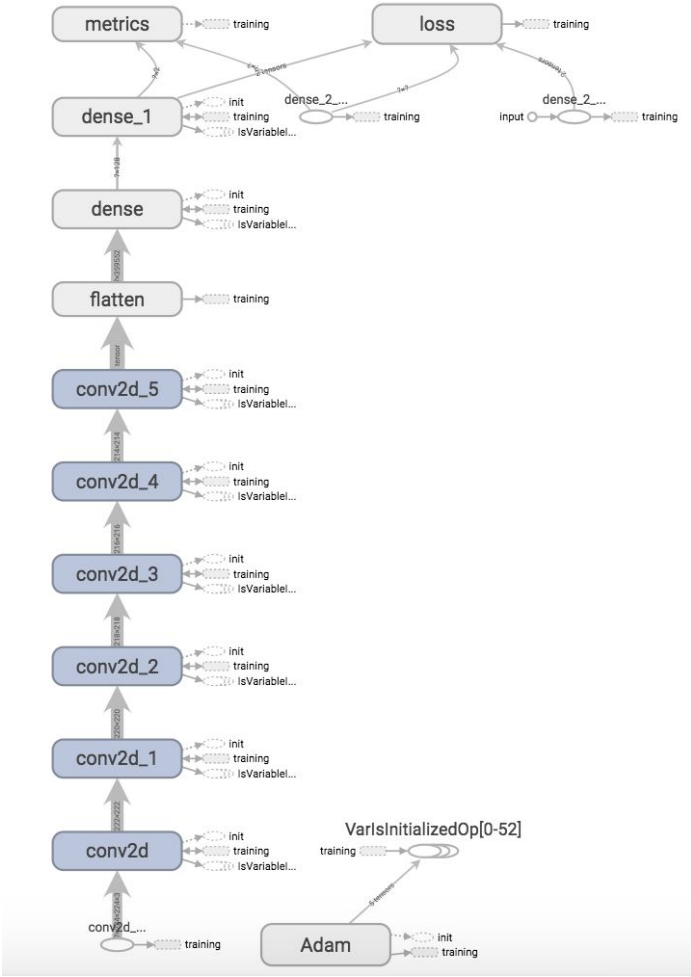
Transfer Learning

```
model = Sequential()
model.add(conv_base)
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dense(num_classes, activation='softmax'))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_resnet_v2 (Model)  (None, 5, 5, 1536)        54336736
_________________________________________________________________
flatten_9 (Flatten)          (None, 38400)             0
_________________________________________________________________
dense_9 (Dense)              (None, 256)               9830656
_________________________________________________________________
dense_10 (Dense)             (None, 2)                 514
=================================================================
Total params: 64,167,906
Trainable params: 64,107,362
Non-trainable params: 60,544
_________________________________________________________________
```

TensorBoard



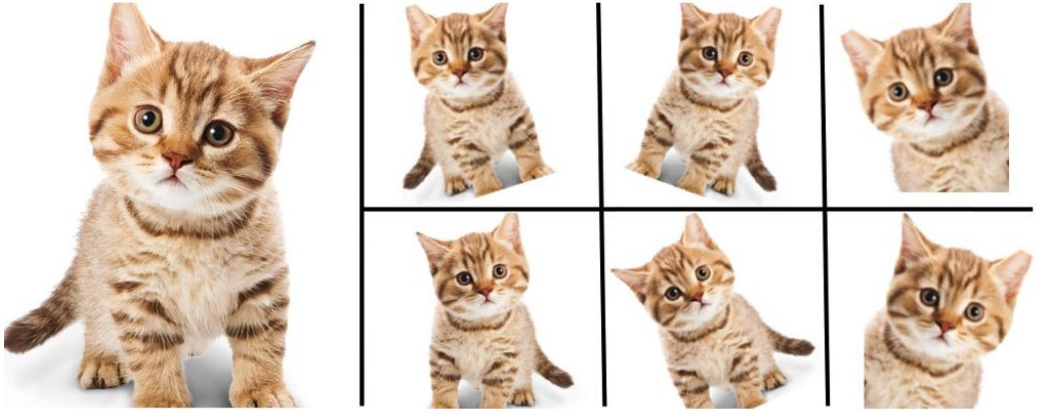


Iteración #3

	Loss	**Accuracy**
Training set	0.258	0.9714
Validation set	1.45	0.65
Testing set	1.34	0.67

¿Cómo podemos obtener mejores resultados?

- Nuestro problema va al paso inicial: tenemos sólo 1000 imágenes.
- Recolectar más datos
- Utilizar GANs para generar más imágenes
- Image Augmentation



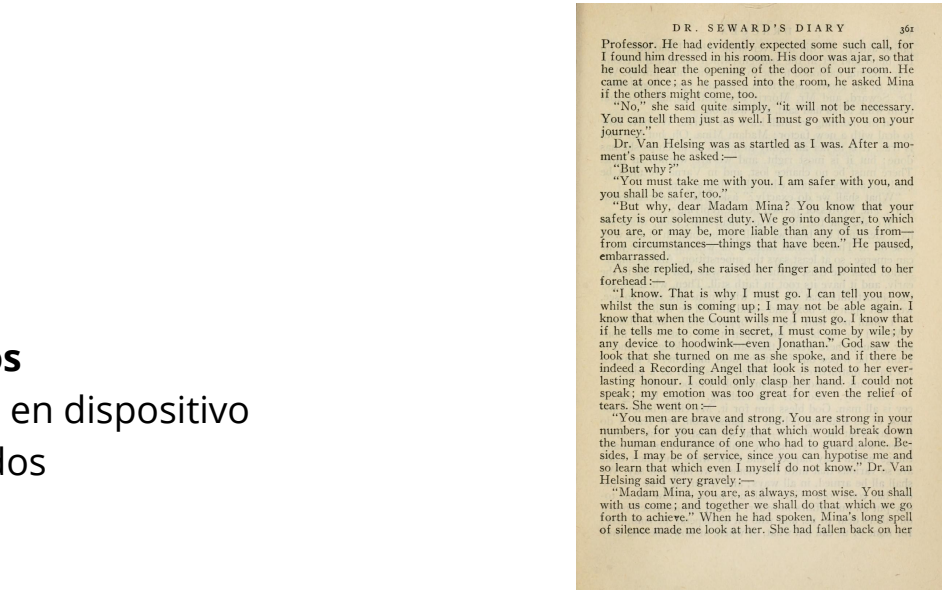
Llevando el proyecto al mundo real

Servicio en la nube

- Firebase ML Kit (beta)
  - APIs ya listos para ser usados
  - Permite tanto on-device como en dispositivo
  - Permite modelos personalizados

Servicio en la nube

- Firebase ML Kit (beta)
  - **APIs ya listos para ser usados**
  - Permite tanto on-device como en dispositivo
  - Permite modelos personalizados
- Reconocimiento de texto



DR. SEWARD'S DIARY 361

Professor. He had evidently expected some such call, for I found him dressed in his room. His door was ajar, so that he could hear the opening of the door of our room. He came at once; as he passed into the room, he asked Mina if the others might come, too.

"No," she said quite simply, "it will not be necessary. You can tell them just as well. I must go with you on your journey."

Dr. Van Helsing was as startled as I was. After a moment's pause he asked:—

"But why?"

"You must take me with you. I am safer with you, and you shall be safer, too."

"But why, dear Madam Mina? You know that your safety is our solemnest duty. We go into danger, to which you are, or may be, more liable than any of us from—from circumstances—things that have been." He paused, embarrassed.

As she replied, she raised her finger and pointed to her forehead:—

"I know. That is why I must go. I can tell you now, whilst the sun is coming up; I may not be able again. I know that when the Count wills me I must go. I know that if he tells me to come in secret, I must come by wile; by any device to hoodwink—even Jonathan." God saw the look that she turned on me as she spoke, and if there be indeed a Recording Angel that look is noted to her everlasting honour. I could only clasp her hand. I could not speak; my emotion was too great for even the relief of tears. She went on:—

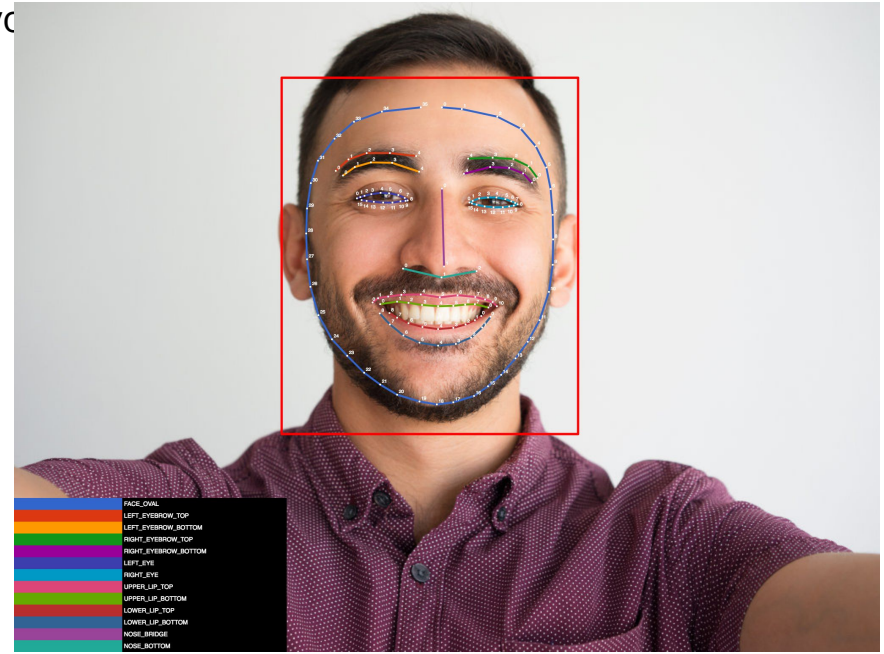
"You men are brave and strong. You are strong in your numbers, for you can defy that which would break down the human endurance of one who had to guard alone. Besides, I may be of service, since you can hypotise me and so learn that which even I myself do not know." Dr. Van Helsing said very gravely:—

"Madam Mina, you are, as always, most wise. You shall with us come; and together we shall do that which we go forth to achieve." When he had spoken, Mina's long spell of silence made me look at her. She had fallen back on her

Block 0	
Text	`Wege der parlamentarischen Demokratie`
Frame	(117.0, 258.0, 190.0, 83.0)
Corner Points	(117, 270), (301.64, 258.49), (306.05, 329.36), (121.41, 340.86)
Recognized Language Code	de
Lines	(3 lines)

Servicio en la nube

- Firebase ML Kit (beta)
  - **APIs ya listos para ser usados**
  - Permite tanto on-device como en dispositivo
  - Permite modelos personalizados
- Detección de cara



Servicio en la nube

- Firebase ML Kit (beta)
  - **APIs ya listos para ser usados**
  - Permite tanto on-device como en dispositivo
  - Permite modelos personalizados
- Labeling de imágenes

Description	sport venue
Knowledge Graph entity ID	/m/0bmgjqz
Confidence	0.9860726

Description	player
Knowledge Graph entity ID	/m/02vzx9
Confidence	0.9797604

Description	stadium
Knowledge Graph entity ID	/m/019cfy
Confidence	0.9635762

Description	soccer specific stadium
Knowledge Graph entity ID	/m/0404y4
Confidence	0.95806926

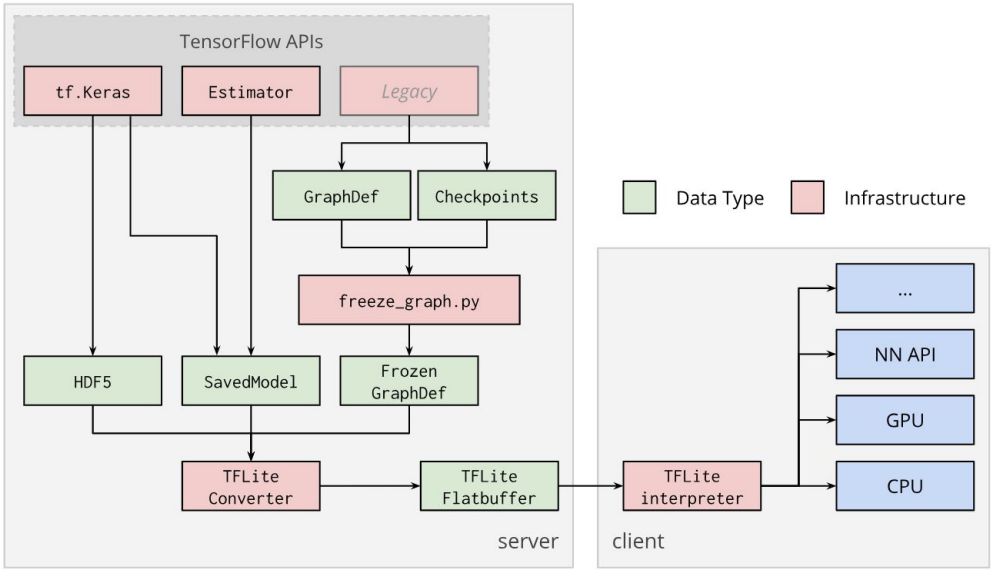
Servicio en la nube

- Firebase ML Kit (beta)
  - **APIs ya listos para ser usados**
  - Permite tanto on-device como en dispositivo
  - Permite modelos personalizados
- Reconocimiento de Landmarks

Result	
Description	Brugge
Geographic Coordinates	51.207367, 3.226933
Knowledge Graph entity ID	/m/0drjd2
Bounding Polygon	(20, 342), (651, 342), (651, 798), (20, 798)
Confidence Score	0.77150935

ML en dispositivo

- TensorFlow Lite
  - Portable para Android, iOS y dispositivos IoT.
  - Modelo más pequeño
  - Aceleración con GPU



- Usar TensorFlow Lite converter
  - Terminal
  - Script de Python

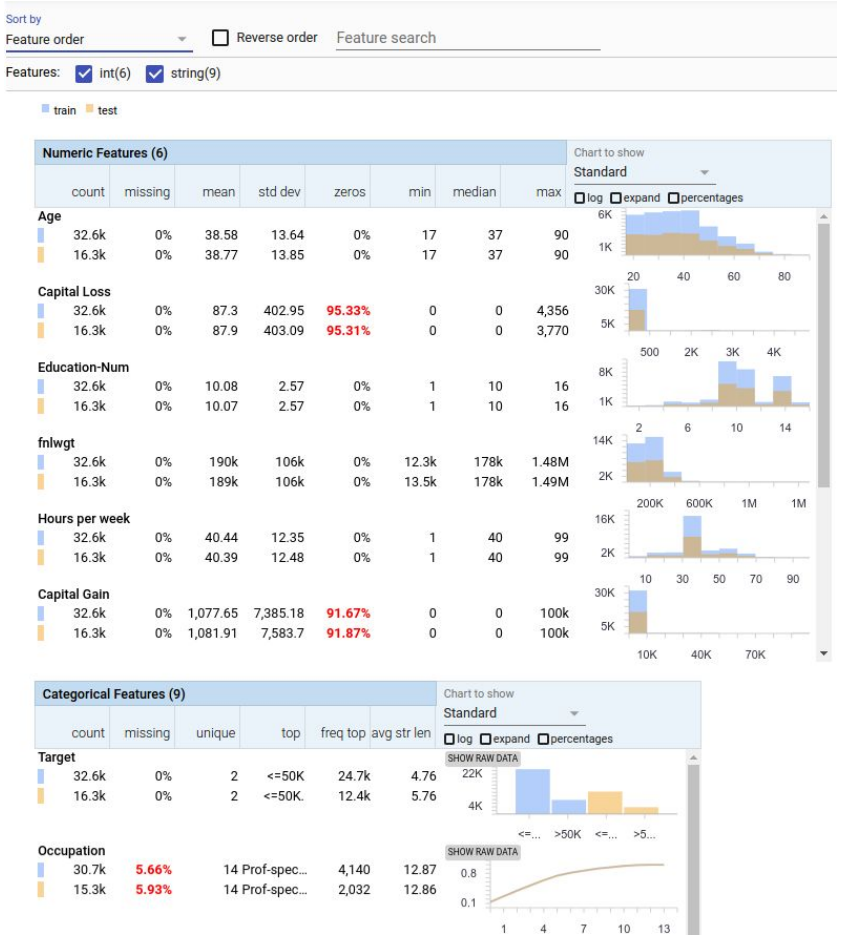
- Ejecución: 3x speedup
- Memoria: 4x compression

ML en dispositivo

- Usar modelos pre-entrenados
  - MobileNet - funciona muy bien para clasificación de imágenes con los recursos limitados de los dispositivos.
  - Inception - clasificación de un total de 1000 clases
  - Smart Reply - respuestas de un mensaje sugeridas según el contexto
- Transfer learning
- Se pueden hostear los modelos en Firebase

TensorFlow en producción (TFX)

- TensorFlow Data Validation (TFDV)
  - Estadísticas de tus datos a gran escala
  - **Visualizador de distribuciones y estadísticas** (Facets)
  - Schema viewer
  - Identificación de anomalías



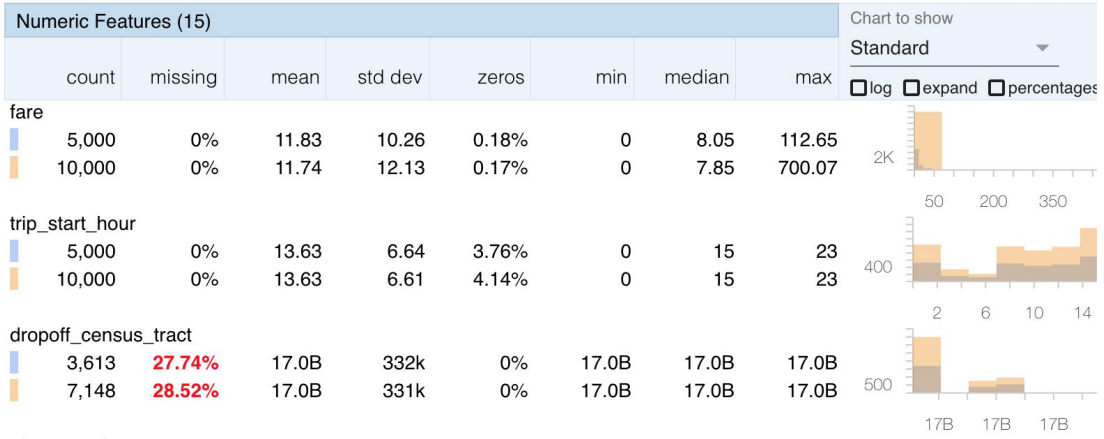
TensorFlow en producción (TFX)

- TensorFlow Data Validation (TFDV)
  - Estadísticas de tus datos a gran escala
  - Visualizador de distribuciones y estadísticas (Facets)
  - **Schema viewer**
  - Identificación de anomalías

Feature name				
'fare'	FLOAT	required		-
'trip_start_hour'	INT	required		-
'pickup_census_tract'	BYTES	optional		-
'dropoff_census_tract'	FLOAT	optional	single	-
'company'	STRING	optional	single	'company'
'trip_start_timestamp'	INT	required		-
'pickup_longitude'	FLOAT	required		-
'trip_start_month'	INT	required		-
'trip_miles'	FLOAT	required		-
'dropoff_longitude'	FLOAT	optional	single	-
'dropoff_community_area'	FLOAT	optional	single	-
'pickup_community_area'	INT	required		-

TensorFlow en producción (TFX)

- TensorFlow Data Validation (TFDV)
  - **Estadísticas de tus datos a gran escala**
  - Visualizador de distribuciones y estadísticas (Facets)
  - Schema viewer
  - Identificación de anomalías



TensorFlow en producción (TFX)

- TensorFlow Data Validation (TFDV)
  - Estadísticas de tus datos a gran escala
  - Visualizador de distribuciones y estadísticas (Facets)
  - Schema viewer
  - **Identificación de anomalías**

Feature name	Anomaly short description	Anomaly long description
'payment_type'	Unexpected string values	Examples contain values missing from the schema: Prcard (<1%).
'company'	Unexpected string values	Examples contain values missing from the schema: 2092 - 61288 Sbeih company (<1%), 2192 - 73487 Zeymane Corp (<1%), 2192 - Zeymane Corp (<1%), 2823 - 73307 Seung Lee (<1%), 3094 - 24059 G.L.B. Cab Co (<1%), 3319 - CD Cab Co (<1%), 3385 - Eman Cab (<1%), 3897 - 57856 Ilie Malec (<1%), 4053 - 40193 Adwar H. Nikola (<1%), 4197 - Royal Star (<1%), 585 - 88805 Valley Cab Co (<1%), 5874 - Sergey Cab Corp. (<1%), 6057 - 24657 Richard Addo (<1%), 6574 - Babylon Express Inc. (<1%), 6742 - 83735 Tasha ride inc (<1%).

TensorFlow en producción (TFX)

- TensorFlow Data Validation (TFDV)
  - Estadísticas de tus datos a gran escala
  - Visualizador de distribuciones y estadísticas (Facets)
  - Schema viewer
  - Identificación de anomalías

- Usos:
  - Validar nuevos datos de inferencia para revisar que los features están bien.
  - Validar nuevos datos de inferencia para revisar que el modelo esté entrenado para esos.
  - Revisar después de hacer feature engineering. (TensorFlow Transform)

TensorFlow en producción (TFX)

- TensorFlow Transform: Librería de preprocesamiento de datos

- Usos:
  - Normalizar entradas
  - Convertir strings a integers generando un vocabulario
  - Convertir floats a integers usando buckets

TensorFlow en producción (TFX)

- TensorFlow Model Analysis
  - Evaluación distribuida de modelo

TensorFlow en producción (TFX)

- TensorFlow Serving
  - Docker con puertos expuestos a REST y a gRPC

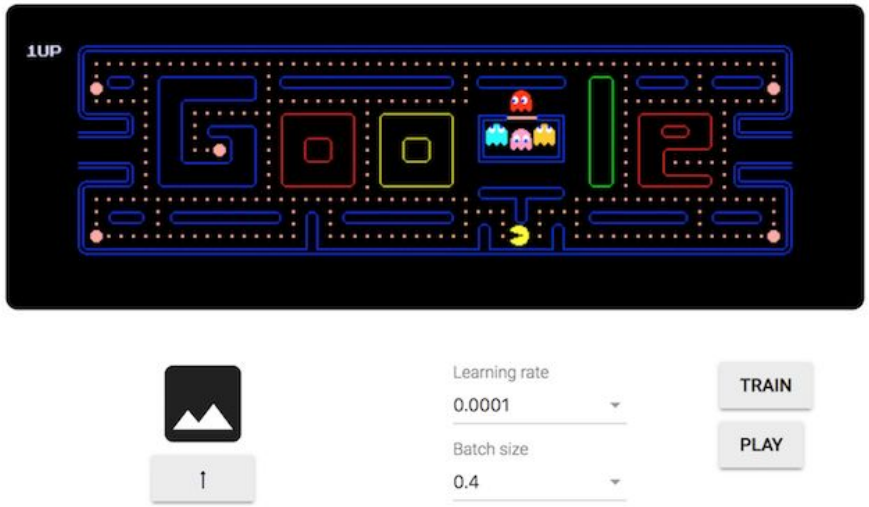
```
docker run -p 8501:8501 \
  --mount type=bind,source=/path/to/my_model/,target=/models/my_model \
  -e MODEL_NAME=my_model -t tensorflow/serving
```

```
curl -d '{"instances": [1.0, 2.0, 5.0]}' \
  -X POST http://localhost:8501/v1/models/half_plus_two:predict
```

```
{ "predictions": [2.5, 3.0, 4.5] }
```

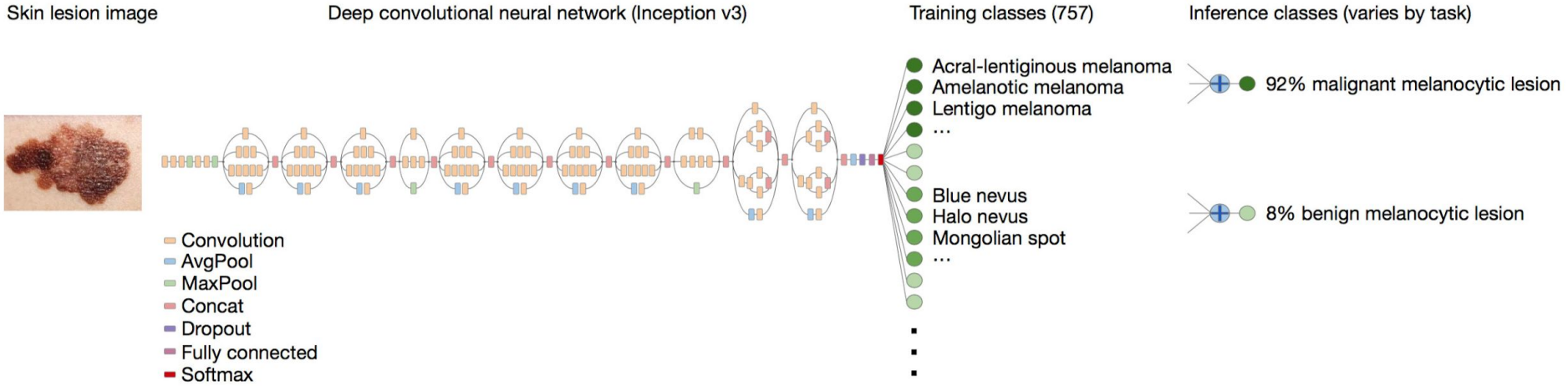
TensorFlow en el Browser

- TensorFlow.js
  - Escribir y entrenar modelos en browser.
  - Utilizar modelos entrenados en Python.
  - Mantener la privacidad de los usuarios



Lanzamiento

Cancer


Skin lesion image
Deep convolutional neural network (Inception v3)
Training classes (757)
Inference classes (varies by task)
Acral-lentiginous melanoma
Amelanotic melanoma
Lentigo melanoma
…
Blue nevus
Halo nevus
Mongolian spot
…
92% malignant melanocytic lesion
8% benign melanocytic lesion
Convolution
AvgPool
MaxPool
Concat
Dropout
Fully connected
Softmax

Deuda técnica de ML

1. Los modelos complejos eliminan encapsulación.
2. Dependencias de datos.
3. Pipelines complejos para preparar los datos.
4. Deuda en la configuración.
5. Falta de flexibilidad a cambios del mundo externo.
6. No hay monitoreo ni testing.

Buenas Prácticas de ML

- No tengas miedo en lanzar un producto que no usa ML.
- Inicia diseñando e implementando las métricas.
- Inicia con un modelo sencillo, implementa bien la infraestructura.
- Has testing de infraestructura y modelo por separado.
- Detecta problemas antes de exportar modelos.
- Inicia con una métrica simple y observable.
- Crea un modelo interpretable.
- Planea lanzar e iterar.
- Y recuerda, tú **no eres un usuario típico**.

Muchas gracias

osanseviero@gmail.com

Fuentes

- https://medium.com/@d3lm/understand-tensorflow-by-mimicking-its-api-from-scratch-faa55787170d
- https://www.tensorflow.org/tutorials/eager/eager_basics#gpu_acceleration
- https://www.quora.com/How-does-TensorFlow-use-GPUs
- http://timdettmers.com/2018/11/05/which-gpu-for-deep-learning/
- https://www.xcelerit.com/computing-benchmarks/insights/benchmarks-deep-learning-nvidia-p100-vs-v100-gpu/
- https://blog.inten.to/hardware-for-deep-learning-part-3-gpu-8906c1644664
- https://arxiv.org/abs/1704.04760

Fuentes

- https://arxiv.org/pdf/1707.07012.pdf
- https://www.tensorflow.org/deploy/distributed
- https://www.tensorflow.org/deploy/distributed
- https://stackoverflow.com/questions/47818822/can-i-use-tensorboard-with-google-colab
- https://www.tensorflow.org/lite/devguide
- https://github.com/tensorflow/model-analysis/tree/master/examples/chicago_taxi
- https://js.tensorflow.org/demos/
- https://ai.google/research/pubs/pub43146