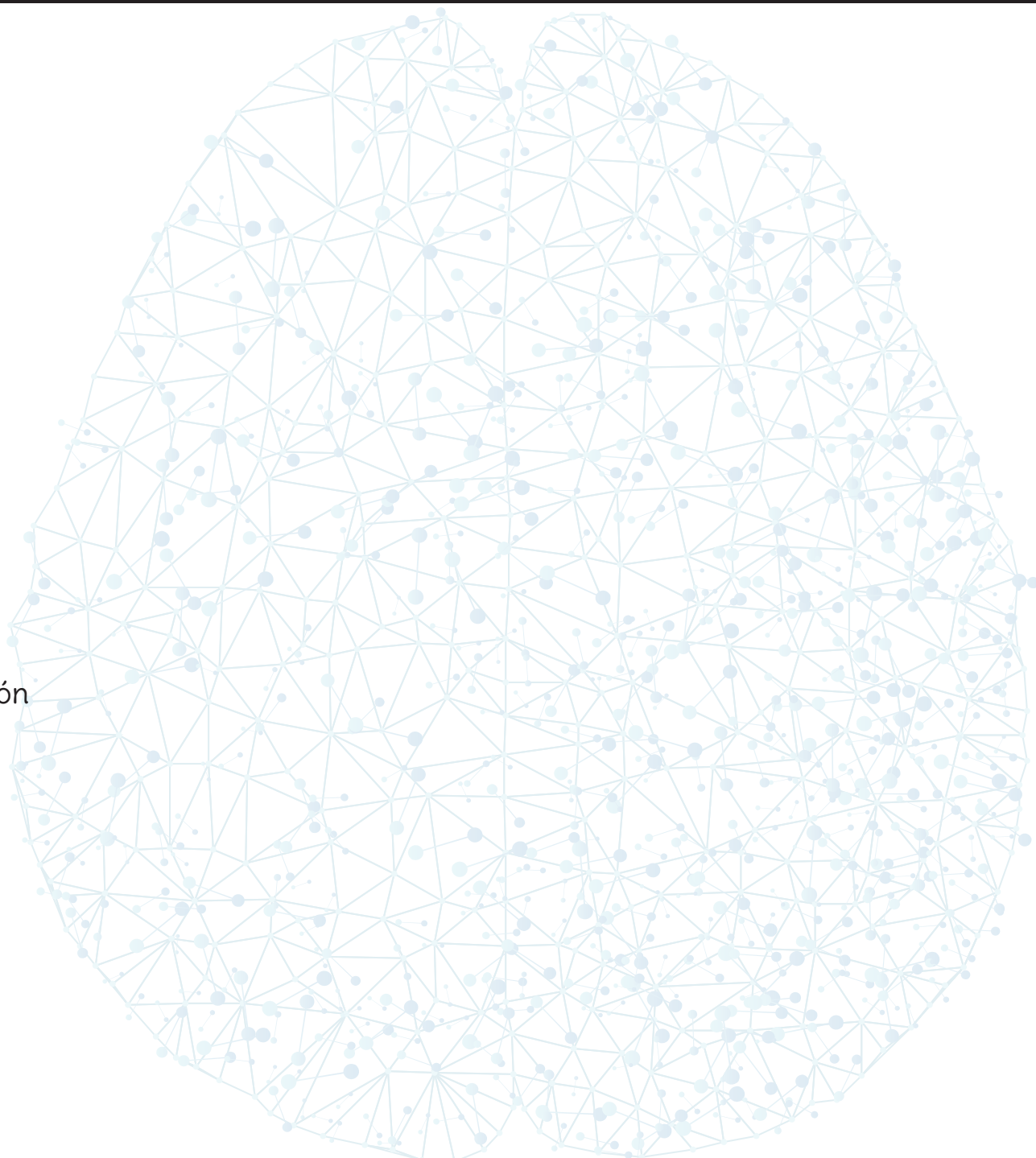




Instituto Tecnológico de Costa Rica  
Escuela de Ingeniería en Computación  
Inteligencia Artificial

TensorFlow

Arlette Calvo Montero  
200410655  
II Semestre

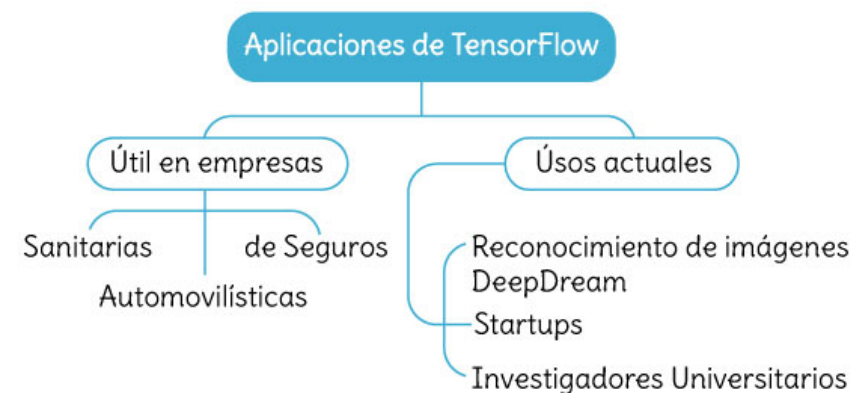


## En qué consiste TensorFlow

Tensor Flow consiste en una biblioteca basada en un sistema de redes neuronales, por lo que tiene la capacidad de relacionar datos de una forma similar a como lo hace el cerebro humano.

Por ejemplo gracias a la capacidad de asociación del sistema de redes neuronales tiene la habilidad de poder relacionar imágenes y textos de manera rápida, además de poder reconocer varias palabras del alfabeto ya que tiene la capacidad de relacionar las letras y fonemas.

Entre sus principales ventajas, se pueden mencionar, que es una herramienta libre, la cual en el momento en que fue liberada su capacidad de reconocimiento y relación era superior al 93% en las imágenes, por otro lado a version liberada en el 2015 mejoró considerablemente el tiempo de respuesta a los 0.7 segundos.



## Documentación de Tutoriales

### • Image Recognition

En qué consiste:

Para el cerebro humano poder hacer una distinción entre objetos o seres a nivel visual, es generalmente una tarea sencilla, sin embargo para una computadora es una tarea más compleja, por lo que Image Recognition provee en un modelo, capaz de poder realizar reconocimiento de imágenes, por medio de la implementación del modelo llamado Inception-v3, el cual ha sido entrenada al intentar clasificar imágenes enteras en 1000 clases, donde como resultado muestra un listado de la siguiente manera: "Cebra", "Dálmata", "Lavavajillas".

## Conclusiones:

Image Recognition, permite poder reconocer imágenes de forma sorprendente y muy acertada, donde al proveer una imagen el muestra un listado con los resultados obtenidos, por ejemplo en este caso la imagen por identificar consiste en un panda, donde al finalizar el análisis muestra un listado de los posibles resultados y además coloca un puntaje correspondiente a la probabilidad de acierto.

Por otro lado el modelo utilizado en este caso fue entrenado con datos del 2012, por lo que posteriormente se podrían realizar más entrenamientos con datos más actuales con el fin de mejorar la capacidad de procesamiento.

Implementar su uso en el desarrollo de aplicaciones es muy factible, ya que esta herramienta únicamente necesita un dato de entrada y él provee un listado con los resultados obtenidos, por lo es de gran ayuda para los desarrolladores, además de que posee excelentes tiempos de respuesta.

## Cómo podría aplicarlo a un problema real:

Esta herramienta puede ser muy útil en muchos sistemas utilizados actualmente, por ejemplo en el manejo de inventarios de una determinada empresa, donde se necesite actualizar la galería correspondiente a sus productos.

En este caso la herramienta es muy útil, ya que se puede desarrollar una plataforma que implemente el uso de este modelo, donde únicamente con colocarle como dato de entrada una imagen, el puede identificar de cual producto se trata y así mismo posteriormente poder clasificarla, con mayor rapidez y acierto.





## Screenshots de resultados:

Para poder realizar este tutorial se debe clonar el repositorio correspondiente, posteriormente buscar la carpeta que contiene los modelos y ejecutar el archivo `classify_image.py`, el cual clasifica una imagen correspondiente a un panda, sin embargo se puede realizar con otra imagen, pero se debe modificar el argumento correspondiente al image file.

```
Cloning into 'models'...
remote: Counting objects: 8301, done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 8301 (delta 23), reused 26 (delta 16), pack-reused 8264
Receiving objects: 100% (8301/8301), 158.47 MiB | 185.00 KiB/s, done.
Resolving deltas: 100% (4621/4621), done.

[(tensorflow) Arlettes-MBP:imagenet arlettecalvo$ python classify_image.py
/Users/arlettecalvo/tensorflow/lib/python3.6/importlib/_bootstrap.py:205: RuntimeWarning: compiletime version 3.5 of module 'tensorflow.python.framework.fast_tensor_util' does not match runtime version 3.6
  return f(*args, **kwargs)
>> Downloading inception-2015-12-05.tgz 100.0%
Successfully downloaded inception-2015-12-05.tgz 88931400 bytes.
2017-11-04 19:11:28.207866: I tensorflow/core/platform/cpu_feature_guard.cc:137]
Your CPU supports instructions that this TensorFlow binary was not compiled to
use: SSE4.1 SSE4.2 AVX AVX2 FMA
2017-11-04 19:11:29.259781: W tensorflow/core/framework/op_def_util.cc:334] Op BatchNormWithGlobalNormalization is deprecated. It will cease to work in GraphDef
version 9. Use tf.nn.batch_normalization().
giant panda, panda, panda bear, coon bear, Ailuropoda melanoleuca (score = 0.891
07)
indri, indris, Indri indri, Indri brevicaudatus (score = 0.00779)
lesser panda, red panda, panda, bear cat, cat bear, Ailurus fulgens (score = 0.0
0296)
custard apple (score = 0.00147)
earthstar (score = 0.00117)
(tensorflow) Arlettes-MBP:imagenet arlettecalvo$
```

## • Reentrenamiento de imagen

### En qué consiste:

Este caso consiste en tomar un modelo completamente entrenado para un conjunto de categorías, donde se toma el último layer y se entrena nuevamente desde cero por medio del “transfer learning”, es decir se hace un reentrenamiento de los pesos existentes para las nuevas clases.

No se refiere a un entrenamiento total, sin embargo dicho entrenamiento “parcial” resulta ser muy efectivo para muchas aplicaciones, además de reducir considerablemente los tiempos.

### Conclusiones:

La implementación del image retraining, puede ser de gran ayuda en el desarrollo de aplicaciones, ya que por medio del transfer learning, donde al tener los layers previamente entrenados, se toma únicamente el último de ellos, se elimina y se hace un reentrenamiento del mismo, reutilizando los layers inferiores.

Lo más importante es que no es necesario realizar un entrenamiento “completo” lo que puede consumir mucho tiempo y recursos, únicamente con reentrenar el layer superior se puede obtener un buen resultado, por ejemplo en este caso, según el entrenamiento realizado la precisión de la prueba final es de 90.4%.

### Cómo podría aplicarlo a un problema real:

Esta herramienta puede ser de gran utilidad en gran variedad de aplicaciones, por ejemplo en sistemas que necesiten el reconocimiento de grupos de animales, donde si se desea que el sistema reconozca a un grupo



de animales en particular, únicamente se debe entrenar el layer superior, en lugar de tener que realizar un entrenamiento completo, lo que podría consumir mucho tiempo y recursos, al hacerlo de esta manera se reduce considerablemente el tiempo y a la vez se obtiene un buen resultado.

### Screenshots de resultados:

La finalidad de este tutorial consiste en la utilización del modelo Inception v3 que se había entrenado previamente, donde lo que se hace es eliminar el layer superior y hace un reentrenamiento del mismo, por medio del aprendizaje por transferencia, donde el aspecto más importante es que dicho entrenamiento es que se las capas inferiores que se han entrenado previamente se pueden reutilizar sin ninguna alteración.

Para la realización de dicho tutorial se utilizó el repositorio de imágenes propuesto en la documentación.

```
./tensorflow/contrib/tensor_forest/kernels/v4/grow_stats.h:385:8: warning: 'InitLeafClassStats' overrides a member function but is not marked 'override' [-Winconsistent-override]
    void InitLeafClassStats(int best_split_index, LeafStat* left_stats,
    ^
./tensorflow/contrib/tensor_forest/kernels/v4/grow_stats.h:196:16: note: overridden virtual function is here
    virtual void InitLeafClassStats(int best_split_index, LeafStat* left_stats,
    ^
./tensorflow/contrib/tensor_forest/kernels/v4/grow_stats.h:478:8: warning: 'InitLeafClassStats' overrides a member function but is not marked 'override' [-Winconsistent-override]
    void InitLeafClassStats(int best_split_index, LeafStat* left_stats,
    ^
./tensorflow/contrib/tensor_forest/kernels/v4/grow_stats.h:196:16: note: overridden virtual function is here
    virtual void InitLeafClassStats(int best_split_index, LeafStat* left_stats,
    ^
3 warnings generated.
INFO: From Linking tensorflow/contrib/factorization/gen_gen_clustering_ops_py_wrappers_cc [for host]:
clang: warning: argument unused during compilation: '-pthread'
INFO: From Linking tensorflow/contrib/tensor_forest/gen_gen_tensor_forest_ops_py_wrappers_cc [for host]:
clang: warning: argument unused during compilation: '-pthread'
INFO: From Linking tensorflow/python/gen_script_ops_py_wrappers_cc [for host]:
clang: warning: argument unused during compilation: '-pthread'
INFO: From Linking tensorflow/python/gen_checkpoint_ops_py_wrappers_cc [for host]:
clang: warning: argument unused during compilation: '-pthread'
INFO: From Linking tensorflow/contrib/layers/gen_sparse_feature_cross_op_py_wrappers_cc [for host]:
clang: warning: argument unused during compilation: '-pthread'
INFO: From Linking tensorflow/contrib/stateless/gen_stateless_random_ops_py_wrappers_cc [for host]:
clang: warning: argument unused during compilation: '-pthread'
INFO: From Linking tensorflow/contrib/tensor_forest/gen_gen_stats_ops_py_py_wrappers_cc [for host]:
clang: warning: argument unused during compilation: '-pthread'
INFO: From Linking tensorflow/contrib/summary/gen_gen_summary_ops_py_wrappers_cc [for host]:
clang: warning: argument unused during compilation: '-pthread'
Target //tensorflow/examples/image_retraining:retrain up-to-date:
  bazel-bin/tensorflow/examples/image_retraining/retrain
INFO: Elapsed time: 3470.871s, Critical Path: 119.68s
(tensorflow) Arlettes-MBP:tensorflow arlettecalvo$
```





```
(tensorflow) Arlettes-MBP:tensorflow arlettecalvo$ bazel-bin/tensorflow/examples/
/image_retraining/retrain --image_dir ~/flower_photos
INFO:tensorflow:Looking for images in 'daisy'
INFO:tensorflow:Looking for images in 'dandelion'
INFO:tensorflow:Looking for images in 'roses'
INFO:tensorflow:Looking for images in 'sunflowers'
INFO:tensorflow:Looking for images in 'tulips'
2017-11-05 01:44:58.892781: I tensorflow/core/platform/cpu_feature_guard.cc:137]
Your CPU supports instructions that this TensorFlow binary was not compiled to
use: SSE4.1 SSE4.2 AVX AVX2 FMA
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/daisy/10172379554_b296050
f82_n.jpg_inception_v3.txt
2017-11-05 01:45:00.050751: W tensorflow/core/framework/op_def_util.cc:334] Op B
atchNormWithGlobalNormalization is deprecated. It will cease to work in GraphDef
version 9. Use tf.nn.batch_normalization().
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/daisy/10172636503_21beded
a75_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/daisy/1031799732_e7f4008c
03.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/daisy/10391248763_1d16681
106_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/daisy/10437754174_22ec990
b77_m.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/daisy/10437770546_8bb6f7b
dd3_m.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/daisy/10437929963_bc13eeb
e0c.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/daisy/10466290366_cc72e33
532.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/daisy/10466558316_a7198b8
7e2.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/daisy/10555749515_13a12a0
26e.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/daisy/10555815624_dc21156
9b0.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/daisy/10555826524_423eb8b
f71_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/daisy/105806915_a9c13e210
6_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/daisy/10712722853_5632165
b04.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/daisy/107592979_aaa9cdf7
8_m.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck/daisy/10770585085_4742b9d
```

```
INFO:tensorflow:2017-11-05 02:14:14.900511: Step 3900: Cross entropy = 0.153549
INFO:tensorflow:2017-11-05 02:14:14.985011: Step 3900: Validation accuracy = 88.0% (N=100)
INFO:tensorflow:2017-11-05 02:14:15.790333: Step 3910: Train accuracy = 98.0%
INFO:tensorflow:2017-11-05 02:14:15.790481: Step 3910: Cross entropy = 0.134891
INFO:tensorflow:2017-11-05 02:14:15.881155: Step 3910: Validation accuracy = 90.0% (N=100)
INFO:tensorflow:2017-11-05 02:14:16.696419: Step 3920: Train accuracy = 97.0%
INFO:tensorflow:2017-11-05 02:14:16.696566: Step 3920: Cross entropy = 0.152133
INFO:tensorflow:2017-11-05 02:14:16.781464: Step 3920: Validation accuracy = 89.0% (N=100)
INFO:tensorflow:2017-11-05 02:14:17.588193: Step 3930: Train accuracy = 96.0%
INFO:tensorflow:2017-11-05 02:14:17.588338: Step 3930: Cross entropy = 0.187534
INFO:tensorflow:2017-11-05 02:14:17.680398: Step 3930: Validation accuracy = 93.0% (N=100)
INFO:tensorflow:2017-11-05 02:14:18.501802: Step 3940: Train accuracy = 98.0%
INFO:tensorflow:2017-11-05 02:14:18.501953: Step 3940: Cross entropy = 0.118826
INFO:tensorflow:2017-11-05 02:14:18.586637: Step 3940: Validation accuracy = 85.0% (N=100)
INFO:tensorflow:2017-11-05 02:14:19.383053: Step 3950: Train accuracy = 94.0%
INFO:tensorflow:2017-11-05 02:14:19.383199: Step 3950: Cross entropy = 0.164838
INFO:tensorflow:2017-11-05 02:14:19.466693: Step 3950: Validation accuracy = 91.0% (N=100)
INFO:tensorflow:2017-11-05 02:14:20.266980: Step 3960: Train accuracy = 99.0%
INFO:tensorflow:2017-11-05 02:14:20.267130: Step 3960: Cross entropy = 0.104074
INFO:tensorflow:2017-11-05 02:14:20.351415: Step 3960: Validation accuracy = 91.0% (N=100)
INFO:tensorflow:2017-11-05 02:14:21.150184: Step 3970: Train accuracy = 99.0%
INFO:tensorflow:2017-11-05 02:14:21.150322: Step 3970: Cross entropy = 0.087620
INFO:tensorflow:2017-11-05 02:14:21.234507: Step 3970: Validation accuracy = 89.0% (N=100)
INFO:tensorflow:2017-11-05 02:14:22.068576: Step 3980: Train accuracy = 94.0%
INFO:tensorflow:2017-11-05 02:14:22.068729: Step 3980: Cross entropy = 0.166022
INFO:tensorflow:2017-11-05 02:14:22.153201: Step 3980: Validation accuracy = 91.0% (N=100)
INFO:tensorflow:2017-11-05 02:14:22.964672: Step 3990: Train accuracy = 99.0%
INFO:tensorflow:2017-11-05 02:14:22.964811: Step 3990: Cross entropy = 0.101018
INFO:tensorflow:2017-11-05 02:14:23.053157: Step 3990: Validation accuracy = 85.0% (N=100)
INFO:tensorflow:2017-11-05 02:14:23.780875: Step 3999: Train accuracy = 99.0%
INFO:tensorflow:2017-11-05 02:14:23.781018: Step 3999: Cross entropy = 0.110239
INFO:tensorflow:2017-11-05 02:14:23.866053: Step 3999: Validation accuracy = 90.0% (N=100)
INFO:tensorflow:Final test accuracy = 90.4% (N=355)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.
(tensorflow) Arlettes-MBP:tensorflow arlettecalvo$
```

```
(tensorflow) Arlettes-MBP:tensorflow arlettecalvo$ bazel build tensorflow/examples/label_image:label_image && \
> bazel-bin/tensorflow/examples/label_image/label_image \
> --graph=/Users/arlettecalvo/tensorflow/tensorflow/tmp/retrained_graph.pb \
> --labels=/Users/arlettecalvo/tensorflow/tensorflow/tmp/retrained_labels.txt \
> --output_layer=final_result:0 \
> --image=/Users/arlettecalvo/tensorflow/tensorflow/flower_photos/daisy/5547758_ee9edfd54_n.jpg
```



## • A Guide to TF Layers: Building a Convolutional Neural Network

### En qué consiste:

Este tutorial consiste en mostrar las herramientas que posee TensorFlow para la creación de una red neuronal convolucional, donde se proporcionan métodos que facilitan la creación de los distintos “layers” necesarios para su elaboración.

Sin embargo es importante conocer en qué consiste y cómo se comporta una red neuronal convolucional. Dichas neuronas tienen pesos que permiten aprender, es decir cada neurona recibe entradas, realiza un producto escalar y posteriormente aplica la función de activación.

Por otro lado lo que diferencia a las redes neuronales convolucionales del resto, es que supone que las entradas son imágenes, por lo que se pueden codificar ciertas propiedades en la arquitectura, por otro lado su mayor aporte consiste en que permiten escalar bien para imágenes de mucha definición, ya que ellas trabajan modelando pequeñas piezas de información y posteriormente combinan esta información en las capas más profundas de la red.

Por ejemplo la primera capa intenta detectar los bordes y establecer patrones de detección de bordes, seguidamente las capas posteriores tratan de combinarlos en formas más simples y finalmente en patrones relacionados con iluminación, escalas, entre otras.

De esta forma las capas finales tratan de hacer coincidir la imagen de entrada con los patrones y así realizar una predicción final.

### Estructura

#### Capa convolucional

Implementa la operación de convolución, la cual recibe como entrada o input la imagen y luego aplica sobre ella un filtro que devuelve un mapa de las características de la imagen original, de esta forma se logra reducir el tamaño de los parámetros.

#### Capa de pooling

Reduce la cantidad de parámetros al quedarse con las características más comunes. Su utilidad principal radica en la reducción de las dimensiones espaciales (ancho x alto). La operación realizada por esta capa también se llama reducción de muestreo. La operación que utiliza es max-pooling, que divide a la imagen de entrada en un conjunto de rectángulos y, respecto de cada subregión, se va quedando con el máximo valor.

#### Capa convolucional

En esta capa cada nodo está conectado a cada nodo en la capa anterior, dicha capa es la encargada de dar el resultado final de la red.

## Conclusiones:

En este tutorial TensorFlow muestra las herramientas necesarias para poder crear una red neuronal convolucional, las cuales tienen como principal característica la utilización de layers convolucionales, lo que hace que esta arquitectura sea adecuada para procesar datos como imágenes de alta definición.

Por otro lado, dicha red neuronal funciona extrayendo las características directamente de las imágenes, es decir las características relevantes se aprenden mientras la red se entrena con la colección de imágenes, de esta forma suelen ser precisas para tareas como clasificación de objetos.

En su aprendizaje tienen la capacidad de detectar las características de las imágenes por medio de capas ocultas, donde cada una de estas capas aumenta la complejidad de las características aprendidas, por ejemplo se puede tener una capa que tenga la habilidad para detectar bordes, y la siguiente capa detecta formas más complejas pertenecientes a la forma del objeto.

## Cómo podría aplicarlo a un problema real:

La implementación de este tipo de red neuronal, puede ser de gran utilidad para tareas como por ejemplo la detección de objetos en una imagen.

Donde se suministra como dato de entrada una imagen por ejemplo tomada por medio de una webcam, y a partir de ella, mediante la utilización de las herramientas que provee TensorFlow para crear la red, se puede detectar un objeto en particular que se encuentre en dicha imagen, incluso se podría reconocer y al mismo tiempo localizar el objeto dentro de la escena.





## Screenshots de resultados:

Para la realización de este tutorial, se fueron implementando cada uno de los pasos propuestos en la documentación, donde se fueron integrando poco a poco cada una de las capas que conforman la estructura de las redes neuronales convolucionales.

```

((tensorflow) Arlette-MBP:layers arlettecalvo$ python cnn_mnist.py
/Users/arlettecalvo/tensorflow/lib/python3.6/importlib/_bootstrap.py:205: RuntimeWarning: compiletime version 3.5 of module 'tensorflow.python.framework.fast_tensor_util' does not match runtime version 3.6
  return f(*args, **kwargs)
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Extracting MNIST-data/train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Extracting MNIST-data/train-labels-idx1-ubyte.gz
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Extracting MNIST-data/t10k-images-idx3-ubyte.gz
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting MNIST-data/t10k-labels-idx1-ubyte.gz
INFO:tensorflow:Using default config.
INFO:tensorflow:Using config: {'_model_dir': '/tmp/mnist_convnet_model', '_tf_random_seed': None, '_save_summary_steps': 100, '_save_checkpoints_steps': None, '_save_checkpoints_secs': 600, '_session_config': None, '_keep_checkpoint_max': 5, '_keep_checkpoint_every_n_hours': 10000, '_log_step_count_steps': 100, '_service': None, '_cluster_spec': <tensorflow.python.training.server_lib.ClusterSpec object at 0x11da2ce10>, '_task_type': 'worker', '_task_id': 0, '_master': '', '_is_chief': True, '_num_ps_replicas': 0, '_num_worker_replicas': 1}
INFO:tensorflow:Create CheckpointSaverHook.
2017-11-07 20:04:32.089236: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2 AVX AVX2 FMA
INFO:tensorflow:Saving checkpoints for 1 into /tmp/mnist_convnet_model/model.ckpt.
INFO:tensorflow:probabilities = [[ 0.09661149  0.08470545  0.09772027  0.0909756
6  0.10413428  0.09479724
0.12373412  0.09395565  0.10386731  0.10949861]
[ 0.09446325  0.08900566  0.10809064  0.08510928  0.10012916  0.1024765
0.11257289  0.11092173  0.09852966  0.09870125]
[ 0.08645773  0.09326914  0.0973303  0.09001478  0.09600638  0.11055008
0.12151684  0.08924329  0.10775803  0.1078807 ]
[ 0.08793498  0.08877057  0.10345381  0.08926101  0.0985187  0.09644746
0.12696566  0.11267462  0.09105628  0.10491689]
[ 0.10393142  0.09618924  0.10657361  0.0814495  0.10489011  0.09277189
0.11726162  0.10073044  0.09432571  0.1018765 ]
[ 0.08273648  0.08454642  0.09524971  0.08302297  0.09357341  0.11278751
0.12499224  0.11391687  0.10753945  0.10163491]
[ 0.08726073  0.08366025  0.10671486  0.08950628  0.10345088  0.09739354
0.11998692  0.10391654  0.09632093  0.11178917]
[ 0.10040507  0.09445368  0.10262921  0.0866302  0.09690864  0.09728413
0.10938949  0.11458721  0.10157906  0.09613331]
[ 0.09880416  0.08582152  0.09651861  0.08852258  0.11568794  0.11069257
0.10381226  0.1004044  0.09798342  0.10175253]
[ 0.09189931  0.08673891  0.10826387  0.08526222  0.11016753  0.09492676
0.10724043  0.10192184  0.09809877  0.11548035]

```

```

[ 0.10134247  0.08438668  0.09920067  0.0846299  0.09683035  0.102452
0.12489346  0.10523134  0.10760296  0.09343018]
[ 0.09512428  0.08941974  0.09842832  0.09133635  0.10849532  0.09640887
0.11377541  0.11061117  0.09376858  0.10263202]
[ 0.09641024  0.08698151  0.10285407  0.09459777  0.10905496  0.09680203
0.10459308  0.1042924  0.09484923  0.10956462]
[ 0.09999481  0.09341143  0.08449947  0.08804601  0.09644812  0.10638045
0.12405583  0.09659092  0.101982  0.10859095]
[ 0.09041615  0.08943769  0.08983482  0.08649927  0.10696669  0.11976649
0.11997219  0.09651117  0.09011312  0.11048248]
[ 0.08797459  0.08020084  0.10216099  0.0844707  0.10015182  0.10761537
0.12180817  0.10362974  0.09942795  0.11255981]
[ 0.0937355  0.09036404  0.10085846  0.08522729  0.10538927  0.10187453
0.11892959  0.1020714  0.09416159  0.1073883 ]
[ 0.09323391  0.10146927  0.09720511  0.09710792  0.10158339  0.09881153
0.11612549  0.09885389  0.09438586  0.10122357]
[ 0.09377895  0.08866724  0.0946934  0.08664039  0.09916418  0.10489836
0.12276738  0.09436199  0.10094507  0.11408299]
[ 0.09182349  0.09763707  0.09611762  0.0874185  0.1153401  0.09897448
0.1159618  0.09591481  0.09909347  0.10171857]
[ 0.08637029  0.08966648  0.09430235  0.08700932  0.11191935  0.10643286
0.11099494  0.10863522  0.10015128  0.10451791]
[ 0.08822019  0.08235773  0.09560703  0.0933839  0.09563401  0.10195183
0.12788232  0.11392076  0.10098041  0.10006187]]
INFO:tensorflow:loss = 2.32745, step = 1
INFO:tensorflow:probabilities = [[ 0.09443525  0.09235686  0.09710532  0.0912212
0.10612607  0.09930597
0.11437572  0.09338839  0.10079223  0.11089294]
[ 0.09423836  0.0846092  0.09247828  0.09399502  0.09644049  0.11113378
0.11662102  0.09680489  0.10285018  0.11082881]
[ 0.08971711  0.08741576  0.10494275  0.09215605  0.10241432  0.09940848
0.11935811  0.10191733  0.0912258  0.11144425]
[ 0.08754249  0.08430976  0.08959094  0.10091976  0.09988995  0.10136399
0.1282281  0.09571458  0.10183493  0.1106056 ]
[ 0.08807259  0.07696068  0.09322581  0.07982317  0.0987344  0.09652867
0.12812594  0.09949179  0.11302944  0.1260075 ]
[ 0.09729845  0.09291861  0.09633313  0.09278407  0.09879669  0.10025654
0.10654815  0.10860372  0.09805005  0.10841064]
[ 0.08499899  0.08250357  0.09328204  0.08887621  0.10849851  0.11003654
0.11804375  0.10524411  0.1068329 ]
[ 0.09132607  0.09115724  0.09108277  0.10529627  0.10630714

```





```

0.09971227 0.09647719 0.11715774 0.09641208]
[ 0.09713524 0.0952493 0.10670291 0.10435195 0.08071009 0.09000757
0.1235824 0.09613356 0.1028542 0.10327286]
[ 0.10072438 0.0845248 0.10251377 0.1036422 0.10091147 0.08732437
0.12490626 0.08923899 0.09821203 0.10800163]
[ 0.10371023 0.0739079 0.09874367 0.09556806 0.09332055 0.09745985
0.11088886 0.11302347 0.10304233 0.11033504]
[ 0.09846732 0.09042361 0.10617527 0.10170634 0.10311354 0.10410955
0.09079629 0.10345495 0.09153593 0.11021724]
[ 0.10443573 0.08107452 0.09991424 0.09136084 0.10977358 0.08571918
0.12642211 0.10373306 0.10327934 0.09428748]
[ 0.09468131 0.08124092 0.09344577 0.10686202 0.11189817 0.10255767
0.1055042 0.1096941 0.10099258 0.09312333]
[ 0.09432245 0.07857399 0.094258 0.09556512 0.10472681 0.09021597
0.13741596 0.10710515 0.09793847 0.09987814]
[ 0.08623835 0.09779352 0.10782824 0.09412313 0.09653948 0.09502088
0.10647948 0.10205358 0.09727883 0.11664452]
[ 0.09229659 0.07439929 0.09846915 0.09240314 0.0918277 0.0903488
0.10837387 0.11509851 0.11637005 0.12041293]] (17.354 sec)
INFO:tensorflow:global_step/sec: 2.74089
INFO:tensorflow:probabilities = [[ 0.10925651 0.07823411 0.10761859 0.0933912
7 0.11340723 0.08325602
0.12526575 0.08821815 0.09527916 0.10607319]
[ 0.10287247 0.09189432 0.08703291 0.09496619 0.0917963 0.10594352
0.1100787 0.10578961 0.09865685 0.11096911]
[ 0.13020016 0.08234882 0.09047787 0.10726811 0.08211603 0.08682269
0.0917752 0.11364348 0.1128705 0.10247713]
[ 0.09252109 0.08415338 0.10931323 0.11241354 0.0988814 0.08129206
0.10645187 0.0932337 0.1121205 0.10961913]
[ 0.10309401 0.10284846 0.10276636 0.08226819 0.0893483 0.08369628
0.11421023 0.10635123 0.11407005 0.10134692]
[ 0.10291781 0.07962814 0.09817061 0.09470601 0.09014932 0.09603243
0.12191842 0.10007419 0.10451189 0.1118912 ]
[ 0.0961044 0.09593052 0.10714141 0.09084704 0.11657295 0.09985255
0.10574231 0.09170352 0.08777511 0.10833018]
[ 0.08934575 0.09054033 0.10092254 0.09972746 0.09554076 0.09913754
0.11087298 0.09671185 0.09154044 0.12566033]
[ 0.09518832 0.08789329 0.08724775 0.0975979 0.09018008 0.10677484
0.10883647 0.10427425 0.11012132 0.11188575]
[ 0.09356147 0.09285165 0.10118181 0.10979062 0.08618987 0.09592341
0.10937332 0.11086674 0.10129164 0.09896936]
[ 0.09129494 0.09956099 0.10728484 0.09224089 0.09054427 0.10199083
0.11120454 0.0909053 0.10757433 0.10739907]
[ 0.09770199 0.10542252 0.10345115 0.0874104 0.10418712 0.09502845
0.10531761 0.09342545 0.09378914 0.11426613]
[ 0.10326364 0.08806758 0.11019574 0.09615611 0.11566193 0.09341284
0.10737017 0.08643411 0.09682646 0.10261137]
[ 0.10198525 0.07830295 0.10589073 0.09314691 0.09085161 0.08728854
0.13428506 0.09435574 0.10848127 0.1054119 ]

```

```

0.00000174 0.00000274 0.00000402 0.00000415]
[ 0.01216274 0.05593186 0.01762053 0.17329983 0.00002993 0.10916392
0.00011636 0.04059076 0.58350796 0.00757607]
[ 0.01069001 0.00015373 0.01111956 0.00072013 0.95476919 0.00133415
0.00660439 0.00038158 0.00007464 0.01415255]
[ 0.00195093 0.00018698 0.00003407 0.0182068 0.00000005 0.9793092
0.00000084 0.0000064 0.00027705 0.0000276 ]
[ 0.00000045 0.0000268 0.99799746 0.00167739 0. 0.00000049
0.00000001 0.00003362 0.0002637 0.00000009]
[ 0.00007087 0.00000018 0.00014622 0.99915898 0.00000011 0.00045618
0.00000025 0.00000002 0.00014846 0.00001881]] (19.961 sec)
INFO:tensorflow:loss = 0.155308, step = 19901 (35.908 sec)
INFO:tensorflow:probabilities = [[ 0.00001286 0.00002142 0.00023233 0.0000007
4 0.99887663 0.00008745
0.00018732 0.00000942 0.00009634 0.00047552]
[ 0.00000385 0.00000732 0.00001476 0.0023517 0.02938445 0.00000566
0.00000025 0.0042118 0.00240133 0.96161884]
[ 0.0001234 0.00114586 0.00190061 0.00007076 0.00071061 0.01276058
0.00000000 0.00000000 0.00000000 0.00000000]
[ 0.99985218 0.00000002 0.00000744 0.00000002 0.00000002 0.00012497
0.00001432 0.00000046 0.00000024 0.00000021]
[ 0.00074183 0.00003069 0.00064578 0.00001318 0.00016548 0.00119033
0.99689972 0.00000002 0.00031139 0.00000166]
[ 0.00002721 0.00006697 0.00286221 0.00024964 0.00010643 0.0121722
0.01317988 0.00000006 0.97103888 0.00029596]
[ 0.00026023 0.00007395 0.00011843 0.0144628 0.00165376 0.81414825
0.00007227 0.00214516 0.0551296 0.11193556]
[ 0.00000012 0.00000141 0.00000088 0.0000208 0.00000005 0.99897003
0.00036429 0. 0.00064211 0.00000027]
[ 0.00000033 0.00000032 0.00000196 0.00000726 0.98858368 0.00001748
0.00000658 0.00007668 0.00002254 0.01128328]
[ 0.99908018 0.00000001 0.00009099 0.00000022 0.00002028 0.00000297
0.00078416 0.00000303 0.00000316 0.00001509]] (18.568 sec)
INFO:tensorflow:global_step/sec: 2.67089
INFO:tensorflow:probabilities = [[ 0.99765778 0.0000007 0.00168171 0.0001913
4 0. 0.00030237
0.00000462 0.00011716 0.00000526 0.00003913]
[ 0.00000338 0.99394751 0.00075775 0.00060535 0.00053064 0.00003069
0.00010213 0.00347937 0.00049417 0.00004893]
[ 0.00002107 0.00000092 0.00001467 0.00001208 0.00002898 0.00000672
0. 0.98921877 0.00000723 0.0106895 ]
[ 0.00000001 0.00000009 0.00000016 0.00000003 0.99993432 0.00000027

```



```

0.00000001 0.00026591 0.00065114 0.99768984]
[ 0.00000478 0.00000539 0.00083254 0.00043797 0.00047278 0.00008781
 0.00000288 0.01613515 0.00005843 0.98196226]
[ 0.00003434 0.00007466 0.00000775 0.00012924 0.00000225 0.01574442
 0.00000795 0.00001691 0.98384029 0.00014215]
[ 0.00010768 0.00001839 0.0000085 0.00001846 0.00007619 0.00149419
 0.99705648 0.00000003 0.00120101 0.00001909]
[ 0.00000006 0.00024906 0.99968171 0.00006638 0. 0.00000033
 0.00000022 0.00000003 0.00000213 0. ]
[ 0.00000806 0.00000026 0.00000107 0.00010628 0.00006109 0.99492466
 0.00004737 0.00000004 0.00460227 0.00024863]
[ 0.00003404 0.99728942 0.00062937 0.00003092 0.00117571 0.00001402
 0.0006135 0.00007107 0.00007822 0.00006373]
[ 0.00000074 0.00000071 0.99898571 0.00071335 0.00011915 0.00000252
 0.00000004 0.00001104 0.00007589 0.00009086]
[ 0.00010063 0.00024309 0.00178383 0.02234573 0.00010673 0.00650512
 0.00086065 0.00005281 0.96766627 0.00033507]
[ 0.00004634 0.00000023 0.00008882 0.00014297 0.00007322 0.00972355
 0.00001595 0.00000159 0.98452234 0.00538499]
[ 0.71203864 0.00009177 0.03700171 0.00947987 0.00931684 0.03096795
 0.00902356 0.00537638 0.05362326 0.13308005]
[ 0.1147263 0.00821958 0.05082263 0.0064401 0.0010412 0.57268721
 0.03282844 0.00337477 0.20980297 0.00005678]
[ 0.00000011 0.99963808 0.00014138 0.00002489 0.00000349 0.00000719
 0.0000101 0.00000394 0.00016948 0.00000139]
[ 0.00000005 0.00003786 0.99363673 0.00561296 0.00045175 0.00000347
 0.00003211 0.00000001 0.00022348 0.00000168]
[ 0.00000276 0.00000186 0.00000134 0.00006277 0.01327347 0.00003576
 0.00000004 0.00947599 0.00000927 0.97713667]] (17.579 sec)
INFO:tensorflow:Saving checkpoints for 20000 into /tmp/mnist_convnet_model/model.ckpt.
INFO:tensorflow:Loss for final step: 0.273784.
INFO:tensorflow:Starting evaluation at 2017-11-08-04:02:38
INFO:tensorflow:Restoring parameters from /tmp/mnist_convnet_model/model.ckpt-20000
INFO:tensorflow:Finished evaluation at 2017-11-08-04:02:51
INFO:tensorflow:Saving dict for global step 20000: accuracy = 0.9694, global_step = 20000, loss = 0.103746
{'accuracy': 0.96939999, 'loss': 0.1037456, 'global_step': 20000}
(tensorflow) Arlettes-MBP:layers arlettecalvo$

```





## • Convolutional Neural Networks

### En qué consiste:

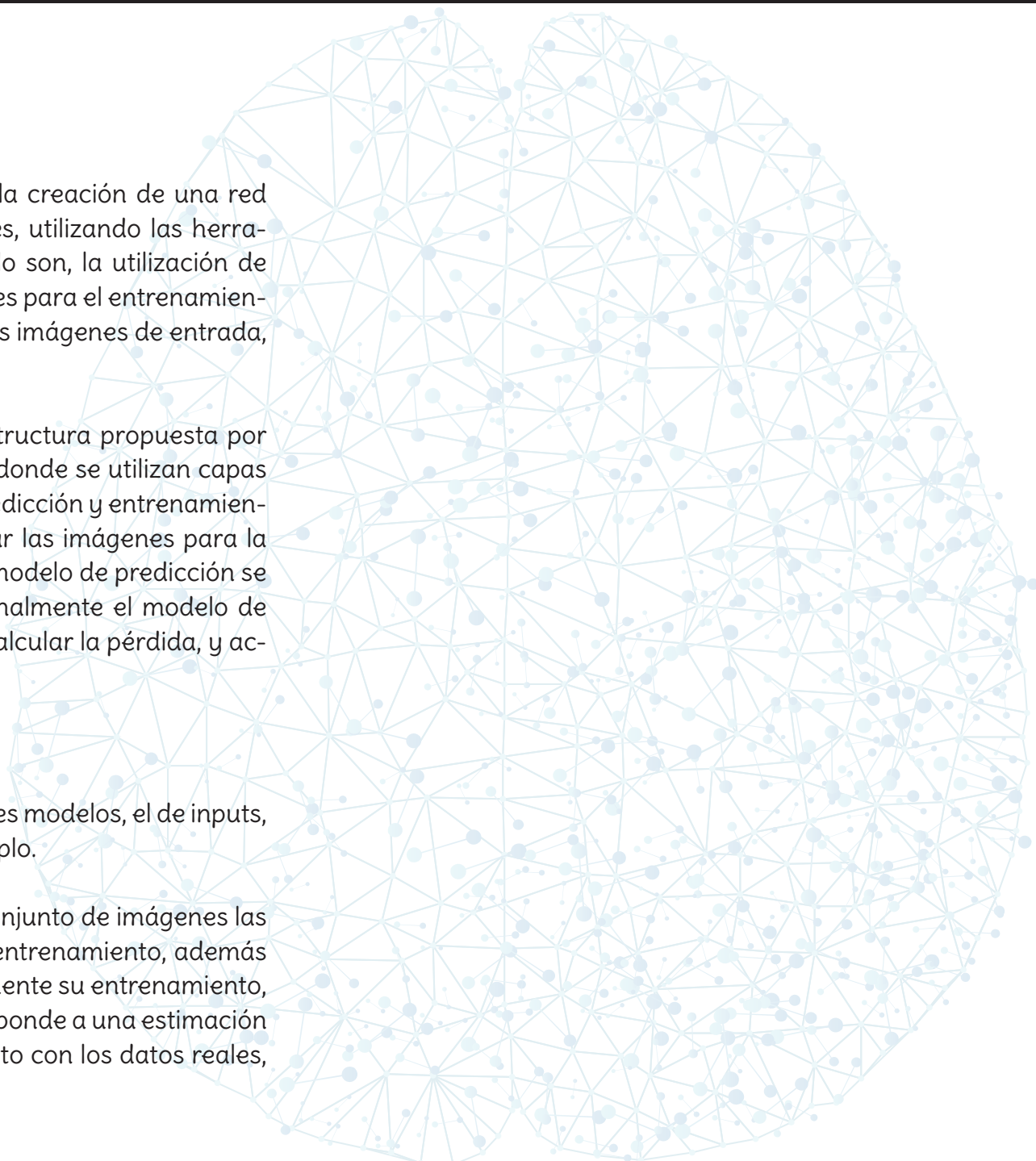
El principal objetivo de este tutorial consiste en la creación de una red neuronal convolucional para reconocer imágenes, utilizando las herramientas proporcionadas por TensorFlow, como lo son, la utilización de componentes matemáticos, además de actividades para el entrenamiento de la red dentro de las cuales podemos citar, las imágenes de entrada, funciones de activación entre otros.

Para la ejecución de este tutorial se utiliza la estructura propuesta por TensorFlow para la creación de este tipo de red, donde se utilizan capas "layers", además se utilizan modelos de inputs, predicción y entrenamiento. El primero de ellos se encarga de preprocesar las imágenes para la evaluación y el entrenamiento, seguidamente el modelo de predicción se encarga de la clasificación de las imágenes y finalmente el modelo de entrenamiento incluye operaciones para poder calcular la pérdida, y actualizaciones de las variables.

### Conclusiones:

En este tutorial se puede ver la utilización de los tres modelos, el de inputs, predicción y entrenamientos aplicados a un ejemplo.

Dentro de la ejecución del tutorial se tienen un conjunto de imágenes las cuales son preprocesadas para su evaluación y entrenamiento, además se cuenta con un proceso de clasificación y finalmente su entrenamiento, donde se calcula el valor de pérdida el cual corresponde a una estimación de que tan cerca están los datos de entrenamiento con los datos reales, con los cuales se calculan los pesos.





Así mismo se cuenta con un optimizador, el cual utiliza los valores de pérdida calculados para poder actualizar los pesos y sesgos en el backpropagation.

## Cómo podría aplicarlo a un problema real:

La utilización de este tipo de red neuronal, puede ser de gran ayuda en aplicaciones que necesiten identificar elementos únicos dentro de una imagen.

Por ejemplo este tipo de red puede ser implementada en sistemas que necesiten reconocimiento facial, ya sea para dar permisos en cuanto a la utilización de alguna aplicación como un sistema de entrada, o simplemente para algún tipo de clasificación de contenido.

## Screenshots de resultados:

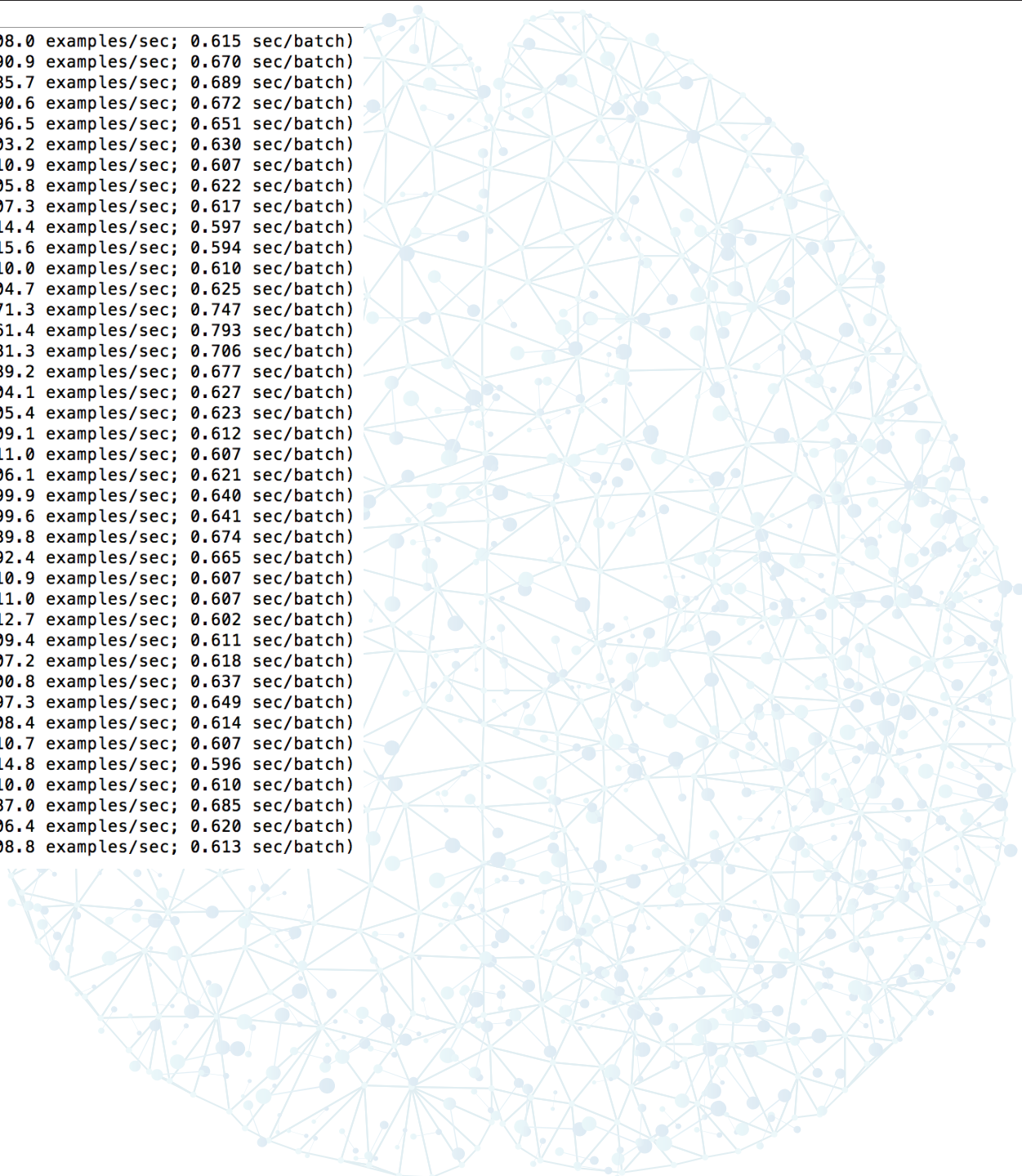
Para la realización de este tutorial, se fueron implementando cada uno de los pasos propuestos en la documentación, donde se ejecutaron cada uno de los archivos segun los modelos propuestos en el tutorial.

```
(tensorflow) Arlettes-MacBook-Pro:cifar10 arlettecalvo$ python cifar10_input.py
/Users/arlettecalvo/tensorflow/lib/python3.6/importlib/_bootstrap.py:205: RuntimeWarning:
[ compiletime version 3.5 of module 'tensorflow.python.framework.fast_tensor_util' does not
match runtime version 3.6
  return f(*args, **kwargs)
(tensorflow) Arlettes-MacBook-Pro:cifar10 arlettecalvo$ python cifar10.py
/Users/arlettecalvo/tensorflow/lib/python3.6/importlib/_bootstrap.py:205: RuntimeWarning:
[ compiletime version 3.5 of module 'tensorflow.python.framework.fast_tensor_util' does not
match runtime version 3.6
  return f(*args, **kwargs)
(tensorflow) Arlettes-MacBook-Pro:cifar10 arlettecalvo$ python cifar10_train.py
/Users/arlettecalvo/tensorflow/lib/python3.6/importlib/_bootstrap.py:205: RuntimeWarning:
[ compiletime version 3.5 of module 'tensorflow.python.framework.fast_tensor_util' does not
match runtime version 3.6
  return f(*args, **kwargs)
>> Downloading cifar-10-binary.tar.gz 100.0%
Successfully downloaded cifar-10-binary.tar.gz 170052171 bytes.
WARNING:tensorflow:From cifar10_train.py:64: get_or_create_global_step (from tensorflow.c
ontrib.framework.python.ops.variables) is deprecated and will be removed in a future vers
ion.
Instructions for updating:
Please switch to tf.train.get_or_create_global_step
Filling queue with 20000 CIFAR images before starting to train. This will take a few minu
tes.
2017-11-15 16:19:21.800318: I tensorflow/core/platform/cpu_feature_guard.cc:137] Your CPU
supports instructions that this TensorFlow binary was not compiled to use: SSE4.1 SSE4.2
AVX AVX2 FMA
2017-11-15 16:19:29.005215: step 0, loss = 4.67 (171.5 examples/sec; 0.746 sec/batch)
2017-11-15 16:19:35.417174: step 10, loss = 4.61 (199.6 examples/sec; 0.641 sec/batch)
2017-11-15 16:19:41.316473: step 20, loss = 4.44 (217.0 examples/sec; 0.590 sec/batch)
```



```
2017-11-15 17:43:30.193038: step 7080, loss = 1.01 (208.0 examples/sec; 0.615 sec/batch)
2017-11-15 17:43:36.897774: step 7090, loss = 0.82 (190.9 examples/sec; 0.670 sec/batch)
2017-11-15 17:43:43.790111: step 7100, loss = 0.95 (185.7 examples/sec; 0.689 sec/batch)
2017-11-15 17:43:50.505953: step 7110, loss = 1.08 (190.6 examples/sec; 0.672 sec/batch)
2017-11-15 17:43:57.020912: step 7120, loss = 1.20 (196.5 examples/sec; 0.651 sec/batch)
2017-11-15 17:44:03.318691: step 7130, loss = 0.98 (203.2 examples/sec; 0.630 sec/batch)
2017-11-15 17:44:09.388819: step 7140, loss = 0.83 (210.9 examples/sec; 0.607 sec/batch)
2017-11-15 17:44:15.609911: step 7150, loss = 0.92 (205.8 examples/sec; 0.622 sec/batch)
2017-11-15 17:44:21.783053: step 7160, loss = 0.92 (207.3 examples/sec; 0.617 sec/batch)
2017-11-15 17:44:27.751959: step 7170, loss = 0.92 (214.4 examples/sec; 0.597 sec/batch)
2017-11-15 17:44:33.689963: step 7180, loss = 0.84 (215.6 examples/sec; 0.594 sec/batch)
2017-11-15 17:44:39.786302: step 7190, loss = 0.99 (210.0 examples/sec; 0.610 sec/batch)
2017-11-15 17:44:46.038237: step 7200, loss = 0.84 (204.7 examples/sec; 0.625 sec/batch)
2017-11-15 17:44:53.512451: step 7210, loss = 0.96 (171.3 examples/sec; 0.747 sec/batch)
2017-11-15 17:45:01.443809: step 7220, loss = 1.09 (161.4 examples/sec; 0.793 sec/batch)
2017-11-15 17:45:08.505452: step 7230, loss = 0.85 (181.3 examples/sec; 0.706 sec/batch)
2017-11-15 17:45:15.271478: step 7240, loss = 0.86 (189.2 examples/sec; 0.677 sec/batch)
2017-11-15 17:45:21.542173: step 7250, loss = 0.76 (204.1 examples/sec; 0.627 sec/batch)
2017-11-15 17:45:27.774732: step 7260, loss = 1.02 (205.4 examples/sec; 0.623 sec/batch)
2017-11-15 17:45:33.894746: step 7270, loss = 0.80 (209.1 examples/sec; 0.612 sec/batch)
2017-11-15 17:45:39.960299: step 7280, loss = 0.88 (211.0 examples/sec; 0.607 sec/batch)
2017-11-15 17:45:46.170581: step 7290, loss = 0.86 (206.1 examples/sec; 0.621 sec/batch)
2017-11-15 17:45:52.574212: step 7300, loss = 1.00 (199.9 examples/sec; 0.640 sec/batch)
2017-11-15 17:45:58.986213: step 7310, loss = 1.07 (199.6 examples/sec; 0.641 sec/batch)
2017-11-15 17:46:05.730515: step 7320, loss = 0.91 (189.8 examples/sec; 0.674 sec/batch)
2017-11-15 17:46:12.381834: step 7330, loss = 1.00 (192.4 examples/sec; 0.665 sec/batch)
2017-11-15 17:46:18.451812: step 7340, loss = 0.79 (210.9 examples/sec; 0.607 sec/batch)
2017-11-15 17:46:24.519091: step 7350, loss = 0.95 (211.0 examples/sec; 0.607 sec/batch)
2017-11-15 17:46:30.537915: step 7360, loss = 1.18 (212.7 examples/sec; 0.602 sec/batch)
2017-11-15 17:46:36.649969: step 7370, loss = 0.91 (209.4 examples/sec; 0.611 sec/batch)
2017-11-15 17:46:42.827087: step 7380, loss = 1.00 (207.2 examples/sec; 0.618 sec/batch)
2017-11-15 17:46:49.200585: step 7390, loss = 1.14 (200.8 examples/sec; 0.637 sec/batch)
2017-11-15 17:46:55.689743: step 7400, loss = 0.91 (197.3 examples/sec; 0.649 sec/batch)
2017-11-15 17:47:01.830533: step 7410, loss = 1.01 (208.4 examples/sec; 0.614 sec/batch)
2017-11-15 17:47:07.904656: step 7420, loss = 0.83 (210.7 examples/sec; 0.607 sec/batch)
2017-11-15 17:47:13.864327: step 7430, loss = 0.92 (214.8 examples/sec; 0.596 sec/batch)
2017-11-15 17:47:19.960133: step 7440, loss = 0.81 (210.0 examples/sec; 0.610 sec/batch)
2017-11-15 17:47:26.805410: step 7450, loss = 0.94 (187.0 examples/sec; 0.685 sec/batch)
2017-11-15 17:47:33.007041: step 7460, loss = 0.86 (206.4 examples/sec; 0.620 sec/batch)
2017-11-15 17:47:39.137718: step 7470, loss = 0.85 (208.8 examples/sec; 0.613 sec/batch)
```

■



## • Vector Representations of Words

### En qué consiste:

Este tutorial consiste en la implementación del modelo propuesto por TensorFlow para poder aprender representaciones de palabras en vectores, es decir “incrustaciones de palabras”.

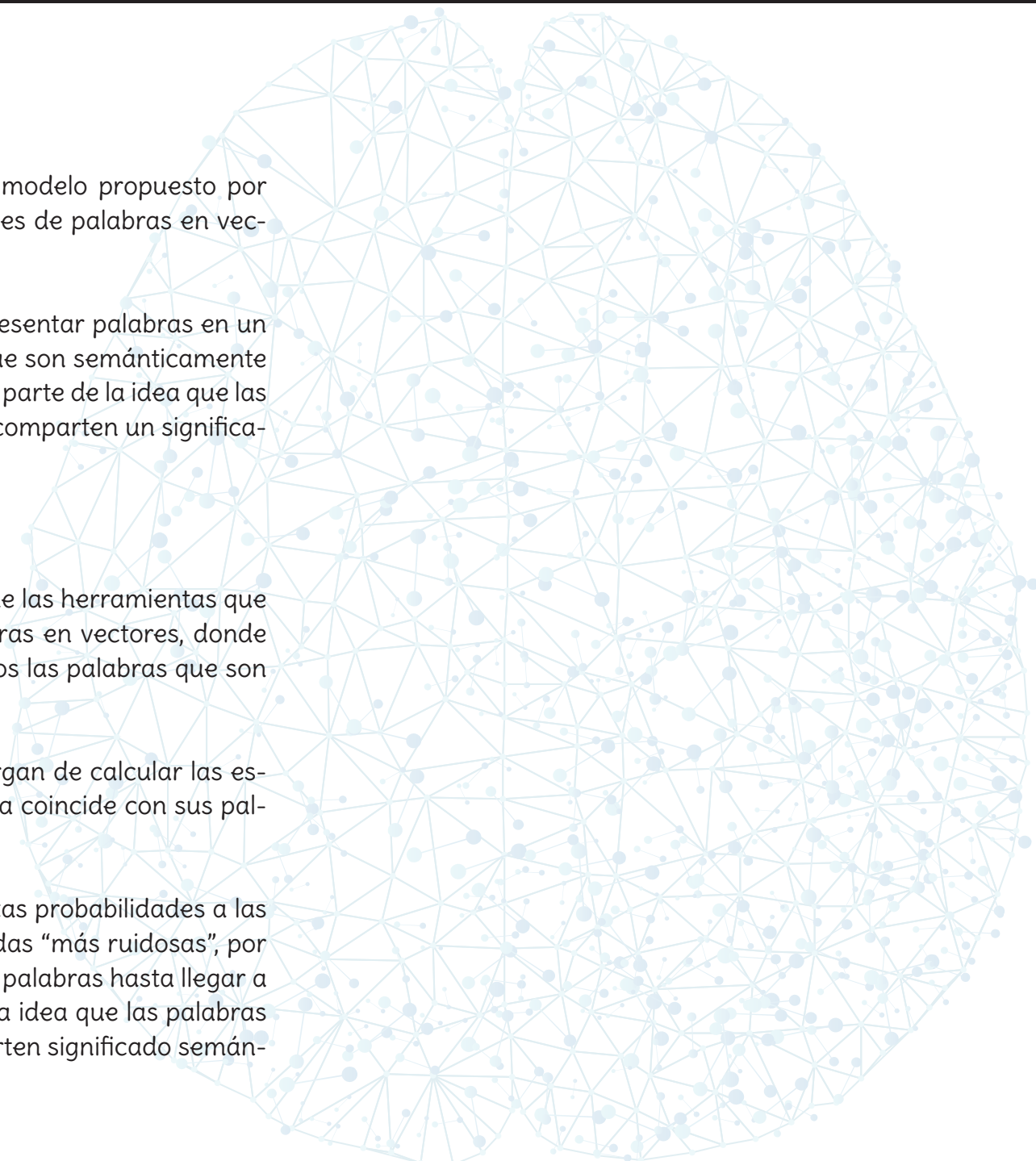
Es así como los modelos vectoriales pueden representar palabras en un espacio vectorial continuo, donde las palabras que son semánticamente similares se asignan a puntos cercanos, ya que se parte de la idea que las palabras que aparecen en los mismos contextos comparten un significado semántico.

### Conclusiones:

En este tutorial se puede ver la implementación de las herramientas que posee TensorFlow para poder representar palabras en vectores, donde la dinámica consiste en asignar a puntos cercanos las palabras que son semánticamente similares.

Por lo que se tienen métodos los cuales se encargan de calcular las estadísticas de la frecuencia con la que una palabra coincide con sus palabras vecinas.

Donde el modelo implementado va asignando altas probabilidades a las palabras reales y bajas a las palabras consideradas “más ruidosas”, por lo que se genera primero una mayor cantidad de palabras hasta llegar a una cantidad más reducida, ya que se parte de la idea que las palabras que se encuentran en contextos similares comparten significado semántico.





## Cómo podría aplicarlo a un problema real:

Una excelente implementación de este tipo de modelos es en sistemas de clasificación de texto, ya que al convertir el texto en un vector puede sumarse y restarse, es decir se puede manipular de otras formas, visto como un vector en el espacio.

Por otro lado a medida que la red neuronal recorre el documento va aprendiendo cómo representar el vocabulario en un formato que puede procesar en sus “layers ocultos” para poder ir prediciendo las palabras, donde el algoritmo aprende sobre las relaciones de cada uno de los términos del vocabulario.

Así mismo al finalizar se cuenta con vectores los cuales pueden sumar y restar, por lo cual es posible realizar ecuaciones de palabras, como por ejemplo si tenemos las palabras: Rey, Hombre y Mujer, al tomar el vector correspondiente a la palabra ‘Rey’ y se agrega a la diferencia resultante al restar Hombre y Mujer, el vector final es cercano a ‘Reina’.

## Screenshots de resultados:

A continuación se presentan algunas de las imágenes generadas al implementar el tutorial.

WORD2VEC 101: WHY IT'S SO IMPORTANT TO DATA SCIENCE

King + (Woman - Man) = ??

King + (Woman - Man) = Queen

```
[(tensorflow) Arlettes-MBP:word2vec arlettecalvo$ python word2vec_basic.py  
/Users/arlettecalvo/tensorflow/lib/python3.6/importlib/_bootstrap.py:205: Runtime  
Warning: compiletime version 3.5 of module 'tensorflow.python.framework.fast_te  
nsor_util' does not match runtime version 3.6  
    return f(*args, **kwargs)  
Found and verified text8.zip  
Data size 17005207  
Most common words (+UNK) [['UNK', 418391], ('the', 1061396), ('of', 593677), ('a  
nd', 416629), ('one', 411764)]  
Sample data [5234, 3081, 12, 6, 195, 2, 3134, 46, 59, 156] ['anarchism', 'origin  
ated', 'as', 'a', 'term', 'of', 'abuse', 'first', 'used', 'against']  
3081 originated -> 5234 anarchism  
3081 originated -> 12 as  
12 as -> 3081 originated  
12 as -> 6 a  
6 a -> 12 as  
6 a -> 195 term  
195 term -> 6 a  
195 term -> 2 of  
2017-11-07 22:49:27.061977: I tensorflow/core/platform/cpu_feature_guard.cc:137]  
Your CPU supports instructions that this TensorFlow binary was not compiled to  
use: SSE4.1 SSE4.2 AVX AVX2 FMA  
Initialized  
Average loss at step 0 : 280.888031006  
Nearest to had: vaccines, reprise, ganglia, fermentable, talal, featureless, add  
s, sanderson,  
Nearest to many: dramatized, detachments, hub, screenings, relinquished, hercule  
, kath, bryan,  
Nearest to called: bravery, marangella, alsatian, jodie, moneo, viola, consolida  
ted, yisrael,  
Nearest to during: technicolor, napalm, deadlines, resolvers, unregistered, folk  
s, resembling, basically,  
Nearest to with: korps, honorum, rium, cree, millennia, skim, ensemble, icd,  
Nearest to into: lambeth, trisha, inclusive, mactutor, overbearing, incitement,  
warming, uncontroversial,  
Nearest to there: prescription, participate, loglan, incarceration, judi, counta  
bly, asher, proteus,  
Nearest to an: antoinette, camden, perpetuated, streaming, irrelevant, christian  
ized, osteoarthritis, obvious,  
Nearest to when: narrative, immaturity, beauchamp, twitching, ethernet, rifles,  
participatory, bah,  
Nearest to d: dalek, stunt, trusting, morph, cho, robbery, kidder, graham,  
Nearest to for: reductase, neutralize, lemay, rifts, slovenians, clearances, som  
me, intv,  
Nearest to if: gamelan, siegfried, micrometer, ammo, durians, rectangular, packe  
d, marsilio,  
Nearest to states: flush, philadelphia, chances, jerusalem, reputation, framewor  
ks, cars, legislated,  
Nearest to first: mohiam, cohen, kossuth, reticent, identical, attempts, esc, cu
```

ception,  
 Nearest to states: flush, cars, philadelphia, agouti, dramatic, jerusalem, reputation, operatorname,  
 Nearest to first: zo, bicycle, attempts, works, on, cohen, u, compared,  
 Nearest to six: nine, eight, three, five, zero, four, seven, two,  
 Nearest to of: and, in, for, dasyprocta, with, nine, agouti, circ,  
 Average loss at step 22000 : 6.97745421338  
 Average loss at step 24000 : 6.88079986274  
 Average loss at step 26000 : 6.74311955643  
 Average loss at step 28000 : 6.319696154  
 Average loss at step 30000 : 5.88112684155  
 Nearest to had: was, has, vaccines, have, were, is, confused, operatorname,  
 Nearest to many: some, these, customer, the, circ, results, dasyprocta, enterprises,  
 Nearest to called: creek, bravery, islet, consolidated, branch, eyebrows, aberdeen, vaccines,  
 Nearest to during: and, in, of, resembling, utterances, napalm, at, ng,  
 Nearest to with: for, in, and, by, from, of, operatorname, or,  
 Nearest to into: from, by, in, was, sabbath, to, and, on,  
 Nearest to there: he, they, prescription, it, often, which, now, operatorname,  
 Nearest to an: the, abet, susima, lighthouse, thereupon, corn, funded, obvious,  
 Nearest to when: and, in, monastery, is, six, that, where, narrative,  
 Nearest to d: b, and, william, worshipper, robbery, increasing, one, rectal,  
 Nearest to for: with, and, in, of, is, on, from, to,  
 Nearest to if: is, circ, rectangular, with, operatorname, siegfried, and, when,  
 Nearest to states: flush, dramatic, cars, agouti, reputation, philadelphia, jerusalem, operatorname,  
 Nearest to first: zo, bicycle, works, operatorname, on, cohen, agouti, compared,  
 Nearest to six: eight, nine, five, four, seven, three, zero, two,  
 Nearest to of: and, in, for, dasyprocta, from, with, nine, at,  
 Average loss at step 32000 : 5.99317011666  
 Average loss at step 34000 : 5.68176041329  
 Average loss at step 36000 : 5.8086677928  
 Average loss at step 38000 : 5.53067963743  
 Average loss at step 40000 : 5.30677920032  
 Nearest to had: has, have, was, vaccines, were, confused, operatorname, talal,  
 Nearest to many: some, these, customer, circ, results, dasyprocta, plymouth, enterprises,  
 Nearest to called: creek, UNK, bravery, islet, consolidated, invisible, and, abitibi,  
 Nearest to during: in, and, utterances, at, resembling, of, current, from,  
 Nearest to with: and, for, in, by, from, zero, on, operatorname,  
 Nearest to into: from, in, sabbath, by, to, was, akita, on,  
 Nearest to there: it, they, he, often, now, prescription, which, operatorname,  
 Nearest to an: the, abet, susima, perpetuated, lighthouse, precursor, hymn, thereupon,  
 Nearest to when: where, zero, in, if, and, is, against, that,  
 Nearest to d: b, william, UNK, worshipper, robbery, zero, and, increasing,  
 Nearest to for: with, zero, of, and, on, airshow, from, in,

Average loss at step 82000 : 4.76378596139  
 Average loss at step 84000 : 4.75491791165  
 Average loss at step 86000 : 4.76127948046  
 Average loss at step 88000 : 4.74573280132  
 Average loss at step 90000 : 4.71888578904  
 Nearest to had: has, have, was, were, vaccines, operatorname, is, thaler,  
 Nearest to many: some, these, several, various, both, results, circ, other,  
 Nearest to called: UNK, see, creek, islet, manure, invisible, eyebrows, bravery,  
 Nearest to during: in, at, after, calypso, from, utterances, while, current,  
 Nearest to with: in, between, by, for, or, and, cebus, operatorname,  
 Nearest to into: from, sabbath, under, through, gathering, by, busan, prayers,  
 Nearest to there: they, it, he, often, now, operatorname, which, michelob,  
 Nearest to an: susima, travolta, perpetuated, hymn, thereupon, the, expressways, amish,  
 Nearest to when: if, where, before, until, after, but, although, as,  
 Nearest to d: b, worshipper, UNK, c, robbery, n, william, carbines,  
 Nearest to for: including, kapoor, in, thaler, with, airshow, after, while,  
 Nearest to if: when, where, callithrix, circ, gadget, nude, before, with,  
 Nearest to states: deceptive, dramatic, flush, cars, thibetanus, operatorname, p  
 rat, hausdorff,  
 Nearest to first: tejada, second, sejm, zo, archie, same, last, upanija,  
 Nearest to six: seven, eight, five, four, nine, three, zero, dasyprocta,  
 Nearest to of: in, thaler, dasyprocta, including, ebola, peacocks, circ, bracing  
 ,  
 Average loss at step 92000 : 4.65660883999  
 Average loss at step 94000 : 4.72485599458  
 Average loss at step 96000 : 4.70201705074  
 Average loss at step 98000 : 4.57996683311  
 Average loss at step 100000 : 4.68917151666  
 Nearest to had: has, have, was, were, vaccines, operatorname, is, ursus,  
 Nearest to many: some, these, several, various, both, most, all, other,  
 Nearest to called: see, chromosphere, creek, eyebrows, gravy, done, invisible, islet,  
 Nearest to during: in, at, after, calypso, while, and, utterances, current,  
 Nearest to with: between, in, by, and, for, or, operatorname, dmd,  
 Nearest to into: from, through, sabbath, under, out, by, gathering, busan,  
 Nearest to there: it, they, he, often, now, operatorname, said, michelob,  
 Nearest to an: susima, travolta, perpetuated, thereupon, hymn, expressways, crick, meditate,  
 Nearest to when: if, where, before, until, after, while, as, although,  
 Nearest to d: b, c, robbery, william, worshipper, n, sanjaks, carbines,  
 Nearest to for: including, kapoor, in, rgya, stenella, airshow, after, during,  
 Nearest to if: when, where, callithrix, though, circ, gadget, nude, is,  
 Nearest to states: deceptive, flush, dramatic, operatorname, thibetanus, cars, adjectives, agouti,  
 Nearest to first: second, tejada, sejm, last, same, upanija, archie, zo,  
 Nearest to six: seven, eight, five, three, four, nine, two, dasyprocta,  
 Nearest to of: in, dmd, thaler, including, dasyprocta, bracing, agouti, ebola,  
 (tensorflow) Arlettes-MBP:word2vec arlettecalvo\$ █