

Conclusiones

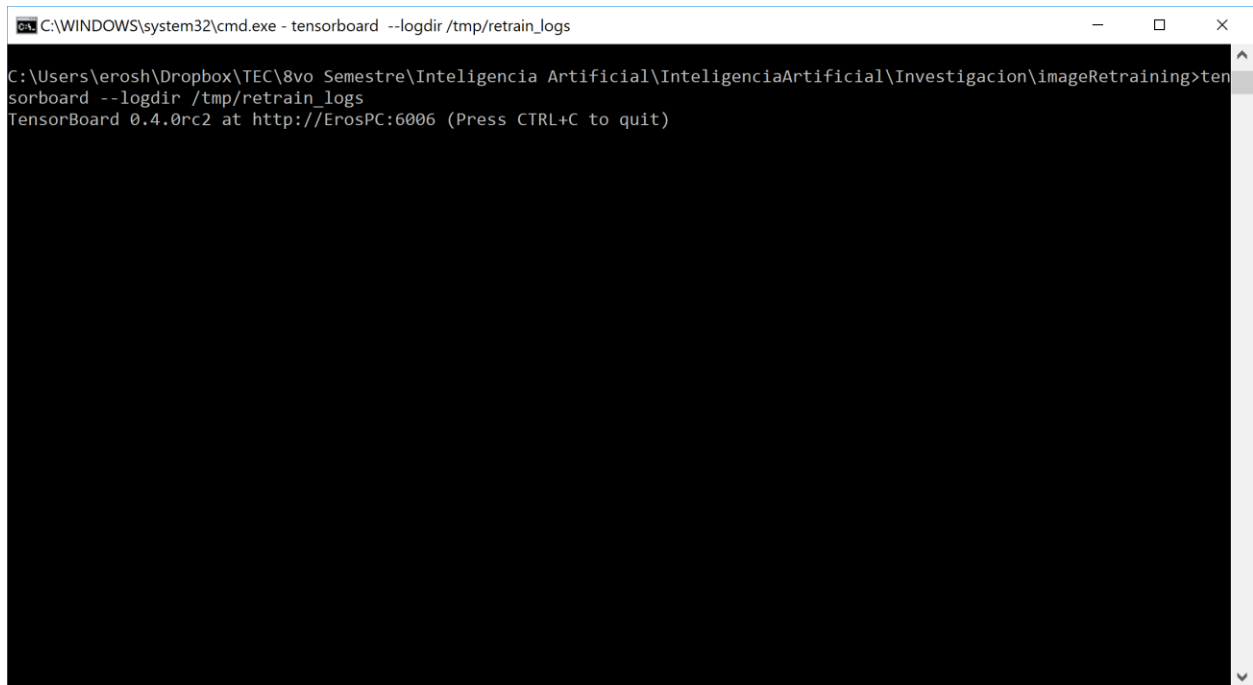
El objetivo de este tutorial, es enseñar al programador a re entrenar una red neuronal que ya previamente posee unos conocimientos base. Esto ocurre porque entrenar una red neuronal desde el comienzo, tomaría mucho tiempo, y aún más si tuviéramos que re entrenarla para ser capaz de reconocer nuevas imágenes que se deseen evaluar. Por lo que, se parte de una red con un entrenamiento base en donde lo que se hará será entrenarla con nuevos sets de imágenes.

Al inicio del tutorial, se parte del punto en utilizar la red neuronal base utilizada en el primer tutorial (Reconocimiento de imágenes) y utilizando un gran conjunto de imágenes de flores (daisy, dandelion, roses, sunflowers, tulips) para re entrenar dicha red en que reconozca estos distintos tipos de flores. En este tutorial, se introduce el concepto de **Cuello de Botella**, que es lo que estará ocurriendo al momento de hacer el re entrenamiento de la red neuronal.

Cuello de botella: Dimensionalidad del input layer, se disminuye, entonces ocurre una especie de cuello de botella en donde la matriz de pesos posee menos columnas que filas o más filas que columnas por lo que los pesos van más conectados hacia la capa de cuello de botella (Ej: Una capa que posea 100 neuronas, pasa a una capa que posea 50 neuronas).

Entonces, lo que ocurre es que el cuello de botella entrena la penúltima capa de la red neuronal, esto se hace para lograr que sean más confiables los resultados de las clasificaciones. Una vez que se realiza este cuello de botella en la penúltima capa, la red se vuelve a entrenar completamente.

En el tutorial, se nos introduce a la bitácora **TensorBoard**, el cual es un ambiente en donde se puede visualizar el entrenamiento de la red neuronal, TensorBoard se puede inicializar desde una ventana de comandos, se inicia un servidor en el puerto 6006 del localhost de la computadora, al accederlo, se pueden visualizar los distintos puntos del entrenamiento:



```
C:\WINDOWS\system32\cmd.exe - tensorboard --logdir /tmp/retrain_logs
C:\Users\erosh\Dropbox\TEC\8vo Semestre\Inteligencia Artificial\InteligenciaArtificial\Investigacion\imageRetraining>ten
sorboard --logdir /tmp/retrain_logs
TensorBoard 0.4.0rc2 at http://ErosPC:6006 (Press CTRL+C to quit)
```

Según la documentación de **TensorFlow**, la mínima cantidad de imágenes para entrenar una red debería ser de aproximadamente **100 imágenes**, cada una de estas, debe ser organizada dependiendo de la categoría a la que pertenecen. En mi caso, **entrené la red para que supiera reconocer comida**, más específicamente Hamburguesas, Pizza, Hot Dogs y Tacos. Por lo que la puse a entrenar con 100 imágenes de cada tipo de comida que menciono.

Cómo podría aplicarse a un problema real

Para darle seguimiento al plan propuesto de reconocimiento de células cancerosas a través de imágenes, el re entrenamiento de la red se puede plantear en la misma área en donde se puedan descubrir nuevas formas/mutaciones/cambios en los mismo en donde lo único que se requiera hacer es tomar un nuevo set de imágenes de estos distintos tipos de células y re entrenar la red neuronal para que sea capaz de reconocer estos nuevos descubrimientos y pueda continuar siendo muy útil en el área.

Screenshots de Resultados

Inicialmente, tomamos una serie de imágenes de flores, y comenzamos a re entrenar la red para que sea capaz de reconocerlas.

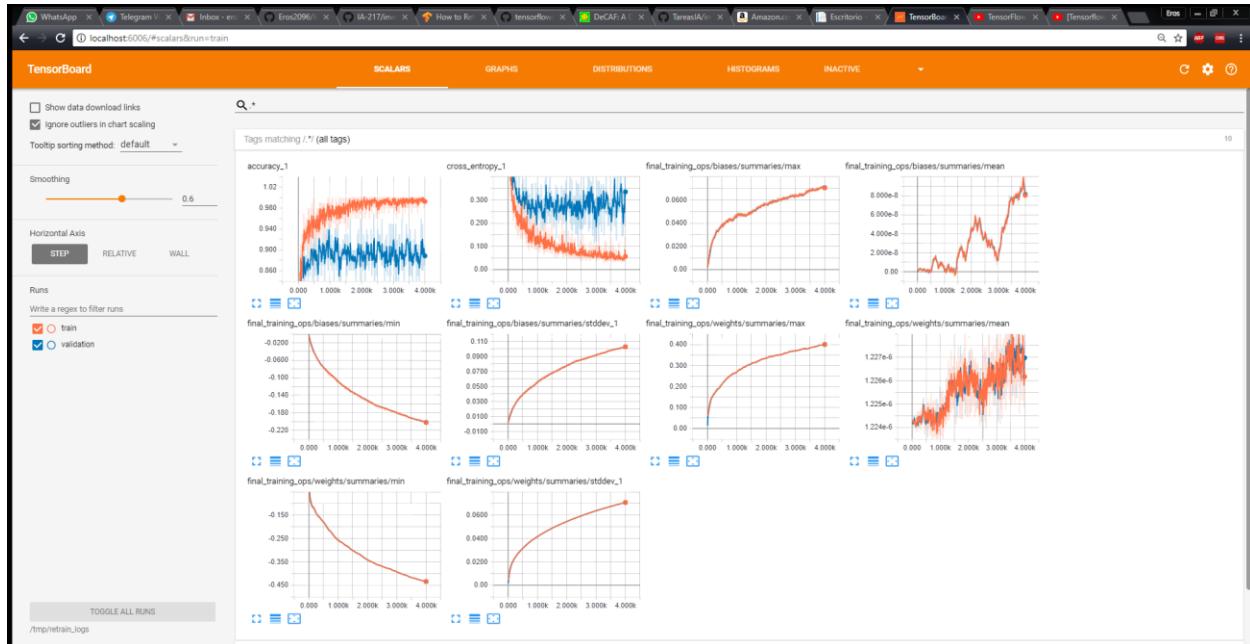
```
C:\WINDOWS\system32\cmd.exe - python retrain.py --image_dir flower_photos
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\1374193928_a52320eafa.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\13826249325_f61cb15f86_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\13901930939_a7733c03f0_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\1392131677_116ec04751.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\1392946544_115acbb2d9.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\13953307149_f8de6a768c_m.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\1396526833_fb867165be_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\13977181862_f8237b6b52.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14021430525_e06baf93a9.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14073784469_ffb12f3387_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14087947408_9779257411_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14114116486_0bb6649bc1_m.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14147016029_8d3cf2414e.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14163875973_467224aaf5_m.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14167534527_781ceb1b7a_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14167543177_cd36b54ac6_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14219214466_3ca6104eae_m.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14221848160_7f0a37c395.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14245834619_153624f836.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14264136211_9531fbc144.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14272874304_47c0a46f5a.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14307766919_fac3c37a6b_m.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14332947164_9b13513c71_m.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14333681205_a07c9f1752_m.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14350958832_29bdd3a254.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14354051035_1037b30421_n.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14372713423_61e2daae88.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14399435971_ea5868c792.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\daisy\14402451388_56545a374a_n.jpg_inception_v3.txt
```

Una vez se termina de entrenar la red con las imágenes de las plantas, el resultado es el siguiente, en donde se obtiene un 90% de precisión. El proceso tomó un total de **4000** ciclos.

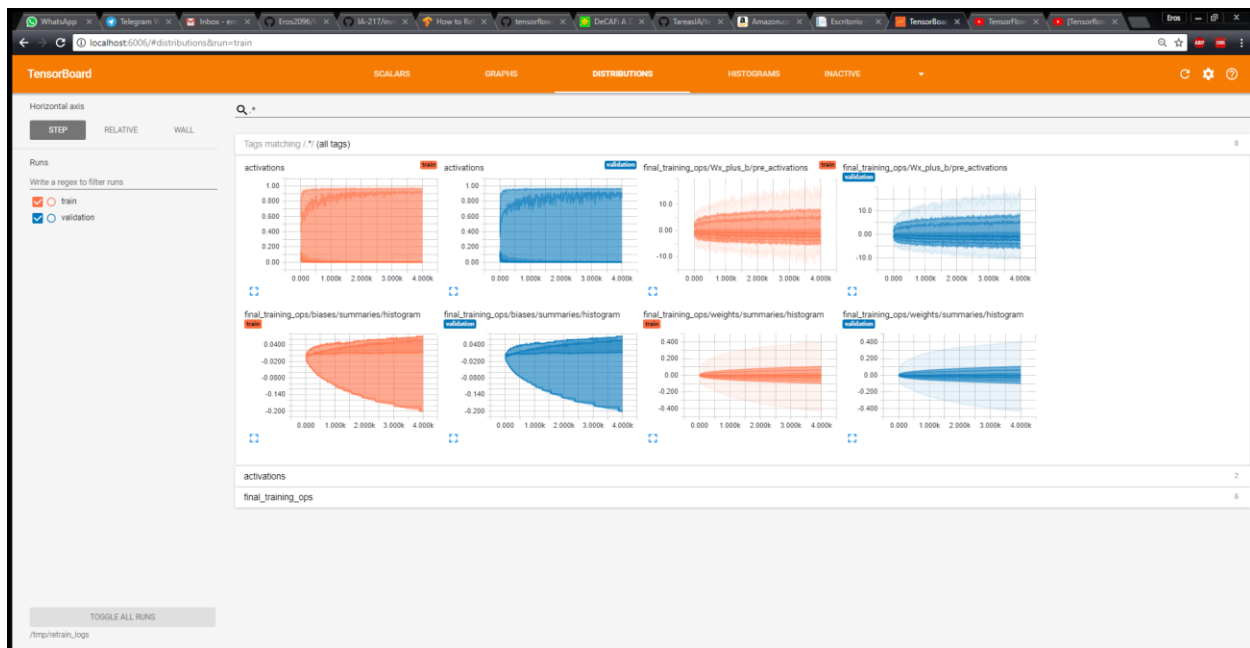
```
C:\WINDOWS\system32\cmd.exe
INFO:tensorflow:2017-11-05 13:46:35.432988: Step 3920: Validation accuracy = 95.0% (N=100)
INFO:tensorflow:2017-11-05 13:46:36.478034: Step 3930: Train accuracy = 100.0%
INFO:tensorflow:2017-11-05 13:46:36.478586: Step 3930: Cross entropy = 0.031375
INFO:tensorflow:2017-11-05 13:46:36.592628: Step 3930: Validation accuracy = 90.0% (N=100)
INFO:tensorflow:2017-11-05 13:46:37.666990: Step 3940: Train accuracy = 99.0%
INFO:tensorflow:2017-11-05 13:46:37.666990: Step 3940: Cross entropy = 0.048058
INFO:tensorflow:2017-11-05 13:46:37.781821: Step 3940: Validation accuracy = 86.0% (N=100)
INFO:tensorflow:2017-11-05 13:46:38.885258: Step 3950: Train accuracy = 100.0%
INFO:tensorflow:2017-11-05 13:46:38.885643: Step 3950: Cross entropy = 0.039504
INFO:tensorflow:2017-11-05 13:46:39.009975: Step 3950: Validation accuracy = 91.0% (N=100)
INFO:tensorflow:2017-11-05 13:46:40.035720: Step 3960: Train accuracy = 100.0%
INFO:tensorflow:2017-11-05 13:46:40.035720: Step 3960: Cross entropy = 0.062749
INFO:tensorflow:2017-11-05 13:46:40.149029: Step 3960: Validation accuracy = 90.0% (N=100)
INFO:tensorflow:2017-11-05 13:46:41.171231: Step 3970: Train accuracy = 99.0%
INFO:tensorflow:2017-11-05 13:46:41.171231: Step 3970: Cross entropy = 0.059214
INFO:tensorflow:2017-11-05 13:46:41.281563: Step 3970: Validation accuracy = 91.0% (N=100)
INFO:tensorflow:2017-11-05 13:46:42.300763: Step 3980: Train accuracy = 98.0%
INFO:tensorflow:2017-11-05 13:46:42.301227: Step 3980: Cross entropy = 0.127859
INFO:tensorflow:2017-11-05 13:46:42.414577: Step 3980: Validation accuracy = 87.0% (N=100)
INFO:tensorflow:2017-11-05 13:46:43.445054: Step 3990: Train accuracy = 99.0%
INFO:tensorflow:2017-11-05 13:46:43.445287: Step 3990: Cross entropy = 0.060248
INFO:tensorflow:2017-11-05 13:46:43.556092: Step 3990: Validation accuracy = 90.0% (N=100)
INFO:tensorflow:2017-11-05 13:46:44.532440: Step 3999: Train accuracy = 99.0%
INFO:tensorflow:2017-11-05 13:46:44.532440: Step 3999: Cross entropy = 0.057748
INFO:tensorflow:2017-11-05 13:46:44.645710: Step 3999: Validation accuracy = 88.0% (N=100)
INFO:tensorflow:Final test accuracy = 90.9% (N=744)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.
```

Visualizando el proceso mediante **TensorBoard**, podemos observar el comportamiento de los pesos de la red durante y después de su entrenamiento:

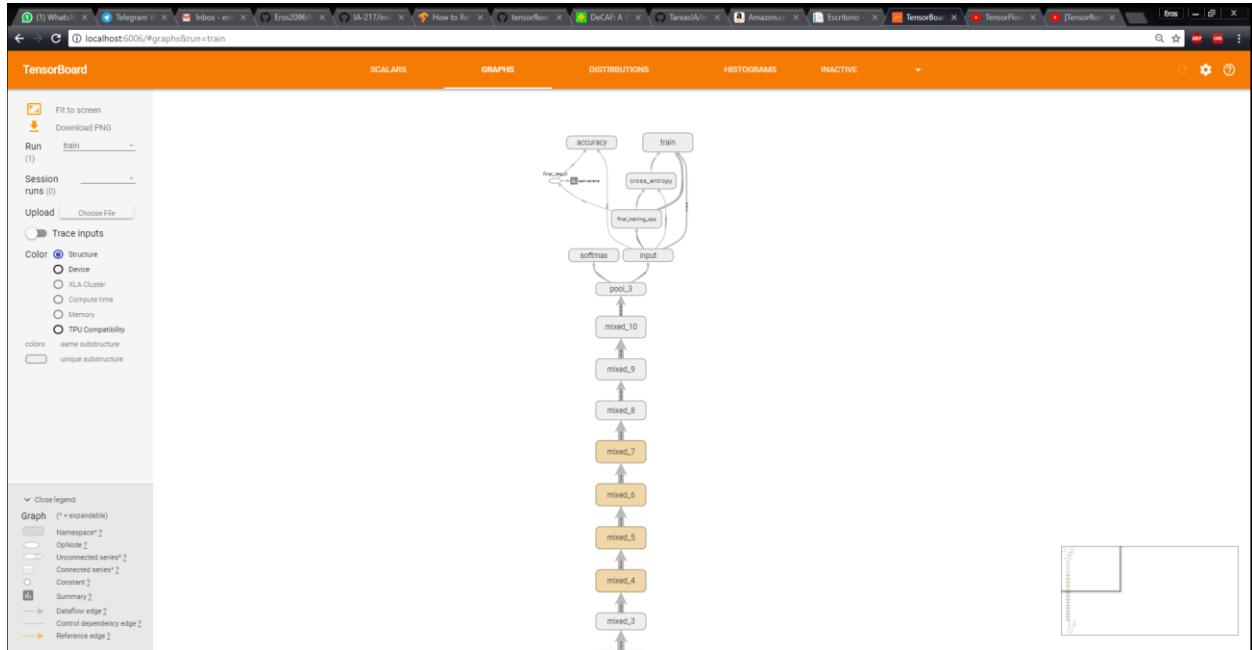
- **Escalares**



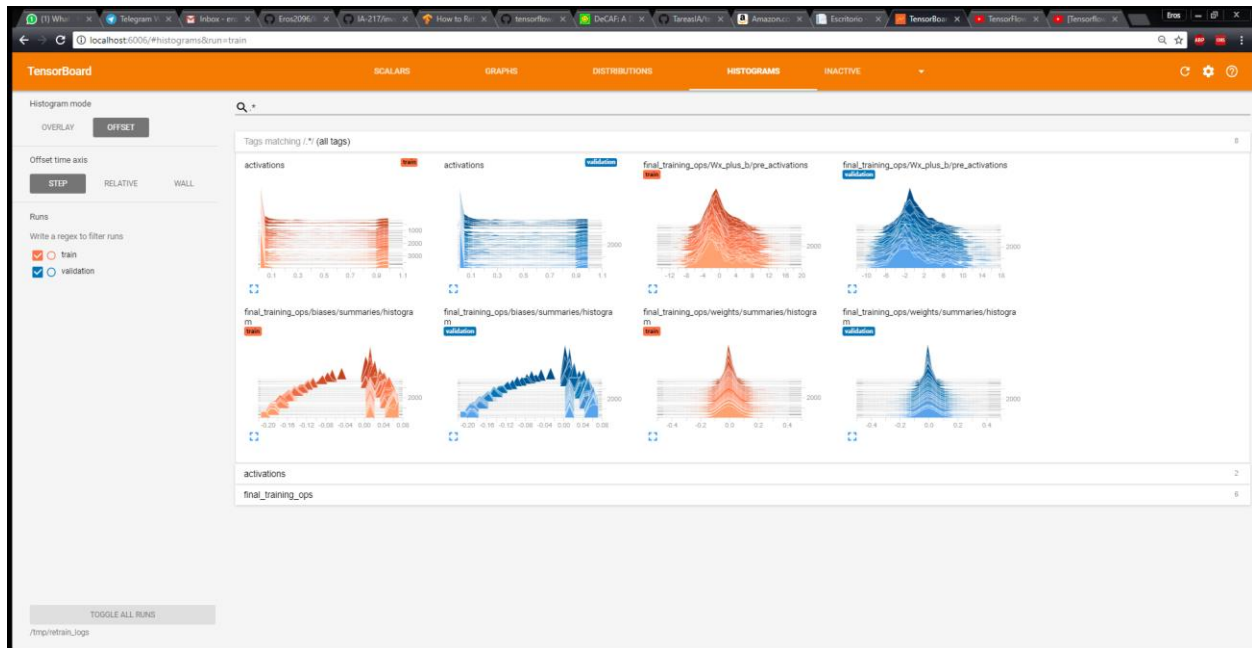
- **Gráficos**




- Grafo Generado



- Histogramas



Una vez entrenada la base, evalué una de las imágenes de las flores y se obtuvo el **resultado correcto**:



```
C:\Users\erosh\Dropbox\TEC\8vo Semestre\Inteligencia Artificial\InteligenciaArtificial\Investigacion\imageRetraining>python label_image.py --image="C:\Users\erosh\Dropbox\TEC\8vo Semestre\Inteligencia Artificial\InteligenciaArtificial\Investigacion\imageRetraining\flower_photos\daisy\21652746_cc379e0eea_m.jpg" --graph C:\tmp\output_graph.pb --labels C:\tmp\output_labels.txt
2017-11-05 17:00:02.829016: I C:\tf_jenkins\home\workspace\rel-win\M\windows\PY\36\tensorflow\core\platform\cpu_feature_guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX2
2017-11-05 17:00:04.043757: W C:\tf_jenkins\home\workspace\rel-win\M\windows\PY\36\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().
daisy (score = 0.99949)
sunflowers (score = 0.00024)
roses (score = 0.00014)
dandelion (score = 0.00009)
tulips (score = 0.00004)
```

Re-entrenamiento Propio

Como fue mencionado anteriormente, se decidió descargar un conjunto de imágenes de comida, Hamburguesa, Pizza, Hot Dogs y Tacos (100 imágenes de cada uno) para entrenar la red neuronal y que fuera capaz de reconocer dichas comidas. Las imágenes fueron descargadas utilizando un programa hecho en Python por el usuario 'hardikvasa'. Incluyo el link a su repositorio:

<https://github.com/hardikvasa/google-images-download>

Se procede a realizar el entrenamiento de la red neuronal:

```
C:\WINDOWS\system32\cmd.exe - python retrain.py --image_dir "C:\Users\erosh\Dropbox\TEC\8vo Se...
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\36.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\38.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\4.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\40.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\41.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\42.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\43.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\44.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\45.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\46.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\47.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\48.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\5.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\51.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\52.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\53.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\54.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\55.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\56.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\59.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\60.jpg_inception_v3.txt
INFO:tensorflow:Creating bottleneck at /tmp/bottleneck\hamburguesa\62.jpg_inception_v3.txt
```

Este proceso de cuello de botella se realiza para cada imagen de cada tipo hasta que terminan todos los ciclos y se obtiene una probabilidad de 93% como resultado. El proceso tomó un total de **4000 ciclos**.

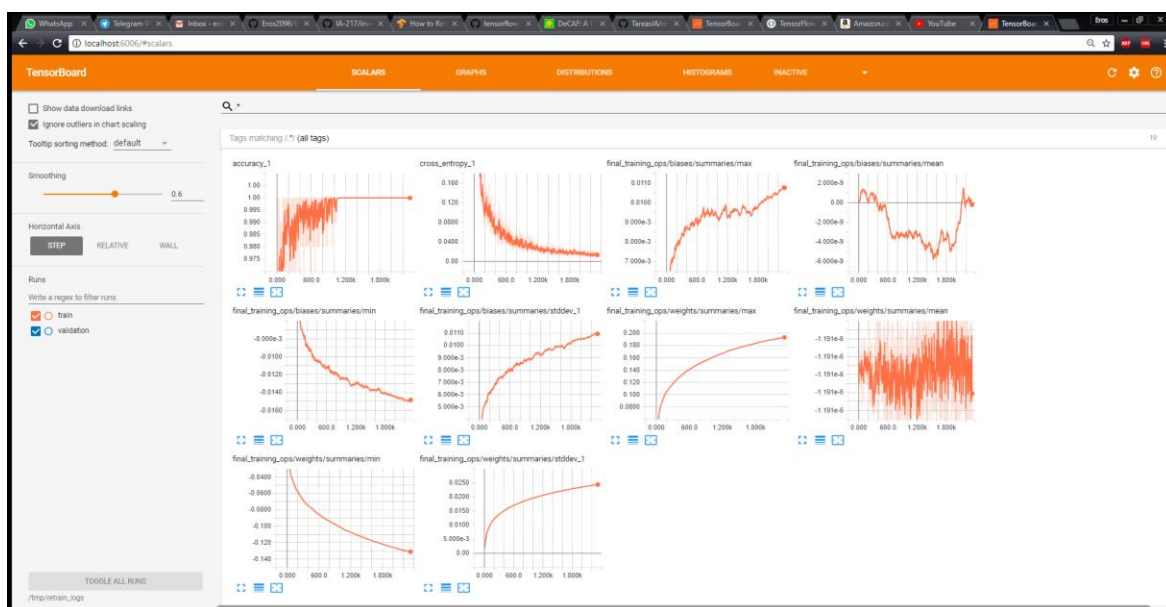
```
C:\WINDOWS\system32\cmd.exe

INFO:tensorflow:2017-11-05 23:49:15.210858: Step 3950: Cross entropy = 0.009766
INFO:tensorflow:2017-11-05 23:49:15.326755: Step 3950: Validation accuracy = 98.0% (N=100)
INFO:tensorflow:2017-11-05 23:49:16.385882: Step 3960: Train accuracy = 100.0%
INFO:tensorflow:2017-11-05 23:49:16.385882: Step 3960: Cross entropy = 0.005912
INFO:tensorflow:2017-11-05 23:49:16.492786: Step 3960: Validation accuracy = 95.0% (N=100)
INFO:tensorflow:2017-11-05 23:49:17.601982: Step 3970: Train accuracy = 100.0%
INFO:tensorflow:2017-11-05 23:49:17.601982: Step 3970: Cross entropy = 0.013813
INFO:tensorflow:2017-11-05 23:49:17.707317: Step 3970: Validation accuracy = 95.0% (N=100)
INFO:tensorflow:2017-11-05 23:49:18.784852: Step 3980: Train accuracy = 100.0%
INFO:tensorflow:2017-11-05 23:49:18.784852: Step 3980: Cross entropy = 0.007489
INFO:tensorflow:2017-11-05 23:49:18.890281: Step 3980: Validation accuracy = 96.0% (N=100)
INFO:tensorflow:2017-11-05 23:49:19.965945: Step 3990: Train accuracy = 100.0%
INFO:tensorflow:2017-11-05 23:49:19.965945: Step 3990: Cross entropy = 0.010274
INFO:tensorflow:2017-11-05 23:49:20.080787: Step 3990: Validation accuracy = 95.0% (N=100)
INFO:tensorflow:2017-11-05 23:49:21.065886: Step 3999: Train accuracy = 100.0%
INFO:tensorflow:2017-11-05 23:49:21.065886: Step 3999: Cross entropy = 0.007391
INFO:tensorflow:2017-11-05 23:49:21.176761: Step 3999: Validation accuracy = 97.0% (N=100)
INFO:tensorflow:Final test accuracy = 93.9% (N=66)
INFO:tensorflow:Froze 2 variables.
Converted 2 variables to const ops.

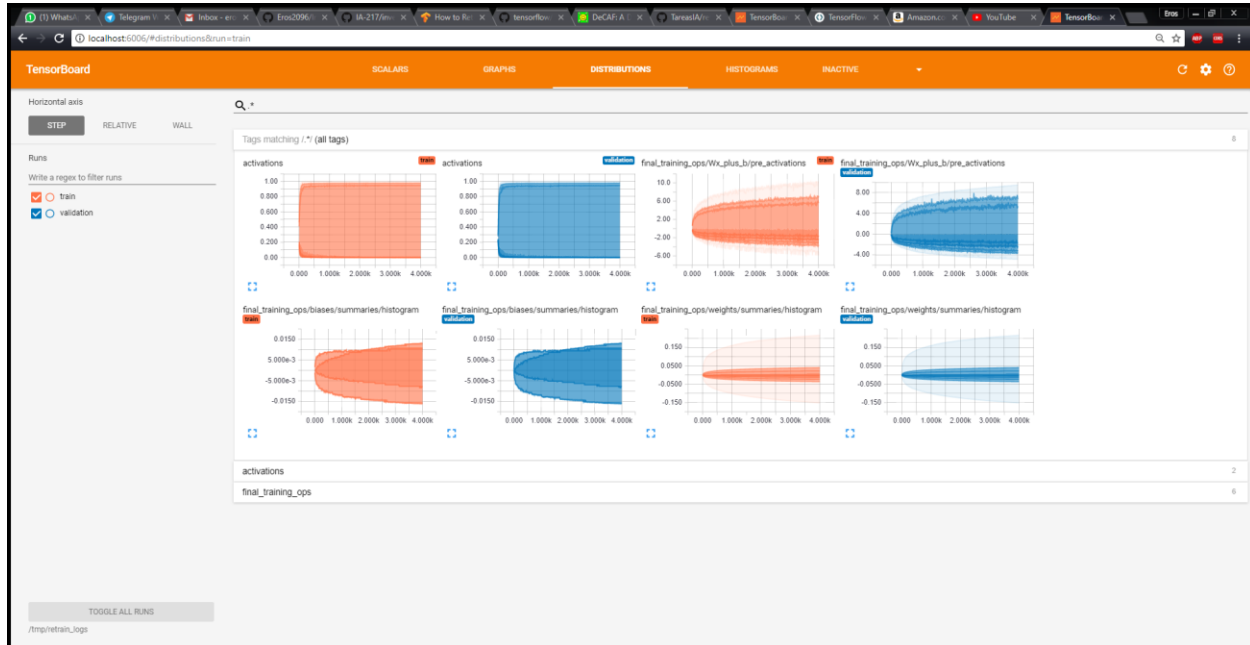
C:\Users\erosh\Dropbox\TEC\8vo Semestre\Inteligencia Artificial\InteligenciaArtificial\Investigacion\imageRetraining>
```

Visualizando el proceso de entrenamiento de las diferentes comidas para la red neuronal mediante TensorBoard, podemos observar el comportamiento de los pesos de la red durante y después de su entrenamiento:

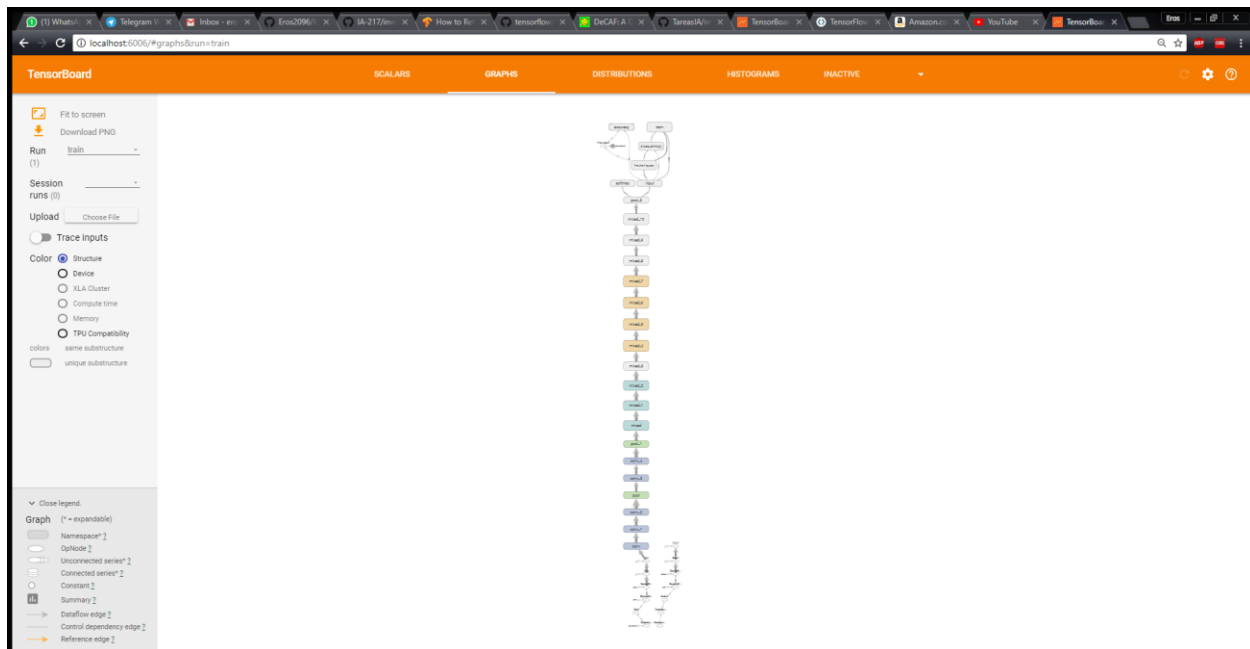
- **Escalares**



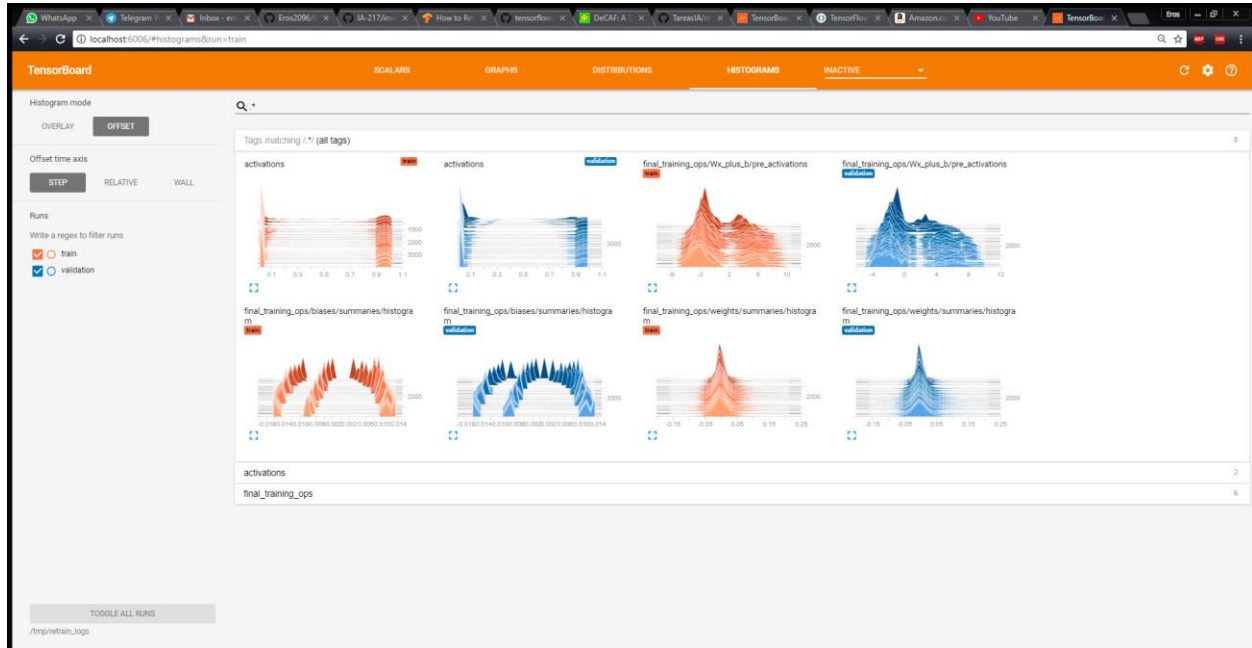
- Gráficos



- Grafo Generado



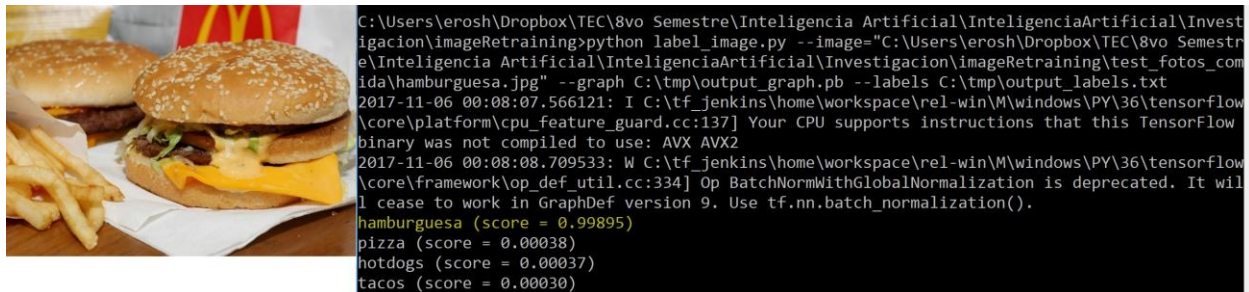
- **Histogramas**



Una vez entrenada la base con las nuevas imágenes de comida, **evalué distintas imágenes para obtener distintos resultados, se ha comprobado que la red ha logrado distinguir claramente cada tipo de comida:**

- **Hamburguesa**

En este caso, se obtuvo el resultado esperado en un 99% (Bastante alto).



- **Hot Dog**

En este caso, se obtuvo el resultado esperado con un 93%.



```
C:\Users\erosh\Dropbox\TEC\8vo Semestre\Inteligencia Artificial\InteligenciaArtificial\Investigacion\imageRetraining>python label_image.py --image="C:\Users\erosh\Dropbox\TEC\8vo Semestre\Inteligencia Artificial\InteligenciaArtificial\Investigacion\imageRetraining\test_fotos_comida\hotDog.jpg" --graph C:\tmp\output_graph.pb --labels C:\tmp\output_labels.txt
2017-11-06 00:10:56.027606: I C:\tf_jenkins\home\workspace\rel-win\M\windows\PY\36\tensorflow\core\platform\cpu_feature_guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX2
2017-11-06 00:10:57.160711: W C:\tf_jenkins\home\workspace\rel-win\M\windows\PY\36\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().
hotdogs (score = 0.93091)
tacos (score = 0.06044)
hamburguesa (score = 0.00460)
pizza (score = 0.00405)
```

- **Pizza**

En este caso, se obtuvo el resultado esperado en un 99% (Bastante alto).



```
C:\Users\erosh\Dropbox\TEC\8vo Semestre\Inteligencia Artificial\InteligenciaArtificial\Investigacion\imageRetraining>python label_image.py --image="C:\Users\erosh\Dropbox\TEC\8vo Semestre\Inteligencia Artificial\InteligenciaArtificial\Investigacion\imageRetraining\test_fotos_comida\pizza.jpg" --graph C:\tmp\output_graph.pb --labels C:\tmp\output_labels.txt
2017-11-06 00:12:33.936807: I C:\tf_jenkins\home\workspace\rel-win\M\windows\PY\36\tensorflow\core\platform\cpu_feature_guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX2
2017-11-06 00:12:35.103591: W C:\tf_jenkins\home\workspace\rel-win\M\windows\PY\36\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().
pizza (score = 0.99836)
tacos (score = 0.00133)
hotdogs (score = 0.00018)
hamburguesa (score = 0.00013)
```

- **Tacos**

En este caso, se obtuvo el resultado esperado en un 99% (Bastante alto).



```
C:\Users\erosh\Dropbox\TEC\8vo Semestre\Inteligencia Artificial\InteligenciaArtificial\Investigacion\imageRetraining>python label_image.py --image="C:\Users\erosh\Dropbox\TEC\8vo Semestre\Inteligencia Artificial\InteligenciaArtificial\Investigacion\imageRetraining\test_fotos_comida\tacos.jpg" --graph C:\tmp\output_graph.pb --labels C:\tmp\output_labels.txt
2017-11-06 00:13:52.750307: I C:\tf_jenkins\home\workspace\rel-win\M\windows\PY\36\tensorflow\core\platform\cpu_feature_guard.cc:137] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX AVX2
2017-11-06 00:13:53.897153: W C:\tf_jenkins\home\workspace\rel-win\M\windows\PY\36\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().
tacos (score = 0.99918)
pizza (score = 0.00042)
hamburguesa (score = 0.00031)
hotdogs (score = 0.00009)
```