# Práctica II

## Intel & Mobile ODT Cervical Cancer Screening

SISTEMAS INTELIGENTES PARA LA GESTIÓN DE LA EMPRESA

CURSO 2016-2017

# Competición

https://www.kaggle.com/c/intel-mobileodt-cervical-cancer-screening

Identificación del tipo fisionómico de la cérvix femenina (parte inferior del útero), con el propósito de establecer el tratamiento preventivo o inicial más adecuado para curar el cáncer de útero.

Se utilizan imágenes imágenes obtenidas con el sistema EVA (Enhanced Visual Assessment) de Mobile ODT.

Introducción (python):
https://www.kaggle.com/philschmidt/intel-mobileodt-cervical-cancer-screening/cervix-eda-model-selection

# Competición

## Evaluación

- Función $logloss$

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

$N$: Número de imágenes en el conjunto de validación

$M$: Número de categorías

$log$: Logaritmo natural (base $e$)

$y_{ij}$: 1 si la observación $i$ pertenece a la clase $j$ ; 0 si no

$p_{ij}$: probabilidad de que la observación $i$ pertenezca a la clase $j$; cumple $\forall i = \{1, \dots, N\} : \sum_{j=1}^{M} y_{ij} = 1$

- Esta formulación de $logloss$ sólo cuenta en el sumatorio las probabilidades asignadas a cada instancia de pertenecer a la clase correcta
  - Se penalizan mucho asignar valores bajos de probabilidad a la clase correcta (falsos negativos)
- Un clasificador perfecto tendrá $logloss = 0$

# Competición

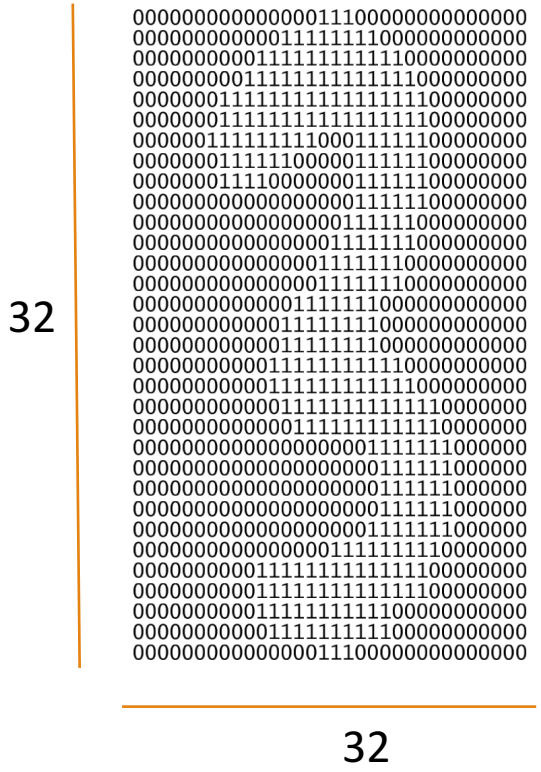
**Evaluación**

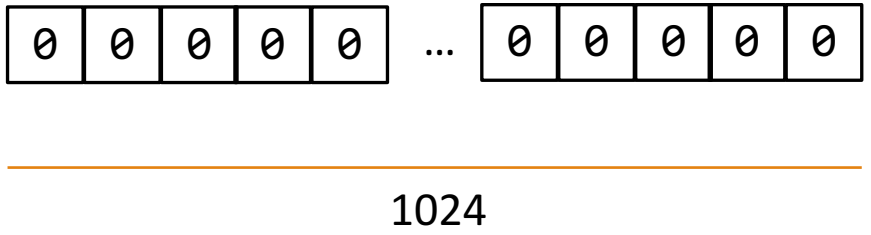
| | $p_{i,1}$ | $p_{i,2}$ | $p_{i,3}$ |
|---|---|---|---|
| Imagen 1 | **0.23** | 0.52 | 0.25 |
| Imagen 2 | **0.70** | 0.15 | 0.15 |
| Imagen 3 | 0.35 | 0.25 | **0.40** |

$$
\begin{aligned}
logloss &= -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}) \\
&= -\frac{1}{3}\times(\log(0.23)+\log(0.70)+\log(0.40)) \\
&= -\frac{1}{3}\times(-1.47-0.357-0.92) \\
&= 0.9156
\end{aligned}
$$

# Aproximaciones a la clasificación de imágenes

1. **Codificación directa**




**Multi-clasificador**

Lan, Y., Lean, L. Handwritten digit recognition using perceptron neural networks. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.27.9135&rep=rep1&type=pdf

# Aproximaciones a la clasificación de imágenes

2. **Pre-procesamiento y extracción de características**

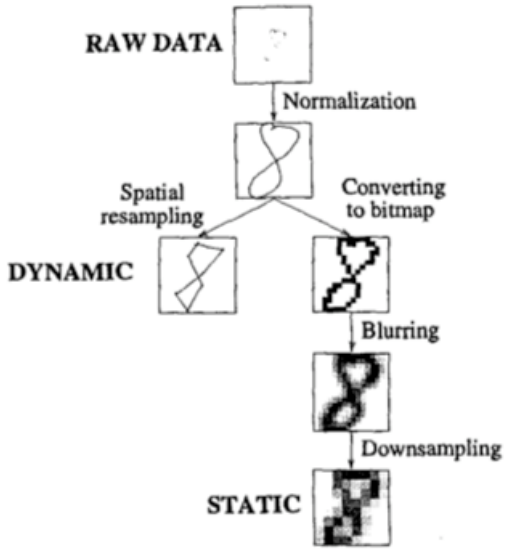



Figure 1: The stages of preprocessing.

**Densidad de píxeles**

**Detector de bordes**

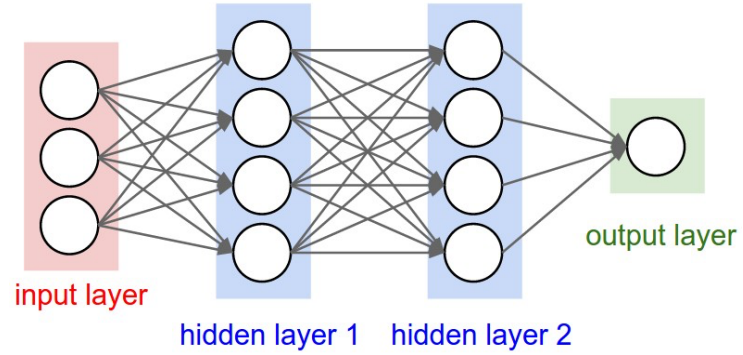
**Detector de líneas**

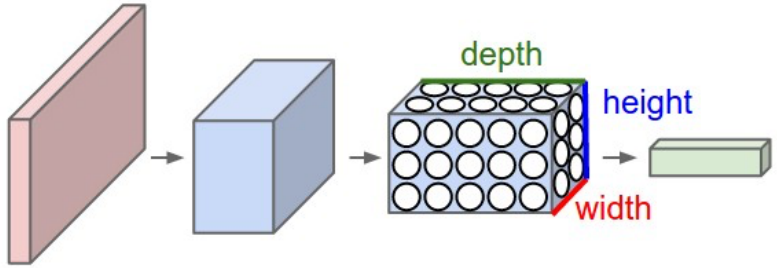
**Multi-clasificador**

Alimoglu, F., Alpaydin, E. Combining multiple representations and classifiers for pen-based handwritten digit recognition. URL: http://ieeexplore.ieee.org/document/620583/?reload=true&section=abstract

# Aproximaciones a la clasificación de imágenes

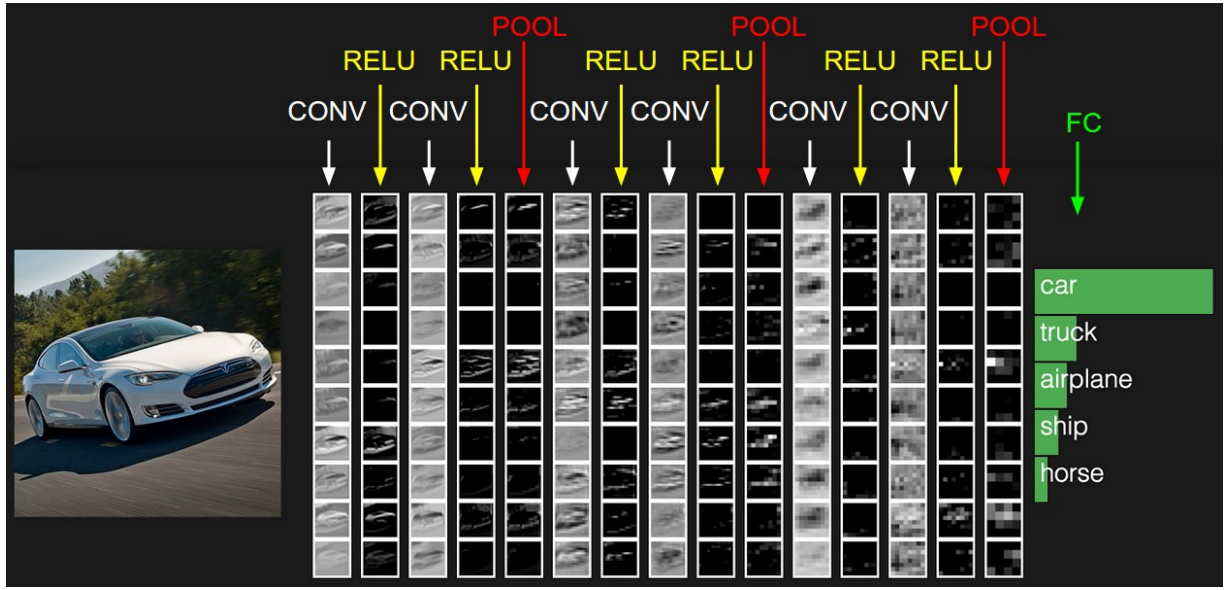
3. Redes neuronales convolucionales (*convolutional neural networks*, CNN)
   http://cs231n.github.io
   https://adeshpande3.github.io/adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/



# Arquitectura de una CNN

1. INPUT (entrada)
2. CONV (convolución)
3. RELU (activación)
4. POOL (reducción de dimensionalidad)
5. FC (totalmente conectada)



# Estrategias

## Solo CNNs

- 3.1. Configurar CNN desde el principio
- 3.2. Utilizar una red ya entrenada
- 3.3. Transfer learning / fine tuning

## CNNs + otros clasificadores

- 3.4. Extracción de características con red ya entrenada
- 3.5. Extracción de características con red propia (desde el principio o *fine tuning*)

>> Aplicar pre-procesamiento: *data augmentation*

>> Creación de *ensembles*

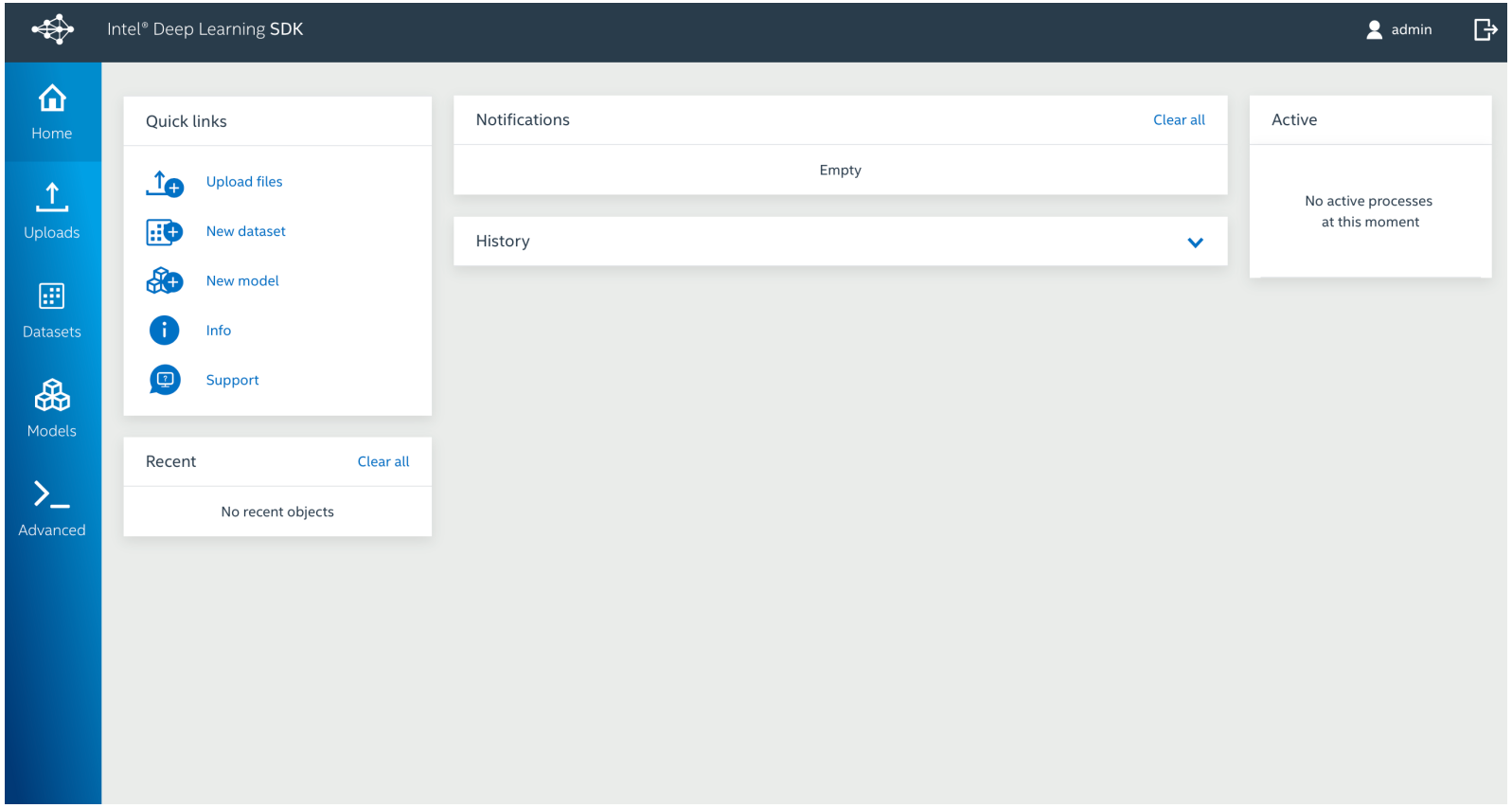
>> Multiclasificación: *one vs one*, *one vs all*

# Intel Deep Learning SDK (3.1, 3.2. 3.3)

https://www.kaggle.com/c/intel-mobileodt-cervical-cancer-screening#Intel-Tutorial

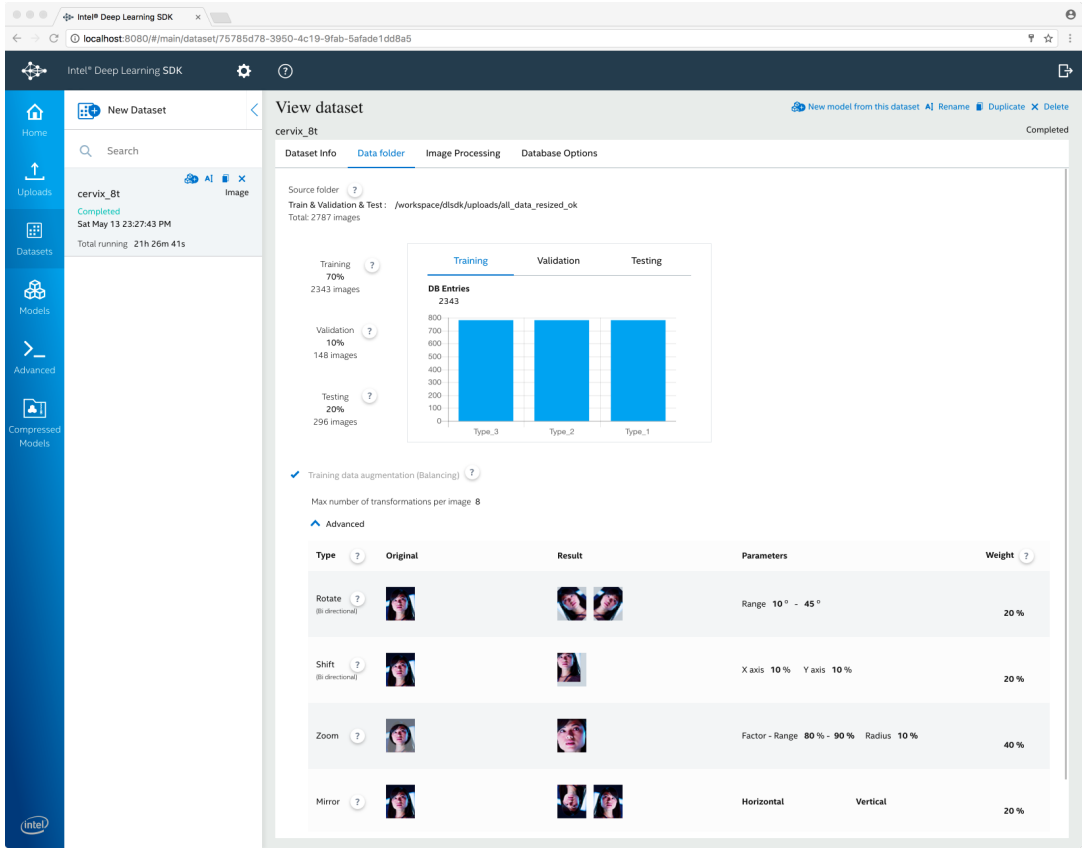
- **Intel Deep Learning SDK 1.0**
  - Requisitos: OS X => Docker 1.12+ , no VirtualBox < 4.3.30
    https://software.intel.com/sites/default/files/managed/ef/08/Intel_Deep_Learing_SDK_Training_Tool_Installation_Guide.pdf
  - Descarga:
    https://software.intel.com/en-us/deep-learning-sdk/documentation
  - Instalación: admin / *password*
    http://localhost:8080/#/main/home

# Intel Deep Learning SDK (3.1, 3.2. 3.3)



# Intel Deep Learning SDK (3.1, 3.2. 3.3)

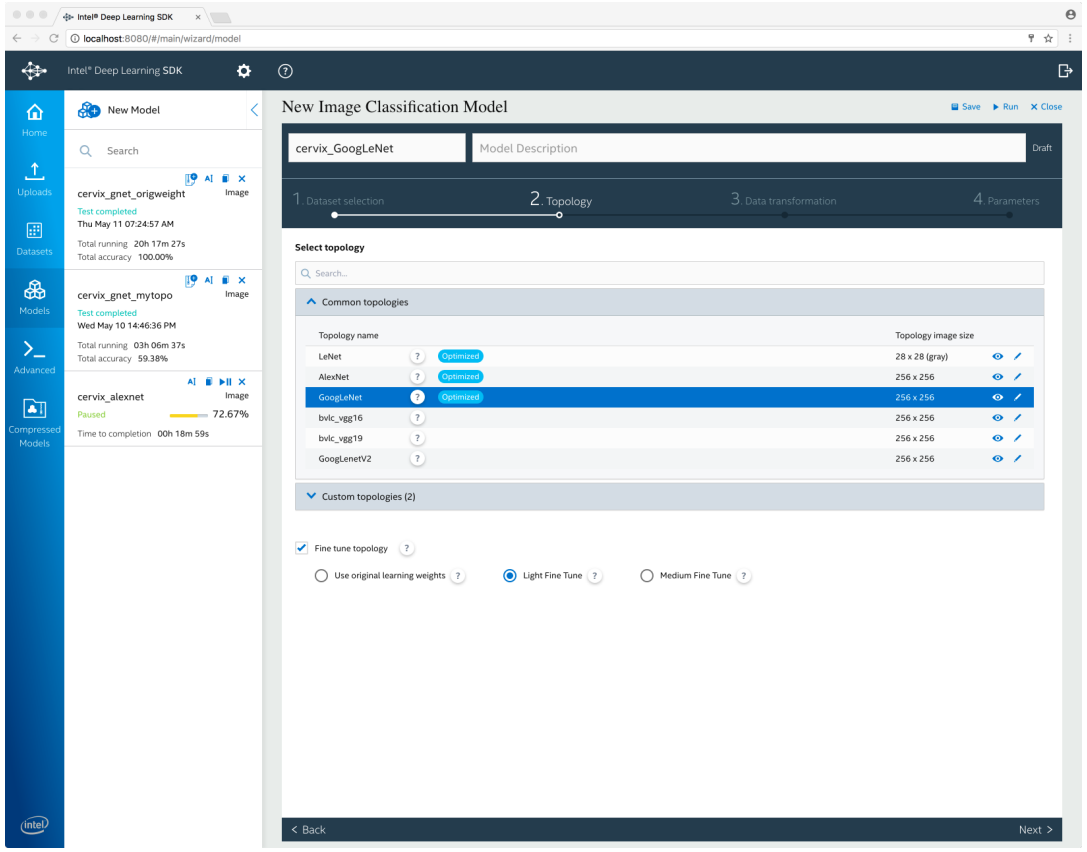
**Pre-procesamiento**
Redimensionado
*Data augmentation*
…




https://consigna.ugr.es/g/QBhkP8vloBl2evp4/all_data_resized.zip

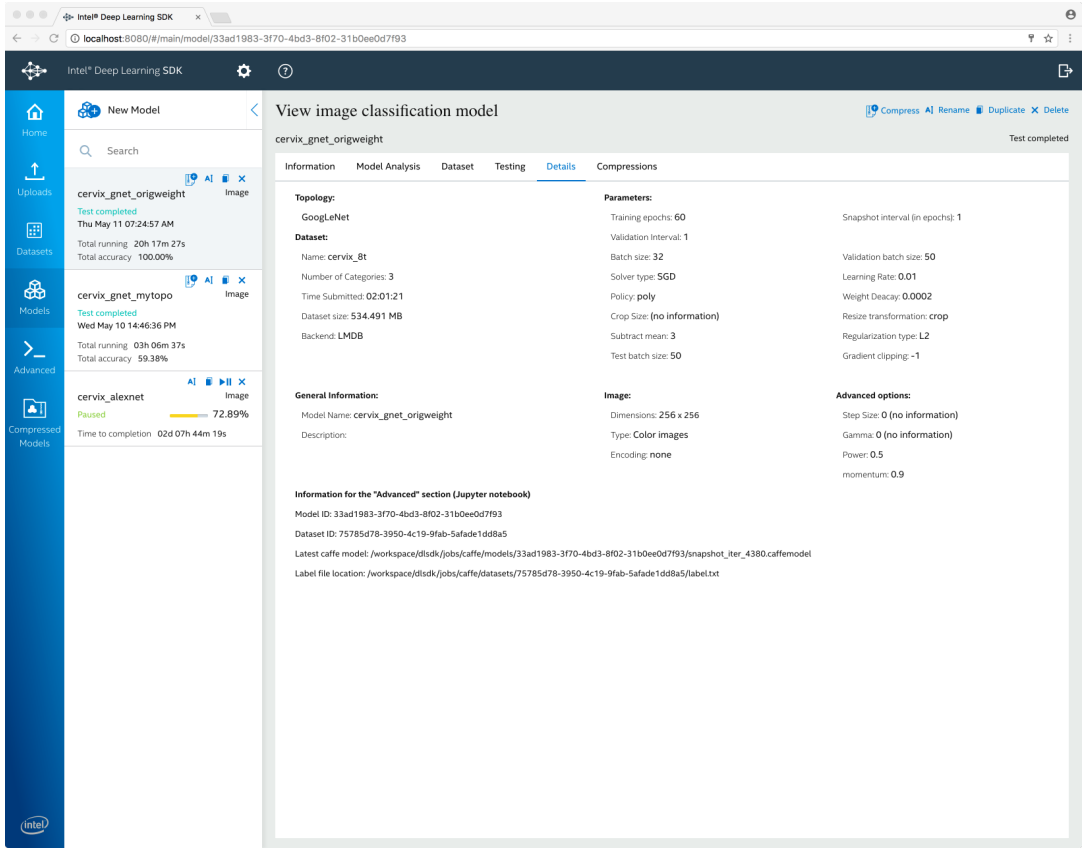
# Intel Deep Learning SDK (3.1, 3.2. 3.3)

**Transfer learning**
Topologías
*Fine tuning*
…

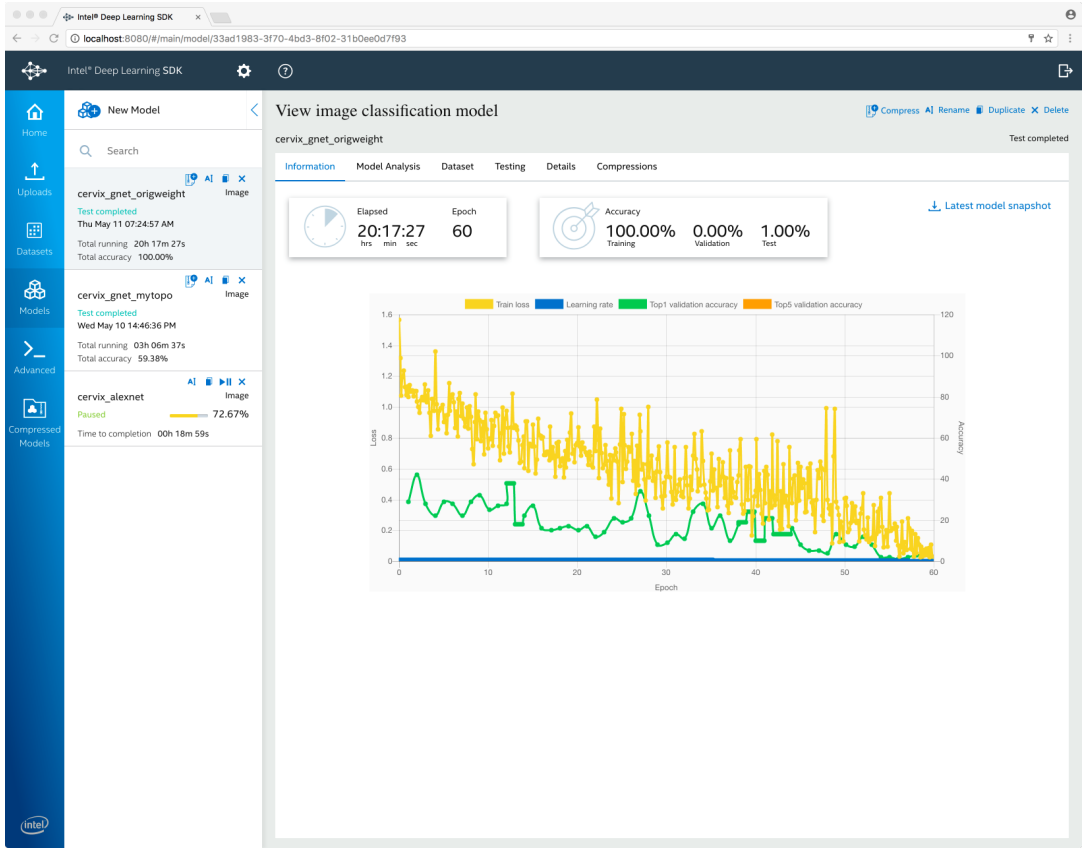


# Intel Deep Learning SDK (3.1, 3.2. 3.3)

***Transfer learning***
Grabar modelo



# Intel Deep Learning SDK (3.1, 3.2. 3.3)



# Intel Deep Learning SDK (3.1, 3.2. 3.3)

```
docker exec –t –i trainingtool–js /bin/bash
```

```
cat /workspace/dlsdk/jobs/caffe/models/<model id>/data_exploration.txt
```

- **model.zip**
  - caffe.log : Log de ejecución del proceso de aprendizaje
  - deploy.protoxt : Modelo para inferencia
  - solver.protoxt : Configuración de aprendizaje
  - train_val.protoxt : Modelo + datos de entrenamiento y validación
  - train_test.protoxt : Modelo + datos de entrenamiento y test
  - train.txt : Etiquetas conjunto de entrenamiento
  - val.txt : Etiquetas conjunto de validación

# MXNET (3.1, 3.2. 3.3, ¿3.4, 3.5?)

**MXNET**
http://mxnet.io

- Biblioteca para Deep Learning con implementación para C++, Python, R, Scala, etc.
- Alternativa a otras bibliotecas: Tensorflow (Google), Caffe (Berkely AI Lab)

- `mx.symbol.Variable`: Crear capa de entrada
- `mx.symbol.Convolution`: Crear capa CONV
- `mx.symbol.Activation`: Crear capa RELU
- `mx.symbol.Pooling`: Crear capa POOL
- `mx.symbol.FullyConnected`: Crear capa FC
- `mx.symbol.SoftmaxOutput`: Crear capa de salida

- `mx.model.FeedForward.create` : Aprender modelo

# MXNET (3.1, 3.2. 3.3, ¿3.4, 3.5?)

**MXNET**
http://mxnet.io/tutorials/index.html#r

- MXNET para MNIST
  **mxnet-MNIST.R**

- MXNET para Cervix + Pre-procesado: (3.1)
  - Blanco y negro
    **mxnet-cervix-bn.R**
  - Color
    **mxnet-cervix-color.R**

- Clasificación de imágenes con MXNET utilizando modelos pre-entrenados (3.2)
  - https://github.com/dmlc/mxnet/blob/master/R-package/vignettes/classifyRealImageWithPretrainedModel.Rmd

- Clasificación de imágenes con MXNET utilizando modelos pre-entrenados y *fine tuning* (3.3)
  - https://statist-bhfz.github.io/cats_dogs_finetune (Punto 4 en adelante)

# Otros (3.1, 3.2. 3.3, 3.4, 3.5)

https://deeplearning4j.org/compare-dl4j-torch7-pylearn

- Tensorflow
  - https://www.tensorflow.org
    - https://www.tensorflow.org/tutorials/image_retraining
- Caffe
  - http://caffe.berkeleyvision.org
    - http://adilmoujahid.com/posts/2016/06/introduction-deep-learning-python-caffe/
- Keras
  - https://github.com/fchollet/keras
    - https://flyyufelix.github.io/2016/10/08/fine-tuning-in-keras-part2.html

# Pre-procesamiento de imágenes en R

**EBImage**
https://bioconductor.org/packages/release/bioc/vignettes/EBImage/inst/doc/EBImage-introduction.html

- Biblioteca para procesamiento de imágenes
- Orígenes en procesamiento de imágenes de células obtenidas con microscopios

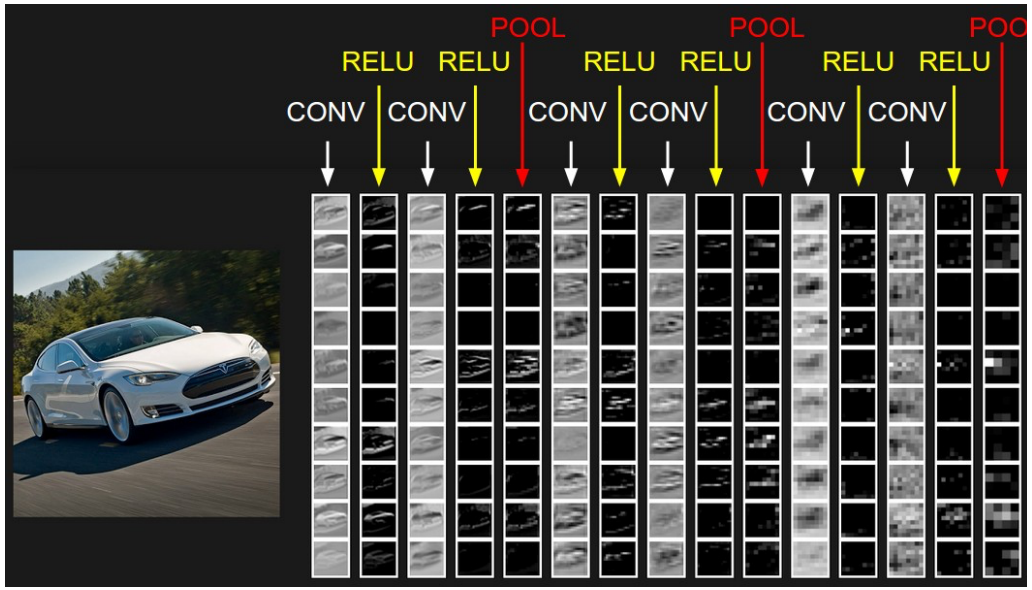
- Operaciones
  - Lectura
  - Visualización de propiedades
  - Gestión de espacios de color
  - Manipulación: negativo, filtrado por umbral
  - Transformaciones: traslación, rotación, redimensión, deformación
  - Filtrados: gaussiano, detección de bordes, eliminación de ruido
  - Operaciones morfológicas: erosión, dilatación
  - Segmentación
  - Manipulación: borrado, rellenado, iluminación

Alternativas: **imager**

# Extracción de características

CNN + otros clasificadores utilizando *feature maps*

Las *feature maps* pueden venir de un modelo pre-entrenado (por ejemplo, con ImageNET) [3.4] o de un modelo propio (idealmente, con *fine tuning*) [3.5]




Feature maps

Multi-clasificador

# Microsoft R (3.4)

**(previamente Revolution R)**

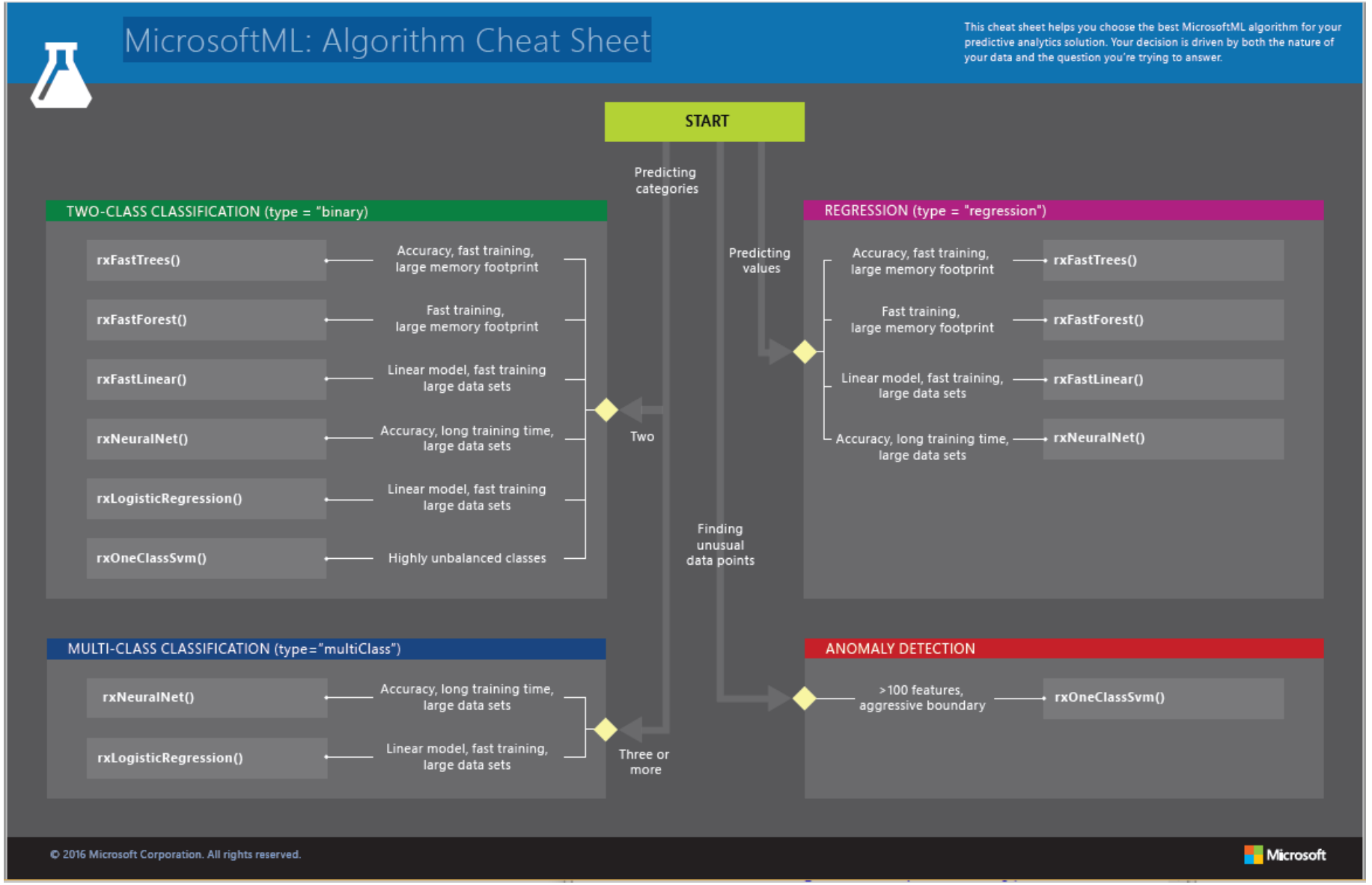
https://blogs.msdn.microsoft.com/rserver

- Extensión de R
  - Integración con plataforma Microsoft
    - R Tools for Visual Studio (https://www.visualstudio.com/es/vs/rtvs/)
    - MicrosoftML (https://msdn.microsoft.com/en-us/microsoft-r/microsoftml-introduction)
    - Azure Machine Learning (https://docs.microsoft.com/es-es/azure/machine-learning/)
    - Alteryx (https://www.r-bloggers.com/using-microsoft-r-with-alteryx/)
  - Herramientas gratuitas
    - Microsoft R Open (https://msdn.microsoft.com/en-us/microsoft-r/r-open)
    - Microsoft R Client (https://msdn.microsoft.com/en-us/microsoft-r/r-client-get-started) [Windows 64-bits, Linux]
    - Microsoft R Server (https://msdn.microsoft.com/en-us/microsoft-r/rserver)
  - Uso con RStudio
    - Microsoft R Client + RStudio (https://msdn.microsoft.com/en-us/microsoft-r/r-client-get-started > Step 2: Configure Your IDE**)**

# Microsoft R (3.4)

**MicrosoftML** (https://msdn.microsoft.com/en-us/microsoft-r/microsoftml/microsoftml)

- Algoritmos de aprendizaje automático
  - **rxFastTrees**
  - **rxFastForest**
  - **rxNeuralNet**
  - **rxLogisticRegression**
  - **rxEnsemble**
- Transformaciones de datos:
  - Crear *pipeline* con transformaciones de los datos
    - **concat**
    - **loadImage**
    - **resizeImage**
    - **extractPixels**
    - Extracción automática de características
      - **featurizeImage**
- Predicción:
  - **rxPredict**


MicrosoftML: Algorithm Cheat Sheet
This cheat sheet helps you choose the best MicrosoftML algorithm for your predictive analytics solution. Your decision is driven by both the nature of your data and the question you're trying to answer.
START
Predicting categories
Predicting values
Finding unusual data points
Two
Three or more
TWO-CLASS CLASSIFICATION (type = "binary)
rxFastTrees()
Accuracy, fast training, large memory footprint
rxFastForest()
Fast training, large memory footprint
rxFastLinear()
Linear model, fast training large data sets
rxNeuralNet()
Accuracy, long training time, large data sets
rxLogisticRegression()
Linear model, fast training large data sets
rxOneClassSvm()
Highly unbalanced classes
REGRESSION (type = "regression")
Accuracy, fast training, large memory footprint
rxFastTrees()
Fast training, large memory footprint
rxFastForest()
Linear model, fast training, large data sets
rxFastLinear()
Accuracy, long training time, large data sets
rxNeuralNet()
MULTI-CLASS CLASSIFICATION (type="multiClass")
rxNeuralNet()
Accuracy, long training time, large data sets
rxLogisticRegression()
Linear model, fast training, large data sets
ANOMALY DETECTION
>100 features, aggressive boundary
rxOneClassSvm()
© 2016 Microsoft Corporation. All rights reserved.
Microsoft

# Microsoft R (3.4)

## Extracción de características con modelo pre-entrenado (no *fine tuning*)

https://blogs.msdn.microsoft.com/rserver/2017/04/12/image-featurization-with-a-pre-trained-deep-neural-network-model/

**rxFeaturize** (https://msdn.microsoft.com/en-us/microsoft-r/microsoftml/packagehelp/rxfeaturize)

- data
- outData
- overwrite
- mlTransforms
  - **loadImage**
  - **resizeImage**
  - **extractPixels**
  - **featurizeImage** (https://msdn.microsoft.com/en-us/microsoft-r/microsoftml/packagehelp/featurizeimage)
    - **dnnModel** {resnet18, resnet50, resnet101, alexnet}

# Microsoft R (3.4)

## Algunas topologías + pre-entrenamiento con *ImageNet*

### ResNet (Microsoft)

K. He, X.Zhang, S. Ren, J. Sun (2015) Deep Residual Learning for Image Recognition. URL: https://arxiv.org/abs/1512.03385

- ResNet-18 (input: 224 x 224, features: 512)
- ResNet-50 (input: 224 x 224, features: 2048) [link]
- ResNet-101 (input: 224 x 224, features: 2048) [link]

### AlexNet (University of Toronto)

A. Krizhevsky, I. Sutskever, G.E. Hinton (2012) ImageNet Classification with Deep Convolutional Neural Networks. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks

- AlexNet (input: 227 x 227, features: 4096)

Más: https://adeshpande3.github.io/adeshpande3.github.io/The-9-Deep-Learning-Papers-You-Need-To-Know-About.html

# Microsoft R (3.4)

**Resultado** **Featurize-Cervix.R** **DeepLearning-Cervix_Classification.R**

| | Path | Label | Feature.0 | Feature.1 | Feature.2 | Feature.3 | Feature.4 | Feature.5 | Feature.6 | Feature.7 | Feature.8 | Feature.9 | Feature.10 | Feature.11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | ../r-workspace/train//Type_1/10.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |
| 2 | ../r-workspace/train//Type_1/1013.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |
| 3 | ../r-workspace/train//Type_1/1019.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |
| 4 | ../r-workspace/train//Type_1/102.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |
| 5 | ../r-workspace/train//Type_1/1024.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |
| 6 | ../r-workspace/train//Type_1/1027.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |
| 7 | ../r-workspace/train//Type_1/1033.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |
| 8 | ../r-workspace/train//Type_1/1040.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |
| 9 | ../r-workspace/train//Type_1/1056.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |
| 10 | ../r-workspace/train//Type_1/1059.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |
| 11 | ../r-workspace/train//Type_1/1061.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |
| 12 | ../r-workspace/train//Type_1/1070.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |
| 13 | ../r-workspace/train//Type_1/1071.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |
| 14 | ../r-workspace/train//Type_1/1077.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |
| 15 | ../r-workspace/train//Type_1/1079.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |
| 16 | ../r-workspace/train//Type_1/109.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |
| 17 | ../r-workspace/train//Type_1/1100.jpg | 1 | 1292353 | 0 | 0 | 0 | 61668144 | 0 | 0 | 3846910 | 3042535 | 43407016 | 200899184 | 0 |

Feature maps

Multi-clasificador

# MXNET (3.1, 3.2. 3.3, 3.4, 3.5)

**MXNET**
http://mxnet.io/tutorials/index.html#r

`model$symbol$get.internals()`: Obtener *features* (https://github.com/dmlc/mxnet/issues/2785)

**Ejemplo en Python:**

- https://github.com/dmlc/mxnet-notebooks/blob/master/python/how_to/predict.ipynb (***Extract features***)