data iku

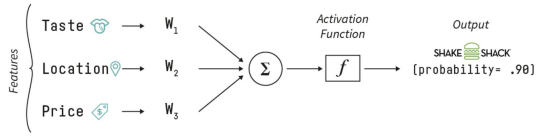
Building an image classification model

Goal

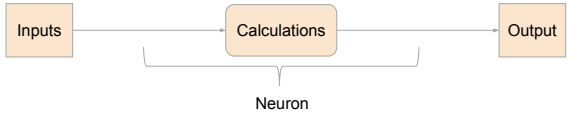
Classify images from the Simpsons to determine which character is in an image

What's a Neural Network?

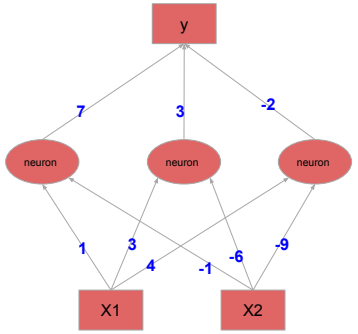
- A logistic regression is a linear model. Each coefficient corresponds to one input



- Neuron = calculation unit parametrized by coefficients called weights, very similar to a logistic regression



What's a Neural Network?



Neural Network = Architecture + Weights

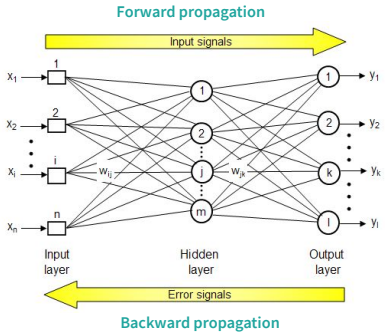
Architecture

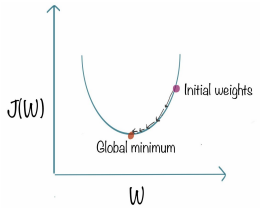
- number of neurons, linked between them, type of neurons …
- different architecture depending on the use case (image, text, …)

Weights

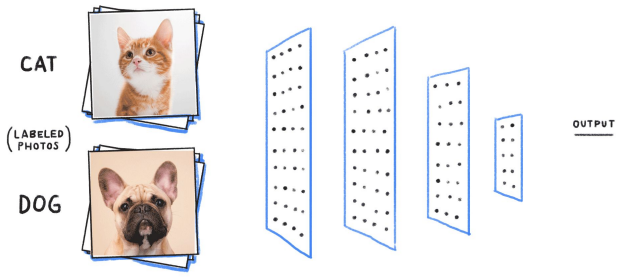
- parameters of the network
- depend on the data the model has been trained on
- "memory of the network"

Training mechanism





What is Deep Learning ?

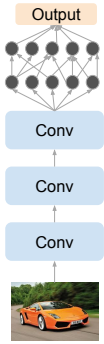


What is Deep Learning ?

The goal is to use labelled training images to learn the parameters (weights) of the function f

Building Neural Networks

Convolutional Neural Nets (CNNs)



Fully connected part

- Multiple fully connected layers
- Classify the image from the pattern identified within the feature extracted

Convolutional part

- Comprised of filters and feature maps
- Multiple convolutional layers to identify patterns
- Used as feature extractor to convert an image into features

Building Neural Networks

Common Issues

"I don't have enough labelled data"

"I don't have GPUs server"

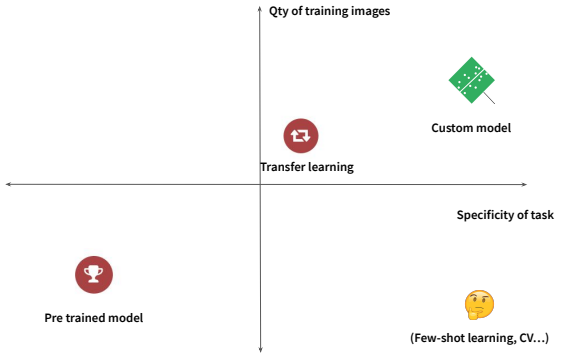
"I don't have a deep learning experts"

"I don't have the time to wait for model training "

3 ways you can do image classification

Using an of the shelf model pre-trained on lots of images	Fine tuning a pre-trained model to your own data by retraining it	Building and training your own Neural Network from scratch
👍 No training, no need to have labelled images, very fast	👍 Needs few images, fast and can perform well on specific use-cases	👍 Very flexible, performs well on specific use cases
👎 Not flexible, low performance on specific use cases	👎 Not flexible, can perform badly on specific use cases	👎 Needs lots of labelled images, long training time

3 ways you can do image classification



Now let's get our hands dirty !

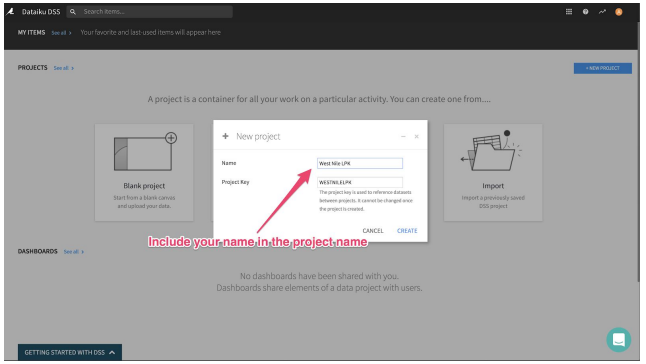
0. Getting started

Connecting to your instances

https://dss**X**-design-bern.training.dataiku.com/

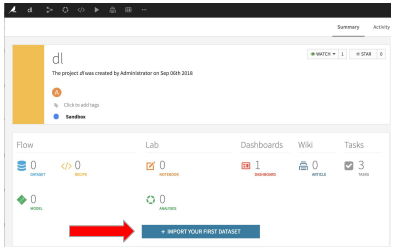
login : **user**Y
password : **obifYi**

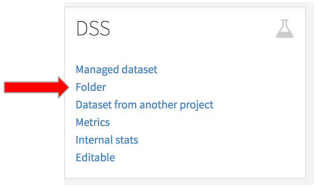
Create a project



Import the data

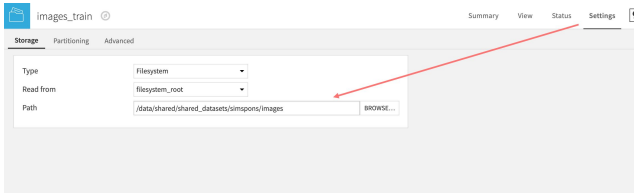
A folder is a directory where you can store every type of files (csv, pdf, jgp, …)
Let's create a folder to store our train and test images.





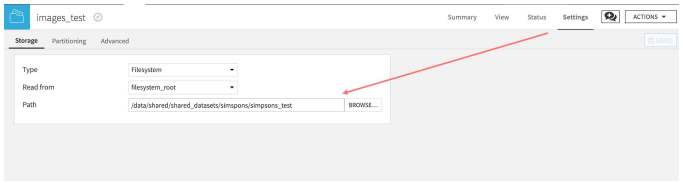
Import the training images

In the folder settings, Read from Filesystem Root
set Path to **/data/shared/shared_datasets/simspons/images**



Import the test images

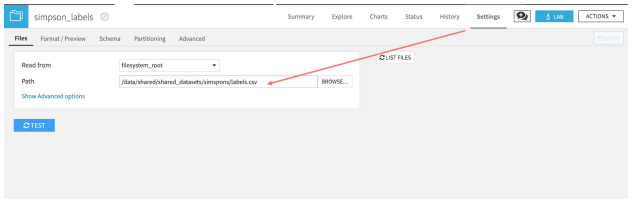
In the folder settings, Read from Filesystem Root
set Path to **/data/shared/shared_datasets/simspons/simpsons_test**



Import the labels dataset

This dataset maps image files to labels i.e simpson character and will be used for training and testing the model

Create a Filesystem dataset, read from Filesystem root :



1. Using a pre-trained model to classify images

Using a pretrained model to classify images

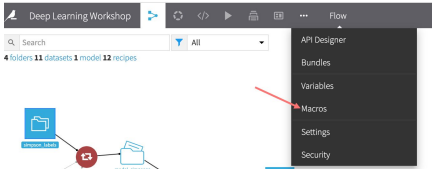
Using existing knowledge of a model trained on a massive image dataset is often a good idea for generic problems and in case of limited resources to train our network

Here, we are going to use a model that was previously trained on the **ImageNet** dataset (over 20,000 categories from more than 14 million annotated images for computer vision research)

Download the pretrained model using a macro

In DSS, Macros execute predefined project-level or instance-level actions.

The « Download pre-trained model » macro will create a managed folder on the flow and will download the pretrained selected model ResNet.



Macro "Download pre-trained model"

Use this macro to download model files to your project.

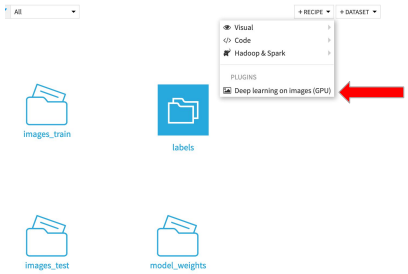
Output folder name: model_weights — Use a different folder for each downloaded model.

Pre-trained model to download: Resnet trained on Imagenet

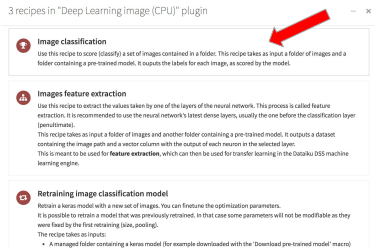
Use the pretrained model to predict test images

In DSS, each time you make a transformation of your data (an aggregation, a join, a python-based preparation), you will be creating a recipe. Recipes have input datasets and output datasets. The transformation to apply is configured in the recipe settings.

Additional toolboxes, called **plugins**, can be installed. We will use the Deep Learning image (GPU) plugin.

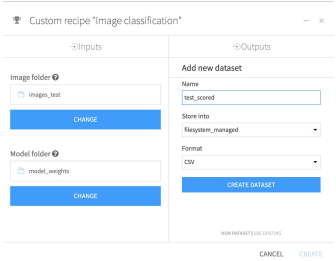


Use the pretrained model to predict test images



The goal is to score the uploaded images :

Choose the Image Classification recipe



Store the output into filesystem_managed

Use the pretrained model to predict test images

Recipe settings

Max number of class labels: 5

Min probability threshold: 0.1 — Class labels less

Use GPU: ☑

List of GPUs to use: 0 — Comma separated list of GPU indexes

Memory allocation rate per GPU: 1

Select the GPU that corresponds to your group !

images	prediction	error
string Text	string Object	bigint Integer
bart_simpson_49.jpg	{}	0
abraham_grampa_simpson_1.jpg	{"volleyball": 0.252938479185104 37}	0
charles_montgomery_burns_21.jpg	{"desk": 0.16564202308654785}	0
lisa_simpson_14.jpg	{"envelope": 0.12832503020763397, "street_sign": …	0
sideshow_bob_48.jpg	{"book_jacket": 0.4522744715213 7756, "comic_bo…	0
bart_simpson_9.jpg	{"comic_book": 0.25136744976604 37, "book_jacket…	0
milhouse_van_houten_9.jpg	{"picket_fence": 0.14136412739753723, "ski": 0.112…	0
homer_simpson_43.jpg	{"racer": 0.18443113565444946, "comic_book": 0.1…	0
marge_simpson_22.jpg	{"drilling_platform": 0.312079519033432, "harvest…	0
moe_szyslak_5.jpg	{"comic_book": 0.5230382680892944}	0
krusty_the_clown_19.jpg	{"toilet_seat": 0.7031659483909607, "potter's_whe…	0
principal_skinner_1.jpg	{"cc{"toilet_seat": 0.7031659483909607, "potter's_wheel": 0.12853971123695374}	

Visualize predicted labels in the output dataset

The labels of the pretrained model are too generic and not adapted to our problem

2. Re-training a pre-trained model on Simpson images

Transfer learning

Transfer learning is a method where a model developed for a task is reused as the starting point for a model on a second learning task

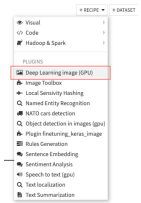
Here, we are going to re-train (i.e adapt) a previously trained model on our specific training dataset

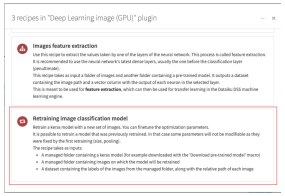
We are not going to retrain all the layers, we will only retrain the last layer of the Neural Network and will capitalize on the knowledge already accumulated by the previous layers.

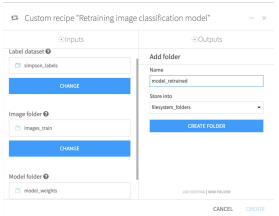
We will do so using the Deep Learning Image classification **plugin's retraining recipe**!

Retrain the Resnet model on the training data

Create a new recipe from the plugin, use model downloaded in step 1 as a basis



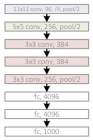




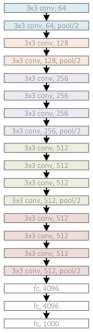
Resnet Background

Computer Vision - Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)



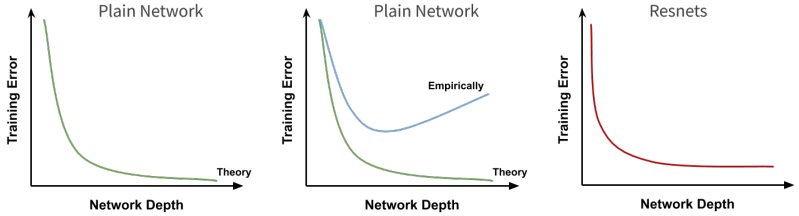
GoogleNet, 22 layers
(ILSVRC 2014)

Reproduced from Kaiming He's 2016 ICML tutorial "Deep learning gets way deep"

Resnet Background

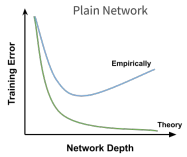
Simply Stacking?

Is the process of *developing better learning networks*, simply a matter of stacking more layers in the architecture (going deeper)?

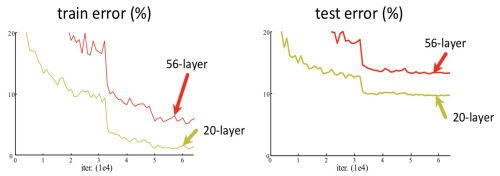


Resnet Background

Simply Stacking?

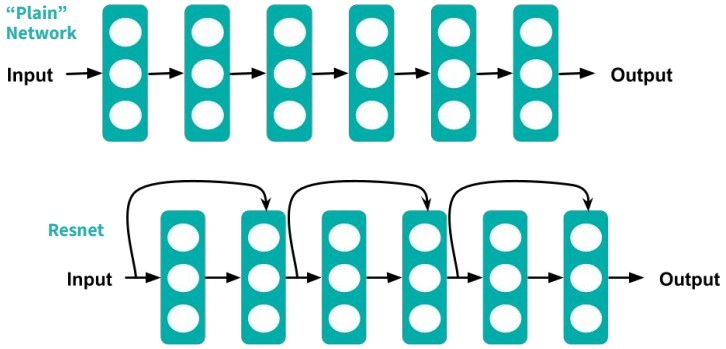


Example of empirical evidence - see He, Zhang, Ren, & Sun (2015)
Deep Residual Learning for Image Recognition in arXiv:1512.03385



- Overly deep plain nets have higher training error
- Runs counter to what we would expect

What is a Residual Network



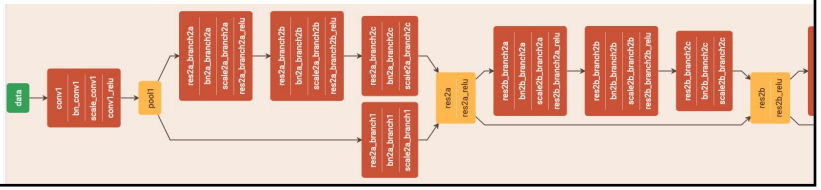
Retrain the Resnet model on the training data

Resent is comprised of 152 layers (compared to 16 in VGG16 and 8 in AlexNet) - i.e. deep

Image sizes: 224 x 224
Batch normalization before each convolution
SGD with mini batch size 256
No dropout

The architecture can be found here

We can start retraining only the last layer, then last layer + last block etc ...



Retrain the model on the training data

Retraining image classification model

Retrain a keras model with a new set of images. You can finetune the optimization parameters.
It is possible to retrain a model that was previously retrained. In that case some parameters will not be modifiable as they were fixed by the first retraining (size, pooling).
The recipe takes as inputs:

- A managed folder containing a keras model (for example downloaded with the 'Download pre-trained model' macro)
- A managed folder containing images on which the model will be retrained
- A dataset containing the labels of the images from the managed folder, along with the relative path of each image

Plugin documentation

DATASET WITH LABELS

Image filename column: path

Describes the relative path to where the image is

Label column: target

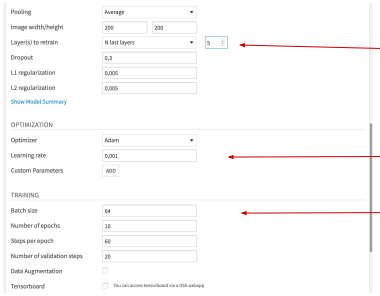
Train ratio

Random Seed

full_path
path
target

Labels of the images from the managed folder

Retrain the model on the training data



Start with last layer only, then increase to 5 to retrain last block too

How big the weight updates will be, start at 0,001

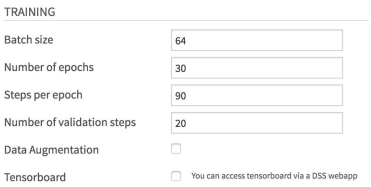
Number of images used for one weight update, start at 32 and increase until you get memory error

Hands-on Part III

Model retraining

Configure the training

- **Batch size** : number of images in one batch (one weight update). Should be as high as possible. But restricted by the GPU memory. Start with 32 and extend until you have a memory error
- **Steps per epochs** : number of weight updates between each evaluation of the model on the validation dataset
- **Number of epochs** : for each epoch, the neural network will see batch size x steps per epochs
- **Number of validations steps** x batch size images to evaluate the validation metrics



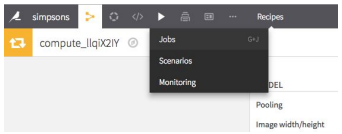
The Simpsons dataset

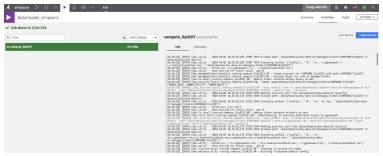
- 7000 images
- *How to fill in the training parameters for the neural network to see one path of the full dataset in one epoch ?*

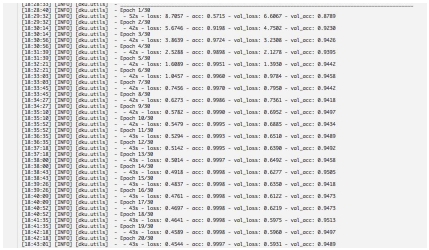
Hands-on Part III

Model evaluation

Check the logs to monitor learning metrics



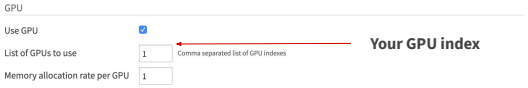




- Epoch number
- loss, acc : evaluation of the loss function and accuracy on the training set
- **val_loss, val_acc** : evaluation of the loss function and accuracy on the validation set → **monitoring**

Retrain the model on the training data

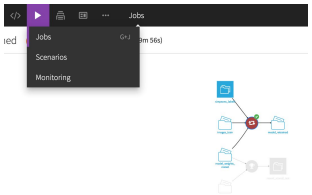
Using GPU will improve dramatically training time, GPUs enable to do parallel operations (matrix multiplications) faster than CPU during training phase



Your GPU index

Retrain the model on the training data

You can review the Job logs to see how your retraining efforts are progressing.

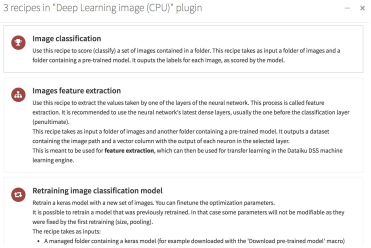


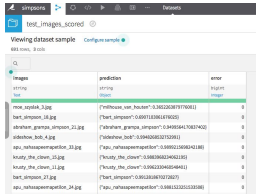

```
[09:53:21] [INFO] [dku.utils]  - Epoch 1/10
[09:57:25] [INFO] [dku.utils]  - 244s - loss: 1.8831 - acc: 0.4331 - val_loss: 1.7453 - val_acc: 0.4500
[09:57:25] [INFO] [dku.utils]  - Epoch 2/10
[09:57:48] [INFO] [dku.utils]  - 24s - loss: 1.0070 - acc: 0.7334 - val_loss: 1.0802 - val_acc: 0.6761
[09:57:48] [INFO] [dku.utils]  - Epoch 3/10
[09:58:12] [INFO] [dku.utils]  - 24s - loss: 0.7581 - acc: 0.8003 - val_loss: 0.8223 - val_acc: 0.7649
[09:58:12] [INFO] [dku.utils]  - Epoch 4/10
[09:58:36] [INFO] [dku.utils]  - 24s - loss: 0.6301 - acc: 0.8438 - val_loss: 0.7801 - val_acc: 0.7744
[09:58:36] [INFO] [dku.utils]  - Epoch 5/10
[09:59:00] [INFO] [dku.utils]  - 24s - loss: 0.5228 - acc: 0.8734 - val_loss: 0.6939 - val_acc: 0.8027
[09:59:00] [INFO] [dku.utils]  - Epoch 6/10
[09:59:23] [INFO] [dku.utils]  - 24s - loss: 0.4558 - acc: 0.8901 - val_loss: 0.6751 - val_acc: 0.8215
[09:59:23] [INFO] [dku.utils]  - Epoch 7/10
[09:59:47] [INFO] [dku.utils]  - 24s - loss: 0.4198 - acc: 0.9010 - val_loss: 0.6349 - val_acc: 0.8129
[09:59:47] [INFO] [dku.utils]  - Epoch 8/10
[10:00:12] [INFO] [dku.utils]  - 24s - loss: 0.3592 - acc: 0.9177 - val_loss: 0.6141 - val_acc: 0.8278
[10:00:12] [INFO] [dku.utils]  - Epoch 9/10
[10:00:35] [INFO] [dku.utils]  - 24s - loss: 0.3166 - acc: 0.9336 - val_loss: 0.6202 - val_acc: 0.8286
[10:00:35] [INFO] [dku.utils]  - Epoch 10/10
[10:00:59] [INFO] [dku.utils]  - 24s - loss: 0.3028 - acc: 0.9362 - val_loss: 0.5871 - val_acc: 0.8341
```

Training Validation

Apply the new model to score the test set

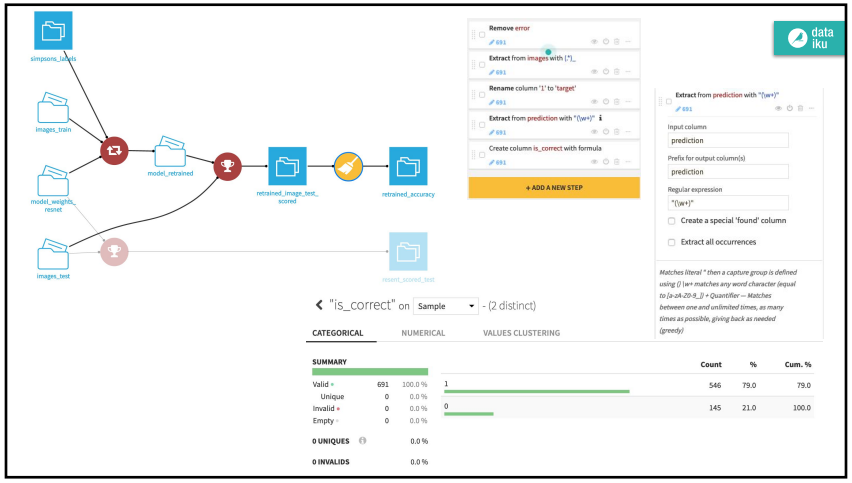
Apply your most accurate model on the test set





BONUS : Compute the accuracy of the model on the test set

- With visual recipes
- In a python/R notebook
- in a dataset metric



Come Again?

We have scored and transferred learning from a state of the art Neural Network architecture to our classification problem … without a single line of code

4. Feature Extraction

Deep Learning

Feature Extraction

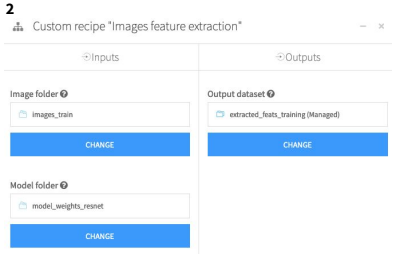
- **Feature extraction**
  - Convert an object in features
  - These features can be used to represent the image in another model
  - Images → Features → ML model
- Essentially we use the trained CNN as an arbitrary feature extractor: An image is pushed through the network, stopping at an arbitrary layer and generating values that can be treated as a feature values for additional ML
- If you don't have any labeled data, extract features from a pretrained model and insert this new information in your data pipeline
- Feature extraction ∴ allows for using a CNN & additional ML in prediction tasks where the classes were not part of the original training for the CNN

1

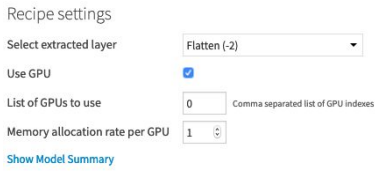
Images feature extraction

Use this recipe to extract the values taken by one of the layers of the neural network. This process is called feature extraction.
It is recommended to use the neural network's latest dense layers, usually the one before the classification layer (penultimate).
This recipe takes as input a folder of images and another folder containing a pre-trained model. It outputs a dataset containing the image path and a vector column with the output of each neuron in the selected layer.
This is meant to be used for **feature extraction**, which can then be used for transfer learning in the Dataiku DSS machine learning engine.

2



3 Use the last layers to represent the CNN as vector of features



Review the resulting feature extractions

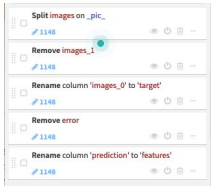
images string Text	prediction string Array	error bigint Integer
lisa_simpson_pic_0094.jpg	[0.006433751899749041, 1.357283592224121, 0.0, …	0
krusty_the_clown_pic_0058.jpg	[0.184280663728714, 0.1969044953584671, 0.0, 0.2…	0
sideshow_bob_pic_0411.jpg	[0.08512356877326965, 1.245836019515991 2, 0.00…	0
apu_nahasapeemapetilon_pic_0323.jpg	[0.1465371996164322, 3.063199758529663, 0.0, 1.7…	0
homer_simpson_pic_0132.jpg	[0.032075051218271255, 0.47969865798950195, 0.…	0
abraham_grampa_simpson_pic_0453.jpg	[0.4685455560684204, 0.9452148079872131, 0.231…	0
ned_flanders_pic_0355.jpg	[0.12418466806411743, 0.7054527401924133, 0.0, …	0

Feature vectors that can be **used as an input** of a machine learning model or in any application to **represent image signal** (similarity between images, recommendation engine, visualization)

Challenge - Prep the Data

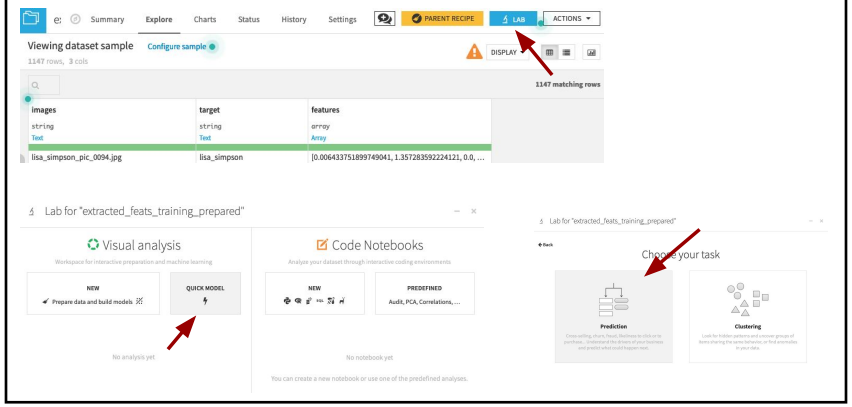
1. Create a prepare recipe that extracts the target based on the image name column

- Split to retrieve the character name
- Rename the columns properly

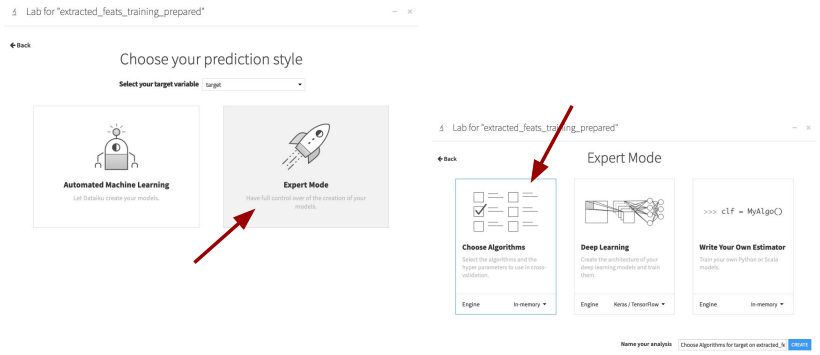


images	target	features
string	string	array
Text	Text	Array
lisa_simpson_pic_0094.jpg	lisa_simpson	[0.006433751899749041, 1.357283592224121, 0.0, …
krusty_the_clown_pic_0058.jpg	krusty_the_clown	[0.18428066372871 4, 0.1969044953584671, 0.0, 0.2…
sideshow_bob_pic_0411.jpg	sideshow_bob	[0.08512356877326965, 1.2458360195159912, 0.00…
apu_nahasapeemapetilon_pic_0323.jpg	apu_nahasapeemapetilon	[0.146537199616432 2, 3.063199758529663, 0.0, 1.7…
homer_simpson_pic_0132.jpg	homer_simpson	[0.03207505121827125 5, 0.4796986579895019 5, 0.…
abraham_grampa_simpson_pic_0453.jpg	abraham_grampa_simpson	[0.46854555606842 04, 0.9452148079872131, 0.231…
ned_flanders_pic_0355.jpg	ned_flanders	[0.12418466806411743, 0.7054527401924133, 0.0, …

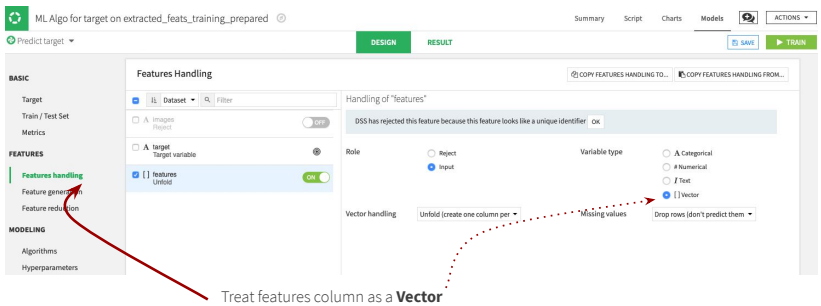
Use Features in New ML Algorithm(s)

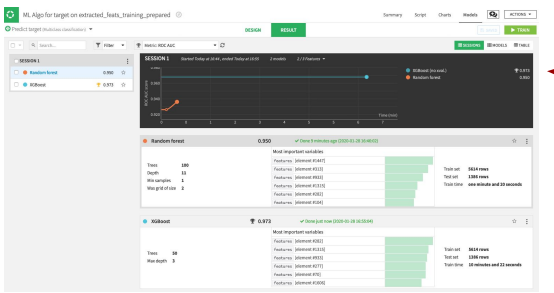


Building ML Model



Building ML Model



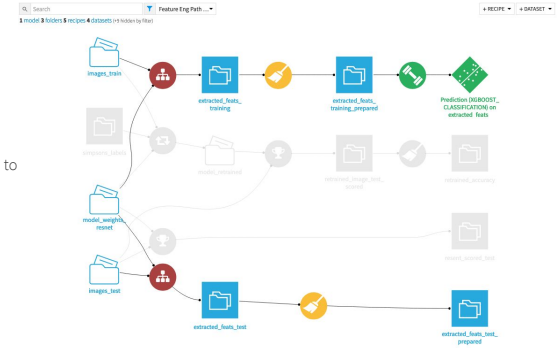


Experiment with some different models

Deploy your chosen model to the flow

Score Test Images with the ML

Reproduce the data flow : extract features from test images and prepare the data



Hint:
SHIFT + click for multi selection to COPY part of the workflow

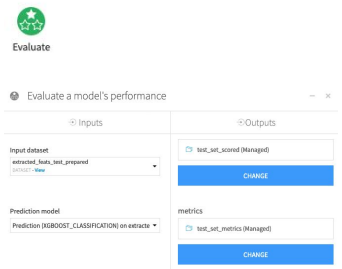
or

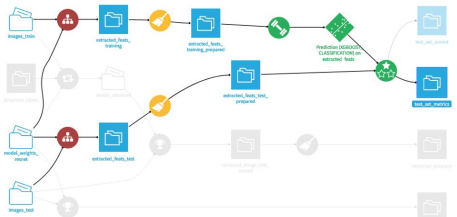
Reproduce recipe step-by-step

Evaluate that prepared test set

data iku

Evaluate model on already-known data





data iku

Hope you had FUN!

data
iku