

Automate Machine Learning with ProActive using Open Source technologies



Summary



ACTIVEeon
SCALE BEYOND LIMITS

1. Overview of the ProActive Machine Learning Open Studio (MLOS)

2. ProActive Machine Learning Open Studio - Technical Report

3. Example of workflows using the MLOS tasks

4. Example of workflows using customized machine learning tasks

Video demonstration is available on this [link](#)



[--Overview--]



Introduction

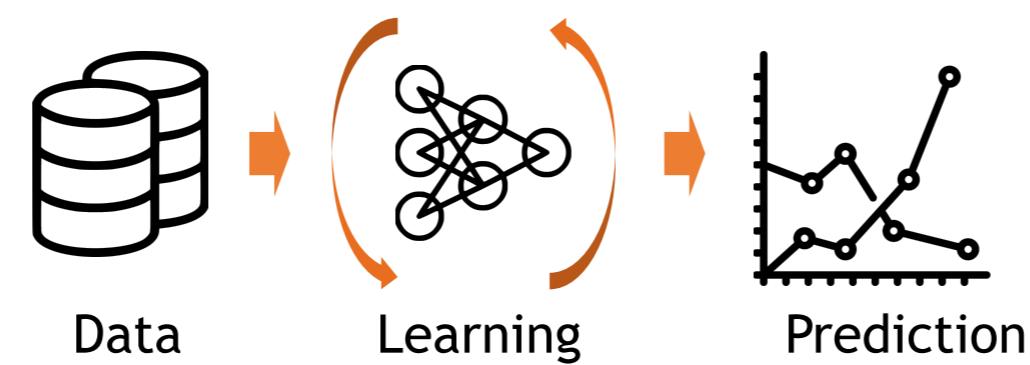
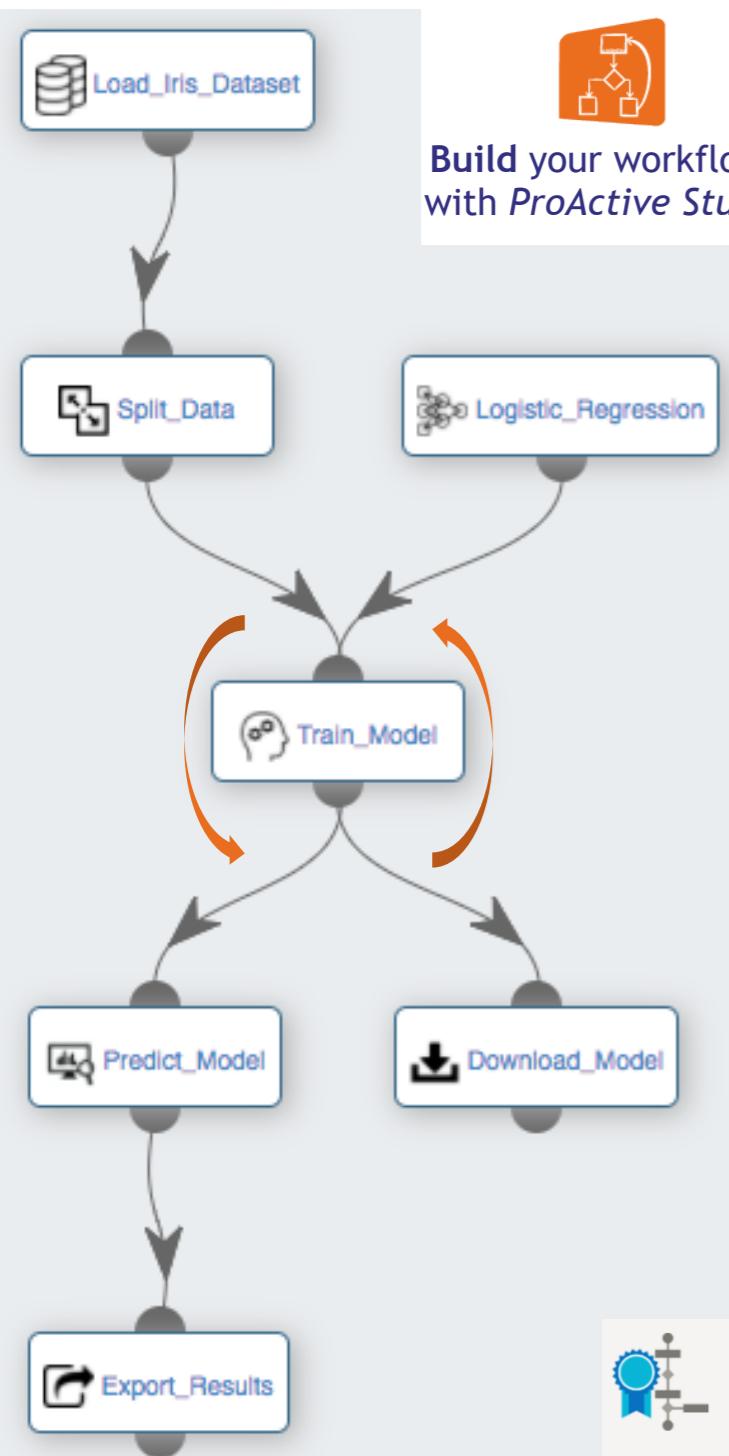


ProActive Machine Learning Open Studio is an interactive visual interface that allows to easily build, deploy, and test predictive on your data. It provides different tasks, for instance, import data, export data, save model, load a trained model, train prediction, and evaluate a model, etc. These tasks enable the user to develop powerful machine learning solutions, such as: fraud detection, text analysis, online offer recommendations, prediction of equipment failures, facial expression analysis, etc. Proactive Machine Learning Studio is open source and allows easy task parallelization, running them on resources matching constraints (Multi-CPU, GPU, data locality, library).

ProActive Machine Learning Studio



ACTIVEeon
SCALE BEYOND LIMITS



- Automate with **intelligence**.
- Compose smart workflows by visual drag-and-drop.
- Enable data scientists to graphically **build and deploy** large-scale experiments.
- Enabled real-time **data visualisation**.
- Use your favorite **machine learning and deep learning libraries**.
- Scaled-up under **local network resources** as well in the cloud.





[-- Technical Report --]



Summary



ACTIVEeon
SCALE BEYOND LIMITS

1. Public Datasets

Load_Boston_Dataset
Load_iris_Dataset

2. Input and Output Data

Download_Model
Export_Data
Import_Data
Load_Model
Log_Parser

3. Data Preprocessing

Add_Data
Add_Label
Filter_Data
Split_Data

3. Feature Extraction

Feature_Vector_Extractor
Time_Series_Feature_Extractor

Summary



4.1 ML Classification

Gaussian_Naive_Bayes

Logistic_Regression

Support_Vector_Machines

4.2 ML Regression

Bayesian_Ridge_Regression

Linear_Regression

Support_Vector_Regresion

3.3 ML Clustering

K_Means

Mean_Shift

4. Train

Train_Clustering_Model

Train_Model

5. Predict

Predict_Clustering Model

Predict_Model



The screenshot shows a user interface for machine learning tasks. The left sidebar has a title bar with a gear icon and the text "machine learning". It includes a "Pin open" button and a list of sections:

- 1. Public Datasets**
 - Load_Boston_Dataset
 - Load_Iris_Dataset
- 2. Input and Output Data**
 - Download_Model
 - Export_Results
 - Import_Data
 - Load_Model
 - Log_Parser
- 3. Data Preprocessing**
 - Add_Data
 - Add_Label
 - Filter_Data
 - Split_Data
- 4. Features Extraction**
 - Feature_Vector_Extractor
 - Time_Series_Feature_Extractor

The main panel on the right lists various machine learning models under numbered sections:

- 5.1 ML Classification**
 - Gaussian_Naive_Bayes
 - Logistic_Regression
 - Support_Vector_Machines
- 5.2 ML Regression**
 - Bayesian_Ridge_Regression
 - Linear_Regression
 - Support_Vector_Regression
- 5.3 ML Clustering**
 - K_Means
 - Mean_Shift
- 6. Train**
 - Train_Clustering_Model
 - Train_Model
- 7. Predict**
 - Predict_Clustering_Model
 - Predict_Model



1. Task Name: Load_Boston_Dataset

2. Task Overview:

Load and return the Boston House-Prices dataset.

Features	Targets	Dimensionality	Samples total
Real, positive	Real 5. -50	13	506

3. How to Use Load_Boston_Dataset

- The Boston House-Prices is a dataset regression, you can only use it with a regression algorithm, such as Linear Regression and Support Vector Regression.
- After this task, you can use the **Split_Data** task to divide the dataset into training and testing sets.



1. Task Name: Load_Iris_Dataset

2. Task Overview:

Load and return the iris dataset.

Features	Classes	Dimensionality	Samples per Class	Samples total
Real, positive	3	4	50	150

3. How to Use Load_Iris_Dataset

- The Iris is a dataset for classification, you can only use it with a classification algorithm, such as Support Vector Machines and Logistic Regression.
- After this task, you can use the **Split_Data** task to divide the dataset into training and testing sets.

Input and Output Data



ACTIVEeon
SCALE BEYOND LIMITS

1. Task Name: Download_Model

2. Task Overview:

Download a trained model on your computer device.

3. How to Use Download_Model

- The **Download_Model** task should be used after the **Train_Model** or **Train_Clustering_Model** tasks.



1. Task Name: Export_Results

2. Task Overview:

Export the results of the predictions generated by a classification, clustering or regression algorithm.

3. Task Variables

OUTPUT_FILE: string HTML or CSV

Converts the prediction results to HTML or CSV file.

4. How to Use Export_Results

- The **Export_Results** task should be used after **Predict_Model** or **Predict_Clustering_Model** tasks.

Input and Output Data



ACTIVEeon
SCALE BEYOND LIMITS

1. Task Name: Import_Data

2. Task Overview:

Load data from external sources.

3. Task Variables

FILE_URL: string

Enter you URL of the CSV file.

FILE_DELIMITER: str

Delimiter to use.

IS_LABELED_DATA: boolean

If your data have label data.

Input and Output Data



ACTIVEeon
SCALE BEYOND LIMITS

Remark: Your CSV file should be in a table format. See the example below.

	preg	plas	pres	skin	test	mass	pedi	age	class
0	1	85	66	29	0	26.6	351	31	0
1	8	183	64	0	0	23.3	672	32	1
2	1	89	66	23	94	28.1	167	21	0
3	0	137	40	35	168	43.1	2288	33	1
4	5	116	74	0	0	25.6	201	30	0
5	3	78	50	32	88	31.0	248	26	1
6	10	115	0	0	0	35.3	134	29	0
7	2	197	70	45	543	30.5	158	53	1
8	8	125	96	0	0	0.0	232	54	1
9	4	110	92	0	0	37.6	191	30	0
10	10	168	74	0	0	38.0	537	34	1
11	10	139	80	0	0	27.1	1441	57	0
12	1	189	60	23	846	30.1	398	59	1
13	5	166	72	19	175	25.8	587	51	1
14	7	100	0	0	0	30.0	484	32	1
15	0	118	84	47	230	45.8	551	31	1
16	7	107	74	0	0	29.6	254	31	1

Rows should be the samples

Columns should be the features



1. Task Name: Load_Model

2. Task Overview:

Load a trained model, and use it to make predictions for new coming data.

3. Task Variables

MODEL_URL: string, default: <https://s3.eu-west-2.amazonaws.com/activeeon-public/models/pima-indians-diabetes.model>
Type the URL to load your trained model.

4. How to Use Load_Model

- The **Load_Model** task should be used before **Predict_Model** or **Predict_Clustering_Model** to make predictions.

Input and Output Data



ACTIVEeon
SCALE BEYOND LIMITS

1. Task Name: Log_Parser

2. Task Overview:

Convert an **unstructured** raw log file into a structured one by matching a group of event patterns.

3. Task Variables:

LOG_FILE: string

refers to the URL of the raw log file that you need to parse.

PATTERNS_FILE: string

refers to the URL of the CSV file that contains the different REGEX expressions of each possible pattern and their corresponding variables.

The csv file must contain three columns:

A. **id_pattern:** integer

specifies the identifier of each pattern

B. **Pattern:** Regex expression

Defines the regex expression of a specific pattern

C. **Variables:** string

refers to the name of each variable included in the pattern.

N.B: Use the symbol '*' for variables that you need to neglect. (e.g. in the example below the 5th variable is neglected)

N.B: All variables specified in each Regex expressions have to be mentioned in the column « Variables » in the right order.

Input and Output Data



ACTIVEeon
SCALE BEYOND LIMITS

STRUCTURED_LOG: string

indicate the extension of the file where you will save the resulted structured logs.

Example of a PATTERN_File

id_pattern	Pattern	Variables
1	([0-9]+) ([0-9]+) ([0-9]+) ([A-Z]+)(.*) Adding an already existing block (.*)	date, time, pid, status, *, id_block
2	([0-9]+) ([0-9]+) ([0-9]+) ([A-Z]+)(.*) Verification succeeded for (.*)	date, time, pid, status, *, id_block



Regex Expressions to match

Variables that need to be extracted

N.B: The order of variables need to be respected :
([0-9]+) refers to date
([0-9]+) refers to time
([0-9]+) refers to pid
([A-Z]+) refers to status
(.*) refers to * (i.e. * means ignore this variable)
(.*) refers to id_block

Data Preprocessing



ACTIVEeon
SCALE BEYOND LIMITS

1. Task Name: Add_Data

2. Task Overview:

Concatenate the new added data to the original input data.

3. Task Variables:

FILE_URL: string

refers to the URL of the data that you need to add.

FILE_DELIMITER: string

Delimiter to use.

IS_LABELED_DATA: boolean

TRUE if the data is labeled.

IGNORE_INDEX: boolean

False if you need to ignore the index of the input data.

Data Preprocessing



ACTIVEeon
SCALE BEYOND LIMITS

1. Task Name: Add_Label

2. Task Overview:

Add a new label column to the original input data.

3. Task Variables

FILE_URL: string

refers to the URL of the file containing the labels that you need to concatenate.

FILE_DELIMITER: string

Delimiters to use.

Data Preprocessing



ACTIVEeon
SCALE BEYOND LIMITS

1. Task Name: Filter_Data

2. Task Overview:

Query the columns of your data with a boolean expression.

3. Task Variables:

Query: string

The query string to evaluate.

FILTERED_FILE_OUTPUT: string

Refers to the extension of the file where the resulted filtered data will be saved.

Options [csv,html]



1. Task Name: Split_Data

2. Task Overview:

Separate data into train and test subsets.

3. Task Variables:

TRAIN_SIZE: float, default: 0.7

This parameter must be float within the range (0, 100), not including the values 0 and 100.

4. How to Use Split_Data

- The input data file must contains at least two rows.
- You should use the **Split_Data** task after the **Import_Data**, **Load_Boston_Dataset** and **Load_Iris_Dataset** tasks.

Feature Extraction



1. Task Name: Feature_Vector_Extractor

2. Task Overview:

Encode **structured** data into numerical feature vectors whereby machine learning models can be applied.

3. Task Variables

SESSION_COLUMN: string

Refers to the ID of the entity that you need to represent.

FILE_OUT_FEATURES: string

Refers to the file name where the resulted features will be saved.

PATTERN_COLUMN: string

Refers to the index of column containing the log patterns.[specific to features extraction from logs]

PATTERNS_COUNT_FEATURES: boolean

Indicates if count features related to log patterns need to be extracted or not.

STATE_VARIABLES: string

Indicates the different variables that need to be considered to extract features according to their content.

COUNT_VARIABLES: string

Refers to the different variables that need to be considered to count their distinct content.

STATE_COUNT_FEATURES_VARIABLES: boolean

Indicates if state and count features need to be extracted or not.



1. Task Name: Time_Series_Feature_Extractor

2. Task Overview:

Extracts features by considering the **temporal** distribution of data.

3. Task Variables:

SESSION_COLUMN: string

Refers to the column name of the entity that you need to represent.

SLIDING_STEP: integer(in Minutes)

Specifies the number of minutes to jump while navigating from a window to another one.

WINDOW_SIZE: integer (in Minutes)

Indicates the window size in Minutes.

START_DATE_TIME: date

Indicates the date starting from it the time series vectors representing each entity [Session] will be extracted.

PATTERN_COLUMN: string

Refers to the column name containing the log patterns.[specific to features extraction from logs]

PATTERNS_COUNT_FEATURES: boolean

True if you want to count the number of detected patterns per entity [Session].

STATE_VARIABLES: string

Indicates the name of columns to use as features by counting the number of occurrence of their distinct content.

Feature Extraction



COUNT_VARIABLES: string

Specifies the name of the columns that you need to add as features by counting their distincts content per entity [Session].

STATE_COUNT_FEATURES_VARIABLES: boolean

True if you need to extract state and count features that you already specified in **COUNT_VARIABLES** and **STATE_VARIABLES**.



1. Task Name: Gaussian_Naive_Bayes

2. Task Overview:

Naive Bayes classifier is a family of simple probabilistic classifier based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

3. Task Variables:

PRIORS: array-like, shape (n_classes)

Prior probabilities of the classes. If specified the priors are not adjusted according to the data.

4. How to Use Gaussian_Naive_Bayes

- The **Gaussian_Naive_Bayes** task should be connected with **Train_Model / Train_Clustering_Model** or **Predict_Model / Predict_Clustering_Model**.



1. Task Name: Logistic_Regression

2. Task Overview:

Logistic Regression is a regression model where the Dependent Variable (DV) is categorical.

3. Task Variables:

PENALTY: string, default: 'l2'

Used to specify the norm used in the penalization. The 'newton-cg', 'sag' and 'lbfgs' solvers support only l2 penalties.

SOLVER: 'newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga', default: 'liblinear'

Algorithm to use in the optimization problem.

MAX_ITERATIONS: int, default: 100

Useful only for the newton-cg, sag and lbfgs solvers. Maximum number of iterations taken for the solvers to converge.

N_JOBS: int, default: 1

Number of CPU cores used. If given a value of -1, all cores are used.

3. How to Use Logistic_Regression

- The **Logistic_Regression** task should be connected with **Train_Model / Train_Clustering_Model** or **Predict_Model / Predict_Clustering_Model**.



1. Task Name: Support_Vector_Machines

2. Task Overview:

Support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification.

3. Task Variables:

C: float, optional (default=1.0)

Penalty parameter C of the error term.

KERNEL: string, optional (default='rbf')

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable.

3. How to Use Support_Vector_Machines:

- The **Support_Vector_Machines** task should be connected with **Train_Model / Train_Clustering_Model** or **Predict_Model / Predict_Clustering_Model**.



1. Task Name: Bayesian_Ridge_Regression

2. Task Overview:

Bayesian linear regression is an approach to linear regression in which the statistical analysis is undertaken within the context of Bayesian inference.

3. Task Variables:

N_ITERATIONS: int, optional (default=300)

Penalty parameter C of the error term.

ALPHA_1: float, default 1.e-6

Hyper-parameter : shape parameter for the Gamma distribution prior over the alpha parameter.

ALPHA_2: float, default 1.e-6

Hyper-parameter : inverse scale parameter (rate parameter) for the Gamma distribution prior over the alpha parameter.

LAMBDA_1: float, default 1.e-6

Hyper-parameter : shape parameter for the Gamma distribution prior over the lambda parameter.

LAMBDA_2: float, default 1.e-6

Hyper-parameter : inverse scale parameter (rate parameter) for the Gamma distribution prior over the lambda parameter.

4. How to Use Bayesian_Ridge_Regression

- The **Bayesian_Ridge_Regression** task should be connected with **Train_Model / Train_Clustering_Model** or **Predict_Model / Predict_Clustering_Model**.

More information about this task can be found [here](#).



1. Task Name: Linear_Regression

2. Task Overview:

Linear regression is a linear approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X .

3. Task Variables:

N_JOBS: int, default:1

Number of jobs to run in parallel.

4. How to Use Linear_Regression

- The **Linear_Regression** task should be connected with **Train_Model / Train_Clustering_Model** or **Predict_Model / Predict_Clustering_Model**.



1. Task Name: Support_Vector_Regression

2. Task Overview:

Support vector regression are supervised learning models with associated learning algorithms that analyze data used for regression.

3. Task Variables:

C: float, optional (default=1.0)

Penalty parameter C of the error term.

KERNEL: string, optional (default='rbf')

Specifies the kernel type to be used in the algorithm. It must be one of 'linear', 'poly', 'rbf', 'sigmoid', 'precomputed' or a callable.

EPSILON: float, optional (default=0.1)

It specifies the epsilon-tube within which no penalty is associated in the training loss function with points predicted within a distance epsilon from the actual value.

4. How to Use Support_Vector_Regression

- The **Support_Vector_Regression** task should be connected with **Train_Model / Train_Clustering_Model** or **Predict_Model / Predict_Clustering_Model**.



1. Task Name: K_Means

2. Task Overview:

K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

3. Task Variables:

N_CLUSTERS: int, optional, default: 8

The number of clusters to form as well as the number of centroids to generate.

MAX_ITERATIONS: int, default: 300

Maximum number of iterations of the k-means algorithm for a single run.

N_JOBS: int, default: 1

The number of jobs to use for the computation. This works by computing each of the `n_init` runs in parallel. If -1 all CPUs are used. If 1 is given, no parallel computing code is used at all

4. How to Use K_Means

- The **K_Means** task should be connected with **Train_Model / Train_Clustering_Model** or **Predict_Model / Predict_Clustering_Model**.



1. Task Name: Mean_Shift

2. Task Overview:

Mean shift is a non-parametric feature-space analysis technique for locating the maxima of a density function.

3. Task Variables:

CLUSTER_ALL: boolean, default True

The number of clusters to form as well as the number of centroids to generate.

N_JOBS: int, default:1

If true, then all points are clustered, even those orphans that are not within any kernel. Orphans are assigned to the nearest kernel. If false, then orphans are given cluster label -1.

4. How to Use Mean_Shift

- The **Mean_Shift** task should be connected with **Train_Model / Train_Clustering_Model** or **Predict_Model / Predict_Clustering_Model**.



1. Task Name: Train_Clustering_Model

2. Task Overview:

Train a clustering model.

3. How to Use Train_Clustering_Model

- The **Train_Clustering_Model** task should be used after a clustering algorithm such as **Bayesian_Ridge_Regression**, **Linear_Regression** or **Support_Vector_Regression**.



1. Task Name: Train_Model

2. Task Overview:

Train a model using a classification or regression algorithm..

3. How to Use Train_Clustering_Model

- The **Train_Model** task should be used after a classification or regression algorithm, such as **Support_Vector_Machines**, **Gaussian_Naive_Bayes** and **Linear_Regression**.

1. Task Name: Predict_Clustering_Model

2. Task Overview:

Generate predictions using a trained model.

3. How to Use Train_Clustering_Model

- The **Predict_Clustering_Model** should be used after the **Train_Clustering_Model** task.

1. Task Name: Predict_Model

2. Task Overview:

Generate predictions using a trained model.

3. How to Use Train_Clustering_Model

- The **Predict_Model** should be used after the **Train_Model** task.

How to build a machine learning workflow using MLOS tasks

Log Analysis Use Case



ACTIVEeon
SCALE BEYOND LIMITS

Build your own Log Analysis Workflow



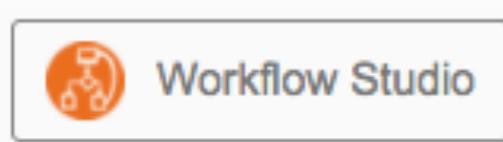
ACTIVEeon
SCALE BEYOND LIMITS

- 1. Create a new Workflow**
- 2. Add Machine Learning Bucket as Extra Catalog Menu**
- 3. Compose your own Log Analysis Workflow using ML bucket**
- 4. Execute your Log Analysis Workflow**
- 5. Preview and Download the Output Results**

Build your own Log Analysis Workflow



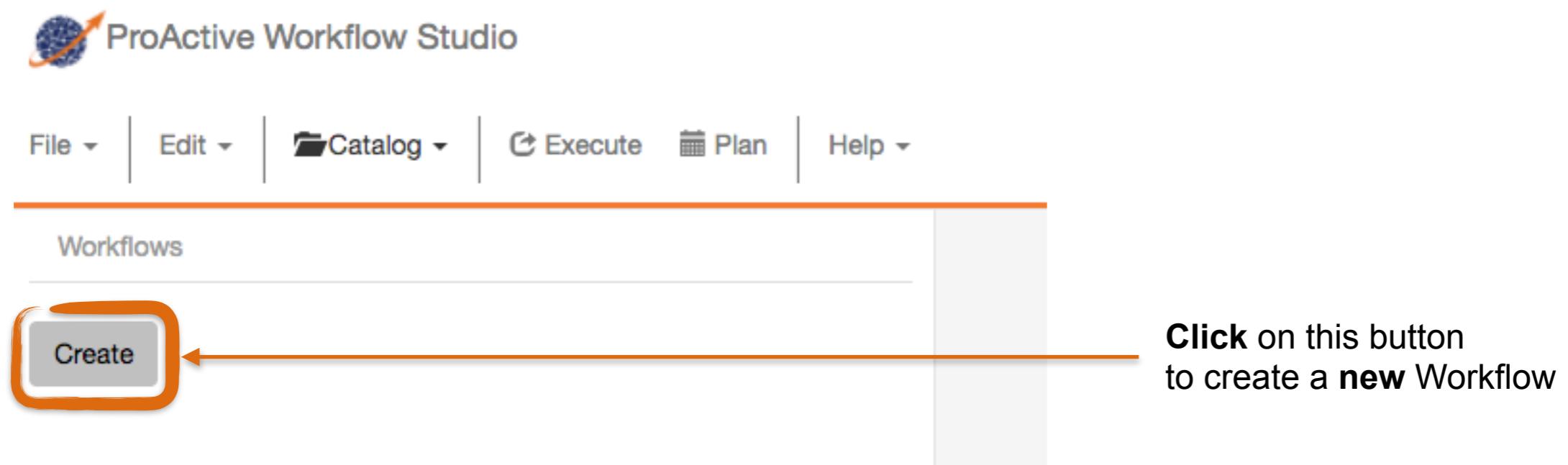
ACTIVEeon
SCALE BEYOND LIMITS



1. Create a new Workflow

Follow these steps to create the workflow:

- Open the [ProActive Studio](#) and Login using the user and password you received by e-mail when you first signed up.
- Create a new Workflow

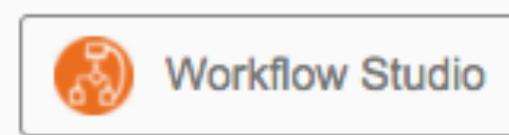


- Fill in the name of the workflow in the left panel, call it Basic_Log_Analysis_Workflow

Build your own Log Analysis Workflow



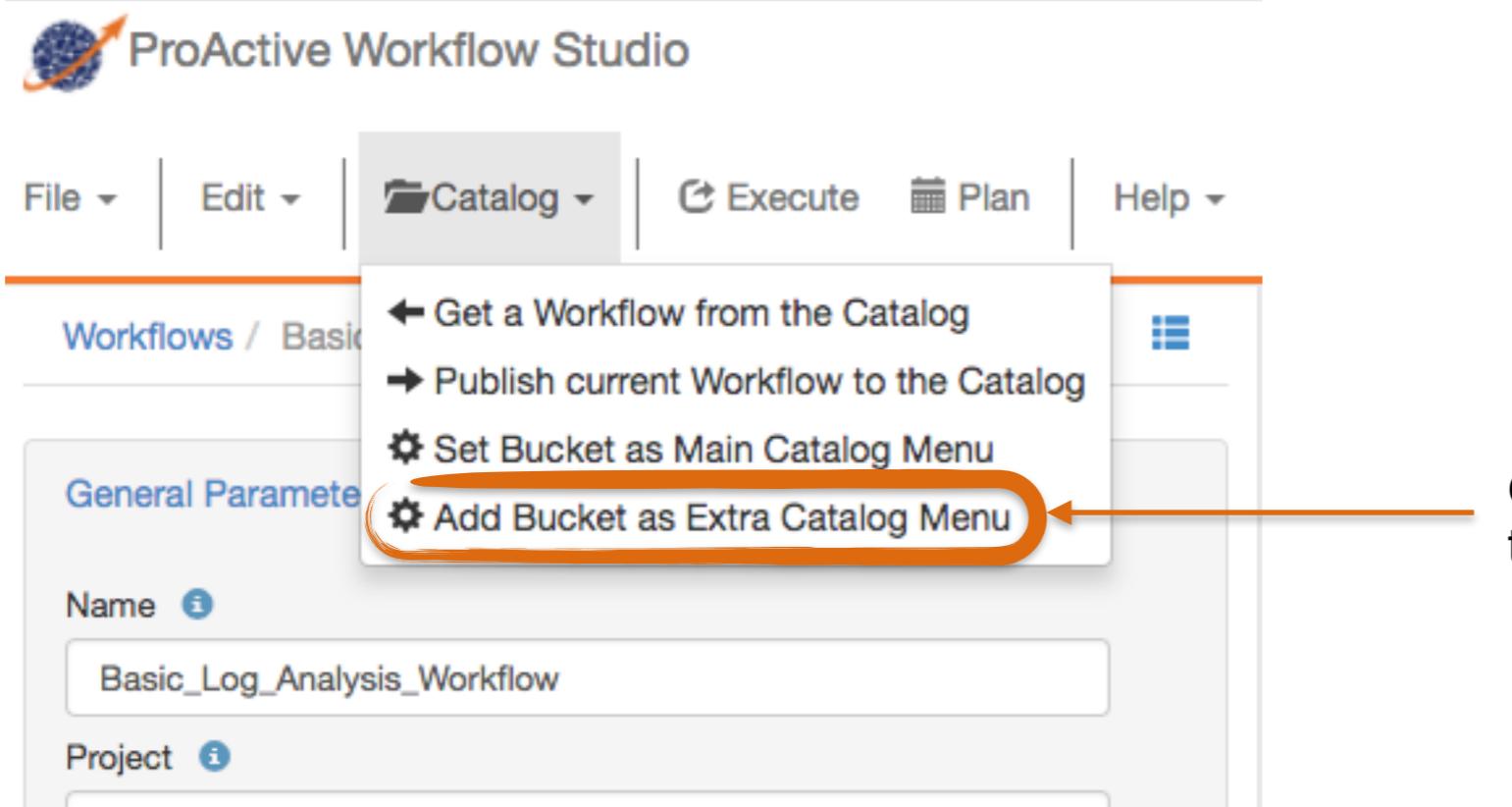
ACTIVEeon
SCALE BEYOND LIMITS



2. Add Machine Learning Bucket as Extra Catalog Menu

Follow these steps to add the machine learning bucket as extra menu:

- Click on Catalog>Add Bucket as Extra Catalog Menu



Click on this button
to add the **ML** bucket

- Click on the machine-learning bucket then on the Select button.
Now the machine-learning bucket should appear as follows:



Build your own Log Analysis Workflow



ACTIVEeon
SCALE BEYOND LIMITS

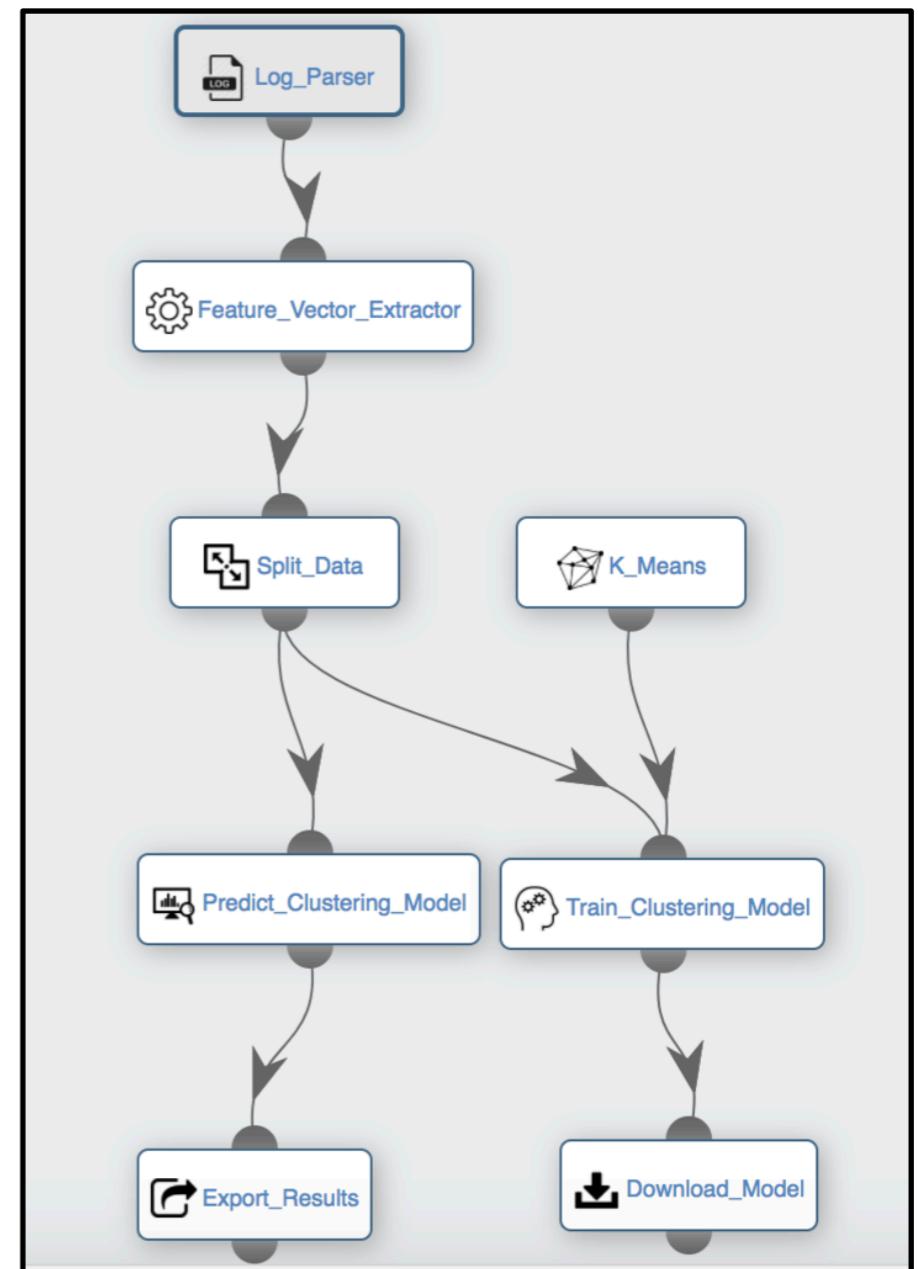


Workflow Studio

3. Compose your own Log Analysis Workflow using ML bucket

Follow these steps to compose your Log Analysis Workflow:

- **Open** the machine learning bucket
- **Drag and drop** the following tasks:
Log_Parser
Feature_Vector_Extractor
Split_Data
K_means
Train_Clustering_Model
Download_Model
Predict_Clustering_Model
Export_Results
- **Link** the different tasks in order to have a workflow similar to the workflow displayed in this image.
- [optional] **Click on** the tasks that you want to personalize by:
modifying its parameters on the General Parameters Section
modifying its code on the Task Implementation Section



Build your own Log Analysis Workflow



ACTIVEeon
SCALE BEYOND LIMITS



Workflow Studio

4. Execute your Log Analysis Workflow

Once your workflow is ready and you are logged in, click on **Execute** to submit your workflow as a job to the Scheduler.

Build your own Log Analysis Workflow



ACTIVEeon
SCALE BEYOND LIMITS



Scheduling & Orchestration

5. Preview and Download the Output Results

Once your workflow is submitted, go to the **scheduler portal** and explore the results of each task.

The screenshot shows the scheduler portal interface. On the left, a table lists workflow tasks. Task 0, 'Log_Parser', is highlighted with a blue selection bar. On the right, a panel titled 'Task Result' displays details for Task 0. At the top of the right panel, there is a 'Preview' button, which is circled with a red outline and labeled '2'. Below the preview button, there are two options: 'Open in browser' and 'Save as file', both of which are also circled with red outlines and labeled '3' and '4' respectively.

ID	Status	Name	Tag	Duration	Nodes	Executions	Node Failures	Visu
0	Finished	Log_Parser		43s 920ms	1	0 / 2	0 / 2	
1	Finished	Feature_vector_Extr...		12s 355ms	1	0 / 2	0 / 2	
2	Finished	Split_Data		3s 105ms	1	0 / 2	0 / 2	
3	Finished	Predict_Clustering_...		2s 706ms	1	0 / 2	0 / 2	
4	Finished	K_Means		508ms	1	0 / 2	0 / 2	
5	Finished	Train_Clustering_Mo...		3s 512ms	1	0 / 2	0 / 2	
6	Finished	Download_Model		569ms	1	0 / 2	0 / 2	
7	Finished	Export_Results		1s 832ms	1	0 / 2	0 / 2	

Remote Visualization
Remote visualization is disabled. Please toggle streaming in output view for a job in order to enable visualization.

Task Result
Task Log_Parser (id: 0) from job Basic_Log_Analysis_Workflow (id: 11)

Open in browser
Save as file

- 1 Click on the task that you want to preview or download its results. For example, click on the Log Parser Task
- 2 Click on Preview
- 3 Click on Open in browser to visualize the results in your browser
- 4 Click on save as file to download the results in your laptop

Example of Workflows using customized machine learning tasks



Introduction



Machine Learning and Computer Vision are two core branches of computer science that allow us to create powerful AI applications. The ProActive Studio provides a fast, easy and practice way to execute different workflows using machine learning and computer vision algorithms. We present useful computer vision applications using deep learning based algorithms by convolutional neural networks for image recognition, object detection, anomaly detection, and image segmentation. In addition, two workflows to deploy and delete machine learning node source are shown. The ProActive Studio uses open source libraries, such as TensorFlow, Keras, Caffe, OpenCV, PyTorch and scikit-learn as a backend for AI workflows.

How does it work



ACTIVEeon
SCALE BEYOND LIMITS

1. Executing the functional Workflows

Image Recognition

TensorFlow_Demo_Prediction.xml returns the flower specie of a given input image.

TensorFlow_Demo_Parallel_Prediction.xml predicts three different images of flower species in parallel.

TensorFlow_Demo_Image_Classification.xml classifies a input image using deep ConvNets (specifically, VGG16)* pre-trained on the ImageNet dataset.

TensorFlow_Demo_Training.xml trains a deep ConvNets (specifically, Inception)* to recognize flower species.

Object Detection

YOLO_Demo_Object_Detection.xml detects real-world objects in a image with the YOLO* library using a pre-trained model.

Anomaly Detection

Anomaly_Detection_Demo.xml checks if exists an anomaly in a certain scene using the YOLO library. However, we show a scene where only people are allowed on a pedestrian street. Anything detected other than people is considered as anomaly.

Image Segmentation

PyTorch_Demo_Obj_Seg_Prediction.xml returns the horse segments of a given input image.

* <https://www.tensorflow.org/>

* <http://www.pyimagesearch.com/2017/03/20/imagenet-vggnet-resnet-inception-xception-keras/>

How does it work



ACTIVEeon
SCALE BEYOND LIMITS

2. Training Workflows

TensorFlow_Demo_Training.xml trains a deep ConvNets (specifically, Inception)* to recognize flower species.

PyTorch_Demo_Obj_Seg_Training.xml trains an image segmentation algorithm using PyTorch* to segment horses. It segments horses in the input image using a deep neural network (DNN).

3. Node Source Deployment (ProActive, Infra, Admin aspects)

Node Source Deploy/Delete

MachineLearning_NodeSource_Deploy.xml deploys a machine learning node source.

MachineLearning_NodeSource_Delete.xml deletes a machine learning node source.

* <https://pjreddie.com/darknet/yolo/>

* http://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html

Contents



ACTIVEeon
SCALE BEYOND LIMITS

1. Executing the functional Workflows

Image Recognition

* **TensorFlow_Demo_Prediction.xml**

* TensorFlow_Demo_Prediction.xml

* TensorFlow_Demo_Image_Classification.xml

Object Detection

* YOLO_Demo_Object_Detection.xml

Anomaly Detection

* Anomaly_Detection_Demo.xml

Image Segmentation

* PyTorch_Demo_Obj_Seg_Prediction.xml

2. Training Workflows

* TensorFlow_Demo_Training.xml

* PyTorch_Demo_Obj_Seg_Training.xml

3. Node Source Deployment (ProActive, Infra, Admin aspects)

* MachineLearning_NodeSource_Deploy.xml

* MachineLearning_NodeSource_Delete.xml

Image Recognition



ACTIVEeon
SCALE BEYOND LIMITS



TensorFlow_Demo_Prediction.xml

Execute your workflow as a job

Set the variables used for the job

CONTAINER_NAME	<input type="text" value="ml"/>	name of the container
PREDICTION_IMAGE	<input type="text" value="2141413229_3f0425f972_n.jpg"/>	name of the input image
MODEL_PATH	<input type="text" value="/tmp/output_graph.pb"/>	pre-trained Tensorflow model
LABEL_PATH	<input type="text" value="/tmp/output_labels.txt"/>	labels from the pre-trained model
PREDICTION_PATH	<input type="text" value="flower_photos/roses/"/>	path of the prediction image
OUTPUT_PATH	<input type="text" value="/out/"/>	output path for the predicted image

Check **Execute**

Image Recognition



ACTIVEeon
SCALE BEYOND LIMITS



TensorFlow_Demo_Prediction.xml

Load a dataset image

Show input image

Predict flower species using a previous trained model

Show the result of the classification (image label and its classification probability)

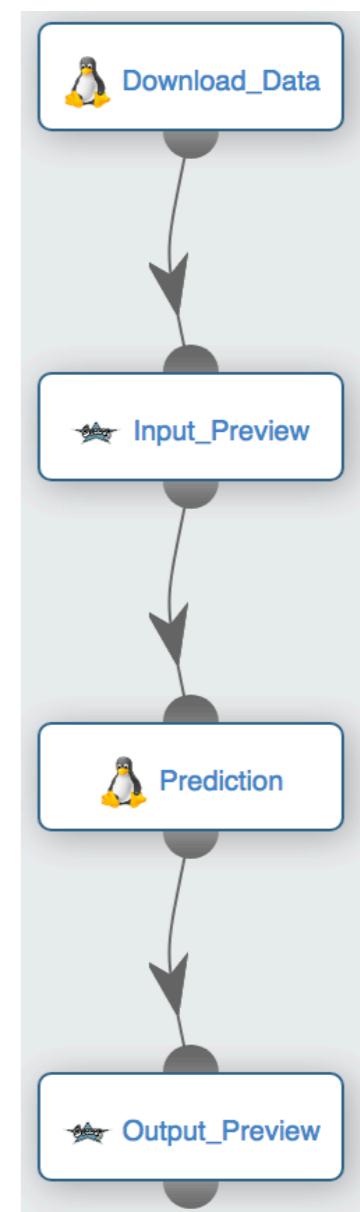


Image Recognition



ACTIVEeon
SCALE BEYOND LIMITS



TensorFlow_Demo_Prediction.xml

Execute your workflow as a job

Set the variables used for the job

CONTAINER_NAME	<input type="text" value="ml"/>	name of the container
TRAINING_SET_ADDRESS	<input type="text" value="http://download.tensorflow.org/example_images/flower_photo"/>	URL of the images to train (compressed file)
TRAINING_SET_NAME	<input type="text" value="flower_photos.tgz"/>	name of the compressed file
TRAINING_SET_PATH	<input type="text" value="flower_photos"/>	path to decompress

Check **Execute**

Image Recognition



ACTIVEeon
SCALE BEYOND LIMITS



TensorFlow_Demo_Prediction.xml

Input image



2141413229_3f0425f972_n.jpg

Prediction result



2141413229_3f0425f972_n.jpg

Contents



ACTIVEeon
SCALE BEYOND LIMITS

1. Executing the functional Workflows

Image Recognition

- * TensorFlow_Demo_Prediction.xml
- * **TensorFlow_Demo_Parallel_Prediction.xml**

- * TensorFlow_Demo_Image_Classification.xml

Object Detection

- * YOLO_Demo_Object_Detection.xml

Anomaly Detection

- * Anomaly_Detection_Demo.xml

Image Segmentation

- * PyTorch_Demo_Obj_Seg_Prediction.xml

2. Training Workflows

- * TensorFlow_Demo_Training.xml
- * PyTorch_Demo_Obj_Seg_Training.xml

3. Node Source Deployment (ProActive, Infra, Admin aspects)

- * MachineLearning_NodeSource_Deploy.xml
- * MachineLearning_NodeSource_Delete.xml

Image Recognition



ACTIVEeon
SCALE BEYOND LIMITS



TensorFlow_Demo_Parallel_Prediction.xml

Execute your workflow as a job

Set the variables used for the job

CONTAINER_NAME	ml	name of the container
MODEL_PATH	/tmp/output_graph.pb	pre-trained Tensorflow model
LABEL_PATH	/tmp/output_labels.txt	labels from the pre-trained model
PREDICTION_PATH	flower_photos/roses/	path of the input images
PREDICTION_IMAGE_1	410425647_4586667858.jpg	name of the first image for prediction
PREDICTION_IMAGE_2	537207677_f96a0507bb.jpg	name of the second image for prediction
PREDICTION_IMAGE_3	685724528_6cd5cbe203.jpg	name of the third image for prediction

Check **Execute**

Image Recognition



ACTIVEeon
SCALE BEYOND LIMITS



TensorFlow_Demo_Parallel_Prediction.xml

Load a dataset image

Show the three input images

Predict three different images of flower species

Show the result of the predictions for the three flower species

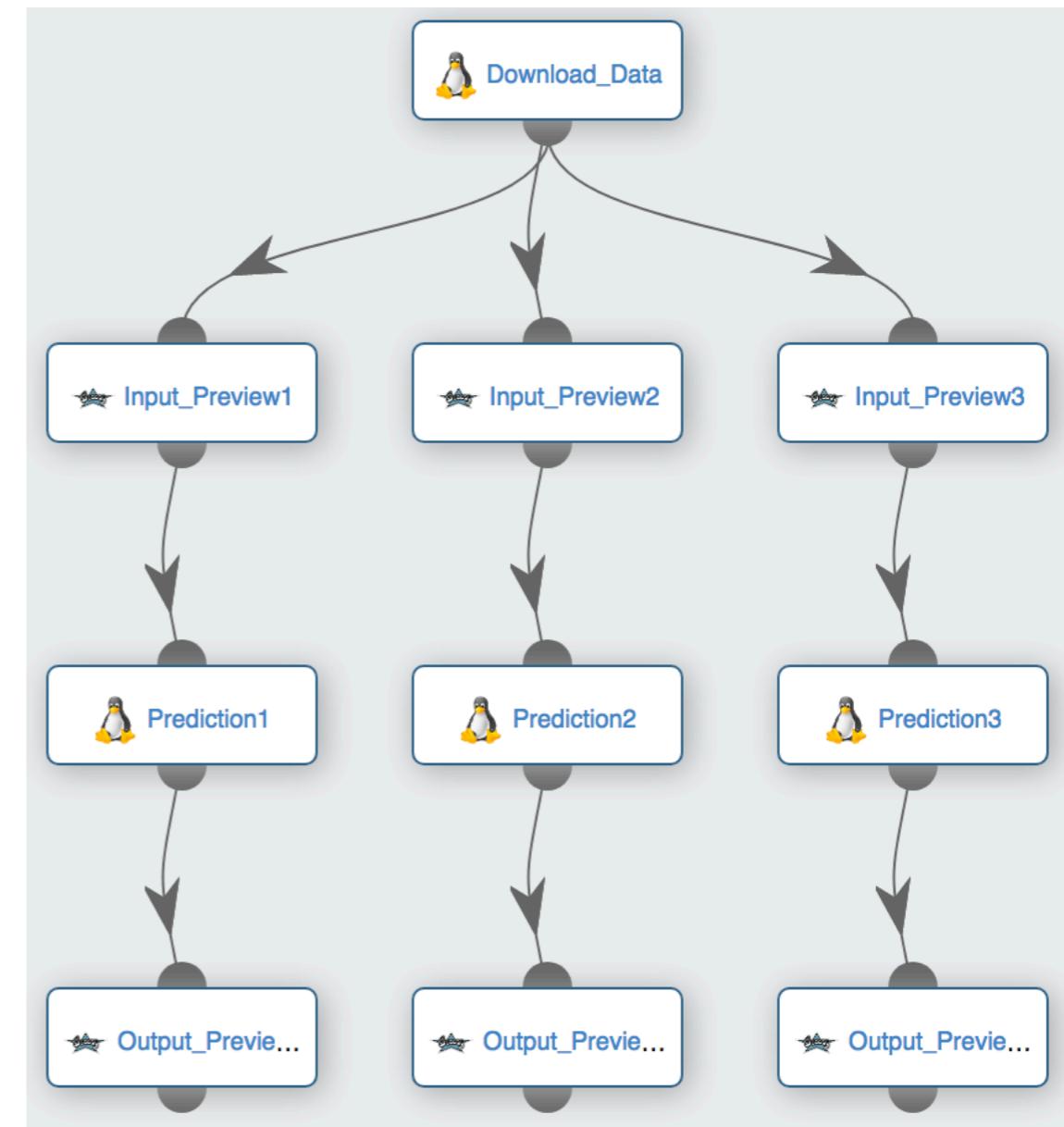


Image Recognition



ACTIVEeon
SCALE BEYOND LIMITS



TensorFlow_Demo_Parallel_Prediction.xml



Output label: roses

Prob: 96%

410425647_4586667858.jpg



Output label: roses

Prob: 99%

537207677_f96a0507bb.jpg



Output label: roses

Prob: 99%

685724528_6cd5cbe203.jpg

Contents



ACTIVEeon
SCALE BEYOND LIMITS

1. Executing the functional Workflows

Image Recognition

- * TensorFlow_Demo_Prediction.xml
- * TensorFlow_Demo_Parallel_Prediction.xml
- * TensorFlow_Demo_Image_Classification.xml**

Object Detection

- * YOLO_Demo_Object_Detection.xml

Anomaly Detection

- * Anomaly_Detection_Demo.xml

Image Segmentation

- * PyTorch_Demo_Obj_Seg_Prediction.xml

2. Training Workflows

- * TensorFlow_Demo_Training.xml
- * PyTorch_Demo_Obj_Seg_Training.xml

3. Node Source Deployment (ProActive, Infra, Admin aspects)

- * MachineLearning_NodeSource_Deploy.xml
- * MachineLearning_NodeSource_Delete.xml

Image Recognition



ACTIVEeon
SCALE BEYOND LIMITS



TensorFlow_Demo_Image_Classification.xml

Execute your workflow as a job

Set the variables used for the job

INPUT_IMAGE	images/dog.jpg
OUTPUT_IMAGE	output/dog.jpg
CONTAINER_NAME	ml

Check **Execute**

→ path of the input image. It is possible to choose different input images, such as ball.jpg, bear.jpg, beer.jpg, car.jpg, and cat.jpg

→ path of the output image

→ name of the container

Image Recognition



ACTIVEeon
SCALE BEYOND LIMITS



TensorFlow_Demo_Image_Classification.xml

Load the input image

Show the input image

Classify the input image using deep ConvNet

Show the result of the classification (image label and its classification probability)

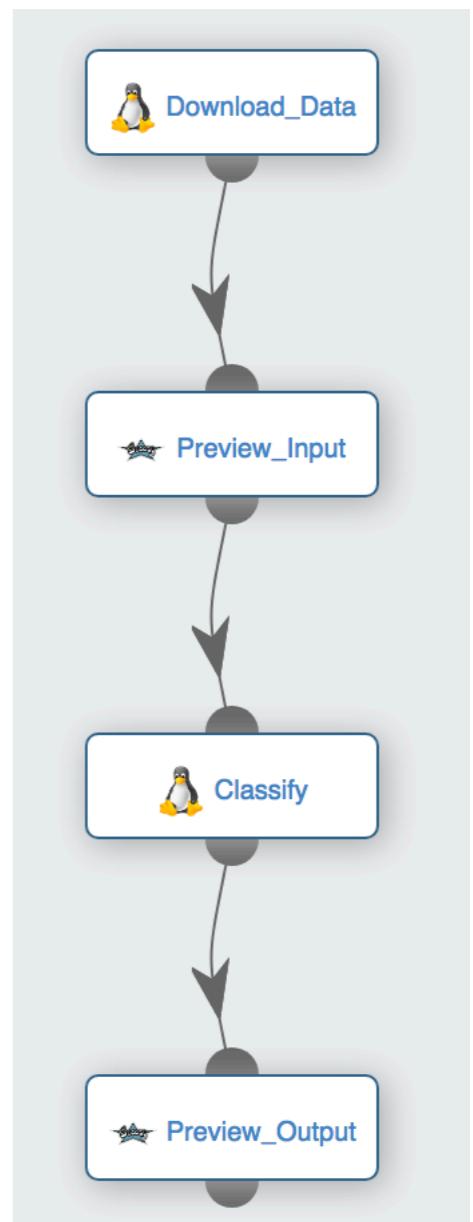


Image Recognition



ACTIVEeon
SCALE BEYOND LIMITS



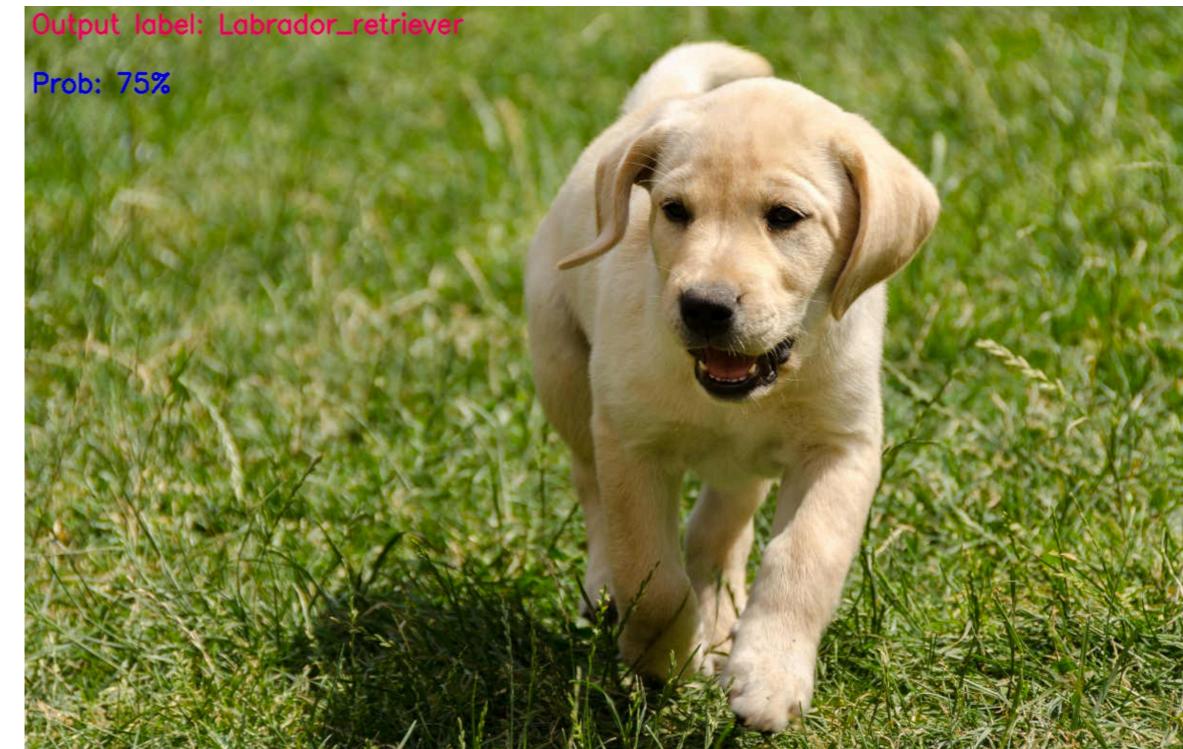
TensorFlow_Demo_Image_Classification.xml

Input image



dog.jpg

Classification result



dog.jpg

Contents



ACTIVEeon
SCALE BEYOND LIMITS

1. Executing the functional Workflows

Image Recognition

- * TensorFlow_Demo_Prediction.xml
- * TensorFlow_Demo_Parallel_Prediction.xml
- * TensorFlow_Demo_Image_Classification.xml

Object Detection

- * **YOLO_Demo_Object_Detection.xml**

Anomaly Detection

- * Anomaly_Detection_Demo.xml

Image Segmentation

- * PyTorch_Demo_Obj_Seg_Prediction.xml

2. Training Workflows

- * TensorFlow_Demo_Training.xml
- * PyTorch_Demo_Obj_Seg_Training.xml

3. Node Source Deployment (ProActive, Infra, Admin aspects)

- * MachineLearning_NodeSource_Deploy.xml
- * MachineLearning_NodeSource_Delete.xml

Object Detection



ACTIVEeon
SCALE BEYOND LIMITS



YOLO_Demo_Object_Detection.xml

Execute your workflow as a job

Set the variables used for the job

INPUT_IMAGE

OUTPUT_IMAGE

CONTAINER_NAME

→ path of the input image

→ path of the output image

→ name of the container

Object Detection



ACTIVEeon
SCALE BEYOND LIMITS



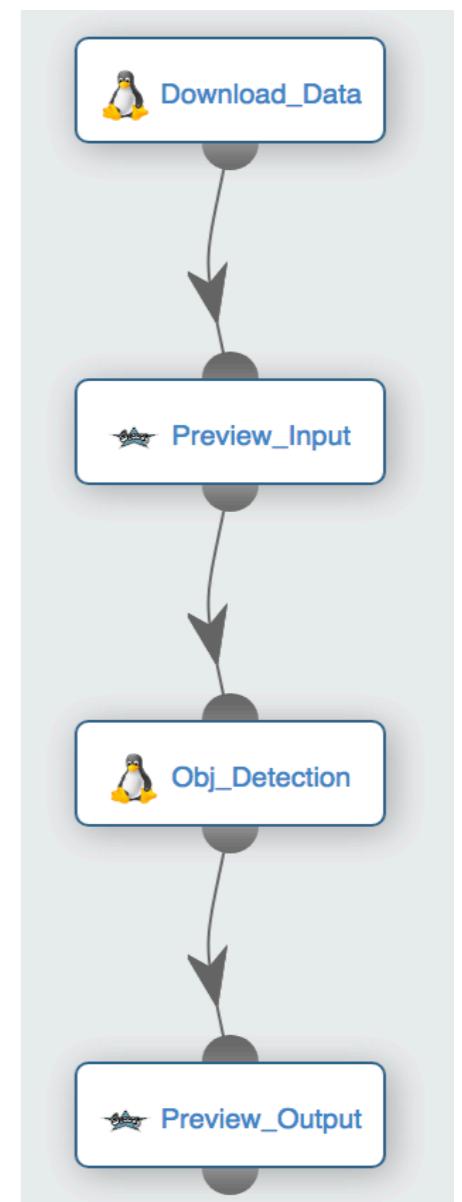
YOLO_Demo_Object_Detection.xml

Load the input image

Show the input image

Detect different objects in the input image using a deep ConvNet

Show the result of the detection (the objects found in the image)



Object Detection



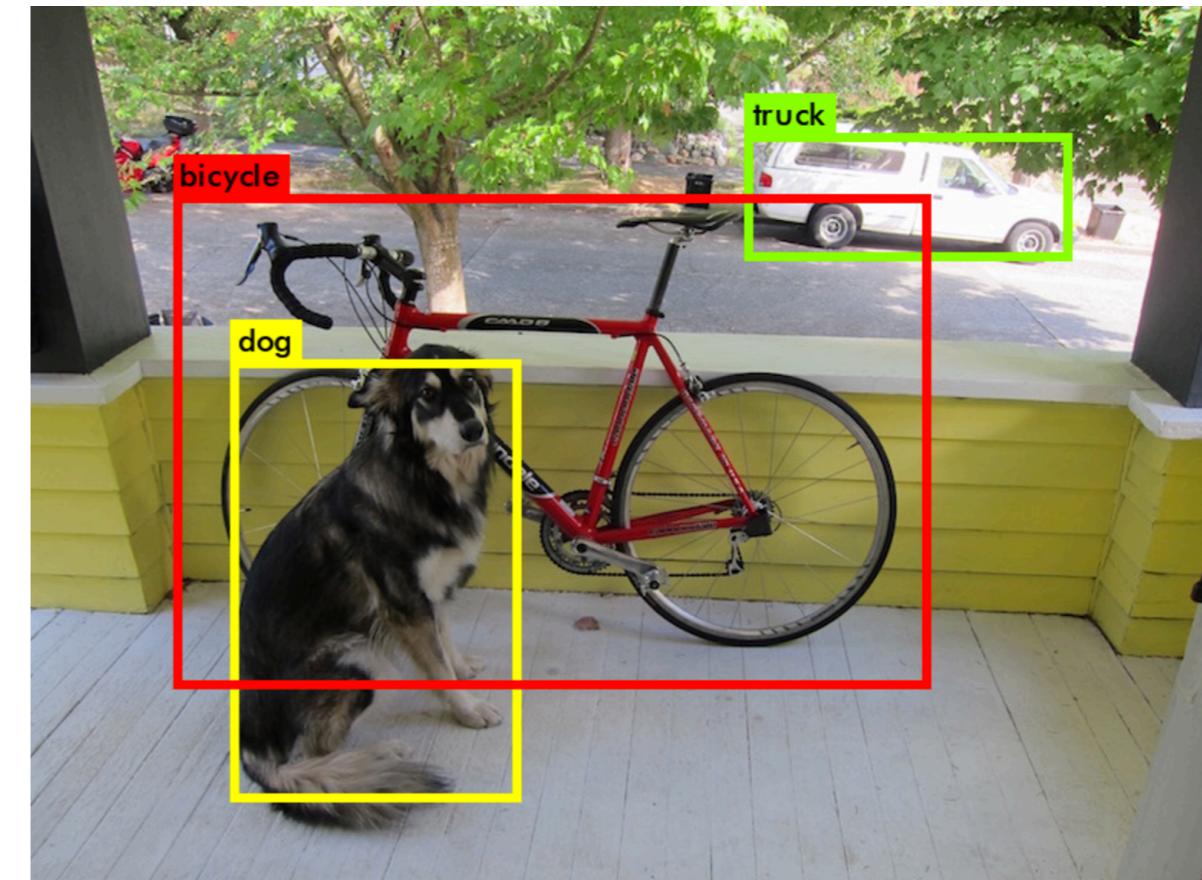
YOLO_Demo_Object_Detection.xml

Input image



dog.jpg

Detection results



dog.jpg

Contents



ACTIVEeon
SCALE BEYOND LIMITS

1. Executing the functional Workflows

Image Recognition

- * TensorFlow_Demo_Prediction.xml
- * TensorFlow_Demo_Parallel_Prediction.xml
- * TensorFlow_Demo_Image_Classification.xml

Object Detection

- * YOLO_Demo_Object_Detection.xml

Anomaly Detection

- * **Anomaly_Detection_Demo.xml**

Image Segmentation

- * PyTorch_Demo_Obj_Seg_Prediction.xml

2. Training Workflows

- * TensorFlow_Demo_Training.xml
- * PyTorch_Demo_Obj_Seg_Training.xml

3. Node Source Deployment (ProActive, Infra, Admin aspects)

- * MachineLearning_NodeSource_Deploy.xml
- * MachineLearning_NodeSource_Delete.xml

Anomaly Detection



ACTIVEeon
SCALE BEYOND LIMITS



Anomaly_Detection_Demo.xml

Execute your workflow as a job

Set the variables used for the job

INPUT_IMAGE_1	<input type="text" value="images/normal.jpg"/>	→ path of the first input image
INPUT_IMAGE_2	<input type="text" value="images/anomaly.jpg"/>	→ path of the second input image
OUTPUT_IMAGE_1	<input type="text" value="output/predictions1.jpg"/>	→ path of the output image 1
OUTPUT_IMAGE_2	<input type="text" value="output/predictions2.jpg"/>	→ path of the output image 2
CONTAINER_NAME	<input type="text" value="ml"/>	→ name of the container

Check **Execute**

Anomaly Detection



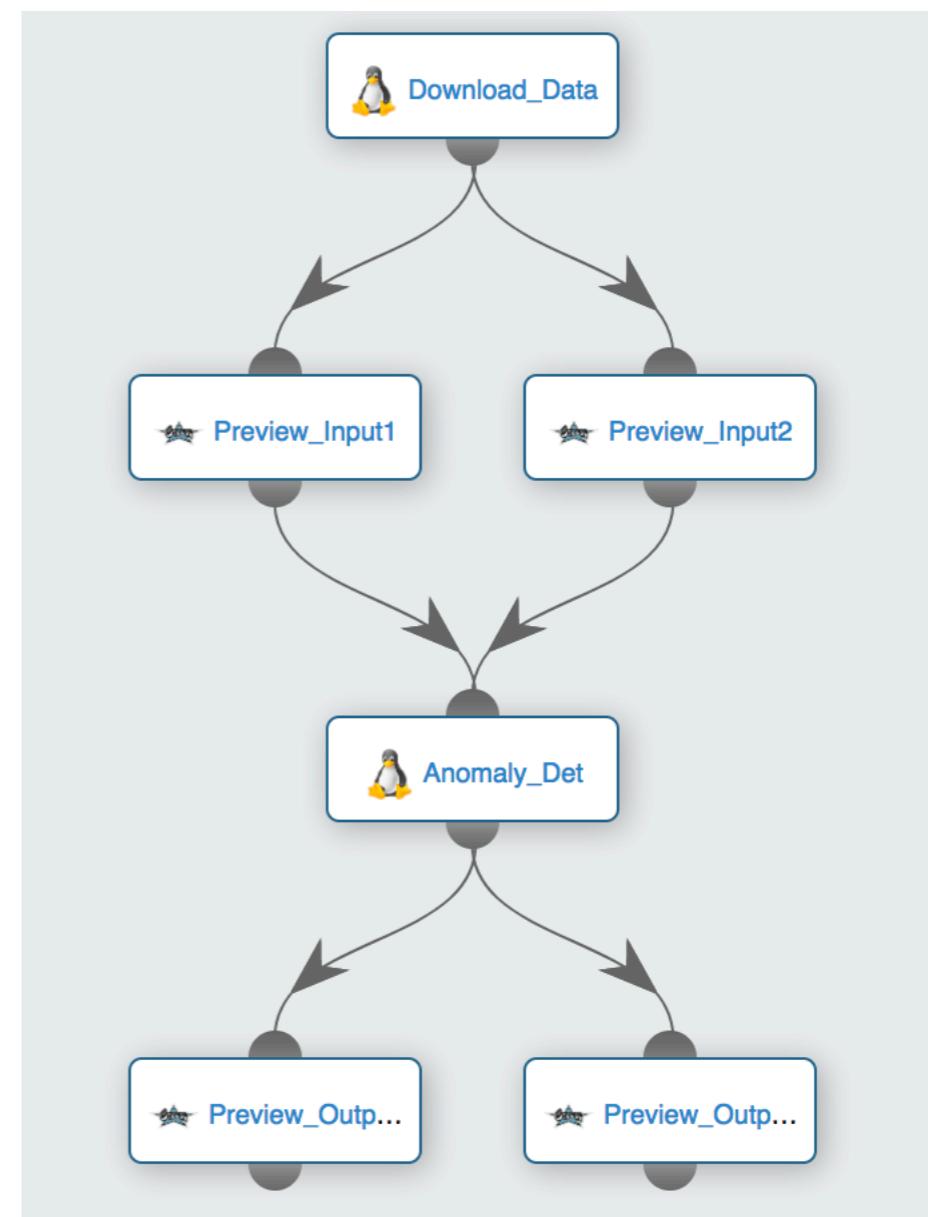
Anomaly_Detection_Demo.xml

Load the dataset

Show the input image1 and the input image 2

Detect different objects in the input image using a deep ConvNet

Show the result of the detection (image 1 and image 2)



Anomaly Detection



Anomaly_Detection_Demo.xml

Input image



normal.jpg

Output image



predictions.jpg

Anomaly Detection



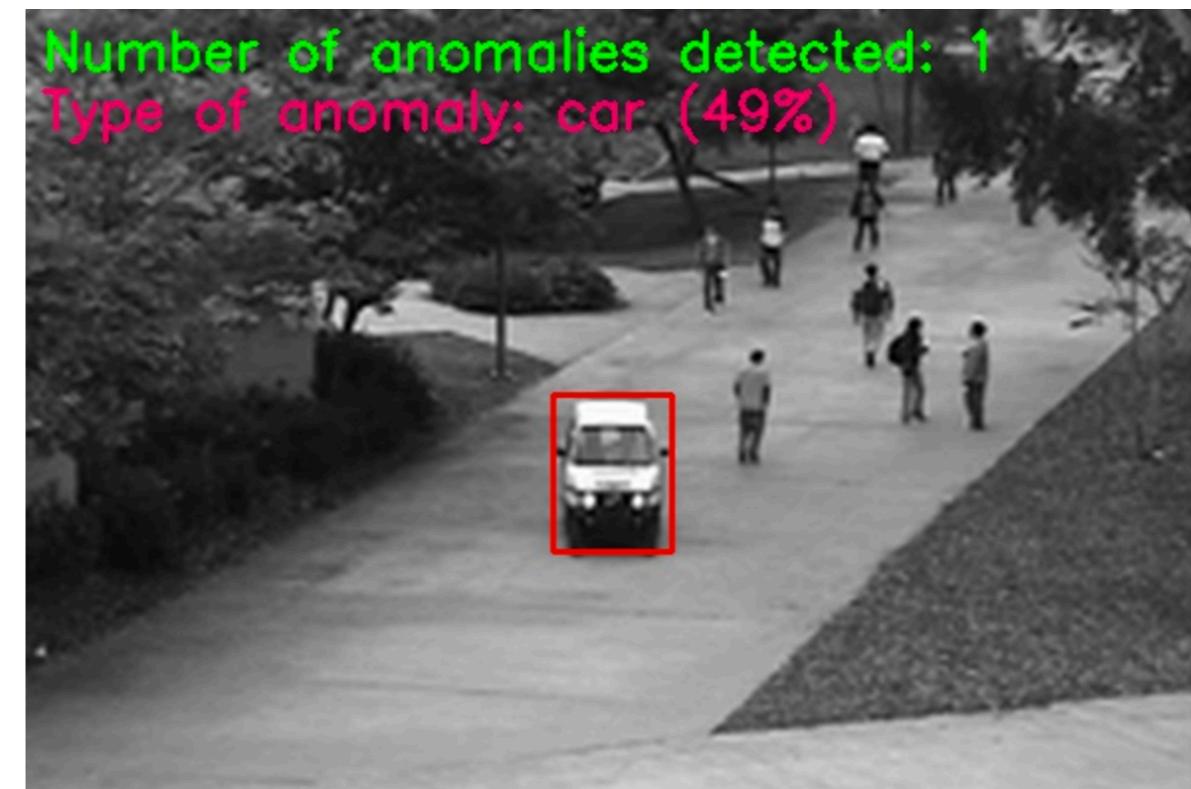
Anomaly_Detection_Demo.xml

Input image



anomaly.jpg

Output image



predictions.jpg

Contents



ACTIVEeon
SCALE BEYOND LIMITS

1. Executing the functional Workflows

Image Recognition

- * TensorFlow_Demo_Prediction.xml
- * TensorFlow_Demo_Parallel_Prediction.xml
- * TensorFlow_Demo_Image_Classification.xml

Object Detection

- * YOLO_Demo_Object_Detection.xml

Anomaly Detection

- * Anomaly_Detection_Demo.xml

Image Segmentation

- * **PyTorch_Demo_Obj_Seg_Prediction.xml**

2. Training Workflows

- * TensorFlow_Demo_Training.xml
- * PyTorch_Demo_Obj_Seg_Training.xml

3. Node Source Deployment (ProActive, Infra, Admin aspects)

- * MachineLearning_NodeSource_Deploy.xml
- * MachineLearning_NodeSource_Delete.xml

Image Segmentation



ACTIVEeon
SCALE BEYOND LIMITS



PyTorch_Demo_Obj_Seg_Prediction.xml

Execute your workflow as a job

Set the variables used for the job

INPUT_IMAGE	horse.jpg	path of the input image
OUTPUT_IMAGE	output.jpg	path of the output image
CONTAINER_NAME	ml	name of the container

Check **Execute**

Image Segmentation



ACTIVEeon
SCALE BEYOND LIMITS



PyTorch_Demo_Obj_Seg_Prediction.xml

Load the input image

Show the input image

Segment the input image using a deep ConvNet

Show the result of the segmentation

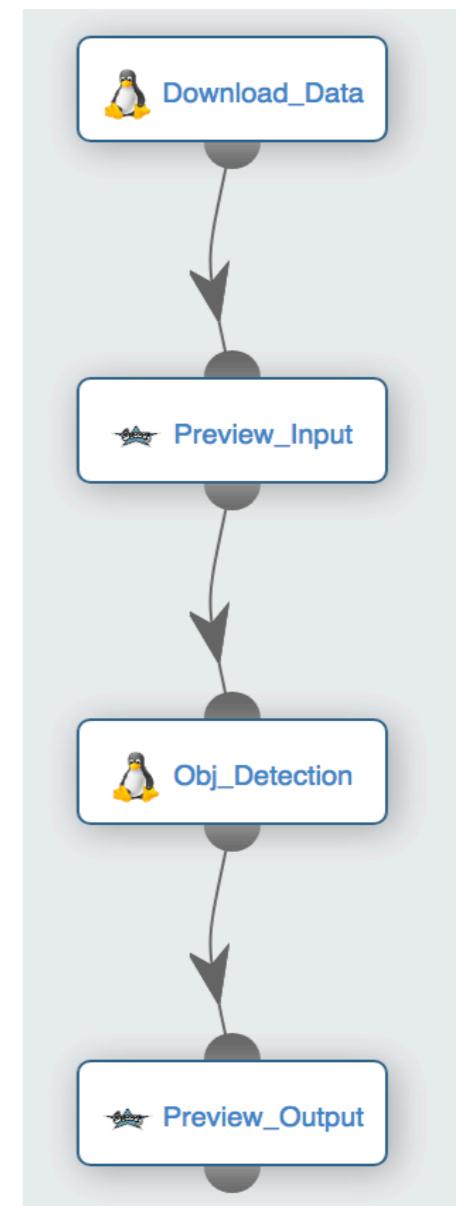


Image Segmentation



PyTorch_Demo_Obj_Seg_Prediction.xml

Input image



horse.jpg

Image segmentation results



horse.jpg

Contents



ACTIVEeon
SCALE BEYOND LIMITS

1. Executing the functional Workflows

Image Recognition

- * TensorFlow_Demo_Prediction.xml
- * TensorFlow_Demo_Parallel_Prediction.xml
- * TensorFlow_Demo_Image_Classification.xml

Object Detection

- * YOLO_Demo_Object_Detection.xml

Anomaly Detection

- * Anomaly_Detection_Demo.xml

Image Segmentation

- * PyTorch_Demo_Obj_Seg_Prediction.xml

2. Training Workflows

- * **TensorFlow_Demo_Training.xml**
- * PyTorch_Demo_Obj_Seg_Training.xml

3. Node Source Deployment (ProActive, Infra, Admin aspects)

- * MachineLearning_NodeSource_Deploy.xml
- * MachineLearning_NodeSource_Delete.xml

Image Recognition



ACTIVEeon
SCALE BEYOND LIMITS



TensorFlow_Demo_Training.xml

Load a dataset image

Train a deep ConvNet to recognize flower species

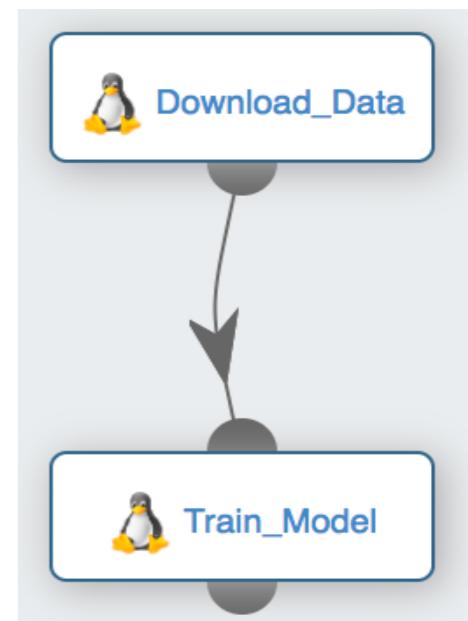


Image Recognition



ACTIVEeon
SCALE BEYOND LIMITS



TensorFlow_Demo_Training.xml

Execute your workflow as a job

Set the variables used for the job

CONTAINER_NAME	ml	name of the container
TRAINING_SET_ADDRESS	http://download.tensorflow.org/example_images/flower_photo.tgz	URL of the images to train (compressed file)
TRAINING_SET_NAME	flower_photos.tgz	name of the compressed file
TRAINING_SET_PATH	flower_photos	path to decompress

Check **Execute**

Image Recognition



ACTIVEeon
SCALE BEYOND LIMITS



TensorFlow_Demo_Training.xml

← → ⌂ 安全 <https://tryqa.activeeon.com/scheduler/> ☆ ☰ :

ProActive Scheduling & Orchestration

Portal Admin Help Submit job Calendar Workflow Studio Resource Manager Cloud Automation Logout song

Executions list

Id	State	Issues	User	Progress	Priority	Duration	Name
110	Finished		song	2 / 2	Normal	52m 26s 537ms	TensorFlow_Demo_Training

<< First < Previous Total number of jobs: 2 Next > Last >>

Details

Tasks Visualization Users Sessions Statistics Usage Job Info Task Info Output Server Logs Preview

Streaming Selected job

```
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:51:54.594754: Step 3900: Cross entropy = 0.158662
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:51:54.879850: Step 3900: Validation accuracy = 84.0% (N=100)
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:51:57.806942: Step 3910: Train accuracy = 97.0%
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:51:57.807042: Step 3910: Cross entropy = 0.115156
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:51:58.090897: Step 3910: Validation accuracy = 89.0% (N=100)
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:01.005608: Step 3920: Train accuracy = 99.0%
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:01.005707: Step 3920: Cross entropy = 0.105359
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:01.288788: Step 3920: Validation accuracy = 90.0% (N=100)
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:04.218535: Step 3930: Train accuracy = 98.0%
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:04.218634: Step 3930: Cross entropy = 0.117230
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:04.500555: Step 3930: Validation accuracy = 92.0% (N=100)
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:07.428914: Step 3940: Train accuracy = 96.0%
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:07.429019: Step 3940: Cross entropy = 0.141235
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:07.711705: Step 3940: Validation accuracy = 93.0% (N=100)
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:10.626573: Step 3950: Train accuracy = 99.0%
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:10.626674: Step 3950: Cross entropy = 0.107701
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:10.909175: Step 3950: Validation accuracy = 85.0% (N=100)
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:13.835726: Step 3960: Train accuracy = 95.0%
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:13.835822: Step 3960: Cross entropy = 0.228663
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:14.118007: Step 3960: Validation accuracy = 90.0% (N=100)
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:17.037750: Step 3970: Train accuracy = 96.0%
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:17.037851: Step 3970: Cross entropy = 0.211066
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:17.316738: Step 3970: Validation accuracy = 92.0% (N=100)
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:20.251709: Step 3980: Train accuracy = 98.0%
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:20.251808: Step 3980: Cross entropy = 0.118425
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:20.536702: Step 3980: Validation accuracy = 93.0% (N=100)
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:23.445544: Step 3990: Train accuracy = 96.0%
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:23.445643: Step 3990: Cross entropy = 0.177713
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:23.71223: Step 3990: Validation accuracy = 92.0% (N=100)
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:26.132723: Step 3990: Train accuracy = 97.0%
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:26.132726: Step 3990: Cross entropy = 0.1522
[110t1@tryqa.activeeon.com:18:52:31] 2017-06-06 16:52:26.647881: Step 3999: Validation accuracy = 90.0% (N=100)
[110t1@tryqa.activeeon.com:18:52:31] Final test accuracy = 91.8% (N=353)
[110t1@tryqa.activeeon.com:18:52:31] Converted 2 variables to const ops.
```

50 mins to train a model

Training set size : 5000 images Accuracy: over 90%

Contents



ACTIVEeon
SCALE BEYOND LIMITS

1. Executing the functional Workflows

Image Recognition

- * TensorFlow_Demo_Prediction.xml
- * TensorFlow_Demo_Parallel_Prediction.xml
- * TensorFlow_Demo_Image_Classification.xml

Object Detection

- * YOLO_Demo_Object_Detection.xml

Anomaly Detection

- * Anomaly_Detection_Demo.xml

Image Segmentation

- * PyTorch_Demo_Obj_Seg_Prediction.xml

2. Training Workflows

- * TensorFlow_Demo_Training.xml
- * **PyTorch_Demo_Obj_Seg_Training.xml**

3. Node Source Deployment (ProActive, Infra, Admin aspects)

- * MachineLearning_NodeSource_Deploy.xml
- * MachineLearning_NodeSource_Delete.xml

Image Segmentation



ACTIVEeon
SCALE BEYOND LIMITS



PyTorch_Demo_Obj_Seg_Training.xml

Execute your workflow as a job

Set the variables used for the job

CONTAINER_NAME → name of the container

NUM_EPOCHS → number of epochs to train

Image Segmentation



ACTIVEeon
SCALE BEYOND LIMITS



PyTorch_Demo_Obj_Seg_Training.xml

Train an image segmentation algorithm with deep neural network to recognise horses.

Load a dataset image

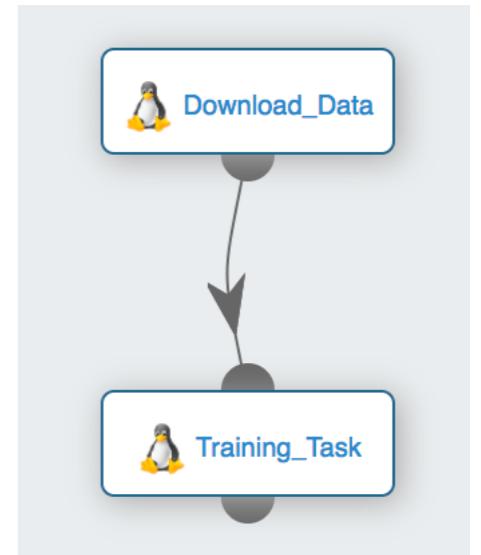


Image Segmentation



ACTIVEeon
SCALE BEYOND LIMITS



PyTorch_Demo_Obj_Seg_Training.xml

10.220.252.9:8080/scheduler/

ProActive Workflow Studio ProActive Scheduler

ProActive Scheduling & Orchestration

Portal Admin Help Submit job Calendar Workflow Studio Resource Manager Cloud Automation Logout admin

<< First < Previous Total number of jobs: 72 Next > > Last >>

Executions list My jobs Pending Running Finished Jobs-centric

Tasks Visualization Users Sessions Job Info Task Info Output Server Logs Preview Streaming Selected job

Download... Training_T...

5 mins to train a model

Training set size: 50 images

```
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse088.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse089.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse089.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse090.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse090.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse091.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse091.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse092.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse092.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse093.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse093.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse094.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse094.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse095.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse095.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse096.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse096.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse097.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse097.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse098.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse098.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse099.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse099.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse100.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse100.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse101.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse101.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse102.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse102.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse103.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse103.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse104.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse104.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse105.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse105.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse106.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse106.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse107.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse107.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse108.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse108.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse109.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse109.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse110.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse110.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse111.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse111.jpg
[355t1@10.220.252.9:15:54:14] inflating: weizmann_horse/JPEGImages_full/horse112.jpg
[355t1@10.220.252.9:15:54:14] inflating: __MACOSX/weizmann_horse/JPEGImages_full/_horse112.ior
```

Contents



ACTIVEeon
SCALE BEYOND LIMITS

1. Executing the functional Workflows

Image Recognition

- * TensorFlow_Demo_Prediction.xml
- * TensorFlow_Demo_Parallel_Prediction.xml
- * TensorFlow_Demo_Image_Classification.xml

Object Detection

- * YOLO_Demo_Object_Detection.xml

Anomaly Detection

- * Anomaly_Detection_Demo.xml

Image Segmentation

- * PyTorch_Demo_Obj_Seg_Prediction.xml

2. Training Workflows

- * TensorFlow_Demo_Training.xml
- * PyTorch_Demo_Obj_Seg_Training.xml

3. Node Source Deployment (ProActive, Infra, Admin aspects)

- * **MachineLearning_NodeSource_Deploy.xml**
- * MachineLearning_NodeSource_Delete.xml



MachineLearning_NodeSource_Deploy.xml

Execute your workflow as a job

Set the variables used for the job

username	<input type="text" value="USERNAME"/>	→ user account name
password	<input type="text" value="PASSWORD"/>	→ user account password
credentials	<input type="text" value="CREDENTIAL_GENERATED_BY_RESOURCE_MANAGER_POI"/>	→ user credentials
rmAddress	<input type="text" value="https://try.activeeon.com/rest/rm/"/>	→ URL address of the resource manager
NodeSourceName	<input type="text" value="MachineLearningNodes"/>	→ name of the node source
nodesNumber	<input type="text" value="4"/>	→ number of nodes
CONTAINER_NAME	<input type="text" value="ml"/>	→ name of the container

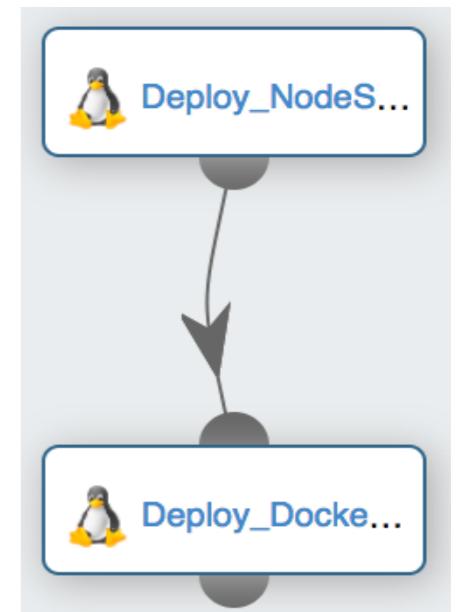
Check **Execute**



MachineLearning_NodeSource_Deploy.xml

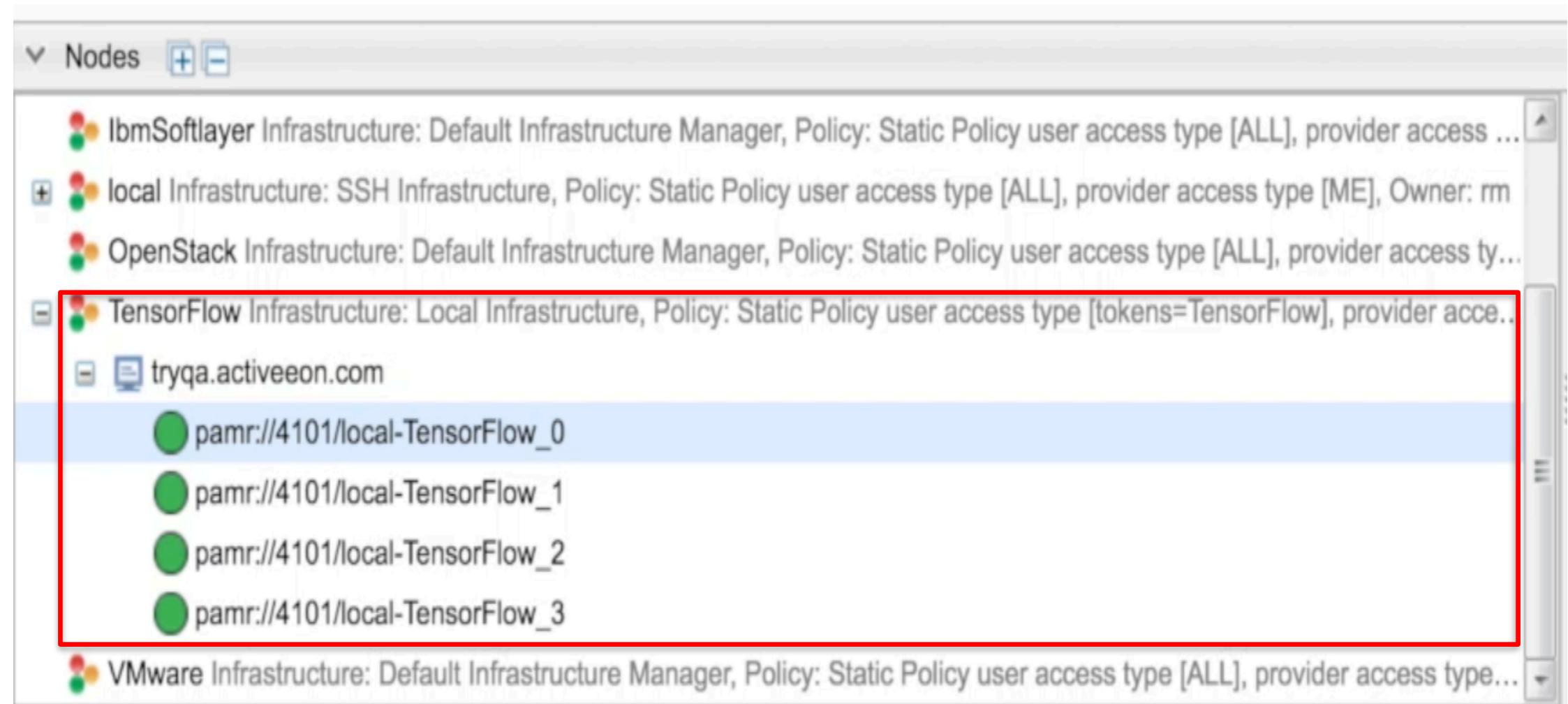
Deploy node source

Deploy docker container





MachineLearning_NodeSource_Deploy.xml



Nodes

- IbmSoftlayer Infrastructure: Default Infrastructure Manager, Policy: Static Policy user access type [ALL], provider access ...
- local Infrastructure: SSH Infrastructure, Policy: Static Policy user access type [ALL], provider access type [ME], Owner: rm
- OpenStack Infrastructure: Default Infrastructure Manager, Policy: Static Policy user access type [ALL], provider access ty...
- TensorFlow Infrastructure: Local Infrastructure, Policy: Static Policy user access type [tokens=TensorFlow], provider acce..**
 - pamr://4101/local-TensorFlow_0
 - pamr://4101/local-TensorFlow_1
 - pamr://4101/local-TensorFlow_2
 - pamr://4101/local-TensorFlow_3
- VMware Infrastructure: Default Infrastructure Manager, Policy: Static Policy user access type [ALL], provider access type...

Contents



ACTIVEeon
SCALE BEYOND LIMITS

1. Executing the functional Workflows

Image Recognition

- * TensorFlow_Demo_Prediction.xml
- * TensorFlow_Demo_Parallel_Prediction.xml
- * TensorFlow_Demo_Image_Classification.xml

Object Detection

- * YOLO_Demo_Object_Detection.xml

Anomaly Detection

- * Anomaly_Detection_Demo.xml

Image Segmentation

- * PyTorch_Demo_Obj_Seg_Prediction.xml

2. Training Workflows

- * TensorFlow_Demo_Training.xml
- * PyTorch_Demo_Obj_Seg_Training.xml

3. Node Source Deployment (ProActive, Infra, Admin aspects)

- * MachineLearning_NodeSource_Deploy.xml
- * MachineLearning_NodeSource_Delete.xml**



MachineLearning_NodeSource_Delete.xml

Execute your workflow as a job

Set the variables used for the job

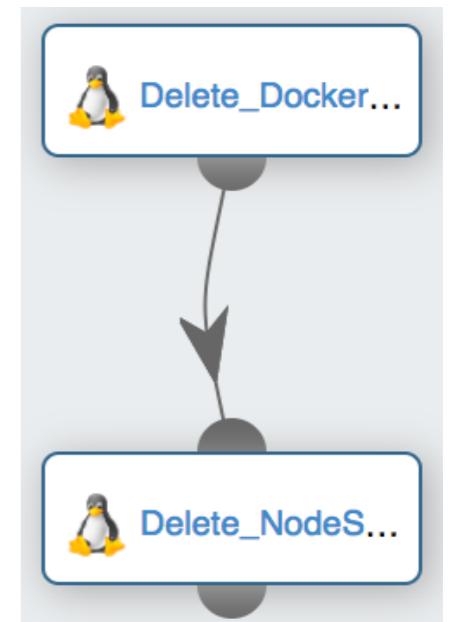
username	<input type="text" value="USERNAME"/>	→ user account name
password	<input type="text" value="PASSWORD"/>	→ user account password
credentials	<input type="text" value="CREDENTIALS_GENERATED_BY_RESOURCE_MANAGER_PC"/>	→ user credentials generated in the resource manager portal
rmAddress	<input type="text" value="https://try.activeeon.com/rest/rm/"/>	→ URL address of the resource manager
NodeSourceName	<input type="text" value="MachineLearningNodes"/>	→ name of the node source
nodesNumber	<input type="text" value="4"/>	→ number of nodes
CONTAINER_NAME	<input type="text" value="ml"/>	→ name of the container



MachineLearning_NodeSource_Delete.xml

Delete docker container

Delete node source





MachineLearning_NodeSource_Delete.xml

Nodes + -

- IbmSoftlayer Infrastructure: Default Infrastructure Manager, Policy: Static Policy user access type [ALL], provider access ...
- local Infrastructure: SSH Infrastructure, Policy: Static Policy user access type [ALL], provider access type [ME], Owner: rm
- OpenStack Infrastructure: Default Infrastructure Manager, Policy: Static Policy user access type [ALL], provider access ty...
- TensorFlow Infrastructure: Local Infrastructure, Policy: Static Policy user access type [tokens=TensorFlow], provider acce...
- tryqa.activeeon.com
 - pamr://4101/local-TensorFlow_0
 - pamr://4101/local-TensorFlow_1
 - pamr://4101/local-TensorFlow_2
 - pamr://4101/local-TensorFlow_3
- VMware Infrastructure: Default Infrastructure Manager, Policy: Static Policy user access type [ALL], provider access type...

Nodes + -

- AmazonAWS Infrastructure: Default Infrastructure Manager, Policy: Static Policy user access type [ALL], provider access...
- AmazonAWS-Spot Infrastructure: Default Infrastructure Manager, Policy: Static Policy user access type [ALL], provider a...
- Azure Infrastructure: Default Infrastructure Manager, Policy: Static Policy user access type [ALL], provider access type [A...
- GoogleCloudCompute Infrastructure: Default Infrastructure Manager, Policy: Static Policy user access type [ALL], provid...
- IbmSoftlayer Infrastructure: Default Infrastructure Manager, Policy: Static Policy user access type [ALL], provider access ...
- local Infrastructure: SSH Infrastructure, Policy: Static Policy user access type [ALL], provider access type [ME], Owner: rm
- OpenStack Infrastructure: Default Infrastructure Manager, Policy: Static Policy user access type [ALL], provider access ty...
- VMware Infrastructure: Default Infrastructure Manager, Policy: Static Policy user access type [ALL], provider access type...