

IBM Maximo Visual Inspection

Version 1.3.0

IBM Maximo Visual Inspection Guide



Note

Before using this information and the product it supports, read the information in [“Notices” on page 195.](#)

This edition applies to IBM® Maximo® Visual Inspection 1.3.0 and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright International Business Machines Corporation 2018, 2020.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

About this document.....	vii
Highlighting.....	vii
ISO 9000.....	vii
Chapter 1. Overview.....	1
Use cases.....	2
What's new.....	6
IBM Maximo Visual Inspection REST APIs.....	6
IBM Maximo Visual Inspection editions.....	7
Trial mode.....	7
Model functionality.....	11
Chapter 2. Concepts.....	13
Chapter 3. Planning.....	15
Chapter 4. License Management in IBM License Metric Tool.....	19
Chapter 5. Installing, upgrading, and uninstalling.....	21
Installation prerequisites.....	21
Installing stand-alone.....	27
Upgrading.....	31
Uninstalling.....	32
Verify the downloaded tar file.....	33
Chapter 6. Checking the application and environment.....	35
Checking the application Docker images in standalone installation.....	35
Checking Kubernetes services status.....	36
Checking Kubernetes node status.....	39
Checking Kubernetes storage status.....	41
Checking application deployment.....	43
Checking system GPU status.....	48
Checking GPU availability.....	49
Chapter 7. Logging in.....	51
Chapter 8. Working with the user interface.....	53
Determining the product version.....	55
Chapter 9. Training and working with models.....	57
Creating and working with data sets.....	57
Data set considerations.....	59
Importing images with COCO annotations.....	61
Search for assets in a data set.....	62
Labeling objects.....	63
Labeling actions.....	66
Training a model.....	68
Working with custom models.....	72
Base models included with IBM Maximo Visual Inspection.....	79

Deploying a trained model.....	80
Preprocessing and post-processing.....	81
Testing a model.....	83
Refining a model.....	84
Automatically labeling objects.....	85
Augmenting the data set.....	87
Importing, exporting, and downloading information.....	88
IBM Maximo Visual Inspection REST APIs.....	90
Understanding metrics.....	91
Chapter 10. Creating and working with project groups.....	95
Chapter 11. Production work flow.....	97
Automatically deploying the newest model.....	98
Chapter 12. Using IBM Maximo Visual Inspection.....	101
Example: Detecting objects in images.....	101
Example: Detecting objects in a video.....	104
Example: Classifying images.....	109
Example: Detecting segmented objects in images.....	110
Example: Detecting actions in a video.....	112
Example: Adding preprocessing and post-processing.....	114
Integrating IBM Maximo Visual Inspection with Maximo Asset Monitor.....	115
Chapter 13. Administering.....	119
Managing users.....	119
Installing a new SSL certificate in IBM Maximo Visual Inspection stand-alone.....	121
Backing up IBM Maximo Visual Inspection.....	121
Monitoring usage metrics.....	122
IBM Maximo Visual Inspection utilities.....	123
Chapter 14. IBM Maximo Visual Inspection Edge.....	129
Planning	129
Installing Docker.....	130
Installing, upgrading, and uninstalling	132
Deploying a trained model	133
Performing an inference.....	136
Inference on embedded edge devices.....	138
Decrypting a trained model.....	140
Chapter 15. Integrating with IBM Maximo Visual Inspection Mobile.....	141
Concepts.....	141
How Maximo Visual Inspection Mobile interacts with IBM Maximo Visual Inspection.....	142
Planning for and installing.....	144
Setting up	144
Device configuration.....	145
Creating inspections.....	148
Dashboard view.....	150
Data Collection vs. Inspection.....	150
Using Maximo Visual Inspection Mobile for labeling and deep learning.....	151
Demo mode.....	151
Handheld mode.....	151
Fixed (mounted) devices for Auto-Capture mode.....	151
Visual Trigger.....	152
Setting up Visual Trigger.....	153
Barcode and OCR Trigger.....	154

Setting up Barcode or OCR Trigger.....	154
Configuring MQTT.....	155
MQTT streaming.....	163
Setting up MQTT streaming.....	163
Troubleshooting.....	164
Chapter 16. Troubleshooting and contacting support.....	167
Troubleshooting common issues - IBM Maximo Visual Inspection standard install.....	167
Troubleshooting common issues - IBM Maximo Visual Inspection Edge.....	183
Gather IBM Maximo Visual Inspection logs and contact support.....	185
Getting fixes from Fix Central.....	187
Contacting IBM Support.....	188
Chapter 17. Notices.....	189
Trademarks.....	190
Terms and conditions for product documentation.....	190
IBM Online Privacy Statement.....	191
Notices.....	195
Trademarks.....	196

About this document

This document provides you with information about installing and using IBM Maximo Visual Inspection to create a dataset that contains images or videos.

Highlighting

The following highlighting conventions are used in this document:

Bold	Identifies commands, subroutines, keywords, files, structures, directories, and other items whose names are predefined by the system. Bold highlighting also identifies graphical objects, such as buttons, labels, and icons that the you select.
<i>Italics</i>	Identifies parameters for actual names or values that you supply.
<code>Monospace</code>	Identifies examples of specific data values, examples of text similar to what you might see displayed, examples of portions of program code similar to what you might write as a programmer, messages from the system, or text that you must type.

ISO 9000

ISO 9000 registered quality systems were used in the development and manufacturing of this product.

Chapter 1. IBM Maximo Visual Inspection overview

IBM Maximo Visual Inspection platform, built on cognitive infrastructure, is a new generation of video/image analysis platforms. The platform offers built-in deep learning models that learn to analyze images and video streams for classification and object detection.

IBM Maximo Visual Inspection includes tools and interfaces for anyone with limited skills in deep learning technologies. You can use IBM Maximo Visual Inspection to easily label images and videos that can be used to train and validate a model. The model can then be validated and deployed in customized solutions that demand image classification and object detection.

The following are the main features of IBM Maximo Visual Inspection:

Streamlined model training

You can use existing models that are already trained as starting point to reduce the time required to train models and improve trained results.

Single-click model deployment

After you create a training model, you can deploy an API with one click. You can then develop applications based on the model that you deployed.

Data set management and labeling

You can manage both raw and labeled data.

Video object detection and labeling assistance

Videos that you import can be scanned for objects and the objects can be automatically labeled.

Architecture overview

The architecture of IBM Maximo Visual Inspection consists of hardware, resource management, deep learning computation, service management, and application service layers. Each layer is built around industry-standard technologies.

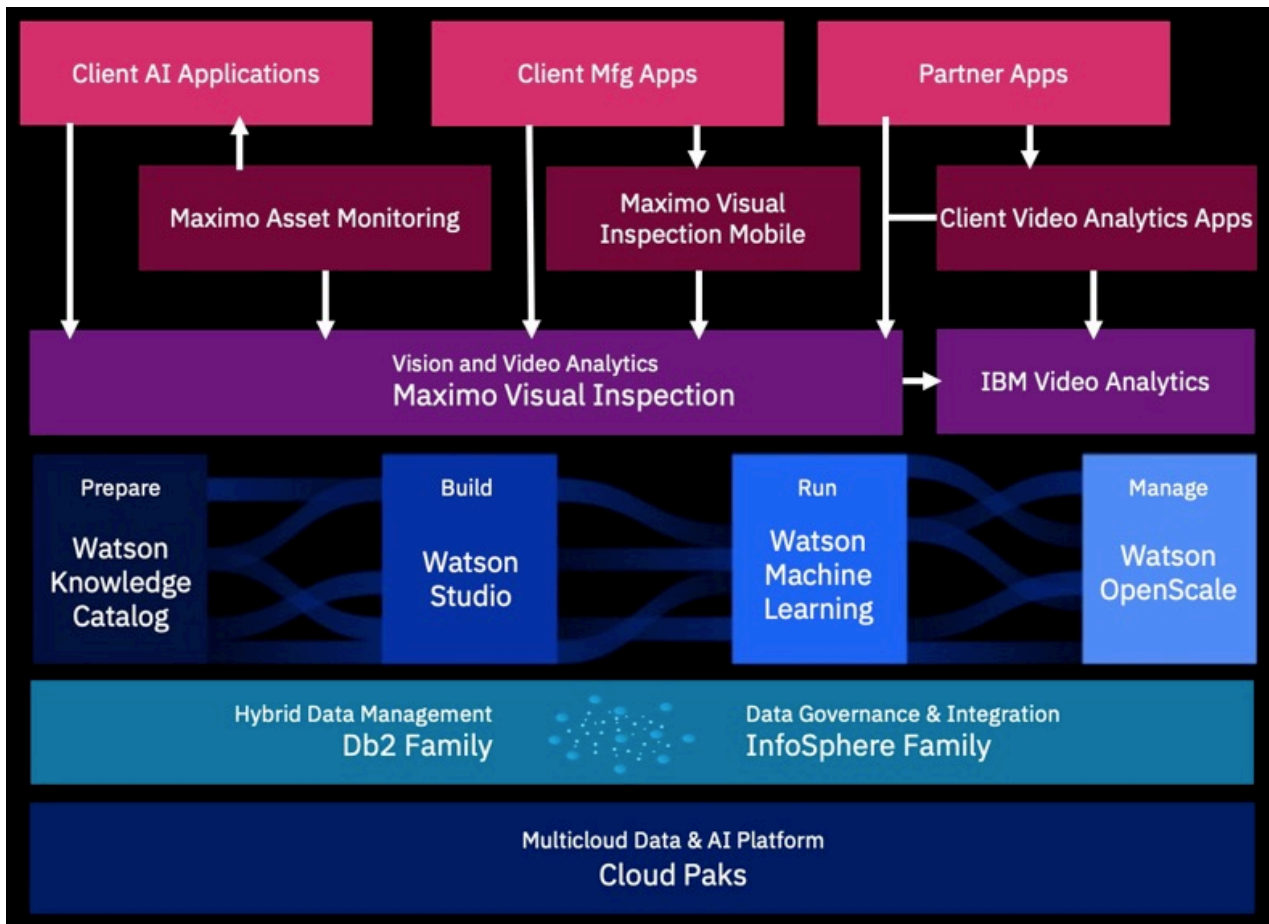


Table 1. Overview of the architecture layers

Architectural Layer	Description
Infrastructure Layer	Consists of hardware systems that support IBM Maximo Visual Inspection, including machines (containers), accelerators (GPUs/FPGAs), storage systems, network.
Resource Management Layer	Coordinates and schedules all computing resources.
Deep Learning Calculation Layer	Consists of deep learning algorithms, including data processing modules, model training, and prediction modules.
Service Management Layer	Manages user projects in a graphical interface, including image preprocessing, management, data set management, training task management, model management.
Application Service Layer	Located on the top of the IBM Maximo Visual Inspection platform, it is responsible for application-related services, including image labeling and preprocessing services, customized image classification services, and customized object detection services.

Note: IBM Maximo Visual Inspection was formerly named IBM Visual Insights (1.2.0) and, prior to that, IBM PowerAI Vision (1.1.5 and earlier).

Use cases

IBM Maximo Visual Inspection includes these main use cases to demonstrate its capabilities:

Static image classification

Determine whether an image belongs to one or more classes of images based on overall image contents. For example, determining the species of bird in an image.

Uragus: 100% Accuracy

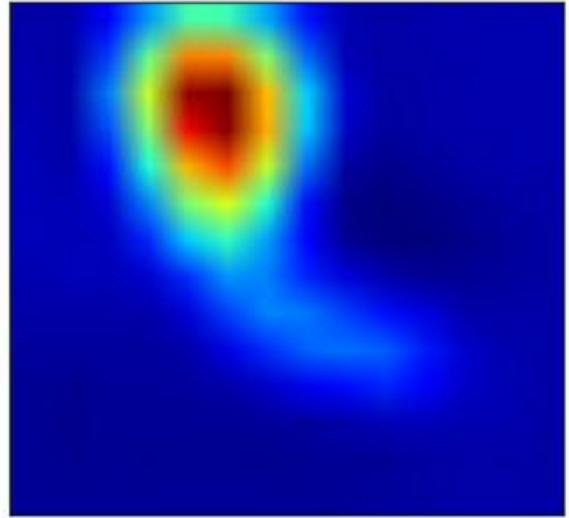


Figure 1. Detecting the overall contents of an image, based on custom training data

Static image object detection

Determine and label the contents of an image based on user-defined data labels. For example, finding and labeling all black cars in an image.



OBJECTS	RESULT
black car 1 objects	1.000

Figure 2. Detecting and labeling instances of objects within an image based on custom training data

Video object detection

Determine and label the contents of an uploaded video or live video stream based on user-defined data labels. For example, finding and labeling all white cars in a video.



Figure 3. Detecting and labeling instances of objects within captured video frames based on custom training data

Static image segmentation

Determine and label the precise location of objects in an image based on user-defined data labels and arbitrary shapes. For example, find and label the precise boundary of all leaves in an image.

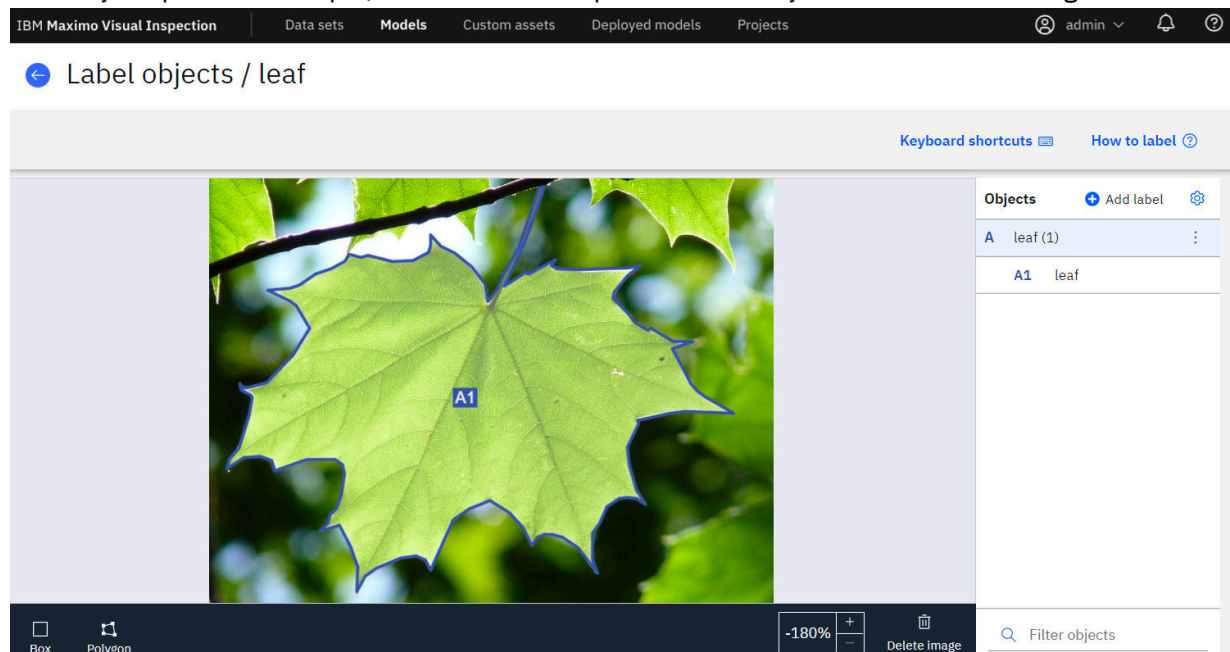


Figure 4. Detecting and labeling the precise edges of an object within an image based on custom training data

Video action detection

Annotate the parts of the video where a specific action is taking place. For example, detect a forehand or backhand stroke in a tennis game.

Auto label an image or video

After deploying a model for object detection, you can improve its accuracy by using the Auto label function. This function uses the labels in the deployed model to generate new labels in the data set; increasing the number of images that are labeled in the data set. The updated data set can be used to train a new, more accurate model.

By default, auto labeled tags are pink, while manually added tags are blue.

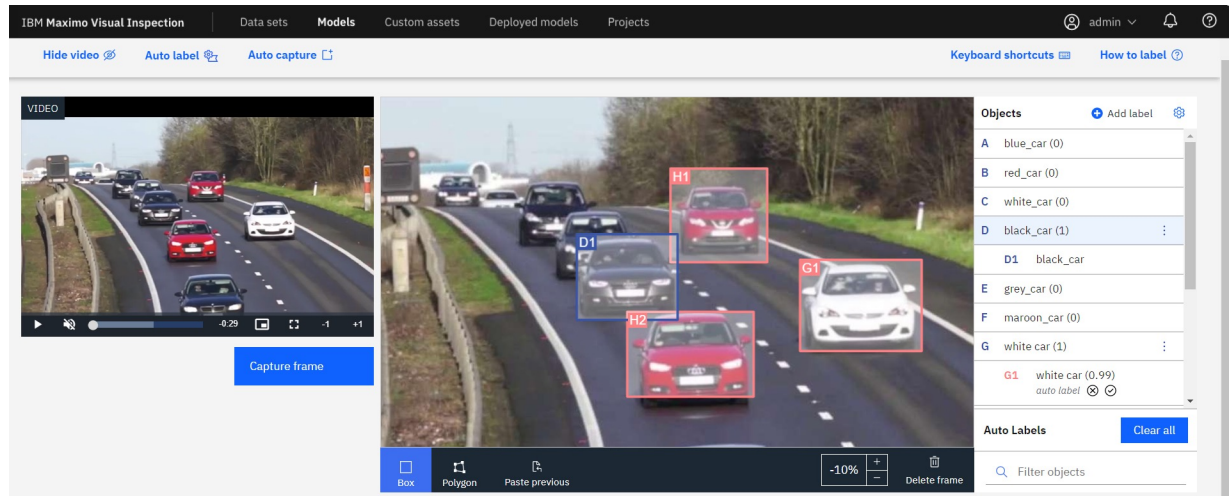


Figure 5. Auto labeled video

Data augmentation

After deploying a model, you can improve the model by using data augmentation to add modified images to the data set, then retraining the model. *Data augmentation* is the use of filters, such as blur and rotate, to create new versions of existing images or frames. Augmentation does not apply to full videos. It can be applied to a video's captured frames just as it is applied to images. When you use data augmentation, a new data set is created that contains all of the existing images, plus the newly generated images, which are marked as augmented.

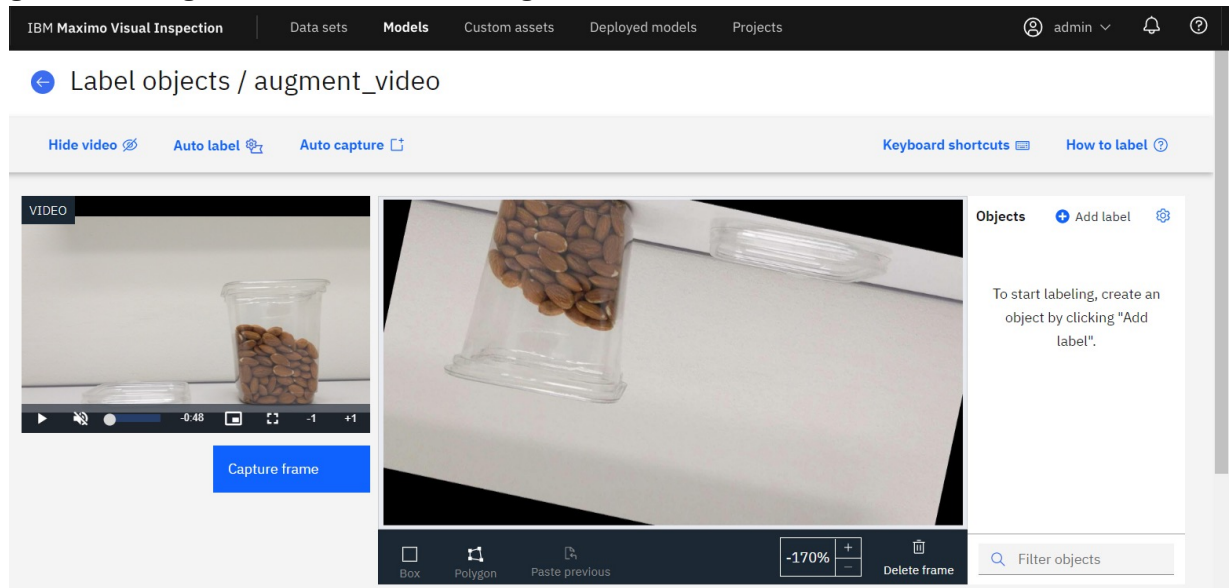


Figure 6. Augmented video

What's new

The following functions, features, and support are added for IBM Maximo Visual Inspection Version 1.3.0:

Product name and portfolio positioning

Formerly known as IBM Visual Insights or IBM PowerAI Vision, this product is renamed to IBM Maximo Visual Inspection to reflect its central role in the IBM AI Applications family. IBM Maximo Visual Inspection is part of the Maximo product family, delivering insights and AI-infused capabilities for smarter assets in the field. This release extends all of the features in the previous versions with new capabilities and functions.

Updated URL

After you upgrade to Version 1.3.0, use the updated URL to access IBM Maximo Visual Inspection:

<https://hostname/visual-inspection/>, where *hostname* is the system on which you installed IBM Maximo Visual Inspection.

Although redirects from the previous product URLs are set up, consider updating bookmarks and files that reference <https://hostname/visual-insights/> or <https://hostname/powerai-vision/>.

Improved support for "transfer learning" with base models

Support for selecting a previously trained model as a base model for the training of a new model is now included for YOLO v3 and Detectron models. For more information, see "Advanced settings" in ["Training a model" on page 68](#).

Integrated support for IBM Maximo Asset Monitor

IBM Maximo Visual Inspection now includes the runtime containers to connect to a Maximo Asset Monitor deployment and provide data on image inferences. For more information, see ["Integrating IBM Maximo Visual Inspection with Maximo Asset Monitor" on page 115](#).

New security policies

Users will now be logged out of the user interface after 60 minutes of inactivity.

On new installations, user accounts are locked if there are too many failed logins. For more information, see ["Managing users" on page 119](#).

User interface improvements

The user interface is updated with the following improvements:

- The data set filters now include support for filtering on images that were created by the automatic saving of inference results. For more information, see ["Preprocessing and post-processing" on page 81](#).
- The GPU usage now indicates the training queue usage if all GPUs are in use. For more information, see ["Training a model" on page 68](#).
- When you test an object detection model with videos, you can now choose to identify objects with bounding boxes or dots. For more information, see ["Testing a model" on page 83](#).

IBM Maximo Visual Inspection REST APIs

You can use REST APIs to work with IBM Maximo Visual Inspection data sets and models, such as performing training and deployment. You can also use them to perform administrative tasks, such as monitoring events. These APIs allow you to bypass the user interface and automate IBM Maximo Visual Inspection processes or solutions.

For information about using the APIs see [IBM Maximo Visual Inspection API documentation](#).

There are also examples of using the APIs for different actions, published [here](#).

IBM Maximo Visual Inspection editions

Each edition of IBM Maximo Visual Inspection has different functionality.

Note: All editions are made available when the product is generally available.

Table 2. IBM Maximo Visual Inspection editions						
Edition	Can be installed as standalone	Model limit	Watermarked images	Can create additional users	Time limit	Renewable
Trial	Y	5	Y		90 days	No - must reinstall
Full	Y	N/A		Y		N/A

IBM Maximo Visual Inspection Trial

IBM Maximo Visual Inspection offers a trial version of the product. It has full functionality, but is not licensed for production use.

The timed trial expires 90 days after you first **start** the application. The time remaining in the time trial is displayed in the user interface as **Days remaining**.

- [“Limitations” on page 7](#)
- [“Installing the trial version” on page 7](#)
- [“What happens when the trial expires?” on page 9](#)
- [“Upgrading to the full version of IBM Maximo Visual Inspection” on page 10](#)

Limitations

This edition of IBM Maximo Visual Inspection works the same as the standard install, with the following exceptions:

- Only five trained models can be on the system at any time. You can, however, export trained models if you want access to them later. See [“Importing, exporting, and downloading IBM Maximo Visual Inspection information” on page 88](#) for instructions.
- Only five models can be deployed at any time.
- Additional users cannot be used with the product. Only the admin user can log in to the user interface and use the product APIs.
- When using a trained model for inference, the resultant images are watermarked.

Installing the trial version



Attention: You cannot install IBM Maximo Visual Inspection stand-alone on the same system that has the following software installed:

- IBM Data Science Experience (DSX)
 - IBM Cloud Private
 - IBM Watson Studio Local Edition
 - Any other Kubernetes based applications
1. You must complete the following installation prerequisites steps before you install IBM Maximo Visual Inspection.
 - a. Complete all steps in the [“Prerequisites for installing IBM Maximo Visual Inspection” on page 21](#) topic.
 - b. Your system must have a proper subscription and repository that provides you with updated packages. For information, see the [Red Hat Subscription Manager](#) documentation.

- c. Turn on Extra Packages for Enterprise Linux (EPEL). For information, see the [EPEL website](#).
 - d. If there was a previous version of the product installed but you do **NOT** wish to migrate the data, delete or move /opt/powerai-vision/volume (for PowerAI Vision) or /opt/vision/volume (for IBM Maximo Visual Insights). This will ensure that data for the previous install, such as data sets and trained models, will not appear in the newer version of the product.
2. Go to the [IBM Maximo Visual Inspection Trial download site](#), then download the .tar file and the .rpm files as instructed.
 3. Unzip and untar the product tar file, and run the installation command for the platform you are installing on. The install files are extracted to visual-inspection-arch-1.3.0.0-ppa, where arch is x86 or ppc, depending on your platform.

RHEL

```
sudo yum install ./<file_name>.rpm
```

Ubuntu

```
sudo dpkg -i ./<file_name>.deb
```

4. Load the IBM Maximo Visual Inspection images from the directory that contains the extracted tar file. The user running the script must have Docker privileges:

```
sudo /opt/ibm/vision/bin/load_images.sh -f ./file_name.tar
```

Note: The installation process can take some time to complete.

5. (RHEL only) Open ports for the firewall to access IBM Maximo Visual Inspection by running this script:

```
sudo /opt/ibm/vision/sbin/firewall.sh
```

6. After the installation is complete, you can start IBM Maximo Visual Inspection by running this script:

```
sudo /opt/ibm/vision/bin/vision-start.sh
```

A user named admin is created with a password of passw0rd. For instructions to change these values, see [“Managing users”](#) on page 119.

Note: The startup script will modify ownership and permissions on /opt/ibm/vision/volume so that the containers can run under a non-root ID and access the data.

You must read and accept the license agreement that is displayed before you can use IBM Maximo Visual Inspection.

It can take several minutes to start IBM Maximo Visual Inspection. To check the status of the startup process, run this script:

```
sudo /opt/ibm/vision/bin/kubect1.sh get pods
```

In the output from the **kubect1.sh get pods** script, you can verify which IBM Maximo Visual Inspection components are available by locating the Deployment section and identifying that the AVAILABLE column has a value of 1 for each component. The following is an example of the output from the **kubect1.sh get pods** script that shows all components are available:


```
$ /opt/ibm/vision/bin/kubect1.sh get pods
```

NAME	READY	STATUS	RESTARTS
vision-cod-infer-ce5c3be1-491a-486c-81ae-2c49bdf17242-86cdtmv8q	1/1	Running	0
vision-edge-connector-7db99dbb7f-vbtpg	1/1	Running	0
vision-edge-mqttbroker-79cb65b865-rhr7w	1/1	Running	0
vision-elasticsearch-6b779bb5f5-9mnck	1/1	Running	0
vision-event-service-86c8c8cd6d-2wq5r	1/1	Running	0
vision-fpga-device-plugin-86htq	1/1	Running	0
vision-keycloak-58f7566896-nwph6	1/1	Running	0
vision-logstash-565d995764-hnnxg	1/1	Running	0
vision-mongodb-b59b56645-wlxgn	1/1	Running	0
vision-postgres-6c57856875-zlknm	1/1	Running	0
vision-service-687b85b97f-spg8n	1/1	Running	1
vision-taskanaly-6dc659c45c-t9rb2	1/1	Running	0
vision-ui-7d885944fc-x8tfc	1/1	Running	0
vision-video-microservice-8457664dc6-dcb2g	1/1	Running	0

After the application startup has completed and the user interface is available, it can be accessed at `https://hostname/visual-inspection/`, where *hostname* is the system on which you installed IBM Maximo Visual Inspection.

What happens when the trial expires?

You can see how much time is left in the trial by reviewing the countdown in the header of the user interface. When the timed trial expires, the product will cease to work, including any running training, inference, import, or export operations. However, if you purchase a license, you will automatically regain access to all of your data sets, models, and so on. When the trial expires, you can upgrade to the full version or remove the trial version. The trial cannot be extended.

If the trial expires and you want to purchase IBM Maximo Visual Inspection, follow the instructions in [“Upgrading to the full version of IBM Maximo Visual Inspection” on page 10](#).

If the trial expires and you do not decide to purchase IBM Maximo Visual Inspection, follow these steps:

1. Remove previously installed images by running the following script:

```
sudo /opt/ibm/vision/bin/purge_image.sh 1.3.0.0
```

Optionally remove all product data by running the following script. This will remove data sets, models, and so on:

```
sudo /opt/ibm/vision/bin/purge_data.sh
```

2. Remove IBM Maximo Visual Inspection by running the following command:

- For RHEL:

```
sudo yum remove visual-inspection
```

- For Ubuntu:

```
sudo dpkg --remove visual-inspection
```

3. Delete the data directory by running the following command:

```
sudo rm -rf /opt/ibm/vision/
```

Upgrading to the full version of IBM Maximo Visual Inspection

When you are ready to purchase IBM Maximo Visual Inspection, you can buy a license from [IBM Maximo Visual Inspection Marketplace](#). Use one of these methods to upgrade to the full version. Your data is not deleted when the product is uninstalled. You will automatically regain access to all of your data sets, models, and so on.

1. Stop the current instance of IBM Maximo Visual Inspection by running the following script:

```
sudo /opt/ibm/vision/bin/vision-stop.sh
```

2. Obtain and install IBM Maximo Visual Inspection:

Install IBM Maximo Visual Inspection from IBM Passport Advantage

- a. Download the product tar file from the [IBM Passport Advantage](#) website.
- b. Unzip and untar the product tar file, and run the installation command for the platform you are installing on. The install files are extracted to `visual-inspection-arch-1.3.0.0-ppa`, where *arch* is x86 or ppc, depending on your platform.

RHEL

```
sudo yum install ./<file_name>.rpm
```

Ubuntu

```
sudo dpkg -i ./<file_name>.deb
```

- c. Load the IBM Maximo Visual Inspection images from the directory that contains the extracted tar file. The user running the script must have Docker privileges:

```
sudo /opt/ibm/vision/bin/load_images.sh -f ./file_name.tar
```

Note: The installation process can take some time to complete.

- d. After the installation is complete, you can start IBM Maximo Visual Inspection by running this script:

```
sudo /opt/ibm/vision/bin/vision-start.sh
```

Related concepts

[Uninstalling IBM Maximo Visual Inspection stand-alone](#)

You must uninstall IBM Maximo Visual Inspection stand-alone on your system, before you can install IBM Cloud Private, IBM Data Science Experience Local, or other Kubernetes-based applications.

Model functionality

You can train these types of models in IBM Maximo Visual Inspection.

<i>Table 3. Types of models</i>								
Model type	GoogLeNet	Faster R-CNN	Tiny YOLO v2	YOLO v3	Detectron	Single Shot Detector (SSD)	Structured segment network (SSN)	Custom model
Description	System default when training for image classification	Optimized for accuracy	Optimized for speed. These models use "you only look once" (YOLO) v2 and will take longer to train than other models in this product. You can choose the accelerator or to deploy to.	Optimized for speed. These models use "you only look once" (YOLO) v3 and will take longer to train than other models in this product. Generates a CoreML asset.	Detectron Mask R-CNN models can use objects labeled with polygons for greater training accuracy. You can disable segmentation for a shorter training time.	Used for real-time inference and embedded devices. It is almost as fast as YOLO but not as accurate as Faster R-CNN.	Used for video action detection	Imported model used for training.
Image classification	Yes	No	No	No	No	No	No	Yes
Object detection	No	Yes	Yes	Yes	Yes	Yes	No	Yes
Action detection	No	No	No	No	No	No	Yes	No

Table 4. Model functionality

Model type	GoogLeNet	Faster R-CNN	tiny YOLO v2	YOLO v3	Detectron	Single Shot Detector (SSD)	Structured segment network (SSN)	Custom model
Deploy multiple models to one GPU	Yes	Yes	No	No	No	No	No	No
Deploy to CPU (Standard install)	No	No	Yes	No	No	No	No	No
Deploy to TensorRT	No	Yes	No	No	No	Yes	No	No
Enable Core ML	Yes	No	Yes	Yes	No	No	No	No
Import / Export model	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Supported on Maximo Visual Inspection Edge	Yes (default)	Yes (default)	Yes	Yes	Yes	Yes	No	Yes - TensorFlow, Keras, and PyTorch
Use for transfer learning	Yes	Yes	No	Yes	Yes	Yes	No	No
Deploy to CPU (Maximo Visual Inspection Edge)	Yes	Yes	Yes	No	No	No	No	No

Chapter 2. IBM Maximo Visual Inspection concepts

IBM Maximo Visual Inspection provides an easy to use graphical user interface (GUI) that you can use to quickly create computer vision-related artificial intelligence (AI) solutions.

You must be familiar with the following concepts before you can start using IBM Maximo Visual Inspection:

Data set

A data set is a collection of images and videos that you uploaded to IBM Maximo Visual Inspection. An example of a data set would be images of cars.

Category

A category is used to classify an image. The image can belong to only a single category. An example of a category for a data set that contains cars would be car manufacturers (Toyota, Honda, Chevy, and Ford).

Custom asset

Custom assets are certain assets that are created outside of IBM Maximo Visual Inspection but that can be used by IBM Maximo Visual Inspection.

Custom model

Also known as a *custom network*. This is a model that was trained outside of IBM Maximo Visual Inspection. For information about what is supported by custom models, see [“Model functionality” on page 11](#).

Custom inference script

Contains files that specify actions to be done outside of IBM Maximo Visual Inspection. For example, to be used with preprocessing, post-processing, or Maximo Asset Monitor. For more information, see [“Preprocessing and post-processing” on page 81](#) and [“Integrating IBM Maximo Visual Inspection with Maximo Asset Monitor” on page 115](#).

Object

An object is used to identify specific items in an image or specific frames in a video. You can label multiple objects in an image or a frame in a video. An example of objects in an image of cars might be wheel, headlights, and windshield.

Project

Project groups allow you to group trained models with the data sets that were used for training. This grouping is optional but is a useful way to organize related data sets. For example, project groups would be useful with a workflow that clones data sets as you refine labels and work toward a more accurate model. For more information about projects, see [Chapter 10, “Creating and working with project groups,” on page 95](#).

Model

A model is a set of tuned algorithms and that produces a predicted output. Models are trained based on the input that is provided by a data set to classify images or video frames, or find objects in images or video frames.

Neural network

A model implementation using a deep learning framework with nodes and weights.

Training concepts

Iteration

A full forward and backward pass using a set of images in the training of the neural network.

Batch

The set of images used in a full forward and backward pass in training of the neural network.

Batch size

The size of the batch of images used in an iteration.

Epoch

The measure for an entire data set passed forward and backward through the neural network one time. Usually the batch size is smaller than the full data set, so an epoch consists of multiple iterations of "Batch size".

Chapter 3. Planning for IBM Maximo Visual Inspection

You must meet the software and hardware requirements and understand the supported file types before you can install IBM Maximo Visual Inspection.

- [“Hardware requirements” on page 15](#)
- [“Software requirements” on page 15](#)
- [“Networking requirements” on page 16](#)
- [“Disk space requirements” on page 16](#)
- [“Supported web browsers” on page 16](#)
- [“Image support” on page 17](#)
- [“Supported video types” on page 17](#)
- [“Deep learning frameworks” on page 18](#)
- [“Limitations” on page 18](#)

Hardware requirements

IBM Maximo Visual Inspection requires the following hardware:

Hardware

The following devices are supported:

- POWER8 S822LC (8335-GTB) or POWER9 AC922 with at least one NVIDIA NVLink capable GPU
- POWER9 IC922 with at least one NVIDIA T4 GPU
- x86 system with at least one NVIDIA Pascal, Volta, or Turing-architecture GPU

Required specifications

Your device must meet these minimum requirements:

- 64 GB of memory
- Ethernet network interface
- 75 GB of storage for installation, and at least 40 GB of storage for runtime. See [“Disk space requirements” on page 16](#) for details.
- The Nvidia GPU must be configured in the "Default" compute mode. The "Exclusive Process" mode will cause trainings to fail. See [nvidia-smi usage](#) for details on compute modes.
- There are multiple options for deploying the model for testing. Deploying a model to a Xilinx FPGA requires the Xilinx Alveo U200 Accelerator card.

Software requirements

You must install the following software before you install IBM Maximo Visual Inspection:

Linux

- Red Hat Enterprise Linux (RHEL) RHEL 7.6 ALT (little endian) for POWER9™
- RHEL 7.7 for x86 and POWER8®.
- Ubuntu 18.04

Note: The Ubuntu Hardware Enablement (HWE) kernel is not supported. Kubernetes services do not function correctly, preventing IBM Maximo Visual Inspection from starting successfully.

NVIDIA CUDA

10.2 or later drivers are required. For information, see the [NVIDIA CUDA Toolkit](#) website.

Docker

- RHEL: docker-1.13.1
- Ubuntu: Docker CE or EE 18.06.01

Networking requirements

Your environment must meet the following networking requirements:

- A default route must be specified on the host system.
 - For instructions to do this on Ubuntu, refer to the IP addressing section in the Ubuntu Network Configuration. Search for the steps to configure and verify the default gateway.
 - For instructions to do this on Red Hat Enterprise Linux (RHEL), refer to [2.2.4 Static Routes and the Default Gateway](#) in the Red Hat Customer Portal.
- For RHEL, docker0 must be in a trusted firewall zone. If it is not in a trusted firewall zone, modify the RHEL settings as follows:

```
sudo nmcli device set docker0 managed yes
sudo nmcli connection modify docker0 connection.zone trusted
sudo systemctl stop NetworkManager.service
sudo firewall-cmd --permanent --zone=trusted --change-interface=docker0
sudo systemctl start NetworkManager.service
sudo nmcli connection modify docker0 connection.zone trusted
sudo systemctl restart docker.service
```

- IPv4 port forwarding must be enabled.

If IPv4 port forwarding is not enabled, run the **/sbin/sysctl -w net.ipv4.conf.all.forwarding=1** command. For more information about port forwarding with Docker, see [UCP requires IPv4 IP Forwarding](#) in the Docker success center.

- IPv6 must be enabled.

Disk space requirements

IBM Maximo Visual Inspection has the following storage requirements for the initial product installation and for the data sets that will be managed by the product.

- /var - The product installation requires at least 75 GB of space in the /var file system for the product Docker images. IBM Maximo Visual Inspection also generates log information in this file system. The installation process requires additional space because all Docker images are extracted to disk requiring about 40 GB of space, then the images are loaded by Docker. When the image is loaded, the extracted image is deleted but while images are being extracted and loaded, space is needed for both copies. All application Docker images are extracted and loaded in parallel.

Recommendation: If you want to minimize the root (/) file system, make sure that /var has its own volume.

- /opt - IBM Maximo Visual Inspection data sets, models, and runtime data are stored in this file system. This file system must have at least five GB of free space, in addition to any data sets, models or other runtime data for IBM Maximo Visual Inspection to operate successfully. The storage needs will vary depending on the data sets and the contents. For example, video data can require large amounts of storage.

Recommendation: If you want to minimize the root (/) file system, make sure that /opt has its own volume. The /opt file system should have at least 40 GB of space, although this value might be more depending on your data sets.

Supported web browsers

The following web browsers are supported:

- Google Chrome Version 60, or later
- Firefox Quantum 59.0, or later

Image support

- The following image formats are supported:
 - JPEG
 - PNG
 - DICOM
- Images with COCO annotations are supported. For details, see [“Importing images with COCO annotations”](#) on page 61.
- IBM Maximo Visual Inspection has limited support for Pascal VOC annotations. Annotations for multiple files residing in a common XML file are not supported. In other words, each annotation XML file can only contain annotations for a single image, identified by the `filename` attribute.

If you have a single XML annotation file containing annotations for multiple images in the data set to be imported, the annotations need to be split out into separate XML files before IBM Maximo Visual Inspection can import the annotations successfully.

- The models used by IBM Maximo Visual Inspection have limitations on the size and resolution of images. If the original data is high resolution, then the user must consider:
 - If the images do not need fine detail for classification or object detection, they should be down-sampled to 1-2 megapixels.
 - If the images do require fine detail, they should to be divided into smaller images of 1-2 megapixels each.
 - There is a 24 GB size limit per upload session. This limit applies to a single .zip file or a set of files. You can, however upload 24 GB of files, then upload more after the original upload completes.
 - Large images will be scaled to the appropriate dimensions for the model as follows:
 - SSD: 512 x 512 pixels
The original aspect ratio is maintained. If necessary, black bands are added to the image to make it fit.
 - Tiny YOLO v2: 416 x 416 pixels
The longest edge is scaled to 416 pixels and, if necessary, black bands are added to the shorter side to make it 416 pixels.
 - YOLO v3: 608 x 608 pixels
 - Faster R-CNN: 1000 x 600 pixels
The original aspect ratio is maintained. If necessary, black bands are added to the image to make it fit.
 - Detectron: 1333 x 800 pixels
 - GoogLeNet: 224 x 224 pixels
 - Action detection: 224 x 224 pixels

Supported video types

The following video formats are supported:

Can be played in the IBM Maximo Visual Inspection GUI:

- Ogg Vorbis (.ogg)
- VP8 or VP9 (.webm)
- H.264 encoded videos with MP4 format (.mp4)

Supported by API only:

- Matroska (.mkv)
- Audio Video Interleave (.avi)

- Moving Picture Experts Group (.mpg or .mpeg2)

Not supported:

Videos that are encoded with the H.265 codec.

Deep learning frameworks

The following frameworks are included with IBM Maximo Visual Inspection.

<i>Table 5. Included frameworks</i>				
Framework	Version	Python 2.7 support	Python3.6 support	Notes
Caffe 2	1.3.1	Yes	No	Supported for Detectron models
IBM Caffe	1.0.0	Yes	No	Supported for GoogLeNet, Faster R-CNN, and Tiny YOLO v2 models
Keras	2.3.1	Yes	Yes	Supported for custom models
TensorFlow	2.1.0	No	Yes	Supported for custom models
TensorRT	7.0.0.11	Yes	Yes	Supported for SSD models
PyTorch	1.3.1	Yes	Yes	Supported for custom models

Limitations

Following are some limitations for IBM Maximo Visual Inspection 1.3.0:

- IBM Maximo Visual Inspection uses an entire GPU when you are training a dataset. Multiple GoogleNet or Faster R-CNN models can be deployed to a single GPU. Other types of models take an entire GPU when deployed. For details about other differences between model types, see [“Model functionality” on page 11](#).

The number of active GPU tasks (model training and deployment) that you can run at the same time depends on the number of GPUs on your server. You must verify that there are enough available GPUs on the system for the desired workload. The number of available GPUs is displayed on the user interface.

- You cannot install IBM Maximo Visual Inspection stand-alone on a system that already has any of these products installed:
 - IBM Data Science Experience (DSX)
 - IBM Cloud Private
 - IBM Watson Studio Local Edition
 - Any other Kubernetes based applications

Chapter 4. License Management in IBM License Metric Tool

The IBM Maximo Visual Inspection product is licensed per *Virtual Server* ([Learn about software licensing - Virtual Server](#)). When the product is installed, a software license metric (SLM) tag file is created to track usage with the IBM License Metric Tool.

The license metric tag is an XML file, with extension `.slmtag`. The IBM License Metric Tool discovers the license metric tag file and provides license consumption reports that, compared with license entitlements, allow IBM to verify license compliance. The tag file is human-readable and can therefore be interpreted by individuals for audit purposes.

The license metric tag file has a standard format and consists of two parts:

Header information

Contains:

SchemaVersion

Identifies the schema version of the license metric tag file.

SoftwareIdentity

Identifies the software identity instance that provides license metric data. Contains:

- **Name**

Name of the software identity - *IBM Maximo Visual Inspection* or *IBM Maximo Visual Inspection Edge*

- **PersistentId**

Unique identifier of the software identity. For IBM Maximo Visual Inspection 1.3.0, the assigned **PersistentId** is:

- **IBM Maximo Visual Inspection** - ebb8d2e1bd62488c8c196f568857ae38
- **IBM Maximo Visual Inspection Edge** - 297aaa94baa441e0ad91a609b24083b7

- **InstanceId**

Identifies the instance of the software identity that provides metrics by the path of the software for which *SLMTag* is generated - `/opt/ibm/vision`.

Metrics information

IBM Maximo Visual Inspection 1.3.0 is licensed per Virtual Server, so the values are:

- **Type** - *VIRTUAL_SERVER*
- **Period** - **StartTime** is the time of install/deploy, **EndTime** is set to date '9999-12-31' so that the IBM License Metric Tool will understand that it as a perpetual license.

Chapter 5. Installing, upgrading, and uninstalling IBM Maximo Visual Inspection

Use the information in these topics to work with the product installation.

Only the most current level of each release of IBM Maximo Visual Inspection should be installed, where version numbers are in the format *version.release.modification*.

After installing IBM Maximo Visual Inspection, you can optionally change the SSL certificate by following the steps in this topic: [“Installing a new SSL certificate in IBM Maximo Visual Inspection stand-alone”](#) on page 121.

Prerequisites for installing IBM Maximo Visual Inspection

Before you can install IBM Maximo Visual Inspection, you must configure Red Hat Enterprise Linux (RHEL), enable the Fedora Extra Packages for Enterprise Linux (EPEL) repository, and install NVIDIA CUDA drivers.

Note: Neither IBM Watson® Machine Learning Community Edition nor IBM Watson Machine Learning Accelerator are required for running IBM Maximo Visual Inspection.

See Chapter 3, “Planning for IBM Maximo Visual Inspection,” on page 15 to ensure that your environment meets all software and hardware requirements.

- [“Red Hat Enterprise Linux operating system and repository setup”](#) on page 21
- [“Ubuntu operating system and repository setup”](#) on page 22
- [“NVIDIA Components: IBM POWER9 specific udev rules \(Red Hat only\)”](#) on page 23
- [“Remove previously installed CUDA and NVIDIA drivers”](#) on page 23
- [“Install the GPU driver \(RHEL\)”](#) on page 24
- [“Install the GPU driver \(Ubuntu\)”](#) on page 25
- [“Verify the GPU driver”](#) on page 25
- [“Install Docker and nvidia-docker2 \(RHEL\)”](#) on page 26
- [“Install Docker and nvidia-docker2 \(Ubuntu\)”](#) on page 26

Red Hat Enterprise Linux operating system and repository setup

1. Enable common, optional, and extra repo channels.

IBM POWER8:

```
sudo subscription-manager repos --enable=rhel-7-for-power-le-optional-rpms
```

```
sudo subscription-manager repos --enable=rhel-7-for-power-le-extras-rpms
```

```
sudo subscription-manager repos --enable=rhel-7-for-power-le-rpms
```

IBM POWER9:

```
sudo subscription-manager repos --enable=rhel-7-for-power-9-optional-rpms
```

```
sudo subscription-manager repos --enable=rhel-7-for-power-9-extras-rpms
```

```
sudo subscription-manager repos --enable=rhel-7-for-power-9-rpms
```

x86:

```
sudo subscription-manager repos --enable=rhel-7-server-optional-rpms
```

```
sudo subscription-manager repos --enable=rhel-7-server-extras-rpms
```

```
sudo subscription-manager repos --enable=rhel-7-server-rpms
```

2. Install packages needed for the installation.

```
sudo yum -y install wget nano bzip2
```

3. Enable the Fedora Project Extra Packages for Enterprise Linux (EPEL) repository:

```
wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
```

```
sudo rpm -ihv epel-release-latest-7.noarch.rpm
```

4. Load the latest kernel or do a full update:

- Load the latest kernel:

- For x86:

```
sudo yum install kernel-devel
sudo yum update kernel kernel-devel kernel-tools kernel-tools-libs
reboot
```

- For POWER:

```
sudo yum install kernel-devel
sudo yum update kernel kernel-devel kernel-tools kernel-tools-libs kernel-bootwrapper
reboot
```

- Do a full update:

```
sudo yum install kernel-devel
sudo yum update
sudo reboot
```

5. Install Docker and configure it so that IBM Maximo Visual Inspection containers can use NVIDIA GPUs. For instructions, see [“Install Docker and nvidia-docker2 \(RHEL\)”](#) on page 26.

Ubuntu operating system and repository setup

1. Install packages needed for the installation

```
sudo apt-get install -y wget nano apt-transport-https ca-certificates curl software-properties-common
```

2. Ensure the kernel headers are installed and match the running kernel. Compare the outputs of:

```
dpkg -l | grep linux-headers kernel-package kernel-headers
```

and

```
uname -r
```

Ensure that the `linux-headers` package version *exactly* match the version of the running kernel. If they are not identical, bring them in sync as appropriate:

- Install missing packages.
- Update down level packages.
- Reboot the system if the packages are newer than the active kernel.

3. Alternatively, do a full update:

```
sudo apt-get update
sudo apt-get dist-upgrade
sudo reboot
```

NVIDIA Components: IBM POWER9 specific udev rules (Red Hat only)

1. Copy the `/lib/udev/rules.d/40-redhat.rules` file to the directory for user overridden rules:

```
sudo cp /lib/udev/rules.d/40-redhat.rules /etc/udev/rules.d/
```

2. Edit the `/etc/udev/rules.d/40-redhat.rules` file:

```
sudo nano /etc/udev/rules.d/40-redhat.rules
```

3. Comment out the entire "Memory hotadd request" section and save the change:

```
# Memory hotadd request
#SUBSYSTEM!="memory", ACTION!="add", GOTO="memory_hotplug_end"
#PROGRAM="/bin/uname -p", RESULT=="s390*", GOTO="memory_hotplug_end"

#ENV{.state}="online"
#PROGRAM="/bin/systemd-detect-virt", RESULT=="none", ENV{.state}="online_movable"
#ATTR{state}=="offline", ATTR{state}="$env{.state}"

#LABEL="memory_hotplug_end"
```

4. Optionally, delete the first line of the file, since the file was copied to a directory where it cannot be overwritten:

```
# do not edit this file, it will be overwritten on update
```

5. Restart the system for the changes to take effect:

```
sudo reboot
```

Remove previously installed CUDA and NVIDIA drivers

Before installing the updated GPU driver, uninstall any previously-installed CUDA and NVIDIA drivers. Follow these steps:

1. Remove all CUDA Toolkit and GPU driver packages.

You can display installed CUDA and driver packages by running these commands:

```
rpm -qa | egrep 'cuda.*(9-2|10-0|10-1)'
```

```
rpm -qa | egrep '(cuda|nvidia).*(396|410|418)\.'
```

Verify the list and remove with **yum remove**.

2. Remove any CUDA Toolkit and GPU driver repository packages.

These should have been included in step 1, but you can confirm with this command:

```
rpm -qa | egrep '(cuda|nvidia).*repo'
```

Use **yum remove** to remove any that remain.

3. Clean the yum repository:

```
sudo yum clean all
```

4. Remove cuDNN and NCCL:

```
sudo rm -rf /usr/local/cuda /usr/local/cuda-9.2 /usr/local/cuda-10.0 /usr/local/cuda-10.1
```

5. Reboot the system to unload the GPU driver:

```
sudo shutdown -i now
```

Install the GPU driver (RHEL)

Install the driver by following these steps:

Note: These instructions are intended for installation on a single Red Hat instance. If the GPU driver must be installed on many Red Hat instances, follow the instructions in this article: [NVIDIA and Red Hat: Simplifying NVIDIA GPU Driver Deployment on Red Hat Enterprise Linux](#).

1. Download the NVIDIA GPU driver:

- Go to [CUDA Toolkit 10.2 Download](#).
- For **Architecture**, select the option that matches your target platform.
- For **Distribution**, select *RHEL*.
- For **Version**, select 7.
- Select the **Installer Type**:
 - For isolated installation, select *rpm (local)*.
 - For smaller downloads with installation over a network, select *rpm (network)*.
- Click **Download** to download the driver.

Important: An rpm file should be downloaded. If a different type of file is downloaded, verify that you chose the correct options and try again.

2. Install CUDA and the GPU driver.

Note:

- For AC922 systems: OS and system firmware updates are required before you install the latest GPU driver.
- A known issue affects the loading of Nvidia kernel modules that use locales other than English. This issue is fixed in `kernel-4.14.0-115.17.1.el7a.ppc64le.rpm`. For more information, see <https://access.redhat.com/errata/RHSA-2020:0174>. For the installation on older kernel levels, change `/etc/locale.conf`:

```
# cat /etc/locale.conf  
LANG="en_US.UTF-8"
```

```
sudo rpm -ivh nvidia-driver-local-repo-rhel7-440.*.rpm
```

```
sudo yum install nvidia-driver-latest-dkms
```

3. Set nvidia-persistenced to start at boot (required for ppc64le, recommended for x86):

```
sudo systemctl enable nvidia-persistenced
```

4. Restart to activate the driver.

5. Verify the setup:

IBM Power

```
docker run --rm nvidia/cuda-ppc64le nvidia-smi
```

x86_64

```
docker run --rm nvidia/cuda nvidia-smi
```


Install the GPU driver (Ubuntu)

Many of the deep learning packages require the GPU driver packages from NVIDIA.

Install the GPU driver by following these steps:

1. Download the NVIDIA GPU driver.

- Go to [CUDA Toolkit 10.2 Download](#).
- For **Architecture**, select the option that matches your target platform.
- For **Distribution**, select *Ubuntu*.
- For **Version**, select *18.04*.
- Select the **Installer Type**:
 - For isolated installation, select *deb (local)*.
 - For smaller downloads with installation over a network, select *deb (network)*.
- Click **Download** to download the driver.

Important: A deb file should be downloaded. If a different type of file is downloaded, verify that you chose the correct options and try again.

2. The driver file name is `NVIDIA-Linux-ppc64le-440.87.01.run`. Give this file execute permission and execute it on the Linux image where the GPU driver is to be installed.

When the file is executed, you are asked two questions. It is recommended that you answer "Yes" to both questions. If the driver fails to install, check the `/var/log/nvidia-installer.log` file for relevant error messages.

3. Install the GPU driver repository and cuda-drivers:

```
sudo dpkg -i nvidia-driver-local-repo-ubuntu1804-440.*.deb
```

```
sudo apt-key add /var/nvidia-driver-local-repo-440.*/*.pub
```

```
sudo apt-get update
```

```
sudo apt-get install cuda-drivers
```

4. Set `nvidia-persistenced` to start at boot

```
sudo systemctl enable nvidia-persistenced
```

5. Reboot the system

Verify the GPU driver

Verify that the CUDA drivers are installed by running the `/usr/bin/nvidia-smi` application.

Example output

```
# nvidia-smi
Fri Mar 15 12:23:50 2019
```

NVIDIA-SMI 418.29 Driver Version: 418.29 CUDA Version: 10.1									
GPU	Name	Persistence-M	Bus-Id	Disp.A	Memory-Usage	Volatile	Uncorr. ECC		
Fan	Temp	Perf	Pwr:Usage/Cap			GPU-Util	Compute M.		
0	Tesla P100-SXM2...	On	00000002:01:00.0	Off	2618MiB / 16280MiB	43%	0	Default	
N/A	50C	P0	109W / 300W						
1	Tesla P100-SXM2...	On	00000003:01:00.0	Off	0MiB / 16280MiB	0%	0	Default	
N/A	34C	P0	34W / 300W						
2	Tesla P100-SXM2...	On	0000000A:01:00.0	Off	5007MiB / 16280MiB	0%	0	Default	
N/A	48C	P0	44W / 300W						
3	Tesla P100-SXM2...	On	0000000B:01:00.0	Off	0MiB / 16280MiB	0%	0	Default	
N/A	36C	P0	33W / 300W						

Processes:					GPU Memory Usage
GPU	PID	Type	Process name		
0	114476	C	/opt/miniconda2/bin/python		2608MiB
2	114497	C	/opt/miniconda2/bin/python		958MiB
2	114519	C	/opt/miniconda2/bin/python		958MiB
2	116655	C	/opt/miniconda2/bin/python		2121MiB
2	116656	C	/opt/miniconda2/bin/python		958MiB

For help understanding the output, see “Checking system GPU status” on page 48.

Install Docker and nvidia-docker2 (RHEL)

Follow these steps to install Docker on RHEL. For full details, refer to <https://github.com/NVIDIA/nvidia-docker#rhel-docker>.

1. Install Docker:

```
sudo yum install docker
```

2. Reboot the system.
3. Add the package repositories:

On x86:

```
distribution=$(cat /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.repo |
sudo tee /etc/yum.repos.d/nvidia-docker.repo
sudo yum install -y nvidia-container-toolkit
sudo systemctl restart docker
```

On IBM Power®:

```
distribution=$(cat /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/docker/$distribution/docker.repo | sudo tee /etc/
yum.repos.d/docker.repo
sudo yum install -y nvidia-container-runtime-hook
sudo systemctl restart docker
```

Install Docker and nvidia-docker2 (Ubuntu)

Use these steps to install Docker and nvidia-docker 2.

1. For Ubuntu platforms, a Docker runtime must be installed. If there is no Docker runtime installed yet, install Docker-CE on Ubuntu.

IBM Power

```
sudo apt-get update
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-
properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=ppc64el] https://download.docker.com/linux/ubuntu
bionic stable"
sudo apt-get update
sudo apt-get install docker-ce=18.06.1~ce~3-0~ubuntu
```

x86_64

```
sudo apt-get update
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-
properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
bionic stable"
sudo apt-get update
sudo apt-get install docker-ce
```

2. Install nvidia-docker 2.

```
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add -
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo
tee /etc/apt/sources.list.d/nvidia-docker.list
sudo apt-get update
sudo apt-get install nvidia-docker2
sudo systemctl restart docker.service
```

3. For each userid that will run docker, add the userid to the docker group:

```
sudo usermod -a -G docker <userid>
```

Users must log out and log back in to pick up this group change.

4. Verify the setup.

IBM Power

```
docker run --rm nvidia/cuda-ppc64le:10.2-base-ubuntu18.04 nvidia-smi
```

x86_64

```
docker run --rm nvidia/cuda nvidia-smi
```

Note:

The **docker run** command must be used with `docker-ce` (in other words, an Ubuntu host) to leverage the GPUs from within a container.

Installing IBM Maximo Visual Inspection stand-alone

You use the command line to install IBM Maximo Visual Inspection stand-alone.

IBM Maximo Visual Inspection stand-alone installation prerequisites

You must complete the following installation prerequisites steps before you install IBM Maximo Visual Inspection.

1. Complete all steps in the “Prerequisites for installing IBM Maximo Visual Inspection” on page 21 topic.
2. Your system must have a proper subscription and repository that provides you with updated packages. For information, see the [Red Hat Subscription Manager](#) documentation.
3. Turn on Extra Packages for Enterprise Linux (EPEL). For information, see the [EPEL](#) website.
4. If there was a previous version of the product installed but you do **NOT** wish to migrate the data, delete or move `/opt/powerai-vision/volume` (for PowerAI Vision) or `/opt/vision/volume` (for

IBM Visual Insights). This will ensure that data for the previous install, such as data sets and trained models, will not appear in the newer version of the product.



Attention: You cannot install IBM Maximo Visual Inspection stand-alone on the same system that has the following software installed:

- IBM Data Science Experience (DSX)
 - IBM Cloud Private
 - IBM Watson Studio Local Edition
 - Any other Kubernetes based applications
- [“Install IBM Maximo Visual Inspection from IBM Passport Advantage” on page 28](#)
 - [“Install IBM Maximo Visual Inspection trial mode” on page 29](#)

Install IBM Maximo Visual Inspection from IBM Passport Advantage

To install IBM Maximo Visual Inspection stand-alone, complete the following steps:

1. Download the product tar file from the [IBM Passport Advantage website](#).
2. (Optional) Verify the downloaded tar file by following the instructions in this topic: [“Verify the downloaded tar file” on page 33](#).
3. Unzip and untar the product tar file, and run the installation command for the platform you are installing on. The install files are extracted to `visual-inspection-arch-1.3.0.0-ppa`, where *arch* is x86 or ppc, depending on your platform.

RHEL

```
sudo yum install ./<file_name>.rpm
```

Ubuntu

```
sudo dpkg -i ./<file_name>.deb
```

4. Load the IBM Maximo Visual Inspection images from the directory that contains the extracted tar file. The user running the script must have Docker privileges:

```
sudo /opt/ibm/vision/bin/load_images.sh -f ./file_name.tar
```

Note: The installation process can take some time to complete.

5. (RHEL only) Open ports for the firewall to access IBM Maximo Visual Inspection by running this script:

```
sudo /opt/ibm/vision/sbin/firewall.sh
```

6. After the installation is complete, you can start IBM Maximo Visual Inspection by running this script:

```
sudo /opt/ibm/vision/bin/vision-start.sh
```

A user named `admin` is created with a password of `passw0rd`. For instructions to change these values, see [“Managing users” on page 119](#).

Note: The startup script will modify ownership and permissions on `/opt/ibm/vision/volume` so that the containers can run under a non-root ID and access the data.

You must read and accept the license agreement that is displayed before you can use IBM Maximo Visual Inspection.

It can take several minutes to start IBM Maximo Visual Inspection. To check the status of the startup process, run this script:

```
sudo /opt/ibm/vision/bin/kubect1.sh get pods
```

In the output from the **kubect1.sh get pods** script, you can verify which IBM Maximo Visual Inspection components are available by locating the Deployment section and identifying that the AVAILABLE column has a value of 1 for each component. The following is an example of the output from the **kubect1.sh get pods** script that shows all components are available:

```
$ /opt/ibm/vision/bin/kubect1.sh get pods
```

NAME	READY	STATUS	RESTARTS
AGE			
vision-cod-infer-ce5c3be1-491a-486c-81ae-2c49bdf17242-86cdtmv8q	1/1	Running	0
14h			
vision-edge-connector-7db99dbb7f-vbtpg	1/1	Running	0
38h			
vision-edge-mqttbroker-79cb65b865-rhr7w	1/1	Running	0
38h			
vision-elasticsearch-6b779bb5f5-9mnck	1/1	Running	0
38h			
vision-event-service-86c8c8cd6d-2wq5r	1/1	Running	0
38h			
vision-fpga-device-plugin-86htq	1/1	Running	0
38h			
vision-keycloak-58f7566896-nwph6	1/1	Running	0
38h			
vision-logstash-565d995764-hnnxg	1/1	Running	0
38h			
vision-mongodb-b59b56645-wlxgn	1/1	Running	0
38h			
vision-postgres-6c57856875-zlknm	1/1	Running	0
38h			
vision-service-687b85b97f-spg8n	1/1	Running	1
38h			
vision-taskanaly-6dc659c45c-t9rb2	1/1	Running	0
38h			
vision-ui-7d885944fc-x8tfc	1/1	Running	0
38h			
vision-video-microservice-8457664dc6-dcb2g	1/1	Running	0
38h			

After the application startup has completed and the user interface is available, it can be accessed at <https://hostname/visual-inspection/>, where *hostname* is the system on which you installed IBM Maximo Visual Inspection.

7. Install any available fix packs. For instructions see [“Getting fixes from Fix Central” on page 187](#).

Install IBM Maximo Visual Inspection trial mode

1. Go to the [IBM Maximo Visual Inspection Trial download site](#), then download the .tar file and the .rpm files as instructed.
2. Unzip and untar the product tar file, and run the installation command for the platform you are installing on.

RHEL

```
sudo yum install ./<file_name>.rpm
```

Ubuntu

```
sudo dpkg -i ./<file_name>.deb
```

The install files are extracted to `visual-insights-arch-1.3.0.0-ppa`, where *arch* is x86 or ppc, depending on your platform.

3. (Optional) Verify the downloaded tar file by following the instructions in this topic: [“Verify the downloaded tar file” on page 33](#).
4. Load the IBM Maximo Visual Inspection images from the directory that contains the extracted tar file. The user running the script must have Docker privileges:

```
sudo /opt/ibm/vision/bin/load_images.sh -f ./file_name.tar
```

Note: The installation process can take some time to complete.

5. (RHEL only) Open ports for the firewall to access IBM Maximo Visual Inspection by running this script:

```
sudo /opt/ibm/vision/sbin/firewall.sh
```

6. After the installation is complete, you can start IBM Maximo Visual Inspection by running this script:

```
sudo /opt/ibm/vision/bin/vision-start.sh
```

A user named admin is created with a password of passw0rd. For instructions to change these values, see [“Managing users”](#) on page 119.

Note: The startup script will modify ownership and permissions on /opt/ibm/vision/volume so that the containers can run under a non-root ID and access the data.

You must read and accept the license agreement that is displayed before you can use IBM Maximo Visual Inspection.

It can take several minutes to start IBM Maximo Visual Inspection. To check the status of the startup process, run this script:

```
sudo /opt/ibm/vision/bin/kubect1.sh get pods
```

In the output from the **kubect1.sh get pods** script, you can verify which IBM Maximo Visual Inspection components are available by locating the Deployment section and identifying that the AVAILABLE column has a value of 1 for each component. The following is an example of the output from the **kubect1.sh get pods** script that shows all components are available:

```
$ /opt/ibm/vision/bin/kubect1.sh get pods
NAME                                     READY   STATUS    RESTARTS
AGE
vision-cod-infer-ce5c3be1-491a-486c-81ae-2c49bdf17242-86cdtmv8q  1/1     Running   0
14h
vision-edge-connector-7db99dbb7f-vbtpg  1/1     Running   0
38h
vision-edge-mqttbroker-79cb65b865-rhr7w  1/1     Running   0
38h
vision-elasticsearch-6b779bb5f5-9mnck  1/1     Running   0
38h
vision-event-service-86c8c8cd6d-2wq5r  1/1     Running   0
38h
vision-fpga-device-plugin-86htq  1/1     Running   0
38h
vision-keycloak-58f7566896-nwph6  1/1     Running   0
38h
vision-logstash-565d995764-hnnxg  1/1     Running   0
38h
vision-mongodb-b59b56645-wlxgn  1/1     Running   0
38h
vision-postgres-6c57856875-zlknm  1/1     Running   0
38h
vision-service-687b85b97f-spg8n  1/1     Running   1
38h
vision-taskanaly-6dc659c45c-t9rb2  1/1     Running   0
38h
vision-ui-7d885944fc-x8tfc  1/1     Running   0
38h
vision-video-microservice-8457664dc6-dcb2g  1/1     Running   0
38h
```

After the application startup has completed and the user interface is available, it can be accessed at <https://hostname/visual-inspection/>, where *hostname* is the system on which you installed IBM Maximo Visual Inspection.

Related concepts

[Logging in to IBM Maximo Visual Inspection](#)

Follow these steps to log in to IBM Maximo Visual Inspection.

Upgrading IBM Maximo Visual Inspection

When upgrading to the latest version of IBM Maximo Visual Inspection, your data from the previous release will not be lost, as long as you are upgrading to the same type of install. For example; from the stand-alone version to the stand-alone version. However, you should undeploy all models before upgrading and redeploy after upgrade.

About this task

Important: Before following these steps, undeploy any deployed models.

Prior to upgrading, is recommended that you back up your environment by following the steps in this topic: [“Backing up IBM Maximo Visual Inspection” on page 121.](#)

Upgrade installation can be performed from IBM PowerAI Vision 1.1.5 or IBM Visual Insights installations.

1. Stop the current instance of the application by running the *stop* script for the product version:

IBM PowerAI Vision

```
sudo /opt/vision/bin/powerai_vision_stop.sh
```

IBM Visual Insights

```
sudo /opt/ibm/vision/bin/vision-stop.sh
```

2. Optionally uninstall the current version of IBM PowerAI Vision or IBM Visual Insights. Doing so will not remove your application data in `/opt/ibm/powerai-vision/volume` (for IBM PowerAI Vision) or `/opt/ibm/vision/volume` (for IBM Visual Insights).
3. Obtain and install IBM Maximo Visual Inspection 1.3.0:

Install IBM Maximo Visual Inspection from IBM Passport Advantage

- a. Download the product tar file from the [IBM Passport Advantage](#) website.
- b. (Optional) Verify the downloaded tar file by following the instructions in this topic: [“Verify the downloaded tar file” on page 33.](#)
- c. Unzip and untar the product tar file, and run the installation command for the platform you are installing on.

RHEL

```
sudo yum install ./<file_name>.rpm
```

Ubuntu

```
sudo dpkg -i ./<file_name>.deb
```

You will be prompted to accept the upgrade of the product if you are running an interactive install.

4. Load the IBM Maximo Visual Inspection images from the directory that contains the extracted tar file. The user running the script must have Docker privileges:

```
sudo /opt/ibm/vision/bin/load_images.sh -f ./file_name.tar
```

Note: The installation process can take some time to complete.

5. Install any available fix packs. For instructions see [“Getting fixes from Fix Central” on page 187.](#)
6. Run the migration script.

```
sudo /opt/ibm/vision/bin/vision-migrate-1.3.0.0.sh
```

7. After the migration completes successfully, start IBM Maximo Visual Inspection:

```
sudo /opt/ibm/vision/bin/vision-start.sh
```

Note: After the application is restarted, the URL to access the application is changed to `https://hostname/visual-inspection/`, where *hostname* is the system on which the product is installed. Redirects are also in place to ensure continuity for prior URLs, which were `https://hostname/visual-insights/` (if upgrading from 1.2.0) or `https://hostname/powerai-vision/` (if upgrading from 1.1.5 or earlier).

8. Remove the prior installation data from `/opt/powerai-vision`.

9. Redeploy trained models as necessary.

- a. Click **Models** from the menu.
- b. Select the model you want to deploy and click **Deploy**.
- c. Specify a name for the model, and for models that were trained with the **Optimized for speed (Tiny YOLO v2)** model, choose the accelerator to deploy to. You can choose GPU, CPU, or Xilinx FPGA - 16 bit (technology preview).
- d. Click **Deploy**. The **Deployed Models** page is displayed. When the model has been deployed, the status column displays **Ready**.
- e. Click the deployed model to get the API endpoint, to view details about the model, such as the owner and the accuracy, and to test other videos or images against the model.

Uninstalling IBM Maximo Visual Inspection stand-alone

You must uninstall IBM Maximo Visual Inspection stand-alone on your system, before you can install IBM Cloud Private, IBM Data Science Experience Local, or other Kubernetes-based applications.

To uninstall IBM Maximo Visual Inspection, complete the following steps:



Warning: If you run the following commands, all the data that you gathered is deleted. Export your data sets and models before you run the following commands.

1. Stop the current instance of IBM Maximo Visual Inspection by running the following script:

```
sudo /opt/ibm/vision/bin/vision-stop.sh
```

2. Remove previously installed images by running the following script:

```
sudo /opt/ibm/vision/bin/purge_image.sh 1.3.0.0
```

Optionally remove all product data by running the following script. This will remove data sets, models, and so on:

```
sudo /opt/ibm/vision/bin/purge_data.sh
```

3. Remove IBM Maximo Visual Inspection by running the following command:

- For RHEL:

```
sudo yum remove visual-inspection
```

- For Ubuntu:

```
sudo dpkg --remove visual-inspection
```

4. Delete the data directory by running the following command:

```
sudo rm -rf /opt/ibm/vision/
```

5. Verify that IBM Maximo Visual Inspection was uninstalled by running the following command:

- For RHEL:

```
rpm -q visual-inspection
```

- For Ubuntu:

```
dpkg -l visual-inspection
```

Verify the downloaded tar file

After downloading a tar file, you can optionally verify it by using the CISO code signing service or with the signing certificate authority.

1. Download the product public key, OCSP public key, and OCSP chain public key:

```
visual-inspect-1.3.0-key.pub
visual-inspect-ocsp-1.3.0-key.pub
vis-inspt-ocspchain-1.3.0-key.pub
```

2. Download the appropriate sig file for your product version and platform:

Table 6. Available signature files		
Version	Platform	File Name and description
IBM Maximo Visual Inspection	Power Systems	visual-inspect-ppc-1.3.0-ppa.sig IBM Maximo Visual Inspection 1.3.0 Power Signature File English
IBM Maximo Visual Inspection	x86	visual-inspect-x86-1.3.0-ppa.sig IBM Maximo Visual Inspection 1.3.0 x86 Signature File English
IBM Maximo Visual Inspection Edge	Power Systems	vis-inspct-edge-ppc-1.3.0-ppa.sig IBM Maximo Visual Inspection Edge 1.3.0 Power Signature File English
IBM Maximo Visual Inspection Edge	x86	vis-inspct-edge-x86-1.3.0-ppa.sig IBM Maximo Visual Inspection Edge 1.3.0 x86 Signature File English
Trial	Power Systems	visual-inspect-trial-ppc-1.3.0.0.sig
Trial	x86	visual-inspect-trial-x86-1.3.0.0.sig

3. Perform the verification:

- To verify the tar file by using the CISO code signing service, run the following command with the appropriate .sig and .tar file and ensure that the output is Verified OK:

```
openssl dgst -sha256 -verify key_file \
-signature sig_file tar_file
```

For example, if you downloaded the install package for Power Systems from Passport Advantage (PPA):

```
openssl dgst -sha256 -verify visual-inspect-1.3.0-key.pub \  
-signature visual-inspect-ppc-1.3.0-ppa.sig visual-inspect-ppc-1.3.0-ppa.tar
```

- The product's certificate validity can be verified using the Online Certificate Status Protocol (OCSP). Run the following command and ensure that the output includes Response verify OK:

```
openssl ocsp -no_nonce -issuer vis-inspt-ocspchain-1.3.0-key.pub -cert vis-inspt-  
ocspchain-1.3.0-key.pub -VAfile vis-inspt-ocspchain-1.3.0-key.pub -text -url http://  
ocsp.digicert.com -respout
```

For example:

```
# openssl ocsp -no_nonce -issuer vis-inspt-ocspchain-1.3.0-key.pub -cert vis-inspt-ocsp-1.3.0-  
key.pub -VAfile vis-inspt-ocspchain-1.3.0-key.pub -text -url http://ocsp.digicert.com -  
respout ocspout  
OCSP Request Data:  
...  
  
Response verify OK  
vis-inspt-ocsp-1.3.0-key.pub: good  
    This Update: Feb 27 18:19:14 2020 GMT  
    Next Update: Mar  5 18:19:14 2020 GMT
```

Chapter 6. Checking the application and environment

After installation of IBM Maximo Visual Inspection, you can check the status of the application and environment by using commands documented in these topics. The Kubernetes commands `helm.sh` and `kubect1.sh` are installed in the bin directory of the product install path. (default: `/opt/ibm/vision`).

Checking the application Docker images in standalone installation

Space limitations or Kubernetes garbage collection activities can result in IBM Maximo Visual Inspection Docker images not being available in the Docker repository on a system.

- [“Using Docker images to validate IBM Maximo Visual Inspection Docker image availability” on page 35](#)
- [“Loading missing images” on page 36](#)

Using Docker images to validate IBM Maximo Visual Inspection Docker image availability

When `load_images.sh` runs successfully, it indicates that the following images were successfully loaded:

```
$ sudo /opt/ibm/vision/bin/load_images.sh -f visual-inspection-images-x86-1.3.0.0.tar
[ INFO ] Waiting for docker loads to complete. This will take some time...
Loaded image: vision-helm:3.2.4
Loaded image: gcr.io/google_containers/hyperkube:v1.16.10
Loaded image: sys-powerai-vision-docker-local.artifactory.swg-devops.com/nginx-ingress-
controller:0.26.1
Loaded image: gcr.io/google_containers/etcd:3.3.15
Loaded image: nvidia/k8s-device-plugin:1.11
Loaded image: coredns/coredns:1.6.2
Loaded image: gcr.io/google_containers/pause:3.1
Loaded image: vision-fpga-device-plugin:1.3.0.0
Loaded image: vision-service:1.3.0.0
Loaded image: vision-dnn-edge:1.3.0.0
Loaded image: vision-dnn-microservices:1.3.0.0
Loaded image: vision-dnn-custom:1.3.0.0
Loaded image: vision-models:1.3.0.0
Loaded image: vision-keycloak:1.3.0.0
Loaded image: vision-usermgt:1.3.0.0
Loaded image: vision-event-service:1.3.0.0
Loaded image: vision-dnn-ssd:1.3.0.0
Loaded image: vision-dnn-yolov3:1.3.0.0
Loaded image: vision-edge-connector:1.3.0.0
Loaded image: postgres:9.6.18
Loaded image: vision-dnn-detectron:1.3.0.0
Loaded image: vision-video-microservice:1.3.0.0
Loaded image: vision-ui:1.3.0.0
Loaded image: vision-logstash:1.3.0.0
Loaded image: vision-mongodb:1.3.0.0
Loaded image: vision-taskanaly:1.3.0.0
Loaded image: vision-dnn-actiondetect:1.3.0.0
Loaded image: vision-preprocessing:1.3.0.0
Loaded image: vision-elasticsearch:1.3.0.0
Loaded image: vision-edge-mqttbroker:1.3.0.0
```

```
[ INFO ] SUCCESS> All images loaded successfully.
```

At any time, these images should also show in the output of `Docker images`:

```
$ docker
images
```

REPOSITORY	TAG	IMAGE ID	CREATED
vision-dnn-edge	1.3.0.0	d84709572f4d	2 days ago
vision-dnn-custom	1.3.0.0	93182056e0b6	2 days ago
vision-preprocessing	1.3.0.0	e9835b962e7d	2 days ago
vision-taskanaly	1.3.0.0	e9b9fa4aa0cf	2 days ago

2.08 GB			
vision-dnn-yolov3	1.3.0.0	18518682786a	2 days ago
13.2 GB			
vision-video-microservice	1.3.0.0	f68ce43c5994	2 days ago
3.45 GB			
vision-dnn-microservices	1.3.0.0	a360e2e0d102	2 days ago
5.82 GB			
vision-dnn-detection	1.3.0.0	25487d2e8916	2 days ago
4.62 GB			
vision-service	1.3.0.0	d5d7ed03edbb	2 days ago
428 MB			
vision-edge-connector	1.3.0.0	539ec7ceb0c9	2 days ago
12.9 MB			
vision-dnn-actiondetect	1.3.0.0	5c20f7062041	2 days ago
4.76 GB			
vision-models	1.3.0.0	f96091c60f1b	2 days ago
1.84 GB			
vision-edge-mqttbroker	1.3.0.0	3c51f411404d	7 days ago
14.2 MB			
vision-event-service	1.3.0.0	9f01fd70c919	7 days ago
82.7 MB			
vision-fpga-device-plugin	1.3.0.0	37a15fd89335	7 days ago
1.03 GB			
vision-elasticsearch	1.3.0.0	88cc7abf1cc6	8 days ago
978 MB			
vision-ui	1.3.0.0	eae409e4b8bc	8 days ago
250 MB			
vision-usermgt	1.3.0.0	2418c3c729b1	8 days ago
166 MB			
vision-mongodb	1.3.0.0	5cda10aa5e31	8 days ago
341 MB			
vision-logstash	1.3.0.0	03ddb2cf8ee5	8 days ago
837 MB			
vision-keycloak	1.3.0.0	62021973a36b	8 days ago
460 MB			
vision-dnn-ssd	1.3.0.0	0f6e351899c2	8 days ago
4.36 GB			
vision-helm	3.2.4	f4bafc7d21a1	4 weeks ago
48.7 MB			
postgres	9.6.18	51e37c2850c7	6 weeks ago
200 MB			
gcr.io/google_containers/hyperkube	v1.16.10	2eb6b85d7591	2 months ago
545 MB			
vision-helm	3.1.2	0162aec7e263	3 months ago
46.8 MB			
~/nginx-ingress-controller (see footnote 1)	0.26.1	29024c9c6e70	9 months ago
483 MB			
gcr.io/google_containers/etcd	3.3.15	b2756210eeab	10 months ago
247 MB			
coredns/coredns	1.6.2	bf261d157914	11 months ago
44.1 MB			

1. Full image name: `sys-powerai-vision-docker-local.artifactory.swg-devops.com/nginx-ingress-controller`

Loading missing images

If any of the IBM Maximo Visual Inspection Docker images are not available in the Docker repository, application failures can occur. In this case, run `load_images.sh` again to load any of the images that are missing.

Checking Kubernetes services status

The Kubernetes infrastructure is used to run the IBM Maximo Visual Inspection application. The **kubectl** command can be used to check the status of these underlying services, using the **--namespace kube-system** option.

- [“Using kubectl get pods to check kube-system” on page 36](#)
- [“Using kubectl describe pods to check kube-system” on page 37](#)

Using kubectl get pods to check kube-system

The **kubectl** command is used to show the detailed status of the Kubernetes pods deployed to run the IBM Maximo Visual Inspection application.

Example output

```
$ /opt/ibm/vision/bin/kubect1.sh get pods --namespace kube-system
NAME                                READY   STATUS    RESTARTS   AGE
coredns-f995d4d4f-nsfqt            1/1     Running   0           38h
nginx-ingress-lb-jwpbk             1/1     Running   0           38h
nvidia-device-plugin-daemonset-lq77n 1/1     Running   0           38h
```

Interpreting the output

- When the Kubernetes system is running correctly, each of the pods should have:
 - In the READY column all pods should be counted - for example, "1/1".
 - A value of "Running" in the STATUS column.
- A STATUS value other than "Running" indicates an issue with the Kubernetes infrastructure.
- A non-0, and growing, value in the RESTARTS column indicates an issue with that Kubernetes pod.

Using kubect1 describe pods to check kube-system

The `kubect1 describe pods` command provides detailed information about each of the pods that provide Kubernetes infrastructure. If the output from a specific pod is desired, run the command `kubect1 describe pod pod_name --namespace kube-system`.

Example output

The output from the command is verbose, so sample output from only one pod is shown:

```

ion/bin/kubect1.sh describe pods --namespace kube-system
Name:                coredns-f995d4d4f-nsfq
Namespace:           kube-system
Priority:             2000000000
Priority Class Name:  system-cluster-critical
Node:                127.0.0.1/127.0.0.1
Start Time:          Tue, 21 Jul 2020 08:27:18 -0500
Labels:              k8s-app=kube-dns
                    pod-template-hash=f995d4d4f
Annotations:         <none>
Status:              Running
IP:                  172.17.0.2
IPs:
  IP:                172.17.0.2
Controlled By:       ReplicaSet/coredns-f995d4d4f
Containers:
  coredns:
    Container ID:  docker://e81b331b90b17657a89f1cf9f8e668167a701ed59de35f3fd02e2415e2bc5a07
    Image:         coredns/coredns:1.6.2
    Image ID:      docker-pullable://docker.io/coredns/
coredns@sha256:12eb885b8685b1b13a04ecf5c23bc809c2e57917252fd7b0be9e9c00644e8ee5
    Ports:         53/UDP, 53/TCP, 9153/TCP
    Host Ports:    0/UDP, 0/TCP, 0/TCP
    Args:
      -conf
      /etc/coredns/Corefile
    State:         Running
      Started:     Tue, 21 Jul 2020 08:27:19 -0500
    Ready:         True
    Restart Count: 0
    Limits:
      memory:      170Mi
    Requests:
      cpu:         100m
      memory:      70Mi
    Liveness:      http-get http://:8080/health delay=60s timeout=5s period=10s #success=1
#failure=5
    Readiness:     http-get http://:8181/ready delay=0s timeout=1s period=10s #success=1
#failure=3
    Environment:   <none>
    Mounts:
      /etc/coredns from config-volume (ro)
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-rsr78 (ro)
Conditions:
  Type                 Status
  Initialized          True
  Ready                True
  ContainersReady      True
  PodScheduled         True
Volumes:
  config-volume:
    Type:              ConfigMap (a volume populated by a ConfigMap)
    Name:              coredns
    Optional:          false
  default-token-rsr78:
    Type:              Secret (a volume populated by a Secret)
    SecretName:        default-token-rsr78
    Optional:          false
QoS Class:            Burstable
Node-Selectors:       kubernetes.io/os=linux
Tolerations:          CriticalAddonsOnly
                    node.kubernetes.io/not-ready:NoExecute for 300s
                    node.kubernetes.io/unreachable:NoExecute for 300s
Events:               <none>

```

Interpreting the output

Significant fields providing status of the Kubernetes pods include:

- The **Status** field should be "Running" - any other status will indicate issues with the environment.
- In the **Conditions** section, the **Ready** field should indicate "True". Any other value indicates that there are issues with the environment.
- If there are issues with any pods, the **Events** section of the pod should have information about issues the pod encountered.

Checking Kubernetes node status

Use these commands to check the status of the nodes in the environment.

- [“kubectl get nodes command” on page 39](#)
- [“kubectl describe nodes command” on page 39](#)

kubectl get nodes command

The `kubectl get nodes` command shows the summary status of the node where IBM Maximo Visual Inspection is deployed.

Example output

```
$ /opt/ibm/vision/bin/kubectl.sh get nodes
NAME          STATUS    ROLES    AGE   VERSION
127.0.0.1     Ready     <none>   38h   v1.16.10
```

Interpreting the output

- STATUS displays "Ready" if the application is deployed and started successfully.
- VERSION displays the Kubernetes level that is deployed to run the IBM Maximo Visual Inspection application.

kubectl describe nodes command

The `kubectl describe nodes` command provides status information regarding the Kubernetes environment used to run the IBM Maximo Visual Inspection application.

Example output

```

$ /opt/ibm/vision/bin/kubect1.sh describe nodes
Name: 127.0.0.1
Roles: <none>
Labels: beta.kubernetes.io/arch=amd64
        beta.kubernetes.io/os=linux
        kubernetes.io/arch=amd64
        kubernetes.io/hostname=127.0.0.1
        kubernetes.io/os=linux
Annotations: node.alpha.kubernetes.io/ttl: 0
              volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp: Tue, 21 Jul 2020 08:27:07 -0500
Taints: <none>
Unschedulable: false
Conditions:
  Type           Status  LastHeartbeatTime             LastTransitionTime
  Reason          -----  -
  MemoryPressure  False   Wed, 22 Jul 2020 22:38:18 -0500 Tue, 21 Jul 2020 08:27:07 -0500
  KubeletHasSufficientMemory kubelet has sufficient memory available
  DiskPressure    False   Wed, 22 Jul 2020 22:38:18 -0500 Tue, 21 Jul 2020 08:27:07 -0500
  KubeletHasNoDiskPressure kubelet has no disk pressure
  PIDPressure     False   Wed, 22 Jul 2020 22:38:18 -0500 Tue, 21 Jul 2020 08:27:07 -0500
  KubeletHasSufficientPID kubelet has sufficient PID available
  Ready           True    Wed, 22 Jul 2020 22:38:18 -0500 Tue, 21 Jul 2020 08:27:17 -0500
KubeletReady
Addresses:
  InternalIP: 127.0.0.1
  Hostname: 127.0.0.1
Capacity:
  cpu: 40
  ephemeral-storage: 951598Mi
  hugepages-1Gi: 0
  hugepages-2Mi: 0
  memory: 65409356Ki
  nvidia.com/gpu: 2
  pods: 400
Allocatable:
  cpu: 40
  ephemeral-storage: 946478Mi
  hugepages-1Gi: 0
  hugepages-2Mi: 0
  memory: 65306956Ki
  nvidia.com/gpu: 2
  pods: 400
System Info:
  Machine ID: 311936e2303b034fe7ef70182235b8cb
  System UUID: 00000000-0000-0000-0000-AC1F6B5AD1FE
  Boot ID: 981cf71f-b0f2-4283-b7aa-8f210e704bbf
  Kernel Version: 3.10.0-1127.8.2.el7.x86_64
  OS Image: Debian GNU/Linux 9 (stretch)
  Operating System: linux
  Architecture: amd64
  Container Runtime Version: docker://1.13.1
  Kubelet Version: v1.16.10
  Kube-Proxy Version: v1.16.10
Non-terminated Pods: (17 in total)
  Namespace           Name
  -----
CPU Requests  CPU Limits  Memory Requests  Memory Limits  AGE
-----
default vision-cod-infer-ce5c3be1-491a-486c-81ae-2c49bdf17242-86cdtmv8q 0
(0%) 0 (0%) 0 (0%) 0 (0%) 14h
default vision-edge-connector-7db99dbb7f-vbtpg 0
(0%) 0 (0%) 0 (0%) 0 (0%) 38h
default vision-edge-mqttbroker-79cb65b865-rhr7w 0
(0%) 0 (0%) 0 (0%) 0 (0%) 38h
default vision-elasticsearch-6b779bb5f5-9mnck 0
(0%) 0 (0%) 0 (0%) 0 (0%) 38h
default vision-event-service-86c8c8cd6d-2wq5r 0
(0%) 0 (0%) 0 (0%) 0 (0%) 38h
default vision-fpga-device-plugin-86htq 0
(0%) 0 (0%) 0 (0%) 0 (0%) 38h
default vision-keycloak-58f7566896-nwph6 0
(0%) 0 (0%) 0 (0%) 0 (0%) 38h
default vision-logstash-565d995764-hnnxg 0
(0%) 0 (0%) 0 (0%) 0 (0%) 38h
default vision-mongodb-b59b56645-wlxgn 0
(0%) 0 (0%) 0 (0%) 0 (0%) 38h
default vision-postgres-6c57856875-zlknm 0
(0%) 0 (0%) 0 (0%) 0 (0%) 38h
default vision-service-687b85b97f-spq8n 0
(0%) 0 (0%) 0 (0%) 0 (0%) 38h
default vision-taskanalyzer-6dc659c45c-t9rb2 0
(0%) 0 (0%) 0 (0%) 0 (0%) 38h
default vision-ui-7d885944fc-x8tfc 0
(0%) 0 (0%) 0 (0%) 0 (0%) 38h

```


Interpreting the output

- Most of the information is informational regarding the system resources (CPUs, GPUs, memory) and version information (OS, Docker, Kubernetes).
- The **Conditions** section can indicate whether there are system resource issues that will affect the running of the application. For example, if any of the **OutOfDisk**, **MemoryPressure**, or **DiskPressure** conditions are **True**, there are insufficient system resources to run IBM Maximo Visual Inspection. For example, the following **Conditions** section shows a system that does not have sufficient disk space available, indicated by **DiskPressure** status of **True**:

```
Conditions:
  Type           Status  LastHeartbeatTime   LastTransitionTime
  Reason          -----  -
  OutOfDisk       False   [...]              [...]
  KubeletHasSufficientDisk  kubelet has sufficient disk space available
  MemoryPressure  False   [...]              [...]
  KubeletHasSufficientMemory kubelet has sufficient memory available
  DiskPressure    True    [...]              [...]
  KubeletHasDiskPressure  kubelet has disk pressure
  Ready          True    [...]              [...]
  KubeletReady                    kubelet is posting ready status
```

- The **Events** section will also have messages that can indicate if there are issues with the environment. For example, the following events indicate issues with disk space that have led to Kubernetes attempting to reclaim resources ("eviction") which can affect the availability of Kubernetes applications:

```
Events:
  Type           Reason          Age         From          Message
  ----           -
  Normal         NodeHasDiskPressure  5m          kubelet, 127.0.0.1  Node 127.0.0.1 status is
now: NodeHasDiskPressure
  Warning        EvictionThresholdMet  3s (x23 over 5m)  kubelet, 127.0.0.1  Attempting to reclaim
nodefs
```

Checking Kubernetes storage status

The IBM Maximo Visual Inspection application requires disk storage for activities including data set storage. The disk space requirements are described using Kubernetes Persistent Volume configuration. The `kubectl` command can be used to examine the `pv` (PersistentVolume) and `pvc` (PersistentVolumeClaims) resources.

Note: The storage requirements described in the **PersistentVolume** and **PersistentVolumeClaims** are not enforced in the standalone deployment. Therefore, the requested space might not be available in the underlying storage of the system. See [“Disk space requirements” on page 16](#) for information about product storage requirements.

- [“Using kubectl get pv and pvc commands” on page 41](#)
- [“Using the kubectl describe pv command” on page 42](#)
- [“Using the kubectl describe pvc command” on page 42](#)

Using kubectl get pv and pvc commands

The `kubectl get pv` and `kubectl get pvc` commands can be used to see what PersistentVolume and PersistentVolumeClaim have been defined for the application.

Example output

```
# /opt/ibm/vision/bin/kubectl.sh get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM
STORAGECLASS  REASON    AGE
vision-data   40Gi     RWX           Retain          Bound  default/vision-data-
pvc          14d
```

```
# /opt/ibm/vision/bin/kubect1.sh get pvc
NAME                STATUS    VOLUME    CAPACITY    ACCESS MODES    STORAGECLASS    AGE
vision-data-pvc     Bound    vision-data  40Gi        RWX              vision-data     14d
```

Interpreting the output

The above output shows information about the Persistent Volume and Persistent Volume Claim for IBM Maximo Visual Inspection. The application currently has a capacity claim of 40G and it is successfully "Bound". If the **STATUS** is not "Bound", the application does not have access to the necessary storage.

Using the kubect1 describe pv command

The `kubect1 describe pv` command is used to see detailed information about the Persistent Volume used by the application.

Example output

```
# /opt/ibm/vision/bin/kubect1.sh describe pv
Name:                vision-data
Labels:              assign-to=vision-data
                    type=local
Annotations:         pv.kubernetes.io/bound-by-controller: yes
Finalizers:          [kubernetes.io/pv-protection]
StorageClass:
Status:              Bound
Claim:               default/vision-data-pvc
Reclaim Policy:      Retain
Access Modes:        RWX
VolumeMode:          Filesystem
Capacity:             40Gi
Node Affinity:       <none>
Message:
Source:
  Type:               HostPath (bare host directory volume)
  Path:               /opt/ibm/vision/volume/
  HostPathType:
Events:              <none>
```

Interpreting the output

The above output shows more details about the Persistent Volume used by the application. The **Source** section has the critical configuration values for **Type** and **Path**. The **Events** section will have information about Error events if there were issues with the Persistent Volume.

Using the kubect1 describe pvc command

The `kubect1 describe pvc` command is used to see detailed information about the Persistent Volume Claim for the application.

Example output

```

$ /opt/ibm/vision/bin/kubect1.sh describe pvc
Name:          vision-data-pvc
Namespace:     default
StorageClass:
Status:     Bound
Volume:     vision-data
Labels:        app=vision
               app.kubernetes.io/managed-by=Helm
               chart=ibm-visual-inspection-prod-3.0.0
               heritage=Helm
               release=vision
Annotations:   meta.helm.sh/release-name: vision
               meta.helm.sh/release-namespace: default
               pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      40Gi
Access Modes:   RWX
VolumeMode:     Filesystem
Mounted By:     vision-cod-infer-ce5c3be1-491a-486c-81ae-2c49bdf17242-86cdtmv8q
               vision-edge-connector-7db99dbb7f-vbtpg
               vision-edge-mqttbroker-79cb65b865-rhr7w
               vision-elasticsearch-6b779bb5f5-9mnck
               vision-event-service-86c8c8cd6d-2wq5r
               vision-logstash-565d995764-hnnxg
               vision-mongodb-b59b56645-wlxgn
               vision-postgres-6c57856875-zlknm
               vision-service-687b85b97f-spg8n
               vision-taskanaly-6dc659c45c-t9rb2
               vision-ui-7d885944fc-x8tfc
               vision-video-microservice-8457664dc6-dcb2g
Events:     <none>

```

Interpreting the output

The above output shows more details about the Persistent Volume Claim used by the application. The **Volume** section references the underlying Persistent Volume, and the **Status** should be "Bound" if it has been successfully allocated to the application. The **Events** section will show if there were issues with the Persistent Volume Claim.

Checking application deployment

IBM Maximo Visual Inspection processes require a Kubernetes environment. Use these commands to verify that the Kubernetes environment was deployed correctly and that all nodes are configured appropriately.

- [“kubect1.sh get pods” on page 43](#)
- [“kubect1 describe pods” on page 44](#)
- [“kubect1 get deployment” on page 46](#)
- [“kubect1 describe deployment” on page 46](#)

kubect1.sh get pods

The `kubect1.sh get pods` command shows the status of the full Kubernetes environment of the IBM Maximo Visual Inspection application.

Example output

```
$ /opt/ibm/vision/bin/kubect1.sh get pods
```

NAME	READY	STATUS	RESTARTS	
vision-cod-infer-ce5c3be1-491a-486c-81ae-2c49bdf17242-86cdtmv8q	1/1	Running	0	
vision-edge-connector-7db99dbb7f-vbtpg	1/1	Running	0	
vision-edge-mqttbroker-79cb65b865-rhr7w	1/1	Running	0	
vision-elasticsearch-6b779bb5f5-9mnck	1/1	Running	0	
vision-event-service-86c8c8cd6d-2wq5r	1/1	Running	0	
vision-fpga-device-plugin-86htq	1/1	Running	0	
vision-keycloak-58f7566896-nwph6	1/1	Running	0	
vision-logstash-565d995764-hnnxg	1/1	Running	0	
vision-mongodb-b59b56645-wlxgn	1/1	Running	0	
vision-postgres-6c57856875-zlknm	1/1	Running	0	
vision-service-687b85b97f-spg8n	1/1	Running	1	
vision-taskanaly-6dc659c45c-t9rb2	1/1	Running	0	
vision-ui-7d885944fc-x8tfc	1/1	Running	0	
vision-video-microservice-8457664dc6-dcb2g	1/1	Running	0	38h

Important fields in the output

STATUS

The value for STATUS should be DEPLOYED after a successful installation.

RESOURCES

The status of individual Kubernetes pods is displayed in this section. The CURRENT and AVAILABLE values for each pod should be equal to or greater than the DESIRED value.

```
RESOURCES:
==> v1beta1/Deployment
NAME                                DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
...
vision-portal                        1        1        1            1         14d
...
```

kubect1 describe pods

The `kubect1.sh describe pods` command shows the detailed information about the Kubernetes pods of the IBM Maximo Visual Inspection application.

Example output

```

$ /opt/ibm/vision/bin/kubect1.sh describe pods
...
Name:          vision-ui-7d885944fc-x8tfc
Namespace:     default
Priority:       0
Node:          127.0.0.1/127.0.0.1
Start Time:    Tue, 21 Jul 2020 08:27:21 -0500
Labels:        app=vision
               chart=ibm-visual-inspection-prod-3.0.0
               component=vision-ui
               heritage=Helm
               pod-template-hash=7d885944fc
               release=vision
               run=vision-ui-deployment-pod
Annotations:   checksum/config: 2e6be9c00aae84e873b79089b7dc6625b3dbaaa9e09607994a8133cd7cbec0fd
               productID: 5737-H10
               productName: IBM Maximo Visual Inspection
               productVersion: 1.3.0.0
Status:        Running
IP:            172.17.0.13
IPs:
  IP:          172.17.0.13
Controlled By: ReplicaSet/vision-ui-7d885944fc
Containers:
  vision-ui:
    Container ID:  docker://38a84e7171c40824b1ecabfbc2853b58f723733dff490b7cca8186619a2f7256
    Image:         vision-ui:1.3.0.0
    Image ID:      docker://
sha256:eae409e4b8bc551ef0ff7aa98e46a46ea6a9540e0a53cc660f6c04143c5f7c49
    Port:          8080/TCP
    Host Port:     0/TCP
    State:         Running
      Started:     Tue, 21 Jul 2020 08:27:59 -0500
    Ready:         True
    Restart Count: 0
    Liveness:      http-get http://:http/visual-inspection/index.html delay=240s timeout=5s
period=10s #success=1 #failure=3
    Readiness:     http-get http://:http/visual-inspection/index.html delay=5s timeout=1s
period=10s #success=1 #failure=3
    Environment:
      CONTEXT_ROOT:    <set to the key 'CONTEXT_ROOT' of config map 'vision-
config'> Optional: false
      DLAAS_API_SERVER: <set to the key 'DLAAS_API_SERVER' of config map 'vision-
config'> Optional: false
      SERVER_HOST_VIDEO_TEST: <set to the key 'SERVER_HOST_VIDEO_TEST' of config map 'vision-
config'> Optional: false
      SERVICE_PORT_VIDEO_TEST: <set to the key 'SERVICE_PORT_VIDEO_TEST' of config map 'vision-
config'> Optional: false
      WEBROOT_VIDEO_TEST: <set to the key 'WEBROOT_VIDEO_TEST' of config map 'vision-
config'> Optional: false
    Mounts:
      /opt/powerai-vision/data from data-mount (rw,path="/data")
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-qmqgr (ro)
Conditions:
  Type            Status
  Initialized      True
  Ready            True
  ContainersReady  True
  PodScheduled     True
Volumes:
  data-mount:
    Type: PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same
namespace)
    ClaimName: vision-data-pvc
    ReadOnly: false
  default-token-qmqgr:
    Type: Secret (a volume populated by a Secret)
    SecretName: default-token-qmqgr
    Optional: false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute for 300s
                  node.kubernetes.io/unreachable:NoExecute for 300s
Events:          <none>

```

Interpreting the output

- Information about the product name and version are given in `productName` and `productVersion`.
- Image and Image ID provide details about the Docker container for the pod.

- The State field should be Running. Any other status indicates problems with the application pod.
- If there are issues with a pod, the Events section of the pod should have information about problems encountered.

kubectl get deployment

The `kubectl get deployment` command shows summary information about the active deployments in the environment. This includes dynamically started pods used for training and deployment of models.

Example output

```
$ /opt/ibm/vision/bin/kubectl.sh get deployment
NAME                                READY    UP-TO-DATE    AVAILABLE    AGE
vision-cod-infer-ce5c3be1-491a-486c-81ae-2c49bdf17242  1/1      1              1             14h
vision-edge-connector                1/1      1              1             38h
vision-edge-mqttbroker               1/1      1              1             38h
vision-elasticsearch                 1/1      1              1             38h
vision-event-service                 1/1      1              1             38h
vision-keycloak                      1/1      1              1             38h
vision-logstash                      1/1      1              1             38h
vision-mongodb                       1/1      1              1             38h
vision-postgres                      1/1      1              1             38h
vision-service                       1/1      1              1             38h
vision-taskanaly                     1/1      1              1             38h
vision-ui                            1/1      1              1             38h
vision-video-microservice            1/1      1              1             38h
```

kubectl describe deployment

The `kubectl describe deployment` command provides verbose status information about each of the deployed nodes in the Kubernetes environment that is being used to run IBM Maximo Visual Inspection.

Example output

The following shows the output from one of the nodes. The full output for all nodes is much longer and has similar entries for each node.

```

$ /opt/ibm/vision/bin/kubect1.sh describe deployment
...
Name: vision-ui
Namespace: default
CreationTimestamp: Tue, 21 Jul 2020 08:27:21 -0500
Labels: app=vision
        app.kubernetes.io/managed-by=Helm
        chart=ibm-visual-inspection-prod-3.0.0
        heritage=Helm
        release=vision
        run=vision-ui-deployment
Annotations: deployment.kubernetes.io/revision: 1
             meta.helm.sh/release-name: vision
             meta.helm.sh/release-namespace: default
Selector: run=vision-ui-deployment-pod
Replicas: 1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType: RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels: app=vision
          chart=ibm-visual-inspection-prod-3.0.0
          component=vision-ui
          heritage=Helm
          release=vision
          run=vision-ui-deployment-pod
  Annotations: checksum/config: 2e6be9c00aae84e873b79089b7dc6625b3dbaaa9e09607994a8133cd7cbec0fd
              productID: 5737-H10
              productName: IBM Maximo Visual Inspection
              productVersion: 1.3.0.0
  Containers:
    vision-ui:
      Image: vision-ui:1.3.0.0
      Port: 8080/TCP
      Host Port: 0/TCP
      Liveness: http-get http://:http/visual-inspection/index.html delay=240s timeout=5s
period=10s #success=1 #failure=3
      Readiness: http-get http://:http/visual-inspection/index.html delay=5s timeout=1s period=10s
#success=1 #failure=3
      Environment:
        CONTEXT_ROOT: <set to the key 'CONTEXT_ROOT' of config map 'vision-
config'> Optional: false
        DLAAS_API_SERVER: <set to the key 'DLAAS_API_SERVER' of config map 'vision-
config'> Optional: false
        SERVER_HOST_VIDEO_TEST: <set to the key 'SERVER_HOST_VIDEO_TEST' of config map 'vision-
config'> Optional: false
        SERVICE_PORT_VIDEO_TEST: <set to the key 'SERVICE_PORT_VIDEO_TEST' of config map 'vision-
config'> Optional: false
        WEBROOT_VIDEO_TEST: <set to the key 'WEBROOT_VIDEO_TEST' of config map 'vision-
config'> Optional: false
      Mounts:
        /opt/powerai-vision/data from data-mount (rw,path="data")
  Volumes:
    data-mount:
      Type: PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same
namespace)
      ClaimName: vision-data-pvc
      ReadOnly: false
  Conditions:
    Type          Status    Reason
    ----          -
    Available     True     MinimumReplicasAvailable
    Progressing   True     NewReplicaSetAvailable
OldReplicaSets: <none>
NewReplicaSet: vision-ui-7d885944fc (1/1 replicas created)
Events: <none>

```

Interpreting the output

- The **Replicas** line shows information regarding how many images are desired and available (similar to the output from `kubect1 get pods`):

Replicas: 1 desired | 1 updated | 1 total | 1 available | 0 unavailable

The "available" value should be equal to the "desired" value.

- The **productVersion** value indicates the level of IBM Maximo Visual Inspection installed:

productVersion: 1.3.0.0

- The **Image** value provides information about the Docker container:

Image: vision-ui:1.3.0.0

- The **Conditions** section has important information about the current status of the image, and any reasons if the status is "failure".

Checking system GPU status

In IBM Maximo Visual Inspection, GPUs are used to train and deploy models. Use these commands to verify that GPUs are set up and available.

`nvidia-smi`

The `nvidia-smi` command is a NVIDIA utility, installed with the CUDA toolkit. For details, see [“Prerequisites for installing IBM Maximo Visual Inspection” on page 21](#). With `nvidia-smi`, you can view the status of the GPUs on the system.

Example output

```
# nvidia-smi
Fri Mar 15 12:23:50 2019
```

NVIDIA-SMI 418.29 Driver Version: 418.29 CUDA Version: 10.1									
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC		
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute	M.		
0	Tesla P100-SXM2...	On	00000002:01:00.0	Off			0		
N/A	50C	P0	109W / 300W	2618MiB / 16280MiB	43%		Default		
1	Tesla P100-SXM2...	On	00000003:01:00.0	Off			0		
N/A	34C	P0	34W / 300W	0MiB / 16280MiB	0%		Default		
2	Tesla P100-SXM2...	On	0000000A:01:00.0	Off			0		
N/A	48C	P0	44W / 300W	5007MiB / 16280MiB	0%		Default		
3	Tesla P100-SXM2...	On	0000000B:01:00.0	Off			0		
N/A	36C	P0	33W / 300W	0MiB / 16280MiB	0%		Default		

Processes:					GPU Memory Usage
GPU	PID	Type	Process name		
0	114476	C	/opt/miniconda2/bin/python		2608MiB
2	114497	C	/opt/miniconda2/bin/python		958MiB
2	114519	C	/opt/miniconda2/bin/python		958MiB
2	116655	C	/opt/miniconda2/bin/python		2121MiB
2	116656	C	/opt/miniconda2/bin/python		958MiB

Interpreting the output

The above output shows the following:

- The system has 4 (0-3) **Tesla P100** GPUs.
- In the last portion of the output, it shows that GPU **0** has a process deployed and running. This can indicate a IBM Maximo Visual Inspection training task or a deployed model. Any GPUs with running jobs are not available for training jobs or deployment of trained models from the user interface. The output also shows multiple processes running on GPU 2, which can indicate that multiple models deployed for inferencing are sharing that GPU resource.
- The output should correctly display the memory configuration of the GPUs. For example, "Unknown error" indicates an issue with the driver setup or configuration. See [“GPUs are not available for training or inference” on page 173](#) for more information.

Checking GPU availability

Use the Nvidia CUDA containers to validate whether the container can access the GPU independently of IBM Maximo Visual Inspection.

If the Nvidia samples cannot run in a container, then the GPUs are not accessible to IBM Maximo Visual Inspection.

Overview

This validation requires you to run two checks:

- Run a GPU check in the container.
- Run a GPU check on the server system.

After you run the two checks to determine their individual status, proceed to the [“Problem resolution” on page 50](#) section.

Running GPU checks in the container

This check runs the following processes:

- The Nvidia CUDA container is run.
- The Nvidia CUDA runtime, which has samples that exercise the GPUs, is installed.
- A sample that uses the GPUs is run to validate GPU availability.

Important: This check requires internet connectivity.

Follow these steps to check the GPU in the container:

1. Run the Nvidia CUDA container:

On Power servers

```
$ docker run -it nvidia/cuda-ppc64le:10.2 /bin/bash
```

On x86 servers

```
$ docker run -it nvidia/cuda:10.2 /bin/bash
```

2. Install the CUDA 10.2 runtime:

```
# apt update; apt install cuda-10.2
```

The output for this command is omitted.

3. Make the deviceQuery sample:

```
# cd /usr/local/cuda-10.2/samples/1_Uutilities/deviceQuery
# make
```

4. Run the deviceQuery sample, and confirm that all system GPUs are visible and usable.

```
root@72642419dd52:/usr/local/cuda-10.2/samples/1_Uutilities/deviceQuery# ./deviceQuery
./deviceQuery Starting...

  CUDA Device Query (Runtime API) version (CUDA static linking)

Detected 4 CUDA Capable device(s)

Device 0: "Tesla P100-SXM2-16GB"
  CUDA Driver Version / Runtime Version      11.0 / 10.2
  ...
```

The information that is returned indicates the number of GPUs that are usable within a container.

These steps can also be used to check whether the GPUs are usable from the system, by installing CUDA Toolkit on the server.

Running GPU checks on the server system

If the previous check identifies issues at the container level, then check whether the GPUs are usable by applications that run directly on the host system. This check helps you determine whether the problem is with the container runtime or the GPU availability on the host.

Follow these steps to check the GPU availability on the host:

1. Install Nvidia CUDA runtime for the host system:
 - a. Go to the [CUDA Toolkit 10.2 Download](#) website.
 - b. Select the options for your architecture, distribution, and version.
 - c. Select an installer type of **rpm (local)**.
 - d. Download and install Nvidia CUDA runtime.
2. Make the deviceQuery sample:

```
# cd /usr/local/cuda-10.2/samples/1_Uutilities/deviceQuery
# make
```

3. Run the deviceQuery sample:

```
/usr/local/cuda-10.2/samples/1_Uutilities/deviceQuery# ./deviceQuery
./deviceQuery Starting...
...
```

Problem resolution

The resolution depends on the outcome of the two checks you ran.

Scenario 1: The issue with GPU usage exists only at the container level

This scenario indicates that the container runtime is not installed correctly.

Check the Nvidia installation instructions of the required Nvidia container runtime for your distribution:

- [Installing on Ubuntu](#)
- [Installing on RHEL 7](#)

Important: For x86 servers, install the `nvidia-container-toolkit` package and its dependencies. For POWER servers (ppc64le), install the `nvidia-container-hook` package and its dependencies.

Scenario 2: The issue with GPU usage exists only on the host

For this scenario, check the following items:

- Make sure that the required Nvidia drives are installed. For more information, see the [Nvidia installation guide](#) website.
- If SELinux is enabled on a Red Hat system, disable it to see whether it is causing the problem. Disable SELinux by running the following command: `setenforce 0`
- For Power Systems, review the blog entry, [What to do with “cudaSuccess \(3 versus 0\) initialization error” on a POWER9 system?](#)

Chapter 7. Logging in to IBM Maximo Visual Inspection

Follow these steps to log in to IBM Maximo Visual Inspection.

Note: IBM Maximo Visual Inspection is supported on these browsers:

- Google Chrome Version 60, or later
- Firefox Quantum 59.0, or later

1. Enter the IBM Maximo Visual Inspection URL in a supported browser:

`https://hostname/visual-inspection/`, where *hostname* is the system on which you installed IBM Maximo Visual Inspection.

2. Enter your user name and password. A default user name (admin) and password (passw0rd) was created at install time. For instructions to change these values, see [“Managing users” on page 119](#).

Related concepts

Managing users

There are two kinds of users in IBM Maximo Visual Inspection: administrators, and everyone else. The way you work with users and passwords differs, depending on how IBM Maximo Visual Inspection is installed.

Chapter 8. Working with the user interface

The IBM Maximo Visual Inspection user interface is made up of these basic parts: the navigation bar, the side bar, the action bar, the data area, and the notification center.

Interface areas

The user interface is made up of several different areas:

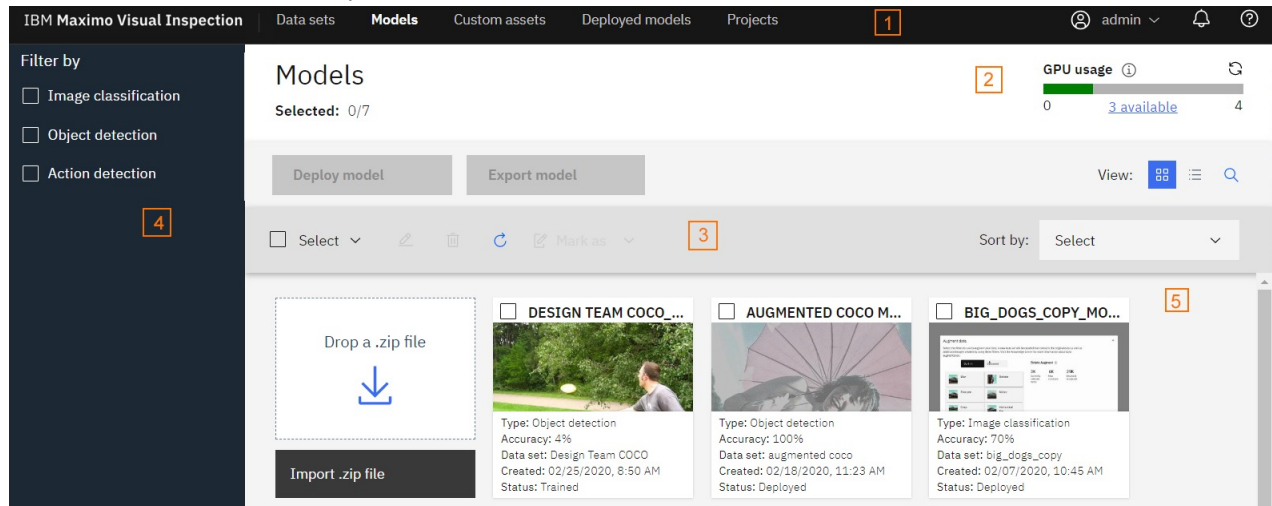


Figure 7. IBM Maximo Visual Inspection user interface

1: The navigation bar

The navigation bar lets you access the notification area (the bell icon), work with your profile, or access the IBM Maximo Visual Inspection Knowledge Center (the question mark icon). If you click the arrow by your user name, you can log out, view your usage metrics, and can see the version number of the product.

2: The header bar

The header bar on the Training, Models, Model details, and Deployed Models pages shows GPU usage details in these categories:

Training

GPUs currently used for training jobs by IBM Maximo Visual Inspection.

Deployed Models

GPUs currently used for deployed models by IBM Maximo Visual Inspection.

External

External GPUs are those that are used for processes outside of IBM Maximo Visual Inspection.

Note: If the output shows "Unknown", then GPUs are in use, but not for IBM Maximo Visual Inspection training or deployment. This either indicates an issue with a GPU in use by a training or deploy job that failed unexpectedly, or there are other applications on the system using GPUs. This could lead to unexpected resource contention and application issues.

3: The action bar

This is where you find the actions that you can take on images, videos, data sets, and models in the current data area. The available actions differ depending on what type of object you are working with.

4: The side bar

Data sets and models have a side bar with filtering options. Filtering helps you specify which objects to include in the data area.

Navigating: If the side bar is long, for example, if you have a data set with a lot of different types of objects, you can scroll through the side bar content. To scroll, hover over the appropriate content and

use your mouse roller or keyboard arrow keys. If the mouse pointer is right over the categories, for example, scrolling moves you through that list. If the mouse pointer is further to the right, on the edge of the side bar, scrolling moves you through all of the content on the side bar.

5: The data area

This is where you find the objects that you can act on. It lists the objects of the selected type, or displays the data included in the data set.

Filtering

With large data sets, you might need to filter the files that are shown in the data area. By default, your whole data set is shown.

Filter by

When you deselect a file type, those files are no longer shown in the data area. Therefore, if you only have Images selected, only images are shown in the data area.

Categories / objects

When you select categories, objects, or both, *all* files of the specified type that belong to any of the selected categories, or contain the selected objects, are shown.

For example, assume you have a data set with two categories: Cats and Dogs. Also assume that you tagged these types of objects: Face, Collar, and Tail. Then if you select Images, the category Dogs, and the object Collar, you will see all images that are dogs *or* contain a collar. This will include images of cats if they have a collar as well as images of dogs with no collar.

Using filtering and "Select all" with video data

When you capture frames from a video, these frames always maintain a child / parent relationship with the original video. That has some selection and filtering implications.

- When using the filter on the side bar, if *any* video frame matches the filter criteria, both the frame and its parent video are selected and are shown in the data area.
- If you click the "Select" box in the action bar, everything in the data area is selected. Therefore, if there is a video shown in the data area, it, and all of its child frames, are selected. Any action performed in this situation applies to all selected images, the video, and all of its child frames.

Example

A user has captured 50 frames from a video file Cars Video. Fourteen frames of the 50 have no labels.

1. The user selects **Unlabeled** in the Objects filter in the sidebar. The 14 frames with no labels and their parent video, Cars Video, are shown in the data area.
2. The user clicks **Select** in the action bar. The frames and the video are all selected.
3. The user clicks the trash can icon, intending to delete the unlabeled frames. However, because the video was selected, it, and the 36 labeled frames, are also deleted.

To delete only the unlabeled frames, the user should click **Select** in the action bar to quickly select all 14 frames, then deselect the video file before clicking the trash can icon.

Deleting items

In general, to delete items, you select and delete the files. However, because video frames always maintain a child / parent relationship with the original video, when you select a video for deletion, the video *and* all of the frames are deleted. You can delete frames and leave the video, but you cannot delete the video and leave the frames.

The notification area

Click the bell icon in the navigation bar to access the notification area. This allows you to view and work with messages. Click the arrow to return to your previous view.

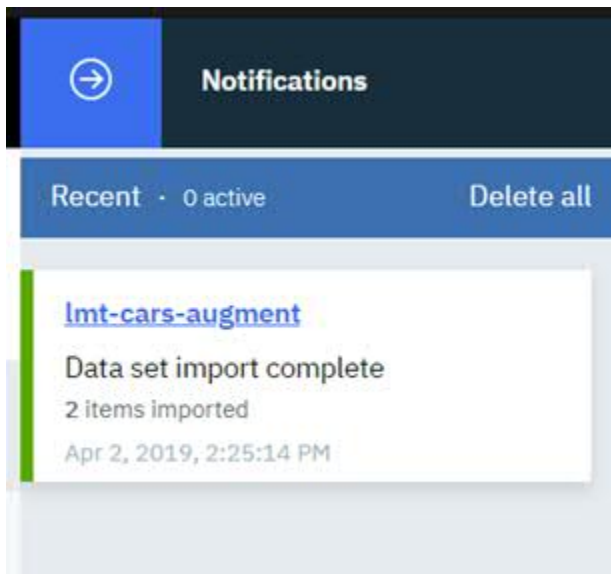


Figure 8. Notification area

Related concepts

Training and working with models

Use these processes to create, deploy, and refine models.

Example: Detecting objects in a video

In this fictional scenario, you want to create a deep learning model to monitor traffic on a busy road. You have a video that displays the traffic during the day. From this video, you want to know how many cars are on the busy road every day, and what are the peak times that have the most cars on the road.

Related tasks

Training a model

After the data set has been prepared, you can train your deep learning model. Trained models can then be deployed for use.

Creating and working with data sets

Before you can work with videos or images, you need to create a data set. A data set is a group of images, videos, or both that you will use to train a deployable model.

Deploying a trained model

Deploy a trained model to get it ready to use within IBM Maximo Visual Inspection or a different program, such as IBM Watson Machine Learning Community Edition. Deploying a model creates a unique API endpoint based on that model for inference operations.

Example: Classifying images

The goal of this example is to train a model to classify images of birds into groups based on their physiological similarities. Once the model is trained with a known dataset, users can upload new data sets to auto classify the birds into their respective categories. We will prepare the data, create a data set, train the model, and test the model.

Determining the product version

The product version is shown on the login screen. To determine which version of IBM Maximo Visual Inspection is installed after logging in, click your user name in the upper right corner of the user interface. The product version is listed in the menu that opens.

Chapter 9. Training and working with models

Use these processes to create, deploy, and refine models.

You can only see and work with objects (data sets, files, trained models, and deployed models) that you own. An object is owned by the user who created it.

Creating and working with data sets

Before you can work with videos or images, you need to create a data set. A data set is a group of images, videos, or both that you will use to train a deployable model.

About this task

To create a data set and add content to it, follow these steps:

Procedure

1. [Log in to IBM Maximo Visual Inspection](#).
2. Click **Data Sets** in the navigation bar to open the **Data Sets** page. There are several ways to create a new data set:
 - To create an empty data set, click **Create new data set**.
 - If you have a previously exported data set, click **Import .zip file**.
 - If you want to copy an existing data set, select the data set and click the "Duplicate" icon.

File considerations:

• Videos

- You can play only the following video types in the IBM Maximo Visual Inspection video player:
 - Ogg Vorbis (.ogg)
 - VP8 or VP9 (.webm)
 - H.264 encoded videos with MP4 format (.mp4)
- Before importing videos for use with action detection models, it is recommended that you prepare them as follows:
 - Cut out long periods of background video without any actions.
 - Transcode videos with FPS greater than 30 down to 30 FPS
 - Crop the video so that actions should take up a large part of the frame.

• Images

- DICOM images are converted to PNG files for storage in the data set, and can then be labeled or augmented like any other image.
- IBM Maximo Visual Inspection has limited support for Pascal VOC annotations. Annotations for multiple files residing in a common XML file are not supported. In other words, each annotation XML file can only contain annotations for a single image, identified by the `filename` attribute.

If you have a single XML annotation file containing annotations for multiple images in the data set to be imported, the annotations need to be split out into separate XML files before IBM Maximo Visual Inspection can import the annotations successfully.

- IBM Maximo Visual Inspection supports importing COCO data sets with the following limitations:

Only "object detection" annotations are supported. You can review the annotation format on the [COCO data format](#) page. When you import images with COCO annotations, IBM Maximo Visual Inspection only keeps the information it will use, as follows:

- IBM Maximo Visual Inspection extracts the information from the images, categories, and annotations lists and ignores everything else.
- Unused annotations are not saved. For example, if there is annotation information for clock, but no image is tagged with a clock, then the clock object (called *category* in COCO) is not saved.
- For COCO annotations that use the RLE format, the entire annotation is ignored.

Note: Images without tags *are* saved.

3. Optionally add the data set to a project group:
 - a) From the Project Groups page, open the relevant project group.
 - b) Click **Add assets to this project**. For Asset type, choose **Data set**, then select your data set.
4. Click the data set you just created to open it. Add images and videos by using **Import file** or by dragging them to the **+** area.

If you do not follow these considerations, your upload will fail and a message will be shown on the screen. For details about why the upload failed, click the bell icon in the page header to open the Notifications center.


Upload considerations:

- You can select multiple image or video files, or a single .zip file that contains images and videos, but you cannot upload a folder that contains images or videos.
- If you import a .zip file into an existing data set, the .zip file cannot contain a directory structure.
- You cannot navigate away from the IBM Maximo Visual Inspection page or refresh until the upload completes. You can navigate to different pages within IBM Maximo Visual Inspection during the upload.
- There is a 24 GB size limit per upload session. This limit applies to a single .zip file or a set of files. You can, however upload 24 GB of files, then upload more after the original upload completes.

Results

Working with data sets

After your data set has been created, select it in the **Data Sets** page to duplicate, rename, delete it, and so on. To work with the images and videos contained in the data set, click the name of the data set to open it.

You can view the metadata associated with an image by opening the image and clicking the **Show metadata** icon (.

To move or copy images and videos between data sets, open the data set, select the images and videos, and click the **Copy** icon in the Action bar. Select **Move** or **Copy**, then the appropriate data set or data sets in the window that opens.

- You can copy images and videos to multiple data sets.
- You can move images and videos to a single data set.
- When you move or copy a video, all captured frames are also moved or copied.
- You cannot move or copy video frames without the video. If you select video frames but not the parent video, you will get an error.

By default, the data sets are displayed as thumbnails. If you select the list view, you will see the information displayed as columns.

Note: If an older data set is imported, the original file names are missing or inaccurate. Additionally, original file names are not available for captured video frames or augmented files.

Working with video data and captured frames

In general, to delete items, you select and delete the files. However, because video frames always maintain a child / parent relationship with the original video, when you select a video for deletion, the

video *and* all of the frames are deleted. You can delete frames and leave the video, but you cannot delete the video and leave the frames.

Related concepts

Creating and working with project groups

Project groups allow you to group trained models with the data sets that were used for training. This grouping is optional but is a useful way to organize related data sets. For example, project groups would be useful with a workflow that clones data sets as you refine labels and work toward a more accurate model. Project groups can be used with a production work flow strategy and automatic model deployment for even more functionality.

Data set considerations

When preparing a data set for training, consider the following information to ensure the best results.

Note: Unless otherwise noted, mentions of "images" refer to both individual images and captured video frames.

- [“What are the limitations on uploaded files?” on page 59](#)
- [“How many images are needed?” on page 60](#)
- [“Special considerations for object detection models” on page 61](#)

What are the limitations on uploaded files?

- The following image formats are supported:
 - JPEG
 - PNG
 - DICOM
- You can play only the following video types in IBM Maximo Visual Inspection:
 - Ogg Vorbis (.ogg)
 - VP8 or VP9 (.webm)
 - H.264 encoded videos with MP4 format (.mp4)
- The models used by IBM Maximo Visual Inspection have limitations on the size and resolution of images. If the original data is high resolution, then the user must consider:
 - If the images do not need fine detail for classification or object detection, they should be down-sampled to 1-2 megapixels.
 - If the images do require fine detail, they should to be divided into smaller images of 1-2 megapixels each.
 - There is a 24 GB size limit per upload session. This limit applies to a single .zip file or a set of files. You can, however upload 24 GB of files, then upload more after the original upload completes.
 - Large images will be scaled to the appropriate dimensions for the model as follows:
 - SSD: 512 x 512 pixels
The original aspect ratio is maintained. If necessary, black bands are added to the image to make it fit.
 - Tiny YOLO v2: 416 x 416 pixels
The longest edge is scaled to 416 pixels and, if necessary, black bands are added to the shorter side to make it 416 pixels.
 - YOLO v3: 608 x 608 pixels
 - Faster R-CNN: 1000 x 600 pixels
The original aspect ratio is maintained. If necessary, black bands are added to the image to make it fit.

- Detectron: 1333 x 800 pixels
- GoogLeNet: 224 x 224 pixels
- Action detection: 224 x 224 pixels
- Images with COCO annotations are supported. For details, see [“Importing images with COCO annotations”](#) on page 61.

How many images are needed?

A data set with a variety of representative objects labeled will train a more accurate model. The exact number of images and objects cannot be specified, but some guidelines recommend as many as 1,000 representative images for each class. However, you might not need a data set this large to train a model with satisfactory accuracy. The number of images required depends on the kind of training you plan on doing:

Image classification

- There must be at least two categories.
- Each category must have at least five images.

Object detection

The data set must contain at least five images with an object labeled for each defined object. For example, if you want to train the data set to recognize cars and you have three images and one video, you must add the "car" label to each image and at least two frames of the video. Labeling five cars in one image is not adequate. If this requirement is not met and you train the model, it will not be trained to recognize that type of object.

For object detection model training, images that do not have any objects labeled are not used in the training of the model.

Tiny YOLO V2 considerations:

- Tiny YOLO v2 requires at least five unique bounding boxes in the data set. If multiple images have the same coordinates for the bounding box resulting in fewer than five uniquely sized boxes, the model will fail to train.
- The data set is split into training and validation images for the model per the Ratio training parameter, see "Model hyperparameters" in [“Training a model”](#) on page 68. The Ratio parameter determines how many images are used for training and how many are used for validation, but the images are selected for each group at random. For a small data set, this can result in insufficient validation images. If that happens, training fails.

It is recommended that the data set contain at least 20 labeled images.

Important: Not all of the images in a data set are used for training. Assuming that you did not change the value for Ratio (an advanced hyperparameter setting) when training your model, 20% of the images are randomly selected and used for validation instead of training. Because of this, it is important that you have enough images of every category or object.

For example, consider a data set to be used for training of an object detection model that has 200 images. With the default configuration for model training, 20% of the images (40 images) will be selected for testing the model. If there is a label **LabelA** used to identify an object in the data set, the following scenarios are possible if the number of images labeled with the object are smaller than the test data set, for example, if there are only 20 images with objects labeled as **LabelA**:

- It is possible that all of the images with **LabelA** are in the "training" data set, and none of the images are actually used for testing of the model. This will result in *unknown* accuracy for **LabelA**, since there are no tests of the accuracy.
- Similarly, it is possible that all 20 images with **LabelA** objects are in the test data set but there are no images used for training. This will result in very low or 0% accuracy for the object because the model was not actually trained with any images containing the **LabelA** objects.

If your data set does not have many images or sufficient variety for training, consider using the [Augmentation feature](#) to increase the data set.

Special considerations for object detection models

Accuracy for object detection models can be more challenging since it includes intersection over union (IoU), especially for models that use segmentation instead of bounding boxes. IoU is calculated by the intersection between a ground truth bounding box and a predicted bounding box, divided by the union of both bounding boxes; where the intersection is the area of overlap, a *ground truth* bounding box is the hand drawn box, and the *predicted bounding box* is the one drawn by IBM Maximo Visual Inspection.

In the case of object detection, the object might have been correctly identified but the overlap of the boundary generated by the model is not accurate resulting in a poor IoU metric. This metric might be improved by more precise object labeling to reduce background "noise", by training the model longer, or both.

Importing images with COCO annotations

Images with Common Objects in Context (COCO) annotations have been labeled outside of IBM Maximo Visual Inspection. You can import (upload) these images into an existing IBM Maximo Visual Inspection data set, along with the COCO annotation file, to inter-operate with other collections of information and to ease your labeling effort.

Only "object detection" annotations are supported. You can review the annotation format on the [COCO data format page](#). When you import images with COCO annotations, IBM Maximo Visual Inspection only keeps the information it will use, as follows:

- IBM Maximo Visual Inspection extracts the information from the images, categories, and annotations lists and ignores everything else.
- Unused annotations are not saved. For example, if there is annotation information for clock, but no image is tagged with a clock, then the clock object (called *category* in COCO) is not saved.
- For COCO annotations that use the RLE format, the entire annotation is ignored.

Note: Images without tags *are* saved.

To import images with COCO annotations into IBM Maximo Visual Inspection, follow these steps:

1. If necessary, create a new data set. The data set must exist before importing the COCO annotated data.
2. Download the images that you want to import.
3. If you downloaded `train2017.zip`, IBM Maximo Visual Inspection cannot train the entire data set. Therefore, you must make a new file that contains just the images you want to train. For example, by running this command:

```
ls train2017 | grep jpg | head -20000 >/tmp/flist
```

4. Download the annotations file for your images. For example, `annotations_trainval2017.zip` contains the annotations for the `train2017` data set. For example, if you downloaded `annotations_trainval2017.zip`, extract the `annotations/instances_train2017.json` file, which is the COCO annotation file for object detection.

If you are using a .json file from a different source, it cannot be called `prop.json`.

5. Create a zip file that contains the annotations file and the images.
 - There can be only one .json file in the zip file. If more than one .json file is discovered, only the first one is used.
 - The .json file cannot be named `props.json` because this is used by IBM Maximo Visual Inspection exported data sets, which use different annotations.
 - The images and the annotation file can reside in different directories.
6. Import the zip file into an existing IBM Maximo Visual Inspection data set.

Note: COCO data sets are created for competition and are designed to be challenging to identify objects. Therefore, do not be surprised if the accuracy numbers achieved when training are relatively low, especially with the default 4000 iterations. However, these data sets will allow you to experiment with segmentation training and inference without having to manually label a lot of images


For details about COCO data sets, refer to the [COCO web site](#).

Search for assets in a data set

There are several ways to find images and videos in a data set. You can search based on file names, metadata, categorization, or tags.

You can use the search bar, filtering, or a combination of these to find assets in a data set. For example, if you filter based on a specific category, you can then use the search bar to search for objects with a specific file name within that category. You could then further narrow your search results by searching for specific metadata associated with the file.

Search bar

Click the magnifying glass to access the search bar. You can add search terms that allow you search for specific file names. To search for metadata, open the search bar, then click the metadata search icon (). You can then enter as many key / value pairs as you want, based on the keys and values that are available in the data set. If you enter multiple pairs, all values must be satisfied for any files that are returned.

Filtering

With large data sets, you might need to filter the files that are shown in the data area. By default, your whole data set is shown.

Filter by

When you deselect a file type, those files are no longer shown in the data area. Therefore, if you only have Images selected, only images are shown in the data area.

Categories / objects

When you select categories, objects, or both, *all* files of the specified type that belong to any of the selected categories, or contain the selected objects, are shown.

For example, assume you have a data set with two categories: Cats and Dogs. Also assume that you tagged these types of objects: Face, Collar, and Tail. Then if you select Images, the category Dogs, and the object Collar, you will see all images that are dogs *or* contain a collar. This will include images of cats if they have a collar as well as images of dogs with no collar.

Using filtering and "Select all" with video data

When you capture frames from a video, these frames always maintain a child / parent relationship with the original video. That has some selection and filtering implications.

- When using the filter on the side bar, if *any* video frame matches the filter criteria, both the frame and its parent video are selected and are shown in the data area.
- If you click the "Select" box in the action bar, everything in the data area is selected. Therefore, if there is a video shown in the data area, it, and all of its child frames, are selected. Any action performed in this situation applies to all selected images, the video, and all of its child frames.

Example

A user has captured 50 frames from a video file `Cars Video`. Fourteen frames of the 50 have no labels.

1. The user selects **Unlabeled** in the Objects filter in the sidebar. The 14 frames with no labels and their parent video, `Cars Video`, are shown in the data area.
2. The user clicks **Select** in the action bar. The frames and the video are all selected.
3. The user clicks the trash can icon, intending to delete the unlabeled frames. However, because the video was selected, it, and the 36 labeled frames, are also deleted.

To delete only the unlabeled frames, the user should click **Select** in the action bar to quickly select all 14 frames, then deselect the video file before clicking the trash can icon.

Labeling objects

One of the most important steps is to ensure that you properly label objects by adding tags to your data.

Requirements

Recommendation: Label and class names should be 64 characters or less. Longer label names are supported but using international characters or very long label names can cause an internal metadata error, resulting in a training failure.

Image classification

- There must be at least two categories.
- Each category must have at least five images.

Object detection

The data set must contain at least five images with an object labeled for each defined object. For example, if you want to train the data set to recognize cars and you have three images and one video, you must add the "car" label to each image and at least two frames of the video. Labeling five cars in one image is not adequate. If this requirement is not met and you train the model, it will not be trained to recognize that type of object.

For object detection model training, images that do not have any objects labeled are not used in the training of the model.

Tiny YOLO V2 considerations:

- Tiny YOLO v2 requires at least five unique bounding boxes in the data set. If multiple images have the same coordinates for the bounding box resulting in fewer than five uniquely sized boxes, the model will fail to train.
- The data set is split into training and validation images for the model per the Ratio training parameter, see "Model hyperparameters" in ["Training a model" on page 68](#). The Ratio parameter determines how many images are used for training and how many are used for validation, but the images are selected for each group at random. For a small data set, this can result in insufficient validation images. If that happens, training fails.

It is recommended that the data set contain at least 20 labeled images.

Note: A data set with a variety of representative objects labeled will train a more accurate model. The exact number of images and objects cannot be specified, but some guidelines recommend as many as 1,000 representative images for each class. However, you might not need a data set this large to train a model with satisfactory accuracy.

If your data set does not have many images or sufficient variety for training, consider using the [Augmentation feature](#) to increase the data set.

- ["Labeling videos" on page 63](#)
- ["Labeling images" on page 65](#)

Labeling videos

1. Select the video from your data set and select **Label Objects**.
2. Capture frames by using one of these options:

- **Auto capture frames** - IBM Maximo Visual Inspection captures a video frame every n seconds, where n is specified in the **Capture Interval (seconds)** field.

Note: Depending on the length and size of the video and the interval you specified to capture frames, the process to capture frames can take several minutes.

- **Manually capture frames** - use **Capture frame** to capture relevant frames.

Note: When you capture frames from a video, these frames always maintain a child / parent relationship with the original video.

3. If required, manually add new frames to an existing data set. This might happen if **Auto capture frames** does not produce enough frames with a specific object type. To manually add new frames, follow these steps:

- a. Play the video and when the frame you want is displayed, click the pause icon.

Tip: You can use the video player's status bar to find a frame you want.

- b. Click **Capture Frame**.

4. Create new object labels for the data set by clicking **Add new** by the Objects list. To add multiple object labels, enter one label, click **Add**, then enter the next until you are done. Label names cannot contain any special characters other than the underscore (_). For example, characters such as these are not allowed: - " / \ | { } () ; ,

Note: If non-ASCII characters are used in the label name, they will not be displayed correctly when using a video to test the deployed model. See [“Testing a model” on page 83](#).

You can rename objects later. However, after you rename an object, you will no longer be able to undo actions done before the rename.

5. Label the objects in the frames by following these steps.

- a. Select the first frame in the carousel.

- b. Select the correct object label.

- c. Choose **Box** or **Polygon** from the bottom left, depending on the shape you want to draw around each object. Boxes are faster to label and train, but less accurate. Only Detectron models support polygons. However, if you use polygons to label your objects, then use this data set to train a model that does not support polygons, bounding boxes are defined and used. Draw the appropriate shape around the object.

Tip: The **Paste previous** button is active if there is at least one frame before the current frame being edited. Clicking **Paste previous** copies all the labels from the previous video frame and paste them into the current frame.

Follow these guidelines when identifying and drawing objects in video frames:

- Do not label part of an object. For example, do not label a car that is only partially in the frame.
- If an image has more than one object, you must label all objects. For example, if you have cars and motorcycles defined as objects for the data set, and there is an image with both cars and motorcycles in it, you must label the cars and the motorcycles. Otherwise, you decrease the accuracy of the model.
- Label each individual object. Do not label groups of objects. For example, if two cars are right next to each other, you must draw a label around each car.
- Draw the shape as close to the objects as possible. Do not leave blank space around the objects.
- You can draw shapes around objects that touch or overlap. For example, if one object is behind another object, you can label them both. However, it is recommended that you only label objects if the majority of the object is visible.
- Use the zoom buttons (+ and -) on the bottom right side of the editing panels to help draw more accurate shapes.

Note: If you are zoomed in on an image and use the right arrow key to move all the way to the right edge, you might have to click the left arrow key several times to start panning in the other direction.

- Shapes cannot extend off the edge of the frame.
- After defining a shape, you can copy and paste it elsewhere in the same image or in a different image by using standard keyboard shortcuts. After pasting the shape, it can be selected and dragged to the desired location in the image. The shape can also be edited to add or remove points in the outline.

Note: To copy and paste a shape from one image to another, both images have to be available in the image carousel. From the data set, select all images that will share shapes, then click **Label objects**. All images will be listed in the image carousel in the left side of the Label objects window.

- After a shape has been defined, you will no longer see the points on the outline. To edit a defined box, exit drawing mode, then edit the points as necessary. To exit drawing mode, do one of the following:

- Click the object name on the right side of the window.
- Alt+click (option +click) inside the defined box.

After moving a defined point, drawing mode is automatically enabled again.

- The video object preview does not support non-ascii labels. This is a limitation of the module that generates the displayed label from the label name. The result of the conversion of non-ascii labels will be a label that is all question marks: "?????".

- **Labeling with polygons**

- After a shape has been defined, you will no longer see the points on the outline. To edit a defined shape, exit drawing mode, then edit the points as necessary. To exit drawing mode, do one of the following:

- Click the object name on the right side of the window.
- Click inside the defined shape.

When you are done editing the shape, click outside the shape to enter drawing mode again.

- To delete a point from an outline, ctrl+click (or cmd+click).
- To add a point to an outline, click the translucent white square between any two points on the outline.
- To move a point on the outline, click it and drag.

Labeling images

Follow these steps to label images in your data set:

1. Create new object labels for the data set by clicking **Add new** by the Objects list. To add multiple object labels, enter one label, click **Add**, then enter the next until you are done. Label names cannot contain any special characters other than the underscore (_). For example, characters such as these are not allowed: -"/ \ | { } () ; ,
 2. Open an image. In the right pane, select the object you want to label.
 3. Choose **Box** or **Polygon** from the bottom left, depending on the shape you want to draw around each object. Boxes are faster to label and train, but less accurate. Only Detectron models support polygons. However, if you use polygons to label your objects, then use this data set to train a model that does not support polygons, bounding boxes are defined and used. Draw the appropriate shape around the object.
- Do not label part of an object. For example, do not label a car that is only partially in the frame.
 - If an image has more than one object, you must label all objects. For example, if you have cars and motorcycles defined as objects for the data set, and there is an image with both cars and motorcycles in it, you must label the cars and the motorcycles. Otherwise, you decrease the accuracy of the model.
 - Label each individual object. Do not label groups of objects. For example, if two cars are right next to each other, you must draw a label around each car.
 - Draw the shape as close to the objects as possible. Do not leave blank space around the objects.
 - You can draw shapes around objects that touch or overlap. For example, if one object is behind another object, you can label them both. However, it is recommended that you only label objects if the majority of the object is visible.
 - Use the zoom buttons (+ and -) on the bottom right side of the editing panels to help draw more accurate shapes.

Note: If you are zoomed in on an image and use the right arrow key to move all the way to the right edge, you might have to click the left arrow key several times to start panning in the other direction.

- Shapes cannot extend off the edge of the frame.
- After defining a shape, you can copy and paste it elsewhere in the same image or in a different image by using standard keyboard shortcuts. After pasting the shape, it can be selected and dragged to the desired location in the image. The shape can also be edited to add or remove points in the outline.

Note: To copy and paste a shape from one image to another, both images have to be available in the image carousel. From the data set, select all images that will share shapes, then click **Label objects**. All images will be listed in the image carousel in the left side of the Label objects window.

- After a shape has been defined, you will no longer see the points on the outline. To edit a defined box, exit drawing mode, then edit the points as necessary. To exit drawing mode, do one of the following:
 - Click the object name on the right side of the window.
 - Alt+click (option +click) inside the defined box.

After moving a defined point, drawing mode is automatically enabled again.

- The video object preview does not support non-ascii labels. This is a limitation of the module that generates the displayed label from the label name. The result of the conversion of non-ascii labels will be a label that is all question marks: "?????".
- **Labeling with polygons**
 - After a shape has been defined, you will no longer see the points on the outline. To edit a defined shape, exit drawing mode, then edit the points as necessary. To exit drawing mode, do one of the following:
 - Click the object name on the right side of the window.
 - Click inside the defined shape.

When you are done editing the shape, click outside the shape to enter drawing mode again.

- To delete a point from an outline, ctrl+click (or cmd+click).
- To add a point to an outline, click the translucent white square between any two points on the outline.
- To move a point on the outline, click it and drag.

Objects panel

Click the settings icon on the right side of the Objects panel to change the labeling settings, such as whether to show object labels inside shapes, hide all shapes except the one being drawn, change the shape opacity, and so on.

As you label objects, they are added to the list in the Objects panel on the right. To work with a labeled object, select it in the Objects panel. You can hide the object outline, rename it, or delete it.

To work with all objects of one type, such as cars, click the three dots to the right of the object title. These actions apply only to the items identified as this type of object in the current image.

Related tasks

Automatically labeling objects

After deploying a model for object detection, you can improve its accuracy by using the Auto label function. This function uses the labels in the deployed model to generate new labels in the data set; increasing the number of images that are labeled in the data set. The updated data set can be used to train a new, more accurate model.

Labeling actions

To label an action in a video, you will mark the start and end of the action, then assign it a tag. These tags exist at the data set level and can be used by any video in the data set. Videos with action tags are marked with an action icon in the lower right corner.

- [“Recommendations” on page 67](#)

- [“Preparing videos for import” on page 67](#)
- [“Steps to label actions” on page 67](#)
- [“Working with action labels” on page 68](#)
- [“Tips” on page 68](#)

Recommendations

There is no minimum number of labels required, but more data will typically give better results.

- Each action label must be in the range of 5 - 1000 frames. The required length of time depends on the video's FPS. For 30 FPS, each action label must be in the range of .166 - 33.367 seconds.

The label's duration is checked based on the frames per second and the selected start and end times. For example, if an action label is marked with a start time of 12.295 seconds and end time of 12.296 seconds for a 30 FPS video, you will get an error message like the following: "Label duration of '100' milliseconds does not meet required duration between '166.83333' milliseconds and '33366.668' milliseconds".

- At least 10 instances of each action tag in the data set are recommended.
- The longer the total labeled action time is, the better your results will be.
- If multiple types of actions are labeled in a data set, the total amount of time for each action type should be similar. For example, if you tag 20 instances of the action "jump" in a data set with a total time of 27 seconds, and you tag 10 instances of the action "drive" in the data set with a total time of 53 seconds, the model will be biased toward the "drive" action.

The total time for each action type is shown in the left pane in the Actions section.

Preparing videos for import

Before importing videos for use with action detection models, it is recommended that you prepare them as follows:

- Cut out long periods of background video without any actions.
- Transcode videos with FPS greater than 30 down to 30 FPS
- Crop the video so that actions should take up a large part of the frame.

Steps to label actions

1. Open the data set that contains the video you want to label.
2. Create an action tag in the data set by expanding **Actions** on the left and clicking **Add action**. If you delete the tag, all instances of that tag are removed from any video in the data set that used that tag.
3. Select the video and click **Label actions**. The existing tags are listed on the right. You can click the settings icon (gear) to choose how the action tags are organized in the Actions list.

Note: If you click **Add label** to create a new tag, it is added to the video and the data set. If the action does not meet the criteria described above, the label will still be created, but the label count will not be increased. The tag is created with a count of 0.

4. Find the start of an action by using the video control bar:
 - Use the slider or play button to get near the part of the video you want.
 - Set the playback rate (1x, .5x, and so on) to control how fast the video plays.
 - Use the +1 and -1 buttons to move forward or backward one frame.
5. Click **+** in **Start time**. Actions cannot overlap. That is, the start time cannot be between an existing Start time and End time.
6. Find the end of the action, then click **+** in **End time**.
7. Specify an action name, either by selecting an existing tag or by entering the name for a new tag, then click **Create action**.

Working with action labels

- To delete one instance of a tag, select the tag in the Action panel and click the trash can.
- To delete all instances of a tag in the current video, select the top level tag in the action panel, click the three vertical dots, and click **Delete actions**.
- To delete a tag and all instances of the tag in every video in the data set, go to the data set main page, expand Actions, and click **Edit**. Click **X**.
- If you select a tag in the Action panel, the video player moves to the start of that action.
- To edit a tag, select the tag in the Actions panel and click the pencil. You can change the values for start time, end time, or action name, then click **Edit action** to save your changes. To cancel editing the tag, click the "X" in the Actions panel.

Tips

- The best actions to label are relatively short directional motions, with minimal camera viewpoint movement.
- Actions should be at least 3-5 frames long. If there are similar, unlabeled actions in a video, this can result in false positives. Any similar actions should be given a different label.
- There should only be one action taking place in the parts of the video that you label. For example, if part of a video shows people driving and people riding bicycles, it should not be labeled with an action.
- Portions of the video that have been labeled are shown as in the associated action color in the video progress bar. Unlabeled portions are blue.

Training a model

After the data set has been prepared, you can train your deep learning model. Trained models can then be deployed for use.

About this task

Procedure

1. From the **Data set** page, click **Train**.
2. In the **Train data set** window, fill out the values as appropriate, then click **Train**:

Type of training

Image classification

Choose this if you want to use the model to categorize images as belonging to one of the types that you defined in the data set.

Note:

- There must be at least two categories.
- Each category must have at least five images.

Optimize model using

System default (GoogLeNet)

Models trained with this model can only be run on a GPU. Under **Training options**, you can enable Core ML. After deploying the model, you can download the generated Core ML assets from the Deployed models page.

Custom packaged model

Select an imported model to use for training.

Object detection

Choose this if you want to use the model to label objects within images.

Note: The data set must contain at least five images with an object labeled for each defined object. For example, if you want to train the data set to recognize cars and you have three images and one video, you must add the "car" label to each image and at least two frames of the video. Labeling five cars in one image is not adequate. If this requirement is not met and you train the model, it will not be trained to recognize that type of object.

Optimize model using

Faster R-CNN

Models optimized for accuracy can only be run on a GPU. It is always enabled for TensorRT. After deploying the model, you can download the TensorRT assets from the Deployed models page.

Tiny YOLO v2

Models optimized for speed can be run anywhere, but might not be as accurate as those optimized for accuracy. These models use "you only look once" (YOLO) v2 and will take several hours to train.

You will choose the accelerator to deploy to when deploying the model. You can choose GPU, CPU, or Xilinx FPGA - 16 bit (technology preview). Under **Training options**, you can enable Core ML. After deploying the model, you can download the generated Core ML assets from the Deployed models page.

YOLO v3

Models optimized for speed can be run anywhere and support CoreML deployment. These models use "you only look once" (YOLO) v3 and may take longer to train for desired accuracy.

A CoreML asset is always generated.

Detectron

Detectron Mask R-CNN models can only be run on a GPU. They can use objects labeled with polygons for greater training accuracy. Labeling with polygons is especially useful for small objects, objects that are at a diagonal, and objects with irregular shapes. However, training a data set that uses polygon labels takes longer than training with rectangular bounding boxes. If you want to use a Detectron model but want a shorter training time, you can disable segmentation and IBM Maximo Visual Inspection will use rectangles instead of polygons. The actual images are not modified, so you can train with segmentation later.

Single Shot Detector (SSD)

Suitable for real-time inference and embedded devices. It is almost as fast as YOLO but not as accurate as Faster R-CNN. It is always enabled for TensorRT. After deploying the model, you can download the TensorRT assets from the Deployed models page.

Custom packaged model

Select an imported model to use for training.

Action detection

Choose this if you want to use this model to label actions within videos.

Optimize model using

Structured segment network (SSN)

Used for action detection models only to detect short activity or actions in videos. It is best at classifying short bursts of time that have a strong sense of direction.

Advanced settings

Base model

You must select a base model when training for image classification with GoogLeNet. You can optionally choose a base model when training for object detection with Faster R-CNN, Detectron, and YOLO v3.

When you specify a base model, IBM Maximo Visual Inspection uses the information in the base model to train the new model. This allows you to transfer learning that has already been

done with one model to a new model, resulting in more accurate training. You can choose a model that is included with IBM Maximo Visual Inspection, or you can choose your own model that you previously trained or imported. For models that were trained in IBM PowerAI Vision versions prior to 1.1.2, the list of associated objects or categories is not shown in the user interface. However, those models are still usable.

The base model's network must have been trained with the same model type. For example, to train a new object detection model with Faster R-CNN, then the base model must be built on Faster R-CNN. Only viable models are listed in the Base model table.

Note: Base models are not available for tiny YOLO v2, SSD, Detectron, and custom models that are used for object detection, or custom models that are used for image classification.

IBM Maximo Visual Inspection comes with several common models such as flowers, food, and so on, that you can use to help classify your data. If you do not select a base model when training with GoogLeNet, General is used. For more information, see [“Base models included with IBM Maximo Visual Inspection” on page 79](#).

Model hyperparameters

For advanced users, these settings are available to help fine-tune the training. The user interface presents ranges for several of the hyperparameters, with lower and upper bounds marked with either a parenthesis, (), or a bracket, []. A parenthesis indicates a non-inclusive bound, and a bracket indicates an inclusive bound. For example, (0-0.5] indicates the value must be greater than 0 and can be up to and including 0.5.

Epochs (Action detection only)

The number of times the entire data set is passed through the training algorithm. Large data sets are divided into smaller parts to fit the GPU memory and processed as batches. One batch is passed through the algorithm during each iteration. Therefore, each epoch is made up of many iterations.

Specifying a large number of epochs can increase the training time substantially especially for larger data sets.

Max iteration (Image classification and object detection only)

The maximum number of times the data is passed through the training algorithm, up to 1,000,000 iterations. In general, the more iterations the model is trained, the more accurate the model will be. However, in many cases the test accuracy will plateau at some point beyond which further iterations do not result in a significant improvement in the accuracy of the model.

The default number of iterations is a placeholder, chosen based on characteristics of the model. For example, tiny YOLO v2 models typically require more iterations to achieve a class and IoU accuracy comparable to other object detection models in the product. Therefore, the default is higher for this model.

Depending on training factors, such as the number of data set samples, number of classes, inter and intra similarity between classes/objects, target accuracy, and so on, the optimal number of iterations may be higher or lower than the default. One strategy for determining the optimal number of iterations is to choose a target accuracy and train the model until the desired accuracy is reached or the loss rate no longer is improving.

Momentum (Object detection only)

This value increases the step size used when trying to find the minimum value of the error curve. A larger step size can keep the algorithm from stopping at a local minimum instead of finding the global minimum.

Ratio

IBM Maximo Visual Inspection automatically “splits” the data set for internal validation of the model’s performance during training. The default Ratio value of 80/20 will result in 80% of the images in the data set (at random) being used for training, and 20% being used for measurement / validation.

Note: Image classification models do not allow a ratio of 100%, as some images are required for validation.

Test iteration (Image classification only)

The number of times data is passed through the training algorithm before possible completion. For example, if this value is 100, and Test interval is 50, the model is run through the algorithm at least 100 times; being tested ever 50 times.

Test interval (Image classification only)

The number of times the model is passed through the algorithm before testing. For example, if this value is 50, the model is tested every 50 iterations. Each of these tests becomes a data point on the metrics graphs.

Learning rate

This option determines how much the weights in the network are adjusted with respect to the loss gradient. A correctly tuned value can result in a shorter training time. However, it is recommended that only advanced users change this value. A learning rate that is too large can result in significant oscillations in the loss function, and in the worst case "exploding gradients" resulting in failure to train the model.

Weight decay

This value specifies regularization in the network. It protects against over-fitting and is used to multiply the weights when training.

3. The model starts to train (if there is a GPU available) or is added to the training queue. Each active training job takes one GPU. If all GPUs are in use, the training job is added to the queue. You can see the status on the Models page. To view details about a model that is being trained or is in the queue, click on the name of the model in the Models page. The time listed in **Queued time** is the time that the user submitted the training job and it was added to the queue.

You can remove the model from the queue by clicking **Stop training**, whether training has started or not. If training has started, you have the option to keep the model.

4. (Optional - *Only supported when training for object detection.*) Stop the training process by clicking **Stop training > Keep Model > Continue**.

You can wait for the entire training model process complete, but you can optionally stop the training process when the lines in the training graph start to flatten out, as shown in the figure below. This is because improvements in quality of training might plateau over time. Therefore, the fastest way to deploy a model and refine the data set is to stop the process before quality stops improving.

Note: Use early stop with caution when training segmented object detection models (such as with Detectron), because larger iteration counts and training times have been demonstrated to improve accuracy even when the graph indicates the accuracy is plateauing. The precision of the label is can still being improved even when the accuracy of identifying the object location stopped improving.

Understanding the model training graph

As IBM Maximo Visual Inspection trains the model, the graph shows the relative performance of the model over time. The model should converge at the end of the training with low error and high accuracy.

In the figure, you can see the Loss CLS line and the Loss Bbox lines start to plateau. In the training graph, the lower the loss value, the better. Therefore, you can stop the training process when the loss value stops decreasing. The training model has completed enough iterations and you can continue to the next step.

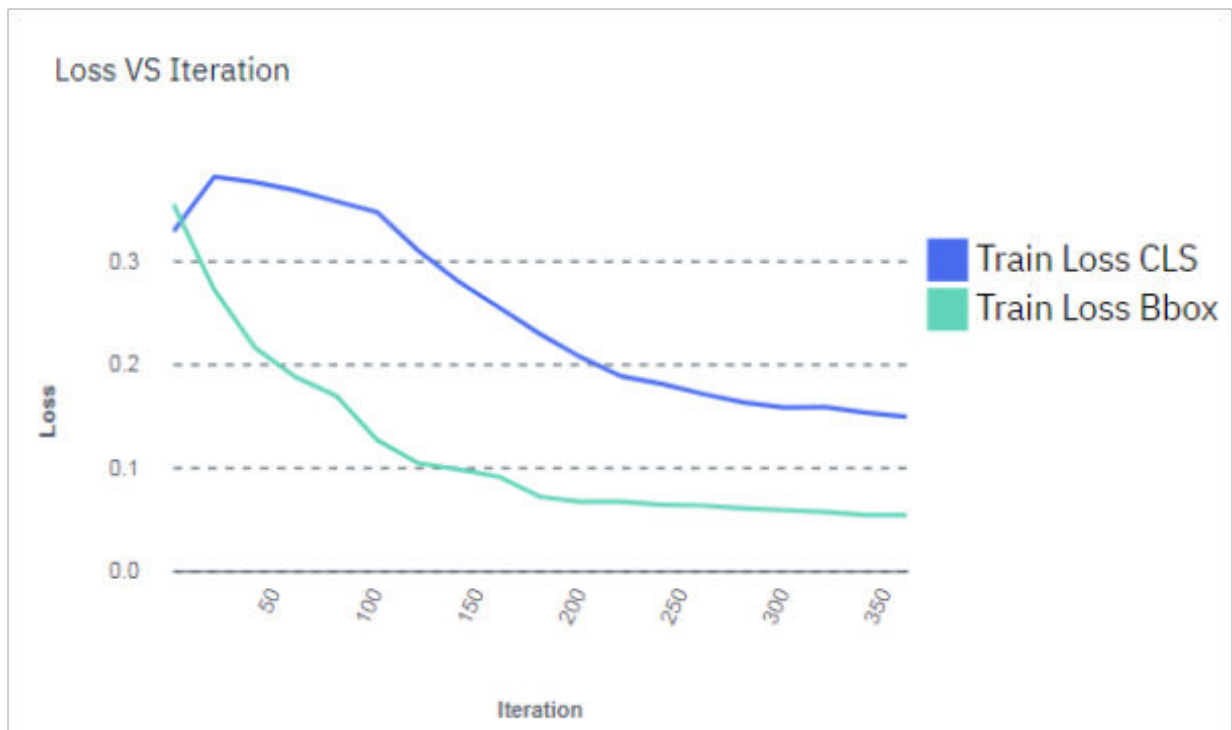


Figure 9. Model training graph

Important: If the training graph converges quickly and has 100% accuracy, the data set does not have enough information. The same is true if the accuracy of the training graph fails to rise or the errors in the graph do not decrease at the end of the training process. For example, a model with high accuracy might be able to discover all instances of different race cars, but might have trouble differentiating between specific race cars or those that have different colors. In this situation, add more images, video frames, or videos to the data set, label them, then try the training again.

The "Loss vs. Iteration" graph is different for the SSD model than for other object detection models. This is because the SSD model combines the "Train Loss Bbox" and "Train Loss CLS" into a single statistic that is presented in the training graph. For SSD models, the "Train Loss Bbox" value is not valid and the graph shows a constant value of '0', while the "Train Loss CLS" tracks a combination of Bbox and CLS loss.

Training time

The total training time depends on the data set size and the number of iterations.

The time required to prepare the data set (including setting up the DNN container usage, partitioning the data set into training and testing portions, and resizing training) and the time to generate model training statistics (accuracy per object, and confusion matrix) is directly related to the size of the data set. With a very large data set, the preprocessing and post-processing steps can take many minutes.

The training time for a model is also directly related to the number of iterations, or the number of times a batch of images is run through a forward and backward pass of the model. The execution time for an iteration varies significantly from model to model, but the more iterations the model is trained, the longer it will take. More training iterations usually also generates a more accurate model.

Related concepts

Understanding metrics

IBM Maximo Visual Inspection provides several metrics to help you measure how effectively your model has been trained.

Working with custom models

You can save time and resources by using your own TensorFlow based custom models (also referred to as *custom networks*) with IBM Maximo Visual Inspection. In general, custom models work the same as any

other model in IBM Maximo Visual Inspection. However, there are some differences you should understand.

When you upload a custom model to the **Custom Models** page, you can use the model to train a data set in IBM Maximo Visual Inspection and generate a IBM Maximo Visual Inspection trained model.

Note: Custom models cannot be used to import pre-trained models into IBM Maximo Visual Inspection. Additionally, transfer learning is not supported with custom models.

Use the information in this topic to prepare a IBM Maximo Visual Inspection trained model by using a custom TensorFlow model: [“Preparing a model that will be used to train data sets in IBM Maximo Visual Inspection” on page 73](#).

This repository has examples with detailed instructions and sample files for using custom models.

Related information

[Examples on github](#)

Preparing a model that will be used to train data sets in IBM Maximo Visual Inspection

If your custom model will be used to train data sets in the IBM Maximo Visual Inspection framework, your custom model must meet the following requirements.

After the model is properly prepared, upload it to IBM Maximo Visual Inspection by opening the **Custom Models** page and clicking **Browse files**. You can then use it to train a data set. Follow these instructions to train a data set; selecting **Custom model**: [“Training a model” on page 68](#).

Custom model requirements:

- It must be TensorFlow or PyTorch based.
- It must conform to Python 3. Any trained custom models from releases prior to Version 1.3.0 will not work if the custom model only supports Python 2.
- It must implement the MyTrain Python class.
 - The MyTrain implementation must reside in a file named `train.py` in the top level directory of the zip file contents.
 - The following import must be added to the `train.py` file in order to define the training callbacks:

```
from train_interface import TrainCallback
```

- The class name must be MyTrain.

MyTrain Template:

```
class MyTrain(TrainCallback):
    def __init__():
        pass
    def onPreprocessing(self, labels, images, workspace_path, params):
        pass
    def onTraining(self, monitor_handler):
        pass
    def onCompleted(self, model_path):
        pass
    def onFailed(self, train_status, e, tb_message):
        pass
```

class MyTrain(TrainCallback)

Use the MyTrain API to prepare a TensorFlow or PyTorch model that will be used to train data sets with IBM Maximo Visual Inspection.

- [“Template” on page 74](#)
- [“def onPreprocessing\(self, labels, images, workspace_path, params\)” on page 74](#)
- [“def onTraining\(self, monitor_handler\)” on page 75](#)
- [“def onCompleted\(self, model_path\)” on page 75](#)
- [“def onFailed\(self, train_status, e, tb_message\):” on page 75](#)

- [“Monitoring and reporting statistics” on page 75](#)

Template

This is a template you can use for the MyTrain API:

```
class MyTrain(TrainCallback):
    def __init__():
        pass
    def onPreprocessing(self, labels, images, workspace_path, params):
        pass
    def onTraining(self, monitor_handler):
        pass
    def onCompleted(self, model_path):
        pass
    def onFailed(self, train_status, e, tb_message):
        pass
```

def onPreprocessing(self, labels, images, workspace_path, params)

Callback for data set preprocessing.

Input

labels (dict)

Image categories and index.

Example: {'safety_vest': 1, 'helmet': 0, 'no_safety_vest': 2, 'no_helmet': 3}

images

- *image classification* (dict): Image path and its category.

Example: {'/dataset/Acridothores/001.jpg': 'Acridothores', '/dataset/Butorides/002.jpg': 'Gallinula', '/dataset/Butorides/003.jpg': 'Butorides'}

- *object detection* (list): List of annotation objects; including the image name and annotation.

Example:

```
[annotation[0] annotation[1] ...]
image filename
annotations[0].filename: /dataset/safety-detection/ee1fba93-a5f0-4c8b-8496-
ce7605914651.jpg
image size [width, height, depth]
annotations[0].size: [450, 330, 3]
bounding box #0 label
annotations[0].objects[0].label: helmet
# bounding box #0 position [xmin, ymin, xmax, ymax]
annotations[0].objects[0].bbox: [111, 16, 205, 106]
annotations[0].objects[1].label: helmet
annotations[0].objects[1].bbox: [257, 42, 340, 140]
annotations[0].objects[2].label: safety_vest
annotations[0].objects[2].bbox: [40, 105, 215, 291]
annotations[0].objects[3].label: safety_vest
annotations[0].objects[3].bbox: [207, 124, 382, 309]
```

workspace_path (string)

Temporary workspace path recommended to be used in all training life cycles.

Example: "/tmp/workspace"

params (dict)

Hyper parameters for training. These parameters are available to the custom model, but they are not required.

- Object detection example:

```
{ 'max_iter' : 4000, 'learning_rate' : 0.001, 'weight_decay' : 0.0005,
  'momentum' : 0.9 , 'traintest_ratio' : 0.8 }
```

- Classification example:

```
{ 'max_iter' : 4000, 'learning_rate' : 0.001, 'weight_decay' : 0.0005,
  'test_iteration' : 100, 'test_interval' : 20}
```

Output:

None

def onTraining(self, monitor_handler)

Callback for training.

Input

monitor_handler (MonitorHandler): Handler for train/test status monitoring.

Output

None

def onCompleted(self, model_path)

Callback for training completed. A training task is terminated either with `onCompleted()` or with `onFailed()`. You need to save the trained model in this callback.

Input

model_path (String): The absolute model path and file.

Output

None

def onFailed(self, train_status, e, tb_message):

Callback for training failed. A train task is terminated either with `onCompleted()` or with `onFailed()`

Input

***train_status* (string)**

Training status when the failure occurred.

***e* (Exception object)**

Programming exception object.

***tb_message* (string)**

Formatted traceback message.

Output

None

Monitoring and reporting statistics

The `onTraining` API passes a `monitor_handler` object. This object provides callbacks to report both training and test messages back to IBM Maximo Visual Inspection. Depending on the type of training being performed, classification or object detection, the appropriate callback must be used.

Object detection callbacks

Use this callback when the custom model is trained for object detection.

- [“def updateTrainMetrics\(current_iter, max_iter, loss_cls, loss_bbox, epoch\)” on page 76](#)
- [“def updateTestMetrics\(mAP\)” on page 76](#)

def updateTrainMetrics(current_iter, max_iter, loss_cls, loss_bbox, epoch)

Handler for status updates from the training process. This should be called **actively** by your custom code to post training status to the IBM Maximo Visual Inspection user interface.

Input

current_iter (int)

Current iteration in the epoch

max_iter (int)

Maximum iterations in one epoch

loss_cls (float)

Training loss of classification

loss_bbox (float)

Training loss of bounding box prediction

epoch (int)

Current training epoch

Example

```
monitor_handler.updateTrainMetrics(current_iter, max_iter, loss_cls,  
    loss_bbox, epoch)
```

Output

None

def updateTestMetrics(mAP)

Handler for status updates from the testing process. This should be called **actively** by your custom code to post testing status to the IBM Maximo Visual Inspection user interface.

Input

mAP (float): Testing mean average precision

Example

```
monitor_handler.updateTestMetrics(mAP)
```

Output

None

Classification callbacks

Use this callback when the custom model is trained for image classification.

- [“def updateTrainMetrics\(current_iter, max_iter, loss, epoch\)” on page 76](#)
- [“def updateTestMetrics\(current_iter, accuracy, loss, epoch\)” on page 77](#)

def updateTrainMetrics(current_iter, max_iter, loss, epoch)

Handler for status updates from the training process. This should be called **actively** by your custom code to post training status to the IBM Maximo Visual Inspection user interface.

Input

current_iter (int)

Current iteration in the epoch

max_iter (int)

Maximum iterations in one epoch

loss (float)

Training loss

epoch (int)

Current training epoch

Example

```
monitor_handler.updateTrainMetrics(current_iter, max_iter, loss, epoch)
```

Output

None

def updateTestMetrics(current_iter, accuracy, loss, epoch)

Handler for status updates from the testing process. This should be called **actively** by your custom code to post testing status to the IBM Maximo Visual Inspection user interface.

Input**current_iter (int)**

Current iteration in the epoch

accuracy (float)

Testing accuracy

loss (float)

Training loss

epoch (int)

Current training epoch

Example

```
monitor_handler.updateTrainMetrics(iter_num, accuracy, loss, epoch_num)
```

Output

None

Preparing a model that will be deployed in IBM Maximo Visual Inspection

If your custom model will be deployed in the IBM Maximo Visual Inspection framework, your custom model must meet the following requirements.

After the model is properly prepared, import it to IBM Maximo Visual Inspection by navigating to the **Models** page and clicking **Import .zip file**. To deploy the model, on the **Models** page, select the model and click **Deploy model**.

Custom model requirements:

- It must be TensorFlow or PyTorch based.
- It must conform to Python 3. Any trained custom models from releases prior to Version 1.3.0 will not work if the custom model only supports Python 2.
- It must implement the MyDeploy Python class.
 - The MyDeploy implementation must reside in a file named `deploy.py` in the top level directory of the zip file contents.
 - The following import must be added to the `deploy.py` file in order to define the deploy callbacks:

```
from deploy_interface import DeployCallback
```

- The class name must be MyDeploy.

MyDeploy Template:

```
class MyDeploy(DeployCallback):
    def __init__(self):
        pass
    def onModelLoading(self, model_path, labels, workspace_path):
        pass
    def onTest(self):
        pass
    def onInference(self, image_url, params):
        pass
    def onFailed(self, deploy_status, e, tb_message):
        pass
```

class MyDeploy(DeployCallback)

Use the MyDeploy API to prepare a TensorFlow or PyTorch model that will be deployed in IBM Maximo Visual Inspection.

Template

This is a template you can use for the MyDeploy API:

```
class MyDeploy(DeployCallback):
    def __init__(self):
        pass
    def onModelLoading(self, model_path, labels, workspace_path):
        pass
    def onTest(self):
        pass
    def onInference(self, image_url, params):
        pass
    def onFailed(self, deploy_status, e, tb_message):
        pass
```

def onModelLoading(self, model_path, labels, workspace_path)

Callback for load model.

Input

model_path (string)

Model path. The model must be decompressed before this callback.

workspace_path (string)

Temporary workspace path recommended to be used in all deploy activities.

labels (dict)

The label index to name mapping.

Example: {1: 'safety_vest', 0: 'helmet', 2: 'no_safety_vest', 3: 'no_helmet'}

Output:

None

def onTest(self)

Test API interface with a custom message. This method is used to test responsiveness of the deployed model, and can return any arbitrary string as a response.

Input

None

Output

message (string): Output message.

def onInference(self, image_url, params)

Inference with a single image.

Input

image_url (string)

Path of the image for inference.

params (dict)

Additional inference options.

heatmap (string)

Request a heat map. This is only supported for classification. Possible values:

- "true" : A heat map is requested.

- "false" : A heat map is not requested.

conf_threshold (float)

Confidence threshold. Value in the range 0.0 - 1.0, to be treated as a percentage. Only results with a confidence greater than the specified threshold are returned. The smaller confidence threshold you specify, the more results are returned. If you specify 0, many, many results will be returned because there is no filter based on the confidence level of the model.

Output (classification)

result({"label": "apple", "confidence": 0.9, "heatmap": "_value_"}): predicted label and its score

label (string): predicted label name

confidence (float): number for certainty. between 0 and 1

heatmap (string): heatmap return

Output (object detection)

result([{"confidence": 0.95, "label": "badge", "ymax": 145, "xmax": 172, "xmin": 157, "ymin": 123}]): predicted results in list

confidence(float): number for certainty. between 0 and 1

label(string): predicted label name

ymax(int): the max Y axis of bounding box

xmax(int): the max X axis of bounding box

ymin(int): the min Y axis of bounding box

xmin(int): the min X axis of bounding box

def onFailed(self, deploy_status, e, tb_message)

Callback for deploy failed. A deploy task is terminated with onFailed().

Input

deploy_status (string)

Deploy status when the failure occurred.

e (Exception object)

Programming exception object.

tb_message (string)

Formatted traceback message.

Output

None

Base models included with IBM Maximo Visual Inspection

You can use a *base model* to help train your model. You can choose your own Faster R-CNN or GoogLeNet model, or select one of the models that is included with IBM Maximo Visual Inspection.

Table 7. Base models included with IBM Maximo Visual Inspection			
Type	Number of images	Size	Source
Action	9532	310M	Stanford 40 actions
Flower	8189	348M	Visual Geometry Group
Food	1503	14.6M	https://ibm.box.com/s/cbocm5pvtuyudaoaypdwl3jaypets1hel
General	ImageNet dataset		ilsvrc12 http://image-net.org/download

Table 7. Base models included with IBM Maximo Visual Inspection (continued)

Type	Number of images	Size	Source
Landscape	1472	22.2M	Proprietary data set
Scene	108754	38G	SUN database
Vehicle	16185	1.9G	https://ai.stanford.edu/%7Ejkruse/cars/car_dataset.html

Deploying a trained model

Deploy a trained model to get it ready to use within IBM Maximo Visual Inspection or a different program, such as IBM Watson Machine Learning Community Edition. Deploying a model creates a unique API endpoint based on that model for inference operations.

About this task

Note:

Models trained in IBM Maximo Visual Inspection can also be exported and deployed using the [Chapter 14, “IBM Maximo Visual Inspection Edge,” on page 129](#).

To deploy the trained model, follow these steps:

Procedure

1. Click **Models** from the menu.
2. Select the model you want to deploy and click **Deploy**.
3. Specify a name for the model, and for models that were trained with the **Optimized for speed (Tiny YOLO v2)** model, choose the accelerator to deploy to. You can choose GPU, CPU, or Xilinx FPGA - 16 bit (technology preview).

If custom inference scripts have been uploaded, you can optionally select **Advanced deployment**. These options allow you to choose a custom inference script and to specify whether inference results are saved.

Notes:

- You can save inference results to a data set without choosing a custom inference script for this deployment. You can select **Advanced deployment** without having custom inference scripts.
- For image classification models trained prior to version 1.2.0.1, categories that are added as a result of an inference are listed by category UUID, rather than category name. To see category names instead, retrain the model, then run the inference again.

For information about uploading custom inference scripts, see [“Preprocessing and post-processing” on page 81](#).

If the option to save inference results to a data set is chosen, then an existing data set must be selected for the results to be saved to. Labels generated in the inference results will be marked as "inferred" when viewing the images in the Label objects view. This can be used to validate the performance of the model.

Note: Deploying a model to a Xilinx FPGA requires the [Xilinx Alveo U200 Accelerator card](#).

GPUs are used as follows:

- Multiple Faster R-CNN and GoogLeNet models are deployed to a single GPU. IBM Maximo Visual Inspection uses packing to deploy the models. That is, the model is deployed to the GPU that has the most models deployed on it, if there is sufficient memory available on the GPU. The GPU group can

be used to determine which deployed models share a GPU resource. To free up a GPU, *all* deployed models in a GPU group must be deleted (undeployed).

Note: IBM Maximo Visual Inspection leaves a 500MB buffer on the GPU.

4. Click **Deploy**. The **Deployed Models** page is displayed. When the model has been deployed, the status column displays **Ready**.
5. Click the deployed model to get the API endpoint, to view details about the model, such as the owner and the accuracy, and to test other videos or images against the model.

For information about using the API see [IBM Maximo Visual Inspection API documentation](#).

Note: When using the API, the smaller confidence threshold you specify, the more results are returned. If you specify 0, many, many results will be returned because there is no filter based on the confidence level of the model.

6. If necessary, you can delete a deployed model.

Related concepts

[Automatically deploying the newest model](#)

If you are using the production work flow within a project group, you can also turn on auto deploy. When auto deploy is turned on, IBM Maximo Visual Inspection automatically deploys a model when it is successfully trained and when it is marked as Production. IBM Maximo Visual Inspection automatically undeploys any deployed models when the associated trained model is marked as Rejected. Additionally, it tracks the latest model marked as Production or Untested and ensures that the latest production-ready model is deployed for a project group.

[Working with the user interface](#)

The IBM Maximo Visual Inspection user interface is made up of these basic parts: the navigation bar, the side bar, the action bar, the data area, and the notification center.

[Understanding metrics](#)

IBM Maximo Visual Inspection provides several metrics to help you measure how effectively your model has been trained.

Related information

[Vision Service API documentation](#)

Preprocessing and post-processing

You can upload customizations that enable you to perform operations before and after each inference operation with no manual intervention.

Note: Action detection models are not supported.

- [“custom.py Template” on page 81](#)
- [“Optional requirements.txt file” on page 82](#)
- [“Deploying a model with preprocessing, post-processing, or both” on page 82](#)

custom.py Template

Use this template to generate one Python file named `custom.py`, which can contain instructions for preprocessing, postprocessing, or both. This file must conform to Python 2 for all model types except custom models, which require Python 3.

Important: This file must be packaged in a zip file with `custom.py` in the top level directory of the zip file.

Other files, such as additional Python files, shell scripts, and images can also reside in the zip file. If the customization script requires Python modules aside from those built-in modules in Python, you can create a `requirements.txt` file, which contains a list of modules to be installed by using pip. The template contains this information:

class CustomInference

The only Python class in the file. It must be named CustomInference and holds the "pre" and "post" callouts.

onPreProcessing

If defined, this function must be in CustomInference. This function will be called before inference is run on the image.

onPostProcessing

If defined, this function must be in CustomInference. This function will be called after inference is run on the image.

```
class CustomInference:
# Callout for inference pre-processing. Will be called before the
# actual inference on the "image"
#
# Input:
#   image:    Image represented as a NumPy array that inference is to be
#             performed on.
#
# params:    To be used for additional parameters. This will be
#             a list of key/value pairs.
#
# Output:
#   image:    Return the image represented as a NumPy array that is to
#             be used for inference. This array may be manipulated
#             in this function, or it may be the same exact NumPy array.
#
def onPreProcessing(self, image, params):
    return image
# Callout for the inference post-processing. Will be called
# after the image has been inferred.
#
# Input:
#   Image:    Image represented as a NumPy array that inference is to be
#             performed on.
#
# results:    JSON of the inference results. The JSON will be
#             dependent on the type of inference.
#
# params:    To be used for additional parameters. This will
#             be a list of key/value pairs
#
# Output:
#   results:  A json object that is a copy of the original
#             inference results. However, if the callout
#             intends to return additional information, that
#             information can be returned in the json results
#             under the key "user".
#
def onPostProcessing(self, image, results, params):
    return results
```

Optional requirements.txt file

If the customization script requires Python modules aside from those built-in modules in Python, you can create a requirements.txt file, which contains a list of modules to be installed by using pip.

Example:

```
sseclient==0.0.19
tflearn==0.3.2
keras==2.2.4
```

Deploying a model with preprocessing, post-processing, or both

To deploy a model that will use additional processing, you will upload the custom zip file, then specify it on the deploy:

1. Navigate to the **Custom assets** page and upload the zip file that contains custom.py. For Asset type, select **Custom inference script**.

2. Navigate to the model you want to deploy and click **Deploy model**. In the upper right corner, select **Advanced deployment**.
3. For **Custom inference script**, select the inference script that you want to use, specify what you want done with the inference results, and click **Deploy**.

Note:

- Inference results can be saved even if you do not choose a custom inference script.
- Inference results for videos are not saved.

Testing a model

After deploying your model, you should test it against other images and videos to make sure that it works as expected.

Procedure

1. Click **Deployed Models** from the menu.
2. Click the deployed model you want to test. The model opens in the **Deployed model** page.
3. Use the **Test Model** area to upload images and videos, one at a time. If you provide a DICOM image, it will be converted to PNG before inferencing.

If you are testing an action detection model, optionally set the following values:

Generate annotated video

Select this option if you want to export the results, including the annotated video. If you are testing with a video, select annotations with dots or bounding boxes. For videos with multiple objects and movement of objects, dots are recommended.

Minimum action duration (frames)

Specify the minimum number of consecutive frames in which an action must be detected, with a confidence higher than the specified **Confidence threshold**, in order for it to be identified in the inference. For example, if this is set to 20 frames and the confidence threshold is 60%, the only actions that are returned are at least 20 frames long and have a confidence level of at least 60%.

Confidence threshold

Specify the minimum confidence level for returned actions. For example, if you set the value to 60%, only actions that have at least a 60% confidence threshold are returned.

4. The results are shown on the bottom of the window.

If you used an image to test an image classification model

The test result displays the uploaded picture with the resultant heat map overlayed, and gives the classification and the confidence of the classification. Multiple classes are returned with the decreasing levels of confidence for the different classes. The heat map is for the highest confidence classification and can help you determine whether the model has correctly learned the features of this classification. To hide classes with a lower confidence level, use the **Confidence threshold** slider.

The red area of the heat map corresponds to the areas of the picture that are of highest relevance. Use the slider to change the opacity of the heat map. Because the heat map is a square, the test image is compressed into a square. This might cause the image to look distorted, but it will reliably show you the areas that the algorithm identified as relevant.

If you used an image to test an object detection model

The identified objects are labeled in the image, with the calculated precision.

If you used a video to test an object detection model

Before providing the video, the annotation of dots or bounding boxes was selected. The video is processed, then the processed video is displayed, with a list of all of the objects on the right. As you watch the processed video, the identified objects are labeled as they appear in the video. Objects are labeled with a dot at the center of the object or bounding box, with the name displayed

next to the annotation. Polygon annotations are not used in the video object test, even if the model is trained for segmentation.

If you click an object in the list, it takes you to that point in the video. Processing the video might take a while, depending on its size.

The inference might take a long time to complete; however, you can run multiple inferences simultaneously. Additionally, you do not have to stay on the deployed model details page. If you leave the page, a notification window opens, where you can watch the progress. Clicking the link in this window loads the inference results section in the deployed model details page.

To download the result, click **Export result** in the Results section. A ZIP file is downloaded to your system. This file contains the original video, a JSON file that contains the result information, and the processed video with object labels added as annotations.

When you close the results area for an inference, the results are not removed. They are saved for seven days, unless you delete them. To access the results of previous inferences, click **Results history** in the Test Model section of the Deployed Models page. You can open or delete any of the saved results.

Note: The video object preview does not support non-ascii labels. This is a limitation of the module that generates the displayed label from the label name. The result of the conversion of non-ascii labels will be a label that is all question marks: "?????".

If you used a video to test an action detection model

The video is processed, then as you watch the processed video, the identified actions are output, along with the confidence and start and end times, as they appear in the video. Processing the video might take a while, depending on its size.

The inference might take a long time to complete; however, you can run multiple inferences simultaneously. Additionally, you do not have to stay on the deployed model details page. If you leave the page, a notification window opens, where you can watch the progress. Clicking the link in this window loads the inference results section in the deployed model details page.

The identified actions are grouped by action tag. To see individual actions that were discovered, expand the action tag. Clicking on an action moves the video preview to the start of that action.

To download the result, click **Export result** in the Results section. A ZIP file is downloaded to your system. This file contains the original video, a CSV file that contains the result information, and if the option to generate the annotated video was selected when the inference operation was started, the processed video with action labels added as annotations.

When you close the results area for an inference, the results are not removed. They are saved for seven days, unless you delete them. To access the results of previous inferences, click **Results history** in the Test Model section of the Deployed Models page. You can open or delete any of the saved results.

5. If you are satisfied with the results, the model is ready to be used in production. Otherwise, you can refine the model by following the instructions in this topic: [“Refining a model” on page 84](#).

Related concepts

[Importing, exporting, and downloading IBM Maximo Visual Inspection information](#)

You can import and export IBM Maximo Visual Inspection models and data sets. This allows you to save them for archiving then use them later, use them on a different IBM Maximo Visual Inspection install, and so on.

Refining a model

After deploying a model, you can improve its accuracy by supplying more data. There are several methods you can use to add more data to the model.

About this task

You can add more data by using any combination of the following options:

Procedure

1. Upload new images or videos to the data set and classify or label them as appropriate.
2. For an existing video, capture more frames and classify or label them as appropriate. Or, for action detection models, label more actions.
3. Use data augmentation. *Data augmentation* is the use of filters, such as blur and rotate, to create new versions of existing images or frames. Augmentation does not apply to full videos. It can be applied to a video's captured frames just as it is applied to images. When you use data augmentation, a new data set is created that contains all of the existing images, plus the newly generated images, which are marked as augmented. For instructions, see [“Augmenting the data set” on page 87](#).
4. For models trained for object detection, you can use the Auto label function to identify more objects in the existing data. See [“Automatically labeling objects” on page 85](#) for instructions.
5. When deploying an object detection model, you can choose **Advanced deployment** and specify that inference results should be saved to a data set. Objects labeled this way have the type "inferred" and the label is green. You can accept or reject the inferred labels. Accepted labels are considered manually added and are changed to blue.

Results

After adding more data, train the model again.

Automatically labeling objects

After deploying a model for object detection, you can improve its accuracy by using the Auto label function. This function uses the labels in the deployed model to generate new labels in the data set; increasing the number of images that are labeled in the data set. The updated data set can be used to train a new, more accurate model.

About this task

It will improve the number of images that are labeled in the dataset which can then be used to train a new model that is more accurate.

Notes:

- You can automatically label images or videos that have not had labels manually added. If any labels have been manually added, that image or frame is skipped.
- Any automatically added labels that are saved or edited are converted to manual labels.
- If images or frames have labels that have only been added through the auto label function, those images and frames are reprocessed. The previous labels are removed and new labels are added.
- If you use a trained Detectron model with segmentation turned on to generate the labels, polygons are used instead of rectangular boxes.
- When also augmenting images, it is recommended that you accept or reject labels before augmenting the data set because auto label confidence levels are not preserved in augmented images.

Automatically labeling objects in a data set

When you auto label a data set, an existing trained model is used to generate labels for images and video frames that have not been manually labeled.

About this task

Notes:

- You can automatically label images or videos that have not had labels manually added. If any labels have been manually added, that image or frame is skipped.
- If images or frames have labels that have only been added through the auto label function, those images and frames are reprocessed. The previous labels are removed and new labels are added.

- When auto labeling a data set, only images and frames are auto labeled. Therefore, any videos that do not have captured frames are skipped. For instructions to automatically add labels to a video, see [“Automatically labeling videos” on page 86](#).

Follow these steps to generate new labels in the data set.

Procedure

1. Open the data set that you want to add more data to and select **Auto label**.
2. Choose the appropriate settings, then click **Auto label**.
3. Labels are added to existing images or video frames that have not been manually labeled. By default, the automatically added labels are light red. For videos, if frames have already been captured, those frames are used for auto labeling. If frames have not been captured, the video is ignored.
4. Review the automatically added labels on the Data Set page. You can manipulate (move or resize) the labels that were automatically generated. You can also save or reject individual labels, or you can reject them all by selecting **Clear all**. Saving or manipulating a label converts it to a manually added label. Rejecting a label deletes it. If you run **Auto label** again, any images or frames that now have manually added labels are skipped.

You can use the confidence filter in the sidebar to review labels by the confidence level assigned by the model used to auto label the data set. For example, you could filter low confidence labels, which are likely wrong, and easily reject them, or filter on high confidence labels to quickly accept labels that are probably accurate.

Note: Auto labels that were created before IBM PowerAI Vision 1.1.5 will not have a confidence value. These auto labels will be treated as 0% confidence. Therefore, when using the confidence filter, they will only show up if the minimum confidence is set to 0.

Automatically labeling videos

When using the auto label function on a data set, only frames and images are processed. Videos are ignored. However, you can run the auto label function on an individual video.

About this task

Note: Any frames that were previously captured by using auto capture and were not manually labeled are deleted before auto labeling. This helps avoid labeling duplicate frames. Manually captured frames are not deleted.

Follow these steps to run the auto label function on a video.

Procedure

1. Open the data set that contains the video.
2. Select the video and click **Label objects**.
3. Click **Auto label**, choose the appropriate settings, then click **Auto label**.

Frames are captured at the specified interval and then the specified trained model is used to process the frames. When an object is identified with the specified confidence threshold, it is labeled. By default, the automatically added labels are light red.

4. Review the automatically added labels on the Data Set page. You can manipulate (move or resize) the labels that were automatically generated. You can also save or reject individual labels, or you can reject them all by selecting **Clear all**. Saving or manipulating a label converts it to a manually added label. Rejecting a label deletes it. If you run **Auto label** again, any images or frames that now have manually added labels are skipped.

You can use the confidence filter in the sidebar to review labels by the confidence level assigned by the model used to auto label the data set. For example, you could filter low confidence labels, which are likely wrong, and easily reject them, or filter on high confidence labels to quickly accept labels that are probably accurate.

Note: Auto labels that were created before IBM PowerAI Vision 1.1.5 will not have a confidence value. These auto labels will be treated as 0% confidence. Therefore, when using the confidence filter, they will only show up if the minimum confidence is set to 0.

Augmenting the data set

After deploying a model, you can improve the model by using data augmentation to add modified images to the data set, then retraining the model. *Data augmentation* is the use of filters, such as blur and rotate, to create new versions of existing images or frames. Augmentation does not apply to full videos. It can be applied to a video's captured frames just as it is applied to images. When you use data augmentation, a new data set is created that contains all of the existing images, plus the newly generated images, which are marked as augmented.

About this task

Notes:

- When also using auto label, it is recommended that you accept or reject labels before augmenting the data set because auto label confidence levels are not preserved in augmented images.
- Augmented images contain any labels and categories identified in the original image.
- The Tiny YOLO v2 model requires that images have unique object anchors - bounding box location and size. Using augmentations with color, noise, sharpen, or blur will generate modified images with identical bounding boxes, which causes a training failure for this type of model.

To augment a data set, follow these steps:

Procedure

1. Open the data set for a deployed model.
2. Select the images to use for augmentation, then click **Augment data**. If you select a video, every captured frame is used for augmentation. If you select some, but not all, frames in a video, only the selected frames are used for augmentation.
3. Choose any combination of filters to apply to your data set, then click **Continue**.

Each filter generates one or more new versions of each selected image; the filters are not cumulative. For example, if you select **Sharpen** and **Flip horizontal**, six new images are generated; one flipped and five sharpened.

When you select a filter, you can see an example of what that filter would do to an image. This sample image is **not** a live preview of the filter. It is an example of what an image might look like with that filter applied. Some filters, such as Blur and Sharpen, have additional settings you can choose.

4. Specify a name for the new data set and click **Create data set**.
5. The new data set, containing the original images, is created immediately. The augmented images are added after all processing completes. After the new data set is created, you can train a model based on the new data set. See this topic for instructions: [“Training a model” on page 68](#).

Augmentation settings

These settings are available when augmenting data.

Each filter generates one or more new versions of each selected image; the filters are not cumulative. For example, if you select **Sharpen** and **Flip horizontal**, six new images are generated; one flipped and five sharpened.

Note: When you select a filter, you can see an example of what that filter would do to an image. This sample image is **not** a live preview of the filter. It is an example of what an image might look like with that filter applied.

Blur

Select the maximum amount of Gaussian and motion blur. Gaussian blur makes the entire image appear out of focus by reducing detail and noise. Motion blur makes the image appear as if it (or the camera) is in motion.

Five new images are generated in the range of each nonzero selection. For example, if Motion = 25 and Gaussian = 10, then five images are generated by applying a motion blur filter in random strengths in the range 0-25, and five additional images are generated by applying a Gaussian blur filter in the range 0-10.

Sharpen

Select the maximum amount of sharpening to apply. Some noise will be introduced. Five new images are generated in the specified range. For example, if Sharpness = 25, five new images are generated by applying the sharpen filter in random strengths in the range of 0-25.

Color

Select the maximum amount of change in the image's brightness, contrast, hue, and saturation. Five new images are generated by using randomly selected values in the selected ranges. The resultant values can be either positive or negative.

For example, if Brightness = 30, Contrast = 15, Hue = 5, and Saturation = 10, five images are generated that have brightness changed by (-30, 30)% , contrast is changed by (-15, 15)%, and so on.

Crop

Select the maximum percentage of the image that should remain. For example, selecting 25 means that at most 25% of the original image remains and 75% is removed. Five new images will be generated that are cropped in the selected range. The crop is centered at a random point.

For example, if Crop = 25, five images are generated cropped to retain 100% - 25% of the original image.

Vertical flip

Create a new image by flipping the existing image across the top edge. That is, the top of the image becomes the bottom.

Horizontal flip

Create a new image by flipping the existing image across the side edge. That is, the left side of the image becomes the right side.

Rotate

Select the maximum value of rotation for the new images. Rotation can be either clockwise or counter-clockwise. Five new images are generated that are rotated by this amount. For example, if this value is 45, five new images are generated that are rotated either clockwise or counter-clockwise by a random number in the range 0-45.

Noise

Select the maximum amount of noise to add to the new images, specified as a percentage of what IBM Maximo Visual Inspection determines to be a reasonable amount of noise for the images to remain usable. Therefore, if you select 100, none of the generated images will have 100% noise added. Instead, the output images will possibly have the maximum amount of noise added while still remaining usable.

Five new images are generated with noise added in the specified range. For example, if this value is 25, five new images are created with a random amount of noise added in the range 0 - 25% of a reasonable amount of noise.

Importing, exporting, and downloading IBM Maximo Visual Inspection information

You can import and export IBM Maximo Visual Inspection models and data sets. This allows you to save them for archiving then use them later, use them on a different IBM Maximo Visual Inspection install, and so on.

- [“Exporting” on page 89](#)
- [“Downloading assets” on page 90](#)
- [“Importing” on page 90](#)

Exporting

When exporting a model or data set in the user interface, a zip file is created and transferred to the browser for saving. A notification is presented to indicate that the download is in progress, since the time to transfer large models and data sets depends on network speeds.

Export a data set

To export a data set, open the Data sets page, open the data set you want to export, then click **Export** in the action bar. The data set is saved in your default download directory as *data_set_name.zip*. This zip file contains the images as well as any tags or categories you have assigned.

Notes:

- When exporting a data set, any objects that are not used in the data set are not contained in the exported data set. Therefore, they are not included when the data set is imported.

For example, if the object or label "car" is defined but is not used in any of the images in the data set, the exported data set does not include the "car" object or label. When the data set is imported, the "car" object or label is not created.
- In IBM PowerAI Vision 1.1.1, any information about augmented images is lost on export. Therefore, if the data set is later imported (regardless of the product version), the augmented images will be in the data set, but they will no longer be marked as augmented.

Export a model

When you export a model, a zip file is generated that contains the model and some additional files, depending on the model type. The generated zip file is not encrypted or password protected.

To export a model, open the Models page, select the model you want to export, then click **Export** in the left pane. The model is saved in your default download directory as *character_string.zip*; where *character_string* is randomly generated by the system.

Note:

If the model is not a Custom model that was imported from the Models page, the exported model is not supported for use outside of IBM Maximo Visual Inspection. It *can* be imported into IBM Maximo Visual Inspection Edge and deployed with IBM Maximo Visual Inspection Edge.

It is not recommended that you use an exported model with an earlier version of the product than it was exported from. Additionally, a model from a prior version will not have support for features that were added to later versions of the product. That is, if you export a model from version x.1 and import it into x.2, features that were added in x.2 will not be supported on the imported model.

Export the results of action detection or object detection inference

You can download the results from an inference that was run in the last seven days on an action detection or object detection model by following these steps:

1. Open the Deployed model page, then click the name of the model.
2. Scroll to the Test Model section and click **Results history**. A window opens that shows all inference results from the past seven days.
3. Select the results that you want to view and click **Load results**.
4. Scroll down to any results that you want to download and click **Export result**.
 - For action detection: A ZIP file is downloaded to your system. This file contains the original video, a CSV file that contains the result information, and if the option to generate the annotated video was selected when the inference operation was started, the processed video with action labels added as annotations.
 - For object detection: A ZIP file is downloaded to your system. This file contains the original video, a JSON file that contains the result information, and the processed video with object labels added as annotations.

Downloading assets

When you train certain types of models, additional downloadable assets are generated. To download these assets, open the Deployed models page and click the appropriate model.

Core ML assets

When you download Core ML assets, the model is downloaded as an .mlmodel file. This file can be deployed onto an Xcode project and can be used for inference directly on the device.

TensorRT assets

When you download TensorRT assets, the model is downloaded as a .tar.gz file.

Importing

Import a data set

1. Navigate to the Data sets page.
2. Drag and drop an exported data set .zip file onto the **Create** box.

Important: After the upload starts, do not close the IBM Maximo Visual Inspection tab or refresh the page. Doing so stops the upload.

3. After the upload completes, the data set has its original name.

Notes:

- In IBM PowerAI Vision 1.1.1, any information about augmented images is lost on export. Therefore, if the data set is later imported (regardless of the product version), the augmented images will be in the data set, but they will no longer be marked as augmented.
- The data set associated with a model is not preserved when it is exported. Therefore, for imported models, the Data set field is set to "Not found".

Import a model

Instead of using IBM Maximo Visual Inspection to train a new model, you can *import* a model that was previously trained with IBM Maximo Visual Inspection, and was then exported. This lets you streamline data processing by offloading training tasks and allowing you to reuse models on multiple systems. After the model is imported to the **Models** page, you can deploy it in IBM Maximo Visual Inspection. Use the information in this topic to prepare a custom model that will be deployed in IBM Maximo Visual Inspection: [“Preparing a model that will be deployed in IBM Maximo Visual Inspection” on page 77.](#)

1. Navigate to the Models page.
2. Drag and drop a previously exported model .zip file onto the **Import** box.

Important: After the upload starts, do not close the IBM Maximo Visual Inspection tab or refresh the page. Doing so stops the upload.

3. After the upload completes, the model has its original name.

IBM Maximo Visual Inspection REST APIs

You can use REST APIs to work with IBM Maximo Visual Inspection data sets and models, such as performing training and deployment. You can also use them to perform administrative tasks, such as monitoring events. These APIs allow you to bypass the user interface and automate IBM Maximo Visual Inspection processes or solutions.

For information about using the APIs see [IBM Maximo Visual Inspection API documentation](#).

There are also examples of using the APIs for different actions, published [here](#).

Understanding metrics

IBM Maximo Visual Inspection provides several metrics to help you measure how effectively your model has been trained.

To understand these metrics, you must understand these terms:

True positive

A *true positive* result is when IBM Maximo Visual Inspection correctly labels or categorizes an image. For example, categorizing an image of a cat as a cat.

False positive

A *false positive* result is when IBM Maximo Visual Inspection labels or categorizes an image when it should not have. For example, categorizing an image of a cat as a dog.

True negative

A *true negative* result is when IBM Maximo Visual Inspection correctly does not label or categorize an image. For example, not categorizing an image of a cat as a dog.

False negative

A *false negative* result is when IBM Maximo Visual Inspection does not label or categorize an image, but should have. For example, not categorizing an image of a cat as a cat.

Of course, for a model in production, the values for true negative / positive and false negative / positive can't accurately be known. These values are the expected values for these measurements.

- [“Metrics for image classification \(Optimized for accuracy\)” on page 91](#)
- [“Metrics for object detection” on page 92](#)
- [“Metrics for object detection using the Tiny YOLO model \(Optimized for speed\)” on page 93](#)
- [“Metrics for object detection using Segmentation \(Optimized for Detectron\)” on page 93](#)
- [“Metrics for custom models” on page 93](#)
- [“Metrics for action detection models” on page 94](#)

Metrics for image classification (Optimized for accuracy)

Accuracy

Measures the percentage of correctly classified images. It is calculated by $(\text{true positives} + \text{true negatives}) / (\text{true positives} + \text{true negatives} + \text{false positives} + \text{false negatives})$.

PR curve (Advanced)

The precision-recall (PR) curve plots precision vs. recall (sensitivity). Because precision and recall are typically inversely related, it can help you decide whether the model is appropriate for your needs. That is, do you need a system with high precision (fewer results, but the results are more likely to be accurate), or high recall (more results, but the results are more likely to contain false positives)?

Precision

Precision describes how "clean" the population of hits is. It measures the percentage of images that are correctly classified. That is, when the model classifies an image into a category, how often is it correct? It is calculated by $\text{true positives} / (\text{true positives} + \text{false positives})$.

Recall

The percentage of the images that were classified into a category, compared to all images that should have been classified into that category. That is, when an image belongs in a category, how often is it identified? It is calculated as $\text{true positives} / (\text{true positives} + \text{false negatives})$.

Confusion matrix (Advanced)

The confusion matrix is used to calculate the other metrics, such as precision and recall. Each column of the matrix represents the instances in a predicted class (those that IBM Maximo Visual Inspection marked as belonging to a category, for example). Each row represents the instances in an actual class. Therefore, each cell measures how many times an image was correctly and incorrectly classified.

You can view the confusion matrix as a table of values or a heat map. A heat map is a way of visualizing the data, so that the higher values appear more "hot" (closer to red) and lower values appear more "cool" (closer to blue). Higher values show more confidence in the model.

This matrix makes it easy to see if the model is confusing classes, or not identifying certain classes.

Metrics for object detection

Accuracy

Measures the percentage of correct image classifications. It is calculated by (true positives + true negatives) / all cases.

Loss vs. Iteration

The Loss vs. Iteration graph presents information about the training loss over the range of iterations used during training. There are two measurements of the "Train Loss":

- **CLS = Classification, Localization, Segmentation:** Combined error measurement of how accurately the trained model can split the original image into smaller regions, select (localize) the most interesting regions, and classify any objects in the region.
- **BBox = bounding box:** Measures how precisely the trained model can locate bounding box coordinates for any recognized object, compared to the test subset.

Mean Average precision (mAP)

The average over all classes of the maximum *precision* for each object at each *recall* value. Precision measures how accurate the model is. That is, the percent of the classified objects that are correct. Recall measures how well the model returns the correct objects. For example, out of 100 images of dogs, how many of them were classified as dogs?

To calculate this, first, the PR curve is found. Then, the maximum precision for each recall value is determined. This is the maximum precision for any recall value greater than or equal to the current recall value. For example, if the precision values range from .35 to .55 (and then never reach .55 again) for recall values in the interval .3 - .6, then the maximum precision for every recall value in the interval .3 - .6 is set to .55.

The mAP is then calculated as the average of the maximum precision values.

IoU (Intersection over union)

The accuracy of the location and size of the image label boxes.

It is calculated by the intersection between a ground truth bounding box and a predicted bounding box, divided by the union of both bounding boxes; where the intersection is the area of overlap, a *ground truth* bounding box is the hand drawn box, and the *predicted bounding box* is the one drawn by IBM Maximo Visual Inspection.

Confusion matrix (Advanced)

The confusion matrix is used to calculate the other metrics, such as precision and recall. Each column of the matrix represents the instances in a predicted class (those that IBM Maximo Visual Inspection marked as belonging to a category, for example). Each row represents the instances in an actual class. Therefore, each cell measures how many times an image was correctly and incorrectly classified.

You can view the confusion matrix as a table of values or a heat map. A heat map is a way of visualizing the data, so that the higher values appear more "hot" (closer to red) and lower values appear more "cool" (closer to blue). Higher values show more confidence in the model.

This matrix makes it easy to see if the model is confusing classes, or not identifying certain classes.

PR curve (Advanced)

The precision-recall (PR) curve plots precision vs. recall (sensitivity). Because precision and recall are typically inversely related, it can help you decide whether the model is appropriate for your needs. That is, do you need a system with high precision (fewer results, but the results are more likely to be accurate), or high recall (more results, but the results are more likely to contain false positives)?

Precision

Precision describes how "clean" the population of hits is. It measures the percentage of objects that are correctly identified. That is, when the model identifies an object, how often is it correct? It is calculated by $\text{true positives} / (\text{true positives} + \text{false positives})$.

Recall

The percentage of the images that were labeled as an object, compared to all images that contain that object. That is, how often is an object correctly identified? It is calculated as $\text{true positives} / (\text{true positives} + \text{false negatives})$.

Metrics for object detection using the Tiny YOLO model (Optimized for speed)

Accuracy

Measures the percentage of correctly classified objects. It is calculated by $(\text{true positives} + \text{true negatives}) / (\text{true positives} + \text{true negatives} + \text{false positives} + \text{false negatives})$.

Metrics for object detection using Segmentation (Optimized for Detectron)

Confusion matrix (Advanced)

The confusion matrix is used to calculate the other metrics, such as precision and recall. Each column of the matrix represents the instances in a predicted class (those that IBM Maximo Visual Inspection marked as belonging to a category, for example). Each row represents the instances in an actual class. Therefore, each cell measures how many times an image was correctly and incorrectly classified.

You can view the confusion matrix as a table of values or a heat map. A heat map is a way of visualizing the data, so that the higher values appear more "hot" (closer to red) and lower values appear more "cool" (closer to blue). Higher values show more confidence in the model.

This matrix makes it easy to see if the model is confusing classes, or not identifying certain classes.

Loss vs. Iteration

The Loss vs. Iteration graph presents information about the training loss over the range of iterations used during training. There are two measurements of the "Train Loss":

- **CLS = Classification, Localization, Segmentation:** Combined error measurement of how accurately the trained model can split the original image into smaller regions, select (localize) the most interesting regions, and classify any objects in the region.
- **BBox = bounding box:** Measures how precisely the trained model can locate bounding box coordinates for any recognized object, compared to the test subset.

PR curve (Advanced)

The precision-recall (PR) curve plots precision vs. recall (sensitivity). Because precision and recall are typically inversely related, it can help you decide whether the model is appropriate for your needs. That is, do you need a system with high precision (fewer results, but the results are more likely to be accurate), or high recall (more results, but the results are more likely to contain false positives)?

Precision

Precision describes how "clean" the population of hits is. It measures the percentage of objects that are correctly identified. That is, when the model identifies an object, how often is it correct? It is calculated by $\text{true positives} / (\text{true positives} + \text{false positives})$.

Recall

The percentage of the images that were labeled as an object, compared to all images that contain that object. That is, how often is an object correctly identified? It is calculated as $\text{true positives} / (\text{true positives} + \text{false negatives})$.

Metrics for custom models

When a custom model is imported and deployed, the following metric is shown:

Accuracy

Measures the percentage of correct categorizations. It is calculated by $(\text{true positives} + \text{true negatives}) / (\text{true positives} + \text{true negatives} + \text{false positives} + \text{false negatives})$.

Metrics for action detection models

Accuracy

Measures the percentage of correctly detected actions. It is calculated by $(\text{true positives} + \text{true negatives}) / (\text{true positives} + \text{true negatives} + \text{false positives} + \text{false negatives})$.

Precision

Precision describes how "clean" the population of hits is. It measures the percentage of actions that are correctly identified. That is, when the model identifies an action, how often is it correct? It is calculated by $\text{true positives} / (\text{true positives} + \text{false positives})$.

Recall

The percentage of the video segments that were labeled as an action, compared to all segments in the video that contain that action. That is, how often is an action correctly identified? It is calculated as $\text{true positives} / (\text{true positives} + \text{false negatives})$.

Confusion matrix (Advanced)

The confusion matrix is used to calculate the other metrics, such as precision and recall. Each column of the matrix represents the instances in a predicted class (those that IBM Maximo Visual Inspection marked as belonging to a category, for example). Each row represents the instances in an actual class. Therefore, each cell measures how many times an image was correctly and incorrectly classified.

You can view the confusion matrix as a table of values or a heat map. A heat map is a way of visualizing the data, so that the higher values appear more "hot" (closer to red) and lower values appear more "cool" (closer to blue). Higher values show more confidence in the model.

This matrix makes it easy to see if the model is confusing classes, or not identifying certain classes.

Chapter 10. Creating and working with project groups

Project groups allow you to group trained models with the data sets that were used for training. This grouping is optional but is a useful way to organize related data sets. For example, project groups would be useful with a workflow that clones data sets as you refine labels and work toward a more accurate model. Project groups can be used with a production work flow strategy and automatic model deployment for even more functionality.

Project groups provide API shortcuts for certain trained model actions. That is, you can deploy or perform inferences on the most recently trained or deployed model without knowing the model ID. Instead, the APIs use project group IDs, which never change. This means that as better performing models are generated, your scripts can act on the latest model without needing to be updated.

Project groups track the latest trained model and the latest deployed model separately. If Production work flow is enabled, you can additionally add tags to denote a trained model (and its deployed instance) as production-ready or as untested. See [Chapter 11, “Production work flow,”](#) on page 97 for more information about these tags. When using project groups with the production work flow, you can use project group APIs to work with these models:

- Latest trained model in a project group
- Latest deployed model in a project group
- Latest trained model that is production-ready in a project group*
- Latest deployed model that is production-ready in a project group *
- Latest trained model that is untested in a project group*
- Latest deployed model that is untested in a project group*

*: Production work flow must be enabled for the project group.

Note: All models trained from any data set in a project group will automatically be associated with that project group.

- [“Working with project groups and project group assets”](#) on page 95
- [“Using the production work flow with project groups”](#) on page 96
- [“Automatically deploying models in project groups”](#) on page 96

Working with project groups and project group assets

Project groups can be created at any point in your work flow. To create a project group, click **Projects** in the navigation bar, then click **+**. After the project group is created, you can add resources (data sets and trained models) to it.

To delete a project group, from the Projects page, select the project name and click the trash can icon. None of the assets in the project are deleted, but they will no longer be associated with any project group.

Working with project group assets

To work with project group assets, navigate to the Projects page and click the name of the project group.

Add an asset

To add a data set or model, click **+**, specify the asset type, and select the asset to add. You can start typing the asset name to filter the available assets.

Note: An administrator can add assets to a project group that was created by a different user. However, the project group owner will not be able to see the added assets because only administrators can see resources created by other users. Because of that, the value for "Total items" on the Projects page might be larger than the number of items shown on a project's details page.

Remove an asset

To remove an asset, navigate to the project group, select all assets that you want to remove, and click remove. The assets are not deleted from IBM Maximo Visual Inspection. Additionally, each asset in a project group is independent. For example, If you remove a data set, none of the models derived from that data set are removed.

Notes:

- Any model trained from a data set in a project group is automatically added to that project group. However, any models that were trained from a data set before it was added to the project group must be added manually.
- Each data set or trained model can be a member of only one project group.

Using the production work flow with project groups

If Production work flow is enabled, project groups keep track of the most recently trained model that is marked Production and the most recently trained model that is unmarked. You can use an API to work with the latest deployed model that is marked Production or is Unmarked (untested). This simplifies your workflow because you never have to update the script to point to a different deployed model, and you do not have to manually track model names.

Note: Because the latest trained model is tracked separately from the latest deployed model, it is possible to train a new model and still be using an older model for inferences. This delineation can be reduced (almost eliminated) if you enable production work flow *and* auto deploy. With both of these flags set, the project group tries to keep the deployed models in sync with the trained model's latest trackers.

For details, see [Chapter 11, “Production work flow,” on page 97](#).

Automatically deploying models in project groups

If you are using the production work flow within a project group, you can also turn on auto deploy. When auto deploy is turned on, IBM Maximo Visual Inspection automatically deploys a model when it is successfully trained and when it is marked as Production. IBM Maximo Visual Inspection automatically undeploys any deployed models when the associated trained model is marked as Rejected. Additionally, it tracks the latest model marked as Production or Untested and ensures that the latest production-ready model is deployed for a project group. For details, see [“Automatically deploying the newest model” on page 98](#).

Related tasks

[Creating and working with data sets](#)

Before you can work with videos or images, you need to create a data set. A data set is a group of images, videos, or both that you will use to train a deployable model.

Chapter 11. Production work flow

You can use an API to enable the production work flow. This helps you track which models are ready for production, which failed testing, and which still need to be tested. If you are using the production work flow and project groups, you can use scripts to deploy or perform inferences on the most recently trained model of a specified status in the project group.

- [“Overview” on page 97](#)
- [“Enabling production work flow” on page 97](#)
- [“Setting a model's status” on page 97](#)
- [“Using the production work flow with project groups” on page 98](#)
- [“Using the production work flow with autodeploy” on page 98](#)
- [“Using the production work flow APIs for inferences” on page 98](#)

Overview

You can mark a trained model as one of the following:

- **Production** - The model has been deployed and tested and is ready for use.
- **Rejected** - The model has been deployed and tested but failed validation and should not be used.
- **Unmarked** - The model has not been deployed and tested. All newly trained models are Unmarked.

Note: In the API, this corresponds to a `production` status value of `untested`.

All states must be set manually; except that newly trained models are assigned a status of Unmarked. There are no rules enforced about state changes, so you can set any status on any trained model.

Enabling production work flow

Set `enforce_pwf` to `true` to enable production work flow. To set `enforce_pwf`, use the HTTP PUT verb to the endpoint `/projects/{project-UUID}` and include a JSON body of `{"enforce_pwf": "true"}`.

CURL example:

```
curl -kXPUT -H "x-auth-token: PAIV-AUTH_TOKEN-STRING"
https://PAIV-SERVER.COMPANY.COM/vision/api/projects/PAIV-PROJECT-UUID
-d '{"enforce_pwf": "true"}
```

For detailed information about the APIs, see [IBM Maximo Visual Inspection API documentation](#).

Setting a model's status

To set a model's status, follow these steps:

1. Navigate to the Models page.

Note: The "Production" status can also be set on the Deployed models page.

2. Select one or more models and click **Mark as**.
3. Select the appropriate status.

Note: If auto deploy is enabled, changing a model's status might result in the model being deployed or undeployed. See [“Automatically deploying the newest model” on page 98](#) for details.

When you set a trained model's status, the associated deployed model (if one exists) will have the same status.

Using the production work flow with project groups

If Production work flow is enabled, project groups keep track of the most recently trained model that is marked Production and the most recently trained model that is unmarked. You can use an API to work with the latest deployed model that is marked Production or is Unmarked (untested). This simplifies your workflow because you never have to update the script to point to a different deployed model, and you do not have to manually track model names.

Note: Because the latest trained model is tracked separately from the latest deployed model, it is possible to train a new model and still be using an older model for inferences. This delineation can be reduced (almost eliminated) if you enable production work flow *and* auto deploy. With both of these flags set, the project group tries to keep the deployed models in sync with the trained model's latest trackers.

Using the production work flow with autodeploy

If you are using the production work flow within a project group, you can also turn on auto deploy. When auto deploy is turned on, IBM Maximo Visual Inspection automatically deploys a model when it is successfully trained and when it is marked as Production. IBM Maximo Visual Inspection automatically undeploys any deployed models when the associated trained model is marked as Rejected. Additionally, it tracks the latest model marked as Production or Untested and ensures that the latest production-ready model is deployed for a project group. For details, see [“Automatically deploying the newest model” on page 98](#).

Using the production work flow APIs for inferences

Use the projects API to do predictions to the latest deployed model with a status of "production" or "untested": `/projects/{id}/models/{status}/predict`, where *status* is latest, production, or untested.

Examples:

Do a prediction on the latest deployed model

If PWF is set to enforce, the latest deployed model with a status of "production" is used.

```
/projects/123-456/models/latest/predict
```

Do a prediction on the latest deployed "production" model

```
/projects/123-456/models/production/predict
```

Do a prediction on the latest deployed "untested" (unmarked) model

```
/projects/123-456/models/untested/predict
```

For detailed information about the APIs, see [IBM Maximo Visual Inspection API documentation](#).

Automatically deploying the newest model

If you are using the production work flow within a project group, you can also turn on auto deploy. When auto deploy is turned on, IBM Maximo Visual Inspection automatically deploys a model when it is successfully trained and when it is marked as Production. IBM Maximo Visual Inspection automatically undeploys any deployed models when the associated trained model is marked as Rejected. Additionally, it tracks the latest model marked as Production or Untested and ensures that the latest production-ready model is deployed for a project group.

Note: If you do not mark a model as "production" or "rejected," and you manually deploy it, it will never be automatically undeployed by IBM Maximo Visual Inspection.

- [“Overview” on page 99](#)
- [“Enabling auto deploy” on page 100](#)

Overview

IBM Maximo Visual Inspection does the following when auto deploy is enabled:

When a model is newly trained

The latest trained model is marked Untested and it is automatically deployed. If there is another model marked Untested that is already deployed, the older deployed model is undeployed. The goal of this process is to help people who want to test the latest trained model.

When a model is marked as Production

When a trained model is marked as Production, it is automatically deployed. The goal of this process is to always keep the latest production-ready model deployed. If it is already deployed, the following happens:

1. The deployed model becomes the current deployed Production model.
2. Any other deployed Production model is undeployed.
3. IBM Maximo Visual Inspection finds the most recent trained model with Untested status and deploys it. If there is no other trained model marked as Untested, attempted inferences to the latest Untested model will fail.

When a model is changed from Untested to Rejected

When a trained model is marked as Rejected, the associated deployed model is undeployed.

When a model is changed from Production to Rejected

When a trained model is marked as Rejected, the associated deployed model is undeployed. IBM Maximo Visual Inspection finds the most recent trained model with Production status and deploys it. If there is no other trained model marked as Production, attempted inferences to the latest Production model will fail.

Although auto deploy tracks and deploys the latest model marked with each status, you can manually deploy additional models from the project group by using the Models page. However, those additional models will not be accessible via the API shortcuts.

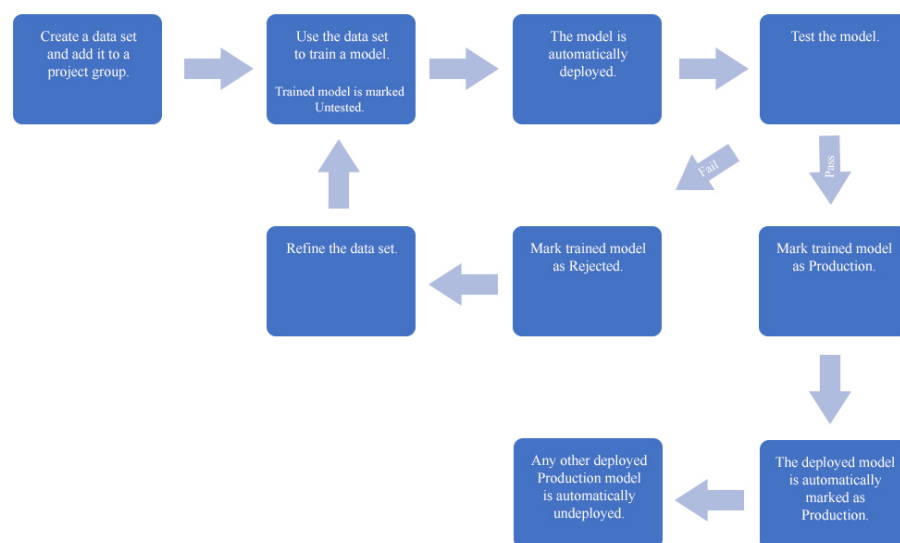


Figure 10. Auto deploy in IBM Maximo Visual Inspection

Enabling auto deploy

To enable auto-deploy, you need to set two pieces of information via the API. You can use CURL to set them.

```
curl -kXPUT -H "x-auth-token: insights-auth-token-value" https://insights-server.your_company.com/visual-insights/projects/{project-UUID} -d '{"enforce_pwf": "true", "auto_deploy": "true"}'
```

Note: Case is ignored for the true string, but the word must be true. For example, it will not work if you set the value as yes.

Chapter 12. Using IBM Maximo Visual Inspection

These fictional examples give step-by-step instructions of how to use IBM Maximo Visual Inspection to accomplish various tasks.

Example: Detecting objects in images

In this fictional scenario, you want to create a deep learning model to determine the make and model of a car caught by a traffic camera.

The image file used in this scenario is available for download here: [Download car image](#).

To create a deep learning model, you will perform the following steps:

1. [“Import images and create a data set” on page 101](#)
2. [“Labeling objects in an image” on page 101](#)
3. [“Training a model” on page 102](#)
4. [“Deploying a trained model” on page 103](#)

Import images and create a data set

First, create a data set and add images to it.

1. [Log in to IBM Maximo Visual Inspection](#).
2. Click **Data Sets** in the navigation bar to open the **Data Sets** page. There are several ways to create a new data set. We will create a new, empty data set.
3. From the **Data set** page, click the icon and name the data set Traffic camera.
4. To add an image to the data set, click the Traffic image data set and click **Import file** or drag the image to the + area.

Important: You cannot navigate away from the IBM Maximo Visual Inspection page or refresh until the upload completes. You can navigate to different pages within IBM Maximo Visual Inspection during the upload.

Labeling objects in an image

The next step is to label objects in the images. For object detection, you must have at minimum five labels for each object. We will create "Black car" and "White car" objects and will label at least five images as black cars, and at least five as white cars.

1. Select the images from your data set and click **Label Objects**.
 2. Create new object labels for the data set by clicking **Add new** by the Objects list. Enter Black car, click **Add**, then enter Black car, then click **OK**.
 3. Label the objects in the images:
 - a. The first image is open in the data area, with thumbnails of all the selected image on the left side. Select the correct object label, for example, "Black car".
 - b. Choose **Box** or **Polygon** from the bottom left, depending on the shape you want to draw around each object. Boxes are faster to label and train, but less accurate. Only Detectron models support polygons. However, if you use polygons to label your objects, then use this data set to train a model that does not support polygons, bounding boxes are defined and used. Draw the appropriate shape around the object.
 - c. Select the thumbnail of the next image to open it. Add the appropriate labels, and continue through the rest of the images.
- Do not label part of an object. For example, do not label a car that is only partially in the image.

- If an image has more than one object, you must label all objects. For example, if you have cars and motorcycles defined as objects for the data set, and there is an image with both cars and motorcycles in it, you must label the cars and the motorcycles. Otherwise, you decrease the accuracy of the model.
- Label each individual object. Do not label groups of objects. For example, if two cars are right next to each other, you must draw a label around each car.
- Draw the shape as close to the objects as possible. Do not leave blank space around the objects.
- You can draw shapes around objects that touch or overlap. For example, if one object is behind another object, you can label them both. However, it is recommended that you only label objects if the majority of the object is visible.
- Use the zoom buttons (+ and -) on the bottom right side of the editing panels to help draw more accurate shapes.

Note: If you are zoomed in on an image and use the right arrow key to move all the way to the right edge, you might have to click the left arrow key several times to start panning in the other direction.

- Shapes cannot extend off the edge of the image.
- After defining a shape, you can copy and paste it elsewhere in the same image or in a different image by using standard keyboard shortcuts. After you paste it, you can refine the shape by moving, adding, or removing points in the outline.

Note: To copy and paste a shape from one image to another, both images have to be available in the image carousel. From the data set, select all images that will share shapes, then click **Label objects**. All images will be listed in the image carousel in the left side of the Label objects window.

- **Labeling with polygons**

- To delete a point from an outline, ctrl+click (or cmd+click).
- To add a point to an outline, click the translucent white square between any two points on the outline.
- To move a point on the outline, click it and drag.

4. After all objects are labeled in all of the image, click **Done editing**.

Training a model

With all the object labels that are identified in your data set, you can now train your deep learning model. To train a model, complete the following steps:

1. From the **Data set** page, click **Train**.
2. Fill out the fields on the **Train Data set** page, ensuring that you select **Object Detection**. We will choose **Accuracy (faster R-CNN)** for **Model selection**.
3. Click **Train**.
4. (Optional - *Only supported when training for object detection.*) Stop the training process by clicking **Stop training > Keep Model > Continue**.

You can wait for the entire training model process complete, but you can optionally stop the training process when the lines in the training graph start to flatten out, as shown in the figure below. This is because improvements in quality of training might plateau over time. Therefore, the fastest way to deploy a model and refine the data set is to stop the process before quality stops improving.

Note: Use early stop with caution when training segmented object detection models (such as with Detectron), because larger iteration counts and training times have been demonstrated to improve accuracy even when the graph indicates the accuracy is plateauing. The precision of the label is can still being improved even when the accuracy of identifying the object location stopped improving.

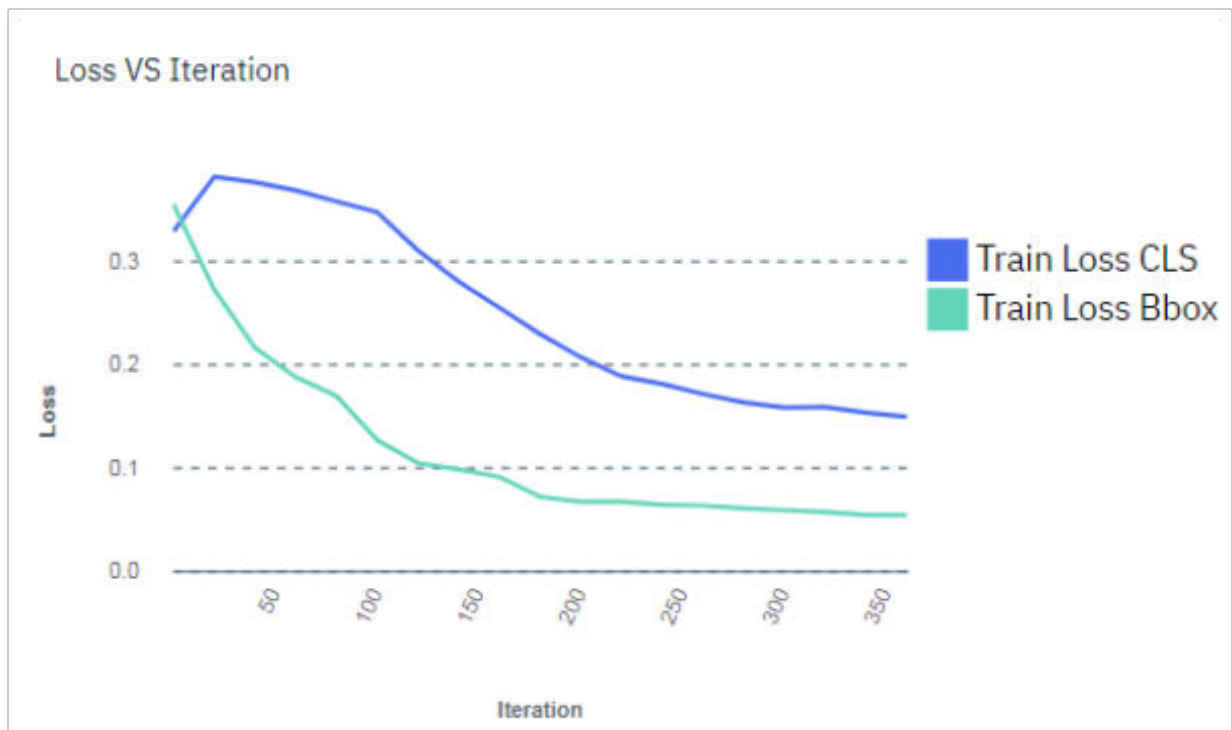


Figure 11. Model training graph

Important: If the training graph converges quickly and has 100% accuracy, the data set does not have enough information. The same is true if the accuracy of the training graph fails to rise or the errors in the graph do not decrease at the end of the training process. For example, a model with high accuracy might be able to discover all instances of different race cars, but might have trouble differentiating between specific race cars or those that have different colors. In this situation, add more images, video frames, or videos to the data set, label them, then try the training again.

Deploying a trained model

To deploy the trained model, complete the following steps. GPUs are used as follows:

- Multiple Faster R-CNN and GoogLeNet models are deployed to a single GPU. IBM Maximo Visual Inspection uses packing to deploy the models. That is, the model is deployed to the GPU that has the most models deployed on it, if there is sufficient memory available on the GPU. The GPU group can be used to determine which deployed models share a GPU resource. To free up a GPU, *all* deployed models in a GPU group must be deleted (undeployed).

Note: IBM Maximo Visual Inspection leaves a 500MB buffer on the GPU.

1. Click **Models** from the menu.
2. Select the model you created in the previous section and click **Deploy**.
3. Specify a name for the model, and click **Deploy**. The **Deployed Models** page is displayed, and the model is deployed when the status column displays **Ready**.
4. Double-click the deployed model to get the API endpoint and test other videos or images against the model. For information about using the API see [IBM Maximo Visual Inspection API documentation](#).

Next steps

You can continue to refine the data set as much as you want. When you are satisfied with the data set, you can train the model again. This time when you train the model, you might want to train the model for a longer time to improve the overall accuracy of the model. The loss lines in the training model graph should converge to a stable flat line. The lower the loss lines are in the training graph the better. After the

training completes, you can deploy the model again. You can double-click the deployed model to get the API endpoint and test other images or images against the model.

Example: Detecting objects in a video

In this fictional scenario, you want to create a deep learning model to monitor traffic on a busy road. You have a video that displays the traffic during the day. From this video, you want to know how many cars are on the busy road every day, and what are the peak times that have the most cars on the road.

The video file used in this scenario is available for download here: [Download car video](#).

To create a deep learning model, you will perform the following steps:

1. [Importing a video](#)
2. [Labeling objects in a video](#)
3. [Training a model](#)
4. [Deploying a model](#)
5. [Automatically label frames in a video](#)

Import a video and create a data set

First, create a data set and add videos to it.

1. [Log in to IBM Maximo Visual Inspection](#).
2. Click **Data Sets** in the navigation bar to open the **Data Sets** page. There are several ways to create a new data set
3. From the **Data set** page, click the icon and name the data set Traffic Video.
4. To add a video to the data set, click the Traffic Video data set and click **Import file** or drag the video to the + area.

Important: You cannot navigate away from the IBM Maximo Visual Inspection page or refresh until the upload completes. You can navigate to different pages within IBM Maximo Visual Inspection during the upload.

Labeling objects in a video

The next step is to label objects in the video. For object detection, you must have at minimum five labels for each object. We will create Car and Motorcycle objects and will label at least five frames in the video with cars and at least five frames with motorcycles.

1. Select the video from your data set and select **Label Objects**.
2. Capture frames by using one of these methods:
 - Click **Auto capture frames** and specify a value for **Capture Interval (Seconds)** that will result in at least five frames. We will select this option and specify 10 seconds.

Note: Depending on the length and size of the video and the interval you specified to capture frames, the process to capture frames can take several minutes.
 - Click **Capture frame** to manually capture frames. If you use this option, you must capture a minimum of five frames from the video.
3. If you used **Auto capture frames**, verify that there are enough of each object type in the video frames. If not, follow these steps to add new frames to the existing data set.

In this scenario, the motorcycle is only in a single automatically captured frame at 40 seconds. Therefore, we must capture at least four more frames with the motorcycle. The motorcycle comes into view at 36.72 seconds. To correctly capture the motorcycle in motion we will create extra frames at 37.79 seconds, 41.53 seconds, and 42.61 seconds.

- a. Play the video. When the frame you want is displayed, click pause.

- b. Click **Capture Frame**.
4. Create new object labels for the data set by clicking **Add new** by the Objects list. Enter **Car**, click **Add**, then enter **Motorcycle**, then click **OK**.

Note: If you later want to delete the label, it must be done at the data set level. It cannot be done from an individual frame or image.

5. Label the objects in the frames:

- Select the first frame in the carousel.
- Select the correct object label, for example, "Car".
- Choose **Box** or **Polygon** from the bottom left, depending on the shape you want to draw around each object. Boxes are faster to label and train, but less accurate. Only Detectron models support polygons. However, if you use polygons to label your objects, then use this data set to train a model that does not support polygons, bounding boxes are defined and used. Draw the appropriate shape around the object.

Note: When **Box** or **Polygon** is selected, you have to hold down the Alt key for non-drawing interactions in the image. This includes trying to select, move, or edit previously drawn shapes in the image, and panning the image by using the mouse. To return to the normal mouse interactions, deselect the **Box** or **Polygon** button.

Review the following tips about identifying and drawing objects in video frames and images:

- Do not label part of an object. For example, do not label a car that is only partially in the frame.
- If an image has more than one object, you must label all objects. For example, if you have cars and motorcycles defined as objects for the data set, and there is an image with both cars and motorcycles in it, you must label the cars and the motorcycles. Otherwise, you decrease the accuracy of the model.
- Label each individual object. Do not label groups of objects. For example, if two cars are right next to each other, you must draw a label around each car.
- Draw the shape as close to the objects as possible. Do not leave blank space around the objects.
- You can draw shapes around objects that touch or overlap. For example, if one object is behind another object, you can label them both. However, it is recommended that you only label objects if the majority of the object is visible.
- Use the zoom buttons (+ and -) on the bottom right side of the editing panels to help draw more accurate shapes.

Note: If you are zoomed in on an image and use the right arrow key to move all the way to the right edge, you might have to click the left arrow key several times to start panning in the other direction.

- Shapes cannot extend off the edge of the frame.
- After defining a shape, you can copy and paste it elsewhere in the same image or in a different image by using standard keyboard shortcuts. After pasting the shape, it can be selected and dragged to the desired location in the image. The shape can also be edited to add or remove points in the outline.

Note: To copy and paste a shape from one image to another, both images have to be available in the image carousel. From the data set, select all images that will share shapes, then click **Label objects**. All images will be listed in the image carousel in the left side of the Label objects window.

- After a shape has been defined, you will no longer see the points on the outline. To edit a defined box, exit drawing mode, then edit the points as necessary. To exit drawing mode, do one of the following:

- Click the object name on the right side of the window.
- Alt+click (option +click) inside the defined box.

After moving a defined point, drawing mode is automatically enabled again.

- The video object preview does not support non-ascii labels. This is a limitation of the module that generates the displayed label from the label name. The result of the conversion of non-ascii labels will be a label that is all question marks: "?????".
- **Labeling with polygons**
 - After a shape has been defined, you will no longer see the points on the outline. To edit a defined shape, exit drawing mode, then edit the points as necessary. To exit drawing mode, do one of the following:
 - Click the object name on the right side of the window.
 - Click inside the defined shape.
 - When you are done editing the shape, click outside the shape to enter drawing mode again.
 - To delete a point from an outline, ctrl+click (or cmd+click).
 - To add a point to an outline, click the translucent white square between any two points on the outline.
 - To move a point on the outline, click it and drag.

The following figure displays the captured video frame at 41.53 seconds with object labels of **Car** and **Motorcycle**. Figure 1 also displays a box around the five frames (four of the frames were added manually) in the carousel that required object labels for the motorcycle that is in each frame.



Figure 12. Labeling objects in IBM Maximo Visual Inspection

Training a model

With all the object labels that are identified in your data set, you can now train your deep learning model. To train a model, complete the following steps:

1. From the **Data set** page, click **Train**.
2. Fill out the fields on the **Train Data set** page, ensuring that you select **Object Detection**. We will choose **Accuracy (faster R-CNN)** for **Model selection**.
3. Click **Train**.
4. (Optional - *Only supported when training for object detection.*) Stop the training process by clicking **Stop training** > **Keep Model** > **Continue**.

You can wait for the entire training model process complete, but you can optionally stop the training process when the lines in the training graph start to flatten out, as shown in the figure below. This is because improvements in quality of training might plateau over time. Therefore, the fastest way to deploy a model and refine the data set is to stop the process before quality stops improving.

Note: Use early stop with caution when training segmented object detection models (such as with Detectron), because larger iteration counts and training times have been demonstrated to improve accuracy even when the graph indicates the accuracy is plateauing. The precision of the label is can still being improved even when the accuracy of identifying the object location stopped improving.

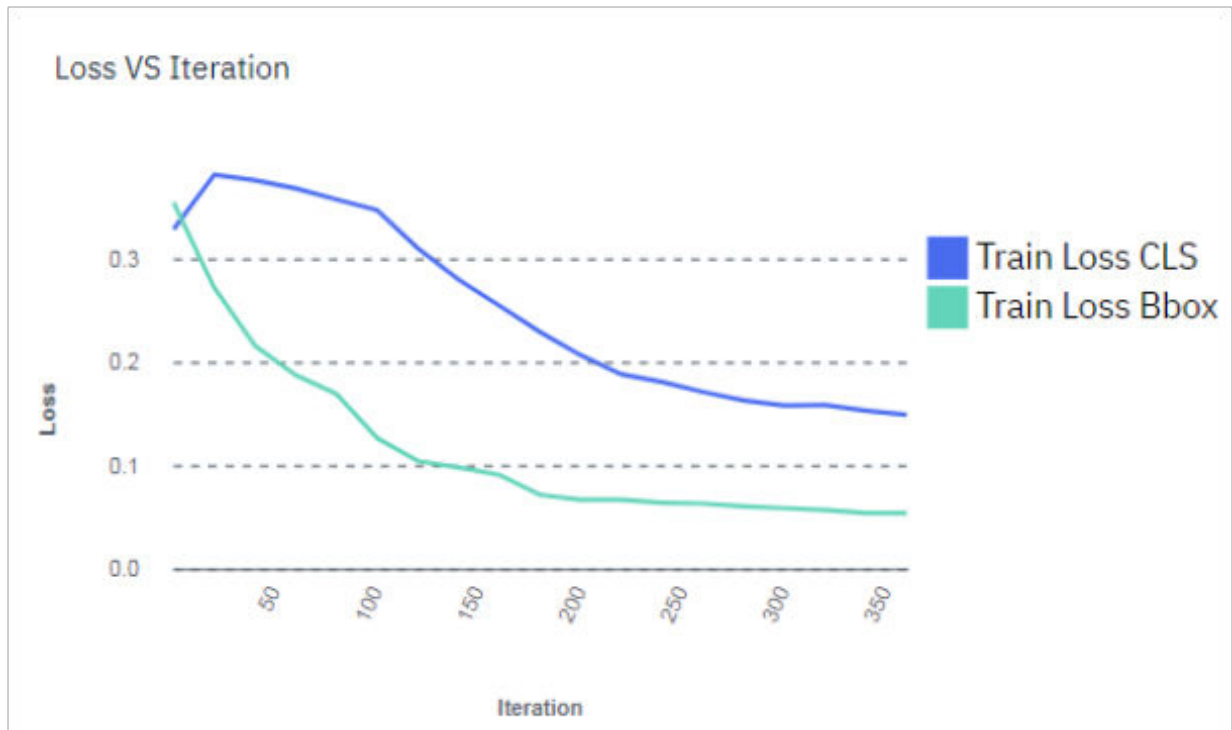


Figure 13. Model training graph

Important: If the training graph converges quickly and has 100% accuracy, the data set does not have enough information. The same is true if the accuracy of the training graph fails to rise or the errors in the graph do not decrease at the end of the training process. For example, a model with high accuracy might be able to discover all instances of different race cars, but might have trouble differentiating between specific race cars or those that have different colors. In this situation, add more images, video frames, or videos to the data set, label them, then try the training again.

Deploying a trained model

To deploy the trained model, complete the following steps. GPUs are used as follows:

- Multiple Faster R-CNN and GoogLeNet models are deployed to a single GPU. IBM Maximo Visual Inspection uses packing to deploy the models. That is, the model is deployed to the GPU that has the most models deployed on it, if there is sufficient memory available on the GPU. The GPU group can be used to determine which deployed models share a GPU resource. To free up a GPU, *all* deployed models in a GPU group must be deleted (undeployed).

Note: IBM Maximo Visual Inspection leaves a 500MB buffer on the GPU.

1. Click **Models** from the menu.
2. Select the model you created in the previous section and click **Deploy**.
3. Specify a name for the model, and click **Deploy**. The **Deployed Models** page is displayed, and the model is deployed when the status column displays **Ready**.

4. Double-click the deployed model to get the API endpoint and test other videos or images against the model. For information about using the API see [IBM Maximo Visual Inspection API documentation](#).

Automatically label frames in a video

You can use the auto label function to automatically identify objects in the frames of a video after a model has been deployed.

In this scenario, you have only nine frames. To improve the accuracy for your deep learning model, you can add more frames to the data set. Remember, you can rapidly iterate by stopping the training on a model and checking the results of the model against a test data set. You can also use the model to auto label more objects in your data set. This process improves the overall accuracy of your final model.

To use the auto label function, complete the following steps:

Note: Any frames that were previously captured by using auto capture and were not manually labeled are deleted before auto labeling. This helps avoid labeling duplicate frames. Manually captured frames are not deleted.

1. Click **Data sets** from the menu, and select the data set that you used to create the previously trained model.
2. Select the video in the data set that had nine frames, and click **Label Objects**.
3. Click **Auto label**.
4. Specify how often you want to capture frames and automatically label the frames. Select the name of the trained model that you deployed in step [3](#), and click **Auto label**. In this scenario, you previously captured frames every 10 seconds. To improve the accuracy of the deep learning model by capturing and labeling more frames, you can specify 6 seconds.
5. After the auto label process completes, the new frames are added to the carousel. Click the new frames and verify that the objects have the correct labels. The object labels that were automatically added are green and the object labels you manually added are in blue. In this scenario, the carousel now has 17 frames.

Next steps

You can manipulate (move or resize) the labels that were automatically generated. You can also save or reject individual labels, or you can reject them all by selecting **Clear all**. Saving or manipulating a label converts it to a manually added label. Rejecting a label deletes it. If you run **Auto label** again, any images or frames that now have manually added labels are skipped.

You can continue to refine the data set as much as you want. When you are satisfied with the data set, you can retrain the model by completing steps [1](#) - [3](#). This time when you retrain the model, you might want to train the model for a longer time to improve the overall accuracy of the model. The loss lines in the training model graph should converge to a stable flat line. The lower the loss lines are in the training graph the better. After the training completes, you can redeploy the model by completing steps [1](#) - [3](#). You can double-click the deployed model to get the API endpoint and test other videos or images against the model.

Related concepts

[Working with the user interface](#)

The IBM Maximo Visual Inspection user interface is made up of these basic parts: the navigation bar, the side bar, the action bar, the data area, and the notification center.

[Understanding metrics](#)

IBM Maximo Visual Inspection provides several metrics to help you measure how effectively your model has been trained.

Related information

[Vision Service API documentation](#)

Example: Classifying images

The goal of this example is to train a model to classify images of birds into groups based on their physiological similarities. Once the model is trained with a known dataset, users can upload new data sets to auto classify the birds into their respective categories. We will prepare the data, create a data set, train the model, and test the model.

About this task

Procedure

1. Prepare the data.

Data preparation consists of gathering two types of data, *training data* and *test data*. Training data is used to teach the neural network features of the object so that it can build the classification model. Test data is used to validate the accuracy of the trained model. Our data will include pictures of different types of birds.

Notes:

- Different images should be used for training data and test data.
 - Images must be in one of these formats:
 - JPEG
 - PNG
 - DICOM
2. Create a data set. Log in to the IBM Maximo Visual Inspection user interface, click **Data Sets** in the navigation bar, click **Create new data set** and name the data set Birds.
 3. Populate the data set.
 - a) In the left pane, expand Categories, click **Add category**. Add the "Acridotheres" category and click **Add**, then click **OK**.
 - b) Upload images of Acridotheres by dragging the images onto the **Drag files here** area.
 - c) In the left pane, click "Uncategorized". The newly uploaded files are shown.
 - d) Click the **Select** box to select the images you just uploaded, then click **Assign category** and choose "Acridotheres".
 - e) Repeat the above steps for the other categories.

Note: To train a model for classification, the data set must meet these requirements:

- There must be at least two categories.
 - Each category must have at least five images.
4. From the **Data set** page, click **Train**. In the Train data set window, choose **Image classification** and keep the default values for all other settings, then click **Train**.
 5. After training is complete, click **Deploy model**.

Important: Each deployed model uses one GPU.

6. Test the trained model. On the Deployed models page, open the model you just deployed. Scroll down to the Test Images area and input a test image.

The test result displays the uploaded picture with the resultant heat map overlayed, and gives the classification and the confidence of the classification. Multiple classes are returned with the decreasing levels of confidence for the different classes. The heat map is for the highest confidence classification and can help you determine whether the model has correctly learned the features of this classification. To hide classes with a lower confidence level, use the **Confidence threshold** slider.

The red area of the heat map corresponds to the areas of the picture that are of highest relevance. Use the slider to change the opacity of the heat map. Because the heat map is a square, the test image is

compressed into a square. This might cause the image to look distorted, but it will reliably show you the areas that the algorithm identified as relevant.

If you are not satisfied with the result, use the information in this topic to refine the model: [“Refining a model” on page 84](#). Otherwise, the model is ready to be used in production.

Related concepts

[Working with the user interface](#)

The IBM Maximo Visual Inspection user interface is made up of these basic parts: the navigation bar, the side bar, the action bar, the data area, and the notification center.

[Understanding metrics](#)

IBM Maximo Visual Inspection provides several metrics to help you measure how effectively your model has been trained.

Related information

[Vision Service API documentation](#)

Example: Detecting segmented objects in images

In this fictional scenario, you want to create a deep learning model to detect segmented objects, such as a bicycle with a rider standing in front of it. To accomplish this, you will import a COCO data set and train a Detectron model.

To create a deep learning model to detect segmented objects, you will perform the following steps:

1. [“Import images and create a data set” on page 110](#)
2. [“Training a model” on page 110](#)
3. [“Deploying a trained model” on page 111](#)

Import images and create a data set

First, create a data set and add images to it.

1. [Log in to IBM Maximo Visual Inspection](#).
2. Click **Data Sets** in the navigation bar to open the Data Sets page. Create a new data set and give it a name.
3. From the [COCO download site](#), click **2017 Train images** to download the `train2017.zip` file.
4. Create a new file that contains just the images that you want from `train2017` by running a command such as the following:

```
ls train2017 | grep jpg | head -20000 >/tmp/flist
```

5. From the [COCO download site](#), click **2017 Train/Val annotations** to download the `annotations_trainval2017.zip` file.
6. From `annotations_trainval2017.zip`, extract the `annotations/instances_train2017.json` file, which is the COCO annotation file for object detection.
7. Add `annotations/instances_train2017.json` to the file of images that you created in step “4” [on page 110](#) and compress them into a zip file.
8. From your new data set, click **Import file** and select the zip file you just created.

Important: You cannot navigate away from the IBM Maximo Visual Inspection page or refresh until the upload completes. You can navigate to different pages within IBM Maximo Visual Inspection during the upload.

Training a model

Because the images are already labeled, you can now train your deep learning model. Training a model uses one GPU:

1. From the **Data set** page, click **Train**.
2. Fill out the fields on the **Train Data set** page. Select **Object Detection** and **Segmentation (Detectron)**.
3. Click **Train**.
4. (Optional - *Only supported when training for object detection.*) Stop the training process by clicking **Stop training** > **Keep Model** > **Continue**.

You can wait for the entire training model process complete, but you can optionally stop the training process when the lines in the training graph start to flatten out, as shown in the figure below. This is because improvements in quality of training might plateau over time. Therefore, the fastest way to deploy a model and refine the data set is to stop the process before quality stops improving.

Note: Use early stop with caution when training segmented object detection models (such as with Detectron), because larger iteration counts and training times have been demonstrated to improve accuracy even when the graph indicates the accuracy is plateauing. The precision of the label is can still being improved even when the accuracy of identifying the object location stopped improving.

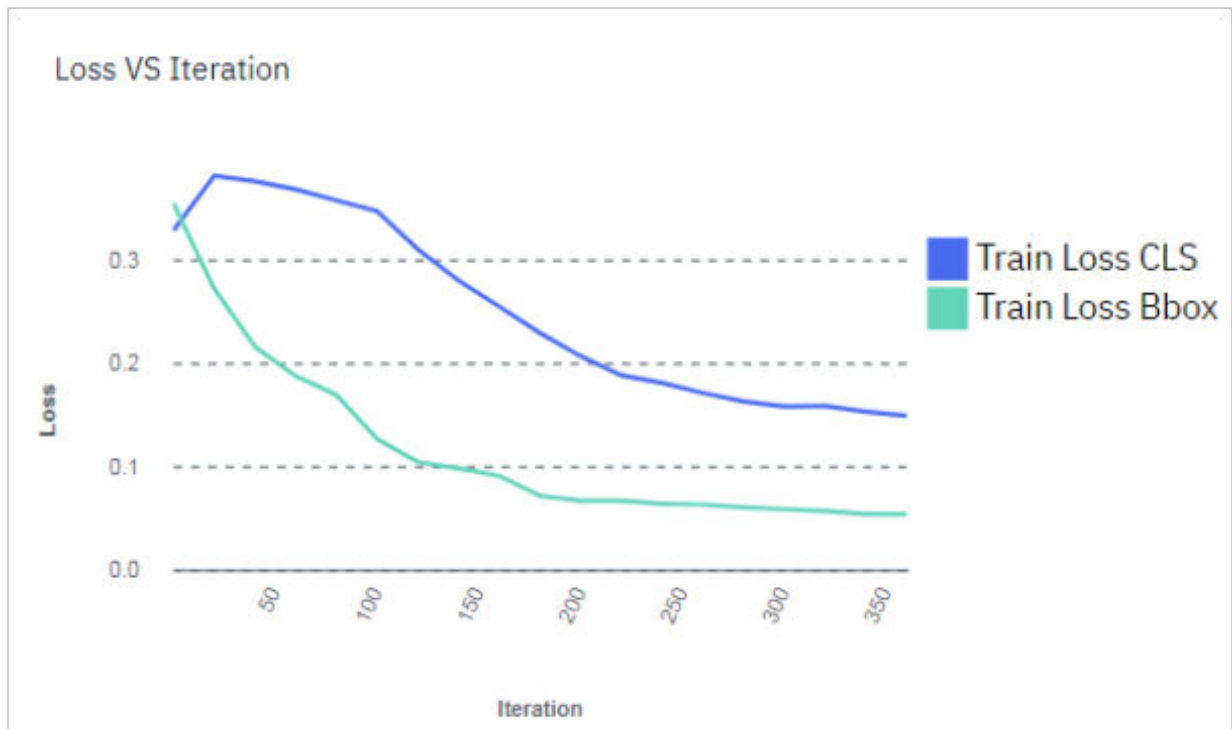


Figure 14. Model training graph

Important: If the training graph converges quickly and has 100% accuracy, the data set does not have enough information. The same is true if the accuracy of the training graph fails to rise or the errors in the graph do not decrease at the end of the training process. For example, a model with high accuracy might be able to discover all instances of different race cars, but might have trouble differentiating between specific race cars or those that have different colors. In this situation, add more images, video frames, or videos to the data set, label them, then try the training again.

Deploying a trained model

To deploy the trained model, follow these steps. Each deployed Detectron model takes one GPU:

1. Click **Models** from the menu.
2. Select the model you created in the previous section and click **Deploy**.
3. Specify a name for the model, and click **Deploy**. The **Deployed Models** page is displayed, and the model is deployed when the status column displays **Ready**.

4. Double-click the deployed model to get the API endpoint and test other images against the model. For information about using the API see [IBM Maximo Visual Inspection API documentation](#).

Next steps

You can continue to refine the data set as much as you want. When you are satisfied with the data set, you can train the model again. This time when you train the model, you might want to train the model for a longer time to improve the overall accuracy of the model. The loss lines in the training model graph should converge to a stable flat line. The lower the loss lines are in the training graph the better. After the training completes, you can deploy the model again. You can double-click the deployed model to get the API endpoint and test other images or images against the model.

Example: Detecting actions in a video

In this fictional scenario, you want to create a deep learning model to determine when a cash register is being opened in a video.

We will assume we have videos named Cashier 1 - Cashier 5, which we will add to a data set named "Open cash register". To create a deep learning model, you will perform the following steps:

1. [Preparing videos for import](#)
2. ["Import videos and create a data set" on page 112](#)
3. ["Labeling actions in a video" on page 112](#)
4. ["Training the model" on page 113](#)
5. ["Deploying a model" on page 114](#)

Preparing videos for import

Before importing videos for use with action detection models, it is recommended that you prepare them as follows:

- Cut out long periods of background video without any actions.
- Transcode videos with FPS greater than 30 down to 30 FPS
- Crop the video so that actions should take up a large part of the frame.

Import videos and create a data set

First, create a data set and add videos to it.

1. [Log in to IBM Maximo Visual Inspection](#).
2. Click **Data Sets** in the navigation bar to open the **Data Sets** page. There are several ways to create a new data set. We will create a new, empty data set.
3. From the **Data set** page, click the icon and name the data set "Open cash register".
4. To add a video to the data set, click the Open cash register data set and click **Import file** or drag the video to the + area. We will assume we have added the Cashier 1 - Cashier 5 videos.

Important: You cannot navigate away from the IBM Maximo Visual Inspection page or refresh until the upload completes. You can navigate to different pages within IBM Maximo Visual Inspection during the upload.

Labeling actions in a video

The next step is to label actions in the videos. We will create the "Open" action and will label it in several videos.

There is no minimum number of labels required, but more data will typically give better results.

- Each action label must be in the range of 5 - 1000 frames. The required length of time depends on the video's FPS. For 30 FPS, each action label must be in the range of .166 - 33.367 seconds.

The label's duration is checked based on the frames per second and the selected start and end times. For example, if an action label is marked with a start time of 12.295 seconds and end time of 12.296 seconds for a 30 FPS video, you will get an error message like the following: "Label duration of '100' milliseconds does not meet required duration between '166.83333' milliseconds and '33366.668' milliseconds".

- At least 10 instances of each action tag in the data set are recommended.
- The longer the total labeled action time is, the better your results will be.
- If multiple types of actions are labeled in a data set, the total amount of time for each action type should be similar. For example, if you tag 20 instances of the action "jump" in a data set with a total time of 27 seconds, and you tag 10 instances of the action "drive" in the data set with a total time of 53 seconds, the model will be biased toward the "drive" action.

The total time for each action type is shown in the left pane in the Actions section.

Follow these steps to label actions. For more information, see [“Labeling actions” on page 66](#):

1. Open the "Open cash register" data set.
2. Create the "Open" action tag in the data set by expanding **Actions** on the left and clicking **Add action**.
3. Select the appropriate video and click **Label actions**. The existing tags are listed on the right.
4. Find the start of an action by using the video control bar:
 - Use the slider or play button to get near the part of the video you want.
 - Set the playback rate (1x, .5x, and so on) to control how fast the video plays.
 - Use the +1 and -1 buttons to move forward or backward one frame.
5. Find the end of the action, then click + in **End time**.
6. Select "Open" for the action name, then click **Save action**.
7. Continue adding actions to videos until you are done.

Training the model

With all the action labels identified in your data set, you can now train your deep learning model by following these steps:

1. From the **Data set** page, click **Train**.
2. Fill out the fields on the **Train Data set** page, ensuring that you select **Action detection**. Leave the default values for all other options.
3. Click **Train**.
4. (Optional - *Only supported when training for object detection.*) Stop the training process by clicking **Stop training > Keep Model > Continue**. You can wait for the entire training model process complete, but you can optionally stop the training process when the lines in the training graph start to flatten out, as shown in the figure below. This is because improvements in quality of training might plateau over time. Therefore, the fastest way to deploy a model and refine the data set is to stop the process before quality stops improving.

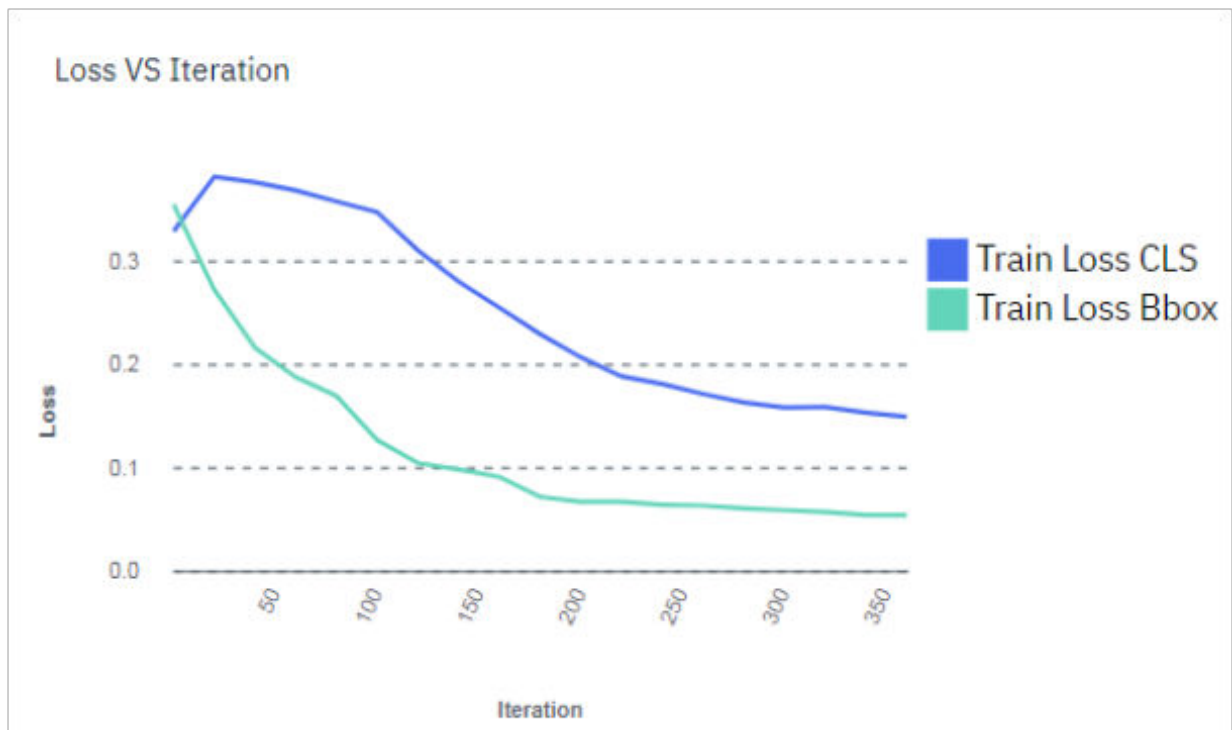


Figure 15. Model training graph

Important: If the training graph converges quickly and has 100% accuracy, the data set does not have enough information. The same is true if the accuracy of the training graph fails to rise or the errors in the graph do not decrease at the end of the training process. For example, a model with high accuracy might be able to discover all instances of different race cars, but might have trouble differentiating between specific race cars or those that have different colors. In this situation, add more images, video frames, or videos to the data set, label them, then try the training again.

Deploying a model

To deploy the trained model, complete the following steps.

Note: Each deployed action detection (SSD) model takes one GPU.

1. Click **Models** from the menu.
2. Select the model you created in the previous section and click **Deploy**.
3. Specify a name for the model, and click **Deploy**. The **Deployed Models** page is displayed, and the model is deployed when the status column displays **Ready**.
4. Double-click the deployed model to get the API endpoint and test other videos or images against the model. For information about using the API see [IBM Maximo Visual Inspection API documentation](#).

Example: Adding preprocessing and post-processing

In this fictional scenario, you want to create a model to detect license plates, then add post-processing that crops everything in the image outside of the license plate.

We will assume that we have a post-processing script that crops the area outside the identified license plates in the image, as well as a model trained to identify license plates. To create a deep learning model and add post-processing, perform the following steps:

1. [“Import the post-processing script” on page 115](#)
2. [“Train the model” on page 115](#)
3. [“Deploy the model” on page 115](#)

4. [“Perform an inference and review results” on page 115](#)

Import the post-processing script

The post-processing script is a .zip file created by using the `custom.py` template. See [“Preprocessing and post-processing” on page 81](#) for details. Navigate to the **Custom assets** page and upload the zip file that contains `custom.py`. For Asset type, select **Custom inference script**. We will name our zip file `crop.zip`.

Train the model

Preprocessing and post-processing can be done on any type of model except action detection. We will be using an object detection model called `license_plates`. For instructions to train a model, see [“Training a model” on page 68](#).

Deploy the model

Deploy the model, specifying the post-processing script:

1. Navigate to the model you want to deploy and click **Deploy model**. In the upper right corner, select **Advanced deployment**.
2. For **Custom inference script**, select the inference script that you want to use, specify what you want done with the inference results, and click **Deploy**. For this example, we will specify to save the inference results to the `cropped_license_plates` data set.

Perform an inference and review results

Use the deployed model API endpoint to perform an inference. After the inference, the `crop` script is called, and the resulting image, with the license plates labeled, is saved to the `cropped_license_plates` data set. When you view the data set in the table view, these images are labeled "Inference result" in the **Created** column.

Integrating IBM Maximo Visual Inspection with Maximo Asset Monitor

Maximo Asset Monitor. Maximo Asset Monitor is a cloud service that enables users to remotely monitor devices at the edge. For example, it can help you notice manufacturing irregularities and take action. This integration allows IBM Maximo Visual Inspection to send inference results to the Maximo Asset Monitor cloud platform for further analysis.

About this task

IBM Maximo Visual Inspection must be installed and running before following these steps. For details about working with Maximo Asset Monitor, refer to the [Maximo Asset Monitor Knowledge Center](#).

Procedure

1. Verify that IBM Maximo Visual Inspection can connect to the internet.
2. Ensure that you have the IBM Watson IoT Platform Organization ID available. If you do not know the ID, log in to the Watson IoT™ Platform . The ID is below your user name.
3. In IBM Maximo Visual Inspection, set up users for Maximo Asset Monitor. It is recommend that you do not use the admin user. Follow these guidelines when creating your user name and password. See [“Managing users” on page 119](#) for instructions.
 - The credentials need to be valid for retrieving the token from Maximo Asset Monitor.
 - You cannot change these credentials later.
4. Create a project in IBM Maximo Visual Inspection and add a data set and model to the project.
5. Configure the Watson IoT Platform instance. Use the Watson IoT Platform Service dashboard to create a new API key and authentication token pair:

- a) In the Service dashboard, navigate to **Apps > Browse API Keys**.
 - b) Click **Generate API Key**.
 - c) Add a comment to identify the API key in the dashboard, for example: Key for enabling Edge support.
 - d) In the Permissions modal, set Role to Operations Application.
 - e) Click **Generate Key**.

Important: You must record the API key and token pair. You will need these to invoke the script `enableEdge.sh` later. The authentication tokens are **non-recoverable**.

Examples:

 - API key (test_api_key): a-9ixbbq-a84ps90Ajs
 - API token (test_api_token): MP\$08VKz!8rXwnR-Q*
 - f) Click **Close**.
6. Use the Watson IoT Platform Service dashboard to configure the Gateway Type and Device ID for connecting the server to the platform instance. The edge solution will run on all the edge gateways of this type. Follow these steps to create a gateway device, noting the Device type, Device ID, and the token associated with it.
 - a) In the Overview dashboard, select Devices from the menu pane, then select **Device Types**.
 - b) From Device Type page, click **Add Device Type**, then fill out these fields:
 - Select **Gateway** and enter the gateway type name.
 - Add a description.
 - Enable **Edge Solution**.
 - From the drop down list, select **Architecture**, then click **Next**.
 - c) On the Device Information modal, enter the gateway type attributes, then click **Next**.
 - d) On the Edge Solutions modal, click **Add Edge Solution**.
 - e) On the Add Edge Solution window, hover over the solution to be configured on Edge devices, click **Select Solution**, then click **Done**.
 - f) Click **Finish**.
 7. Register the edge device.
 - a) In the Overview dashboard, navigate to **Devices > Add Device**.
 - b) In the Identity model, select **Device Type** and select the gateway type registered in step “6” on [page 116](#).
 - c) Enter the device ID and click **Next**.
 - d) In the Device Information modal, enter the device attributes and click **Next**.
 - e) In the Permissions, specify the appropriate role and click **Next**.
 - f) In the Security modal, enter the authentication token and click **Next**.
 - g) Verify the information in the Summary page, then click **Finish**. A Device details page loads.
 - h) The Device details displays items such as Organization ID, Device Type, and device credentials. Copy the 5 items; they are needed later when you update the configuration map for the IBM Maximo Visual Inspection instance.
 8. Configure and connect IBM Maximo Visual Inspection to Watson IoT Platform . The sub-steps depend on your installation:

Standalone

- a. If IBM Maximo Visual Inspection is not started, start it with the following command:

```
sudo /opt/ibm/vision/bin/vision-start.sh
```

- b. Run the Maximo Asset Monitor configuration script:

```
sudo /opt/ibm/vision/bin/vision-config-mam.sh
  -ak '<API Key>' -at '<API Token>' -dt '<Device Token>' -di '<Device ID>' -t
  '<Device Type>'
```

Where the options are:

-h,	--help	Display this usage message and exit.
-o <val>,	--orgId <val>	WIoT Organization ID. Optional item.
-ak <val>,	--apiKey <val>	WIoT API Key with "Operations Application"
access role. Required.		
-at <val>,	--apiToken <val>	WIoT API token. Required.
-dt <val>,	--deviceToken <val>	Device Token from IoT Platform. Required.
-di <val>,	--deviceId <val>	Device ID from IoT Platform. Required.
-t <val>,	--deviceType <val>	Device Type from IoT Platform. Required.
-d <val>,	--domain <val>	WIoT Domain. Defaults to
'internetofthings.ibmcloud.com'		
-u <val>,	--url <val>	URL of the Maximo Visual Inspection Instance.

Cloud Instance

- a. Use the following command to get the name of the vision-edge-mqttbroker pod:

```
kubectl get pods --namespace=<namespace> | grep edge-mqttbroker
```

- b. Run the following command to create the configuration file for the instance, replacing `<mqttbroker_pod_name>` with the pod name that you determined in the previous step:

```
kubectl exec <mqttbroker_pod_name> -- /bin/bash -c "cat <<-EOF > /etc/wiotp-edge/
env.conf
#!/bin/bash
export WIOTP_INSTALL_ORGID='<orgid>'
export WIOTP_INSTALL_DEVICE_TYPE='<device_type>'
export WIOTP_INSTALL_DEVICE_ID='<device_id>'
export WIOTP_INSTALL_DEVICE_TOKEN='<device_token>'
export WIOTP_API_KEY='<api_key>'
export WIOTP_API_TOKEN='<api_token>'
export WIOTP_INSTALL_DOMAIN='<domain>'
export WIOTP_MVI_URL='<mvi_url>'
EOF
"
```

Where the options are:

```
WIOTP_INSTALL_ORGID is the Watson IoT Platform organization ID
WIOTP_INSTALL_DEVICE_TYPE is the Device type from Watson IoT Platform
WIOTP_INSTALL_DEVICE_ID is the Device ID from Watson IoT Platform
WIOTP_INSTALL_DEVICE_TOKEN is the Device token from Watson IoT Platform
WIOTP_API_KEY is the Watson IoT Platform API key
WIOTP_API_TOKEN is the Watson IoT Platform API token
WIOTP_INSTALL_DOMAIN is the Watson IoT Platform Domain
WIOTP_MVI_URL is the URL of the Maximo Visual Inspection instance
```

- c. Restart the following pods to read in the configuration:

```
kubectl delete pod --selector=run=vision-<namespace>-edge-mqttbroker-dp
kubectl delete pod --selector=run=vision-<namespace>-edge-connector-dp
kubectl delete pod --selector=run=vision-<namespace>-event-service-dp
```

9. Deploy a model from the project with the Maximo Asset Monitor custom asset.

- From the Projects page, open your project.
- Open the model to deploy and click **Deploy model**.
- On the Deploy model window, select the **Advanced Deployment** switch.
- Select the option to save inference results and select a data set from the drop down menu.
- Select the "Register Inference Results in Monitor" custom inference script from the menu.
- Click **Deploy**.

10. Test Maximo Asset Monitor by submitting an image to the deployed model. If successful, events appear under the **State** tab of the Watson IoT Platform Service user interface.

Chapter 13. Administering IBM Maximo Visual Inspection

Use this information to administer IBM Maximo Visual Inspection, such as stopping, starting, and determining the status of the pods.

Start or stop IBM Maximo Visual Inspection

There are several situations when you might need to stop and start IBM Maximo Visual Inspection. For example, when upgrading or performing maintenance on the product or on the system, when troubleshooting a problem, and so on. Use these commands to start or stop IBM Maximo Visual Inspection, as appropriate:

```
/opt/ibm/vision/bin/vision-stop.sh
```

```
/opt/ibm/vision/bin/vision-start.sh
```

Determine the status of IBM Maximo Visual Inspection pods

When troubleshooting a problem with IBM Maximo Visual Inspection, you might need to check the status of the Docker pods that are part of IBM Maximo Visual Inspection. For example, if the product does not start, if it is returning errors, or if actions are not completing. Run `kubectl get pods` to see the status. For example:

```
$ /opt/ibm/vision/bin/kubectl get pods
```

NAME		READY	STATUS	RESTARTS	AGE
vision-elasticsearch-fbfc584f9-n7jqc	1/1	Running	0	14d	
vision-fpga-device-plugin-qkfxj	1/1	Running	0	14d	
vision-keycloak-5df778c997-95h8s	1/1	Running	0	14d	
vision-logstash-84cc5cbcc4-9s28j	1/1	Running	0	14d	
vision-mongodb-79d964cfb9-zp6cp	1/1	Running	0	14d	
vision-postgres-85b6ddc9b6-7565q	1/1	Running	0	14d	
vision-service-8657f8878c-nlf2k	1/1	Running	0	10d	
vision-taskanalyzer-589447ffcf-r6wjn	1/1	Running	0	14d	
vision-ui-659dbc4657-npvh9	1/1	Running	0	14d	
vision-video-microservice-5db68fc8fd-s8bg2	1/1	Running	0	14d	

If one or more pods is not running, try stopping and restarting IBM Maximo Visual Inspection.

Managing users

There are two kinds of users in IBM Maximo Visual Inspection: administrators, and everyone else. The way you work with users and passwords differs, depending on how IBM Maximo Visual Inspection is installed.

IBM Maximo Visual Inspection uses Keycloak for user management and authentication. All users and passwords are maintained by Keycloak and stored in a Postgres database. A default user name of `admin` with a password of `passw0rd` are created at install time. You can add, remove, or modify users by using the `kubectl` command.

- [“Types of users” on page 119](#)
- [“IBM Maximo Visual Inspection installed as stand-alone” on page 120](#)

Types of users

Non-administrator users

Users other than the administrator can only see and edit resources that they created.

Administrator

The administrator user (admin) can see and manage all resources in IBM Maximo Visual Inspection regardless of who owns it. A default user name of `admin` with a password of `passw0rd` are created at install time. You can add, remove, or modify users by using the `kubectl` command. You should be aware of the following considerations when working with admin users:

Data sets

- The administrator can see and edit all data sets. That is, this user can add and delete files, create labels, assign categories, duplicate, rename, and delete the data set.
- If the administrator uploads a file to a different user's data set, it is listed as being owned by the data set owner.
- If the administrator duplicates a data set, the duplicate data set is owned by the administrator.

Models

- The administrator can see, rename, and delete all models, including after they are deployed.
- If the administrator trains a model, the training task and the generated model is owned by the administrator.
- If the administrator deploys a model, the deployed model is owned by the administrator.

Project groups

An administrator can add assets to a project group that was created by a different user. However, the project group owner will not be able to see the added assets because only administrators can see resources created by other users. Because of that, the value for "Total items" on the Projects page might be larger than the number of items shown on a project's details page.

IBM Maximo Visual Inspection installed as stand-alone

If you installed IBM Maximo Visual Inspection stand-alone, you can use the `vision-users.sh` script in the `/opt/ibm/vision/bin/` directory to create, delete, modify, and list users.

Usage

```
vision-users.sh [command] [ --user name ] [ --password password ]
```

Command

Specifies the action to take.

create

Create a user in the IBM Maximo Visual Inspection instance. The `user` argument is required for this operation. You can set the password by one of these methods:

- Specify it with the command by using the `password` argument.
- Store it in the environment variable, `VISION_USER_PASSWORD`.

delete

Delete a user from the IBM Maximo Visual Inspection instance. The `user` argument is required for this operation.

list

List the currently created users for a specified IBM Maximo Visual Inspection instance.

modify

Modifies the user's password. The `user` argument is required for this operation. You can set the new password by one of these methods:

- Specify it with the command by using the `password` argument.
- Store it in the environment variable, `VISION_USER_PASSWORD`.

unlock

Unlock a user account that was locked because of too many failed login attempts. The `user` argument is required for this operation.

Name

The user name on which the command is to operate on.

Password

Optionally set a user's password when creating or modifying a user.

Installing a new SSL certificate in IBM Maximo Visual Inspection stand-alone

IBM Maximo Visual Inspection ships with a self-signed certificate that is used by default, but this can be replaced with a certificate generated for IBM Maximo Visual Inspection for secure communications. If you want to use your own certificate, follow these steps to update the IBM Maximo Visual Inspection configuration.

About this task**Procedure**

1. Shut down IBM Maximo Visual Inspection:

```
sudo /opt/ibm/vision/bin/vision-stop.sh
```

2. Edit `/opt/ibm/vision/bin/config.sh` and specify the following information:

- **TLS_CERT_PATH** - Path to your custom PEM encoded public key certificate.
- **TLS_KEY_PATH** - Path to the private key associated with the **TLS_CERT_PATH** certificate.
- **INGRESS_HOSTS** - The host names defined in your certificate that you wish to use to access IBM Maximo Visual Inspection.

3. Start IBM Maximo Visual Inspection:

```
$ sudo /opt/ibm/vision/bin/vision-start.sh
```

Backing up IBM Maximo Visual Inspection

It is important to occasionally back up IBM Maximo Visual Inspection by shutting it down, then creating a backup of the data volume.

About this task

All IBM Maximo Visual Inspection data is stored in the data volume for the installation. For the standalone installation, this is `/opt/ibm/vision/volume`:

- **data** - This directory contains a sub-directory for each user with the data sets, dnn-sources (imported custom models), and trained-models the user has created.
- **run** - This directory contains runtime information for usage metrics, user registry, and metadata for the data sets.

Backing up application data

To back up the critical application data, create a backup of the data volume for the installation. For example:

Procedure

1. Shut down IBM Maximo Visual Inspection. Wait until everything is stopped and all pods are destroyed.

To verify that everything is stopped, run `kubectl get pods`, as described in “[Checking Kubernetes node status](#)” on [page 39](#). `kubectl` will not run when IBM Maximo Visual Inspection is stopped:

```
# kubectl get pods
The connection to the server localhost:8080 was refused - did you specify the right host or port?
```

2. Create a backup of the data volume by using your preferred method. For example:

```
$ cd /opt/ibm/vision; tar -zpcvf vision-volume.tgz /opt/ibm/vision/volume /opt/ibm/vision/run
```

Note: For OpenShift installations, the physical volume defined for the deployment should be backed up.

3. Restart IBM Maximo Visual Inspection.

Results

Alternatively, to protect your data while IBM Maximo Visual Inspection is running, you can download each data set and model manually. If they are imported later, they will have different UUIDs.

Restoring a backup

To restore a back up prepared using the above instructions provided, follow these steps:

1. Stop IBM Maximo Visual Inspection.
2. (Optional) Back up the current run (volume) directories in case there are any issues:

```
$ cd /opt/ibm/vision; mv run volume <backup_dir>
```

3. Restore the previously created backup:

```
$ cd /opt/ibm/vision; tar -zxf vision-backup.tgz
```

4. Restart IBM Maximo Visual Inspection.

Monitoring usage metrics

Administrators can view the IBM Maximo Visual Inspection usage metrics to see how many times an inference API was called for models that have been deployed. This might include models that were deployed previously but are no longer deployed.

To access the **Usage metrics** page, click your user name then click **Usage metrics**.

All metrics are kept from the time that IBM Maximo Visual Inspection 1.1.4 (or later) was installed, but you can use the date range selector to filter the data you want displayed.

The following inference metrics are gathered, depending on the model type:

Action detection

The number of times inferencing was performed for a deployed model. This is the duration of the video multiplied by the frame rate because inferencing is run once per frame.

Auto label

The number of images or frames labeled.

Note: When you run auto label on a video, IBM Maximo Visual Inspection automatically captures frames, then labels the frames. These frames are also counted under Total files. For more information about auto labeling, see “[Automatically labeling videos](#)” on [page 86](#) and “[Automatically labeling objects in a data set](#)” on [page 85](#).

Inferences

The number of times the deployed model was used for inferencing with images. When inferencing is done on a video, it is counted under **Video object detection** or **Action detection**.

Video object detection

The number of times inferencing was performed for a deployed model. This is the duration of the video multiplied by the frame rate because inferencing is run once per frame.

The following file metrics are gathered:

Augmented

The number of files generated by augmenting the data set.

Cloned

The number of files generated by duplicating the data set.

Frames captured

The number of files generated by manually or automatically capturing frames.

Uploaded

The number of files uploaded to a data set. For zip files, the zip file itself is not counted, but each file in the zip file is counted, excluding any .json control files.

Note: The file metrics are not available for data sets created prior to 1.1.5.

The following export metrics are gathered:

Exported data sets

The number of data sets exported by the user.

Exported models

The number of models exported by the user.

Note: The export metrics are not available for data sets or models exported prior to 1.1.5.

IBM Maximo Visual Inspection utilities

IBM Maximo Visual Inspection includes these utilities for working with the product.

- [“Administer” on page 123](#)
- [“Troubleshooting” on page 126](#)
- [“Cleanup and uninstall” on page 126](#)

Administer**accept-vision-license****Usage**

```
# accept-vision-license.sh
```

Description

The `accept-vision-license.sh` utility is used to accept the product license. The environment variable `IBM_POWERAI_VISION_LICENSE_ACCEPT` can be set to yes or no to automatically accept or reject the license. Otherwise, the license is presented along with a prompt to accept:

```
Press Enter to continue viewing the license agreement, or, Enter "1" to accept the
agreement, "2" to decline it or "99" to go back to the previous screen, "3"
Print, "4" Read non-IBM terms.
```

Note: This is called by the startup script `vision-start.sh` the first time the application is started to ensure that the license is accepted. If not accepted, the product will not start.

Requirements

The user must have sudo/root permissions.

check-vision-license**Usage**

```
# check-vision-license.sh
```

Description

Checks whether the product license has already been accepted.

- If the license is accepted, the utility silently exits with success (0).
- If the license has not been accepted, the utility prints an error and exits with error (1).

Requirements

The user must have sudo/root permissions.

config.sh**Usage**

```
# config.sh
```

Description

This file can be used to specify the following configuration values for the application:

EXTERNAL_IP

An IP address for the web portal if it is different from the system host name.

TLS_CERT_PATH, TLS_KEY_PATH, INGRESS_HOSTS

Specifies custom TLS certificates to be used by the application. See [“Installing a new SSL certificate in IBM Maximo Visual Inspection stand-alone” on page 121](#) for details.

Requirements

None - not an executable.

gpu_setup.sh**Usage**

```
# gpu_setup.sh
```

Description

Utility that checks the availability of GPUs on the system and the Docker setup to verify that it supports using GPUs in Docker containers. It is called by the `vision-start.sh` startup script.

Requirements

The user must have sudo/root permissions.

helm.sh**Usage**

```
# helm.sh [ command ]
```

Description

A wrapper for the Kubernetes Helm utility, which works with deployment charts. The `helm.sh` utility can be used to check the status of the IBM Maximo Visual Inspection deployment. See [“Checking application deployment” on page 43](#) for details. For information about the Helm utility, see [Using Helm](#).

Requirements

None.

kubect1.sh**Usage**

```
# kubect1.sh [ command ]
```

Description

A wrapper for the Kubernetes `kubect1` utility, which works with pods and deployments. The `kubect1.sh` utility can be used to check the status of the IBM Maximo Visual Inspection deployment. See these topics for details:

- [“Checking Kubernetes services status” on page 36](#)
- [“Checking Kubernetes node status” on page 39](#)
- [“Checking application deployment” on page 43](#)

For information about the `kubectl` utility, see [Overview of kubectl](#).

Requirements

None.

load_images.sh

Usage

```
# load_images.sh -f [ <vision-images-release>.tar ]
```

Description

Utility to load the IBM Maximo Visual Inspection Docker images, which are provided with the product installation package in the `<vision-images-release>.tar` file. The `load_images.sh` utility requires approximately 75 Gb of free space in the `/var` file system to extract and load the Docker images. Images are loaded in parallel, so if there are space limitations on the system, errors will only be output after all images have attempted to load. The `docker images` command can be used to validate that all images have been loaded. See [“Checking the application Docker images in standalone installation” on page 35](#) for details.

Requirements

The user must have Docker group permissions.

port.sh

Usage

```
# port.sh
```

Description

A file that can be used to specify configuration values for the ports used by the application. This is only required if there are multiple web services running on the system.

POWERAI_VISION_EXTERNAL_HTTPS_PORT

Specifies the SSL port that the IBM Maximo Visual Inspection user interface will use. The default port is 443.

Requirements

None - not an executable.

vision-start.sh

Usage

```
# vision-start.sh [ -nD ]
```

Description

Used to start the IBM Maximo Visual Inspection application and required Kubernetes services. This startup script runs some checks of system requirements that require elevated privileges, such as GPU availability. You can optionally specify the following flags:

-n or --nocheck

Suppress checks of the system environment. For example, SELinux contexts on GPU devices are checked and fixed if they are found to be incorrect. By default, checks are run and any issues found are fixed.

-D or --debug

Output debug information by using the `-x` bash flag.

Requirements

The user must have `sudo`/root permissions.

vision-stop.sh

Usage

```
# vision-stop.sh
```

Description

Used to stop the IBM Maximo Visual Inspection application and required Kubernetes services.

Requirements

The user must have sudo/root permissions.

Troubleshooting

collect_logs.sh

Usage

```
# collect_logs.sh
```

Description

Utility for collecting system logs and information, and IBM Maximo Visual Inspection application logs and information. The utility creates a single tar .gz file with the logs and configuration files that can be provided to IBM support to investigate issues.

Requirements

The user must have sudo/root permissions.

Cleanup and uninstall

purge_data

Usage

```
# purge_data.sh
```

Description

Remove log and runtime data used by the IBM Maximo Visual Inspection application. This **does not** remove the data sets and models created by application users. This data is in `<install_dir>/volume`, and must be removed manually.

Requirements

The user must have the required file system permissions.

purge_images

Usage

```
# purge_image.sh <release_tag>
```

Description

All IBM Maximo Visual Inspection Docker images matching the tag will be removed from the Docker repository. This script can be used to clean up images from a prior IBM Maximo Visual Inspection installation after an upgrade, or to remove IBM Maximo Visual Inspection images when uninstalling the product.

For example, to remove Docker images for the 1.3.0.0 release from the Docker repository, run this command:

```
# purge_images 1.3.0.0
```

You can use the `docker images` command to see what containers are in your Docker repository that are be associated with previous releases and can be purged.

Requirements

The user must have Docker group permissions.

Chapter 14. IBM Maximo Visual Inspection Edge

With a IBM Maximo Visual Inspection Edge, you can quickly and easily deploy multiple models that were trained in IBM Maximo Visual Inspection. These models are portable and can be used by many users and on different systems. This allows you to make trained models available to others, such as customers or collaborators.

Planning for IBM Maximo Visual Inspection Edge

Ensure that the following requirements are met before installing IBM Maximo Visual Inspection Edge.

- “Hardware requirements” on page 129
- “Platform requirements” on page 130
- “Software requirements” on page 130

Hardware requirements

Disk space requirements

- Installation - The IBM Maximo Visual Inspection Edge install package contains Docker containers for deployment on all supported platforms and requires 15 GB to download. Only the images needed for the platform will be installed by the `load_images.sh` operation, but this requires at least 70 GB available in the file system used by Docker, usually `/var/lib/docker`.
- Deploying a model - Models are extracted into the `/tmp` directory before loading. The size of the model depends on the framework, but at least 1 GB should be available in `/tmp` before deploying a model.

GPU model requirements

IBM Maximo Visual Inspection Edge is supported only on NVIDIA Tesla GPUs: T4, V100, and P100.

GPU memory requirements

- For deployment, the amount of memory required depends on the type of model you want to deploy. To determine how large a deployed model is, run `nvidia-smi` from the host after deployment. Find the corresponding PID that correlates to the model you deployed and look at the Memory Usage.

Example:

```
$ nvidia-smi
Tue Feb 26 09:12:59 2019
```

NVIDIA-SMI 418.29 Driver Version: 418.29 CUDA Version: 10.1									
GPU	Name	Persistence-M	Bus-Id	Disp.A	Memory-Usage	Volatile	Uncorr.	ECC	
Fan	Temp	Perf	Pwr:Usage/Cap			GPU-Util	Compute	M.	
0	Tesla P100-SXM2...	On	00000002:01:00.0	Off	1853MiB / 16280MiB	0%	Default	0	
N/A	36C	P0	39W / 300W						
1	Tesla P100-SXM2...	On	00000003:01:00.0	Off	4179MiB / 16280MiB	0%	Default	0	
N/A	38C	P0	42W / 300W						
2	Tesla P100-SXM2...	On	0000000A:01:00.0	Off	3351MiB / 16280MiB	73%	Default	0	
N/A	63C	P0	243W / 300W						
3	Tesla P100-SXM2...	On	0000000B:01:00.0	Off	10MiB / 16280MiB	0%	Default	0	
N/A	35C	P0	31W / 300W						

Processes:					GPU Memory Usage
GPU	PID	Type	Process name		
0	15735	C	/opt/miniconda2/bin/python		958MiB
0	16225	C	python		885MiB
1	39541	C	python		2253MiB
1	86043	C	/opt/miniconda2/bin/python		958MiB

	1	86299	C	/opt/miniconda2/bin/python	958MiB	
	2	103835	C	/opt/miniconda2/bin/python	3341MiB	
+-----+-----+-----+-----+-----+-----+						

- A custom model based on TensorFlow will take all remaining memory on a GPU. However, you can deploy it to a GPU that has at least 2 GB memory.

Platform requirements

- IBM Maximo Visual Inspection Edge can be deployed on x86 and IBM Power Systems platforms.
- YOLO v3, Detectron, and SSD models require Nvidia GPUs. Other models can be deployed in CPU only environments.

Software requirements

Linux

- Red Hat Enterprise Linux (RHEL) RHEL 7.6 ALT (little endian) for POWER9
- RHEL 7.7 for x86 and POWER8.
- Ubuntu 18.04

Note: The Ubuntu Hardware Enablement (HWE) kernel is not supported. Kubernetes services do not function correctly, preventing IBM Maximo Visual Inspection from starting successfully.

NVIDIA CUDA

- x86 - 10.1 or later drivers. For information, see the [NVIDIA CUDA Toolkit](#) website.
- ppc64le - 10.2 or later drivers are required. For information, see the [NVIDIA CUDA Toolkit](#) website.

Docker

- RHEL: docker-1.13.1
- Ubuntu: Docker CE or EE 18.06.01
- When running Docker, nvidia-docker 2 is supported. For RHEL 7.6, see [Using nvidia-docker 2.0 with RHEL 7](#).

Unzip

The unzip package is required on the system to deploy the zipped models.

Installing Docker on IBM Maximo Visual Inspection Edge

Follow these instructions to install Docker on IBM Maximo Visual Inspection Edge.

- [Installing Docker on Red Hat Enterprise Linux \(RHEL\)](#)
- [Installing Docker on Ubuntu](#)

Install Docker and nvidia-docker2 (RHEL)

Follow these steps to install Docker on RHEL. For full details, refer to <https://github.com/NVIDIA/nvidia-docker#rhel-docker>.

1. Install Docker:

```
sudo yum install docker
```

2. Reboot the system.
3. Add the package repositories:

On x86:

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.repo |
sudo tee /etc/yum.repos.d/nvidia-docker.repo
```

```
sudo yum install -y nvidia-container-toolkit
sudo systemctl restart docker
```

On IBM Power:

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/docker/$distribution/docker.repo | sudo tee /etc/
yum.repos.d/docker.repo
sudo yum install -y nvidia-container-runtime-hook
sudo systemctl restart docker
```

Install Docker and nvidia-docker2 (Ubuntu)

Use these steps to install Docker and nvidia-docker 2.

1. For Ubuntu platforms, a Docker runtime must be installed. If there is no Docker runtime installed yet, install Docker-CE on Ubuntu.

IBM Power

```
sudo apt-get update
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-
properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=ppc64el] https://download.docker.com/linux/ubuntu
bionic stable"
sudo apt-get update
sudo apt-get install docker-ce=18.06.1~ce~3-0~ubuntu
```

x86_64

```
sudo apt-get update
sudo apt-get install apt-transport-https ca-certificates curl gnupg-agent software-
properties-common
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu
bionic stable"
sudo apt-get update
sudo apt-get install docker-ce
```

2. Install nvidia-docker 2.

```
curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add -
distribution=$(. /etc/os-release;echo $ID$VERSION_ID)
curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo
tee /etc/apt/sources.list.d/nvidia-docker.list
sudo apt-get update
sudo apt-get install nvidia-docker2
sudo systemctl restart docker.service
```

3. For each userid that will run docker, add the userid to the docker group:

```
sudo usermod -a -G docker <userid>
```

Users must log out and log back in to pick up this group change.

4. Verify the setup.

IBM Power

```
docker run --rm nvidia/cuda-ppc64le:10.2-base-ubuntu18.04 nvidia-smi
```

x86_64

```
docker run --rm nvidia/cuda nvidia-smi
```

Note:

The **docker run** command must be used with `docker-ce` (in other words, an Ubuntu host) to leverage the GPUs from within a container.

Installing, upgrading, and uninstalling IBM Maximo Visual Inspection Edge Version 1.3.0

Follow these steps to install, upgrade, or uninstall IBM Maximo Visual Inspection Edge.

- [“Upgrading IBM Maximo Visual Inspection Edge” on page 132](#)
- [“Installing from IBM Passport Advantage” on page 132](#)
- [“Uninstalling IBM Maximo Visual Inspection Edge” on page 133](#)

Upgrading IBM Maximo Visual Inspection Edge

IBM Maximo Visual Inspection Edge does not have any application state. A new version can be installed without any required upgrade steps.

The prior version of the application can be safely uninstalled before installing the new version - see [“Uninstalling IBM Maximo Visual Inspection Edge” on page 133](#).

To reduce disk space usage, the docker containers required for the prior version of the product can be purged - see [“Uninstalling IBM Maximo Visual Inspection Edge” on page 133](#).

Installing from IBM Passport Advantage

1. Download the product tar file from the [IBM Passport Advantage](#) website.
2. Optionally verify the downloaded product tar file by following the steps in this topic: [“Verify the downloaded tar file” on page 33](#)
 - a. Download these files:

For x86

```
vis-inspect-edge-x86-1.3.0-ppa.sig  
visual-inspect-1.3.0-key.pub  
visual-inspect-ocsp-1.3.0-key.pub  
vis-inspt-ocspchain-1.3.0-key.pub
```

For Power Systems :

```
vis-inspect-edge-ppc-1.3.0-ppa.sig  
visual-inspect-1.3.0-key.pub  
visual-inspect-ocsp-1.3.0-key.pub  
vis-inspt-ocspchain-1.3.0-key.pub
```

3. Decompress the product tar file, change directories to the newly created directory, then run the installation command. For example, using the downloaded package for Power:

```
$ tar -xvf vis-inspect-edge-ppc-1.3.0-ppa.tar  
vis-inspect-edge-ppc-1.3.0-ppa/  
vis-inspect-edge-ppc-1.3.0-ppa/visual-inspection-edge-ppc64le-containers-1.3.0.0.tar  
vis-inspect-edge-ppc-1.3.0-ppa/visual-inspection-edge-1.3.0.0-472.22b7a1d.ppc64le.rpm  
vis-inspect-edge-ppc-1.3.0-ppa/visual-inspection-edge_1.3.0.0-472.22b7a1d_ppc64e1.deb  
$ cd vis-inspect-edge-ppc-1.3.0
```

Run the installation command for the platform you are installing on:

RHEL

```
sudo yum install ./<file_name>.rpm
```

Ubuntu

```
sudo dpkg -i ./<file_name>.deb
```

4. Load the product Docker images with the container tar file:

```
/opt/ibm/vision-edge/bin/load_images.sh -f <tar_file>
```

The file name has this format: `visual-inspection-edge-<arch>-containers-<release>.tar`, where `<arch>` is x86 or ppc, and `<release>` is the product version being installed.

IBM Maximo Visual Inspection Edge will be installed at `/opt/ibm/vision-edge`.

Uninstalling IBM Maximo Visual Inspection Edge

Follow these steps to uninstall IBM Maximo Visual Inspection Edge:

1. It is recommended that deployed models be undeployed, see [“Stopping a deployed model” on page 135](#).
2. To recover disk space, run `docker rmi` to remove the "deploy" docker images with the current release tag. For example, to remove all deploy images for the 1.3.0.0 release:

```
$ for deploy_img in $(docker images --format '{{.Repository}}:{{.Tag}}' | egrep 'vision-dnn-[a-z]*[-]*deploy[-]*[a-z0-9]*\.:1.3.0.0'); do docker rmi ${deploy_img} ; echo "Purged image ${deploy_img}." ; done
```

3. Optionally also uninstall the package:

- For RHEL:

```
sudo yum remove visual-inspection-edge
```

- For Ubuntu:

```
sudo dpkg --remove visual-inspection-edge
```

Deploying a trained model on IBM Maximo Visual Inspection Edge Version 1.3.0

After training and exporting a model via IBM Maximo Visual Inspection, you can deploy it in IBM Maximo Visual Inspection Edge.

See "Exporting a model" in [“Importing, exporting, and downloading IBM Maximo Visual Inspection information” on page 88](#) for instructions to export a model.

The following types of models of models trained in IBM Maximo Visual Inspection can be deployed:

- Object detection using Faster R-CNN (default), Tiny YOLO v2, Detectron, Single Shot Detector (SSD) (POWER only; x86 deployment not supported), custom TensorFlow or PyTorch models, and Keras models.
- Image classification using GoogLeNet (default) and custom TensorFlow or PyTorch models.
- [“Deploying a trained model” on page 133](#)
- [“Deployment output” on page 134](#)
- [“Stopping a deployed model” on page 135](#)

Deploying a trained model

To deploy a model, run this command:

```
/opt/ibm/vision-edge/bin/deploy_zip_model.sh
```

Note: The first time you run this command, you are prompted to accept the license agreement.

Usage:

```
./deploy_zip_model.sh -m <model-name> -p <port> -g <gpu> -t <time-limit> zipped_model_file
```

model-name

The docker container name for the deployed model.

port

The port to deploy the model to.

gpu

The GPU to deploy the model to. If specified as -1, the model will be deployed to a CPU.

Note: YOLO v3, Detectron, and SSD models cannot be deployed to a CPU.

time-limit

(Optional) Specify the time out limit for model deployment in seconds. The default value is 180 seconds.

zipped_model_file

The full path and file name of the trained model that was exported from IBM Maximo Visual Inspection. It can be an image classification model or an object detection model, but must be in zip format.

Examples:

```
/opt/ibm/vision-edge/bin/deploy_zip_model.sh --model dog --port 6001 --gpu 1 ./
dog_classification.zip
/opt/ibm/vision-edge/bin/deploy_zip_model.sh --m car -p 6002 -g -1 /home/user/mydata/car.zip
/opt/ibm/vision-edge/bin/deploy_zip_model.sh -m coco -p 6001 -g 1 /home/user/model/new_models/
cdb-coco-30k_model.zip
```

Deployment output

There are several different results you might see when you deploy a model. For example:

Success

If a model is deployed successfully, it reports back with the message "Successfully deployed model."

```
/opt/ibm/vision-edge/bin/deploy_zip_model.sh -m coco -p 6001 -g 1 /home/user/model/
new_models/cdb-coco-30k_model.zip

Successfully deployed model.

Deployed in 22 seconds
```

Failure

If the deployment fails, it reports back with log information from the docker container, including error messages regarding the failure. Some possible error examples follow. See [“Troubleshooting known issues - IBM Maximo Visual Inspection Edge” on page 183](#) for details about dealing with errors.

- Ran out of GPU memory

```
root@hostname ~]# /opt/ibm/vision-edge/bin/deploy_zip_model.sh -m user_detectron_cars8 -p
7018 -g 1 /root/inference-only-testing/cars_detectron_model.zip
Deployment failed. Here are logs before the failure:
File "/opt/detectron/detectron/core/test_engine.py", line 331, in
initialize_model_from_cfg
    model, weights_file, gpu_id=gpu_id,
File "/opt/detectron/detectron/utils/net.py", line 112, in
initialize_gpu_from_weights_file
    src_blobs[src_name].astype(np.float32, copy=False))
File "/usr/local/lib/python2.7/dist-packages/caffe2/python/workspace.py", line 321, in
FeedBlob
    return C.feed_blob(name, arr, StringifyProto(device_option))
RuntimeError: [enforce fail at context_gpu.cu:359] error == cudaSuccess. 2 vs 0. Error
at: /tmp/pytorch/caffe2/core/context_gpu.cu:359: out of memory
root      : INFO      Callback message: {'msgId':
'6ef7e371-1209-47b3-94c3-940640324ac8', 'msgReturnCode': 'ErrModelLoading', 'msgDesc':
'Traceback (most recent call last):\n File "/opt/DNN/dnn/deploy_process.py", line 165,
in modelLoading\n self.caller.onModelLoading()\n File "/opt/DNN/dnn_impl/
cod_detectron/deploy_service.py", line 64, in onModelLoading\n self.model =
infer_engine.initialize_model_from_cfg(self.deploy)\n File "/opt/detectron/detectron/
core/test_engine.py", line 331, in initialize_model_from_cfg\n model, weights_file,
gpu_id=gpu_id,\n File "/opt/detectron/detectron/utils/net.py", line 112, in
initialize_gpu_from_weights_file\n src_blobs[src_name].astype(np.float32, copy=False))
```

```
\n File "/usr/local/lib/python2.7/dist-packages/caffe2/python/workspace.py", line 321,
in FeedBlob\n     return C.feed_blob(name, arr, StringifyProto(device_option))
\nRuntimeError: [enforce fail at context_gpu.cu:359] error == cudaSuccess. 2 vs 0. Error
at: /tmp/pytorch/caffe2/core/context_gpu.cu:359: out of memory \n', 'msgState':
'aborted', 'msgTime': 1551801403956}
root      : INFO      Wait 5s for messaging completed...
[root@hostname ~]#
```

- Invalid GPU ID specified

```
[root@hostname ~]# /opt/ibm/vision-edge/bin/deploy_zip_model.sh -m user_detectron_cars8 -
p 7018 -g 5 /root/inference-only-testing/cars_detectron_model.zip
Deployment failed. Here are logs before the failure:
Failed building wheel for nvidia-ml-py
Running setup.py clean for nvidia-ml-py
Failed to build nvidia-ml-py
Installing collected packages: nvidia-ml-py
Running setup.py install for nvidia-ml-py: started
Running setup.py install for nvidia-ml-py: finished with status 'done'
Successfully installed nvidia-ml-py-375.53.1
You are using pip version 8.1.1, however version 19.0.3 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
Cannot find gpu 5.
[root@hostname ~]#
```

- Processing was interrupted:

```
/usr/bin/docker-current: Error response from daemon: Conflict. The container name "/"
decrypt" is already in use by container
ec0932898a65b82ed47504c8baa2507046d7bb0fcf460405d6201d3088bc9731.
You have to remove (or rename) that container to be able to reuse that name.
```

To fix the problem, run these commands:

```
docker stop decrypt
docker rm decrypt
```

- Tried to deploy a Detectron model on a CPU:

```
[root@hostname ~]# /opt/ibm/vision-edge/bin/deploy_zip_model.sh -m user_detectron_cars8 -
p 7018 -g -1 /root/inference-only-testing/cars_detectron_model.zip
Deployment failed. Here are logs before the failure:
Failed building wheel for nvidia-ml-py
Running setup.py clean for nvidia-ml-py
Failed to build nvidia-ml-py
Installing collected packages: nvidia-ml-py
Running setup.py install for nvidia-ml-py: started
Running setup.py install for nvidia-ml-py: finished with status 'done'
Successfully installed nvidia-ml-py-375.53.1
You are using pip version 8.1.1, however version 19.0.3 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
We currently do not support CPU mode for Detectron models.
[root@hostname ~]#
```

- Deployment times out:

```
[root@hostname ~]# /opt/ibm/vision-edge/bin/deploy_zip_model.sh -t 15 -m
user_custom_cars3 -p 7008 -g -1 /root/inference-only-testing/cars_keras-
frncnn_custom_model.zip
Deployment timed out at 15 seconds
```

If the deployment times out, increase the time limit by using the `-t` option.

Stopping a deployed model

To stop the deployed model, run the following commands. When you stop the deployed model, the GPU memory is made available.

```
docker stop <model-name>
docker rm <model-name>
```

Example 1:

```
docker stop dog
docker rm dog
```

Example 2:

```
docker stop car
docker rm car
```

Performing an inference with IBM Maximo Visual Inspection Edge

Inference can be done by using the deployed model with a local image file or a URL to an uploaded image file.

- [“Performing an inference” on page 136](#)
- [“Inference output” on page 137](#)

Performing an inference

Use this information to perform an inference.

Optional Parameters:

confthre

Confidence threshold. Specify a value in the range [0.0,1.0], treated as a percentage. Only results with a confidence greater than the specified threshold are returned. The smaller confidence threshold you specify, the more results are returned. If you specify 0, many, many results will be returned because there is no filter based on the confidence level of the model. The default value is 0.5.

containRle

This option is only available for Detectron models. If this is true, the inference output will include RLEs of the segments. The default value is false.

containPolygon

This option is only available for Detectron models. If it is set to true, the polygon for the segments is included in the output. The default value is true.

GET method:

Required Parameters:

imageurl

The URL address of the image. The URL must start with http:// or https://.

Example:

```
curl -G -d "imageurl=https://ibm.box.com/shared/static/
i98xa4dfpff6jwv0lxmcu4lybr8b5kxj.jpg&confthre=0.7&containPolygon=false&containRle=true" http://
localhost:5000/inference
```

POST method:

Required Parameters:

imagefile

The name of the image file to be used for inference.

Example:

```
curl -F "imagefile=@$DIR/data/bird.jpg" \
-F "confthre=0.7" \
-F "containPolygon=false" \
-F "containRle=true" \
http://localhost:5000/inference
```


Example 1 - Classification:

```
curl -F "imagefile=@/home/testdata/cocker-spaniel-dogs-puppies-1.jpg" http://localhost:6001/inference
```

Example 2 - Object detection:

```
curl -G -d "imageurl=https://assets.imgix.net/examples/couple.jpg" http://localhost:6002/inference
```

Example 3 – Object detection of a Tiny YOLO model with confidence threshold:

```
curl -F "imagefile=@/home/testdata/Chihuahua.jpeg" -F "confthre=0.8" http://localhost:6001/inference
```

Note: Confidence threshold works for Faster R-CNN, Detectron, and Tiny YOLO object detection models and GoogLeNet image classification models.

Example 4 - Object detection of a Detectron model that contains polygon segments instead of RLEs (default setting)

```
curl -F "imagefile=@/home/user/model/new_models/pics/cars.jpg" -F "confthre=0.98" http://localhost:6001/inference
```

Example 5 - Object detection of a Detectron model that contains RLE segments instead of a polygon:

```
curl -F "imagefile=@/home/user/model/new_models/pics/cars.jpg" -F "confthre=0.98" -F "containRle=true" -F "containPolygon=false" http://localhost:6001/inference
```

Inference output

IBM Maximo Visual Inspection Edge can deploy image classification and object detection models.

Image classification model

A successful classification will report something similar to the following:

Example 1 output - success

```
{"classified": {"Cocker Spaniel": 0.93}, "result": "success"}
```

The image has been classified as a Cocker Spaniel with a confidence of .93.

Example 1 output - fail

```
{"result": "fail"}
```

The image could not be classified. This might happen if the image could not be loaded, for example.

Object detection model

A successful detection will report something similar to the following:

Example 2 output - success

```
{"classified": [{"confidence": 0.94, "ymax": 335, "label": "car", "xmax": 576, "xmin": 424, "ymin": 160, "attr": []}], "result": "success"}
```

The cars in the image are located at the specified coordinates. The confidence of each label is given.

Example 2 output - success

```
{"classified": [], "result": "success"}
```

Object detection was carried out successfully, but there was nothing to be labeled that has confidence above the threshold.

Example 2 output - fail

```
{"result": "fail"}
```

Objects could not be detected. This might happen if the image could not be loaded, for example.

Example 4 output - success

The output includes a rectangle and polygon.

```
{
  "classified": {
    "confidence": 0.9874554872512817,
    "ymax": 244,
    "label": "car",
    "xmax": 391,
    "xmin": 291,
    "ymin": 166,
    "polygons": [
      [[325, 170], [322, 172], [318, 172], [311, 178], [311, 181], [300, 189], [297, 189], [289, 195], [289, 232], [297, 238], [297, 240], [304, 246], [307, 246], [315, 240], [322, 240], [325, 238], [369, 238], [372, 240], [387, 240], [394, 235], [394, 198], [387, 192], [387, 189], [383, 187], [383, 184], [376, 178], [376, 175], [372, 172], [369, 172], [365, 170]]
    ],
    "result": "success"
  }
}
```

Example 5 output - success

The output includes a rectangle and rle.

[illegible]

Inference on embedded edge devices

Using edge computing for your inference models helps save processing time by removing latency issues, ensures security, and also decreases bandwidth usage. This topic describes how to use IBM Maximo Visual Inspection with embedded edge devices.

- A full end-to-end use case is available if you use the DeepRed FPGA by V3 Technology that takes camera input, analyzes the video, and outputs the video with bounding-boxes on an HDMI attached device. See the section [“Inference on DeepRED” on page 138](#) for details.
- If you want to create your own solution or have a different FPGA board, then use the information in this section: [“Inference with a custom solution” on page 139](#).

Inference on DeepRED

DeepRED is an embedded artificial intelligence development system that supports IBM Maximo Visual Inspection. It lets you quickly deploy the trained model for testing and production.

Generate an IP core for use with DeepRED:

1. Perform customization. For DEEPRED this is not optional. If they want to use DEEPRED, they need to change the ZC706 to DEEPRED in the configmap (both instances of it). They should not add custom DSP_NUM, etc.
2. Perform optional customization.
 - a. On the IBM Maximo Visual Inspection host operating system, run the appropriate command.
 - For a standard install:

```
$ /opt/ibm/vision/bin/kubect1.sh edit configmap vision-config
```
 - b. Find the row beginning EMB_COD_IMAGE in the configuration file and replace ZC706 with DEEPRED:

```
"EMB_COD_IMAGE": ["DEEPRED,DEEPRED,vision-dnn-edge:1.3.0.0"],
```
 - c. Save and exit.
3. Restart IBM Maximo Visual Inspection by running the appropriate command. The deleted pods will automatically restart.
 - For a standard install:

```
$ /opt/ibm/vision/bin/kubect1.sh delete pod -l app=vision
```

4. Train your model.

- On the **Train data set** page, for **Type of training**, select **Object detection**.
- Under **Advanced options**, choose **Optimized for speed**

5. Copy the IP core file for compilation. The generated FPGA IP core is named *UUID*-ipcore.zip, where *UUID* is the UUID of the trained model. It is stored in the following location:

- For a standard install: /opt/ibm/vision/volume/data/trained-models.

Inference with a custom solution

Using a custom solution requires appropriate hardware and software, as well as FPGA development skills. You must be able to:

- Take an existing IP core and use Vivado to merge it into a custom solution. Refer to the [PIE DNN Accelerator IP Integration Guide.pdf](#) for instructions to integrate the generated DNN IP core into your project.
- Set up and use Vivado, Petalinux, and other software.

Environment requirements

- A chip set that can provide enough BRAM, such as Xilinx 7035 or later
- A board with PL side (not just PS side) DRAM so that it can provide sufficient bandwidth between the FPGA and DRAM.

Follow these steps to generate an IP core for use with a custom solution. The examples included are for a ZC706 card:

1. Perform optional customization.

By default, IBM Maximo Visual Inspection is configured to use the following resources on a ZC706 card. However, you can customize these values.

```
DSP_NUM=700
RAM18E_NUM=800
DDR_BANDWIDTH=80000.0
DDR_DATA_WIDTH=512
FPGA_TYPE=xc7z045ffg900-2
```

a. On the IBM Maximo Visual Inspection host operating system, run the appropriate command.

- For a standard install:

```
$ /opt/ibm/vision/bin/kubect1.sh edit configmap vision-config
```

b. Find the row beginning EMB_COD_IMAGE in the configuration file and input your custom values.

For example, for a ZC706 card, replace **ZC706** with the appropriate values for your card: "EMB_COD_IMAGE": ["ZC706,**ZC706**,vision-dnn-edge:1.3.0.0"], as shown here:

```
"EMB_COD_IMAGE": ["ZC706,DSP_NUM=700:RAM18E_NUM=800:DDR_BANDWIDTH=80000.0:
DDR_DATA_WIDTH=512:FPGA_TYPE=xcvu9p12fsgd2104e,vision-dnn-edge:1.3.0.0"],
```

c. Save and exit.

2. Restart IBM Maximo Visual Inspection by running the appropriate command. The deleted pods will automatically restart.

- For a standard install:

```
$ /opt/ibm/vision/bin/kubect1.sh delete pod -l app=vision
```

3. Train your model.

- On the **Train data set** page, for **Type of training**, select **Object detection**.
- Under **Advanced options**, choose **Optimized for speed**

4. Copy the IP core file for compilation. The generated FPGA IP core is named *UUID-ipcore.zip*, where *UUID* is the UUID of the trained model. It is stored in the following location:

- For a standard install: `/opt/ibm/vision/volume/data/trained-models`.

Decrypting a trained model

Models trained and exported by version 1.1.4 and earlier versions of IBM PowerAI Vision are encrypted and are intended for deployment in IBM Maximo Visual Inspection or IBM Maximo Visual Inspection Edge. Starting with version 1.1.5, trained and exported models are not encrypted.

You can decrypt a model that was trained with IBM PowerAI Vision 1.1.4 or earlier by running `decrypt_zip_model`. This will allow data scientists to understand the weights and networks configured by IBM Maximo Visual Inspection and possibly use that information to further train the model. The decrypted model can also be used to port these models to edge devices not supported by IBM Maximo Visual Inspection.

Usage: `/opt/ibm/vision-edge/bin/decrypt_zip_model.sh [-h|--help] | [[-o string] model_file.zip]`

output

Specifies the file name for the output decrypted model.

model_file

A trained model exported from IBM PowerAI Vision 1.1.4 or earlier.

Example:

```
/opt/ibm/vision-edge/bin/decrypt_zip_model.sh -o car_frcnn_decrypted.zip  
car_frcnn.zip
```

This will generate a new zip file `car_frcnn_decrypted.zip`, which is not password protected.

Chapter 15. Integrating with IBM Maximo Visual Inspection Mobile

Maximo Visual Inspection Mobile is a native iOS/iPadOS mobile app that brings the capabilities of IBM Maximo Visual Inspection to the edge and rapidly enables visual inspections on mounted or handheld devices. Maximo Visual Inspection Mobile uses the models trained on IBM Maximo Visual Inspection and performs inferencing using the integrated camera on an iOS/iPadOS device. The app can run models remotely or can use Core ML models that are exported from IBM Maximo Visual Inspection, which enables local inferencing on-device without requiring network connectivity.

Overview of Maximo Visual Inspection Mobile

Maximo Visual Inspection Mobile has the following capabilities:

- Perform local and remote inferencing on object detection and image classification models trained on IBM Maximo Visual Inspection.
- Take photos and upload to IBM Maximo Visual Inspection for model training and refinement
- Monitor inspection performance for to each device and its location via an integrated reporting dashboard
- Provide remote management and configuration of Maximo Visual Inspection Mobile instances running on remote devices

Latest enhancements

The past few versions of the Maximo Visual Inspection Mobile contain these features and improvements:

- Expanded model support to include YOLO v3.
- AI-based triggering of inspections, known as *Visual Trigger*. For more information, see [“Visual Trigger” on page 152](#).
- Support for barcode reader and optical character recognition (OCR). For more information, see [“Barcode and OCR Trigger” on page 154](#).
- Sending inference results to a secondary inspection (also known as *pipelining*). For more information, see [“Creating inspections” on page 148](#).
- Ability to add AND/OR logic rules to inference results. For more information, see [“Creating inspections” on page 148](#).
- Customizable MQTT notification messages. For more information, see [“Configuring MQTT” on page 155](#).
- Enhanced remote manager, including device battery status.
- Enhanced dashboard features.

Follow the guidance in the following topics carefully. Otherwise, the product might not work properly.

Concepts

The following terminology is used in Maximo Visual Inspection Mobile.

Collect mode

Using the app to take pictures that are uploaded to a data set to be used to train or refine a model.

Dashboard

A dashboard displays images from a single IBM Maximo Visual Inspection data set within a project. Additionally, it displays Pass / Fail metrics for the displayed images.

Fixed (mounted) device

A device that is mounted in a single position, such as when monitoring a production line. Maximo Visual Inspection Mobile is used with only infrequent human intervention and can also be managed remotely by other Maximo Visual Inspection Mobile instances.

Handheld device

A device that is not mounted but is held, such as when taking photos of a damaged roof. Maximo Visual Inspection Mobile is used manually.

Inspect mode

Using the app to take pictures that are sent to a trained model for labeling and deep learning.

Inspection

An inspection is a group of settings that define specific IBM Maximo Visual Inspection elements to interact with. It ties together one IBM Maximo Visual Inspection trained model and one data set that exist in the same IBM Maximo Visual Inspection *project* (also called project group). The data set is used as the target of uploads when the photos are taken on the app. The model is used when the app is in Inspect mode. The inspection also specifies a trigger string that external systems can use to identify the inspection when sending messages to capture photos.

Managed device

Managed devices have apps installed using MDM (managed apps). Managed apps can contain preconfigured settings that are controlled by the management party. The MDM server can remove managed apps and their associated data on demand, or specify whether the apps should be removed when the MDM profile is removed. Additionally, the MDM server can prevent managed app data from being backed up to iTunes and iCloud.

Single App Mode

Single App Mode, sometimes called "Single App Lock", is a feature for supervised devices that restricts the device to running only one app. While this mode is enabled, the selected app will stay in the foreground.

Supervised device

Supervision gives an organization more control over their iOS and iPadOS, allowing restrictions such as disabling AirDrop or Apple Music, or placing the device in Single App Mode.

Related concepts

[Creating and working with project groups](#)

Project groups allow you to group trained models with the data sets that were used for training. This grouping is optional but is a useful way to organize related data sets. For example, project groups would be useful with a workflow that clones data sets as you refine labels and work toward a more accurate model. Project groups can be used with a production work flow strategy and automatic model deployment for even more functionality.

Related information

[Apple Device Management](#)

How Maximo Visual Inspection Mobile interacts with IBM Maximo Visual Inspection

Maximo Visual Inspection Mobile is closely tied to IBM Maximo Visual Inspection, as it uses IBM Maximo Visual Inspection models and data sets.

The following figure illustrates how Maximo Visual Inspection Mobile works with IBM Maximo Visual Inspection and other elements in your environment.

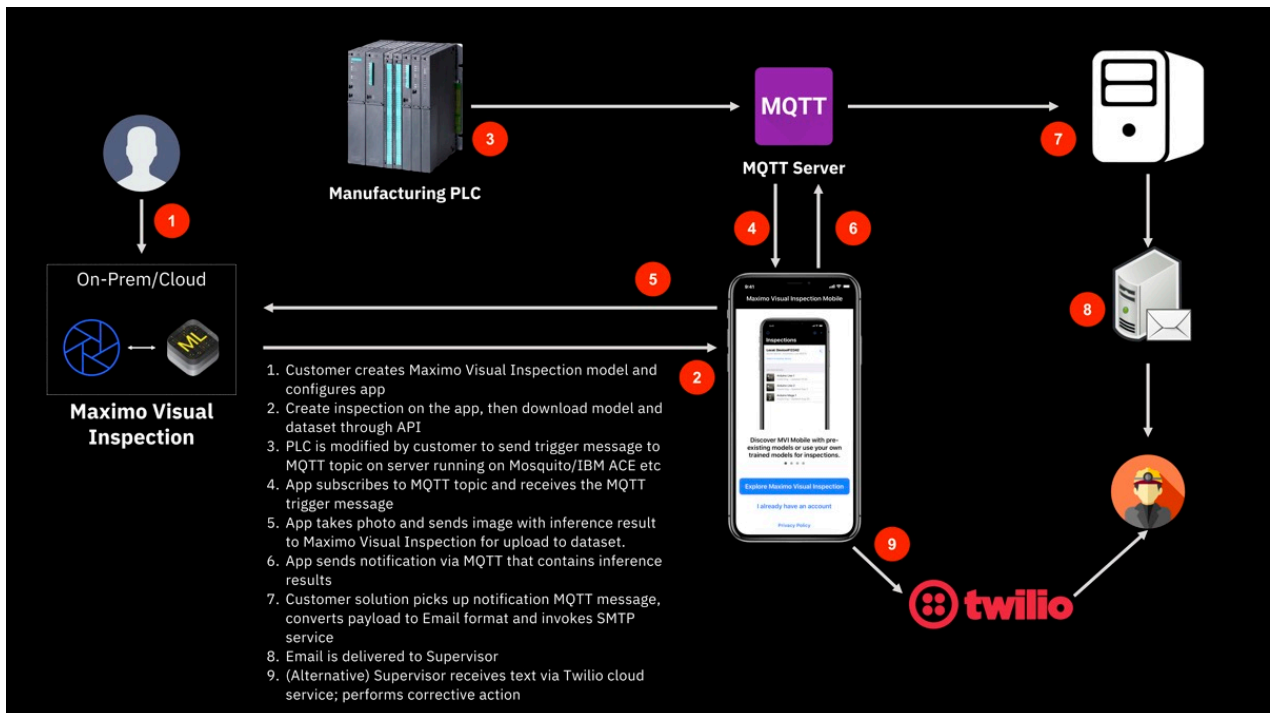


Figure 16. Maximo Visual Inspection Mobile and IBM Maximo Visual Inspection environment

Impacts to IBM Maximo Visual Inspection

In general, you can continue to work in IBM Maximo Visual Inspection while Maximo Visual Inspection Mobile users run inferences and upload models with no impact to IBM Maximo Visual Inspection. However, there are some ways that Maximo Visual Inspection Mobile will affect IBM Maximo Visual Inspection:

- When images are uploaded to an IBM Maximo Visual Inspection data set after inferencing, image metadata is stored with the image in the following format:

Metadata format

```
PhotoType__MMDDYYYYHHMMSS__TriggerDate__TriggerReference__TriggerString__
InspectionName__PASSORFAIL__Location__DeviceName__ModelType__LabelNames__
LabelConfidences__LabelThresholds__LabelExpectedCounts__LabelBoundingBoxes__
AboveOrBelow__PASSorFAIL__ANDOR__PRIMARYorSecondary__FailedLabels___.jpeg
```

Object classification example

```
INSPECTION__20200504143248__20200504040__MANUALREFDATA__arduino__Arduino__
FAIL__boston__iphone__Classification__ArduinoMega__ArduinoUno__0.0__99.97__
75__75__0__0__0__0__0__0__0__0__0__0__PASS_FAIL__OR__OR__1__1__arduino_mega.jpeg
```

Object detection example

```
INSPECTION__20200501214556__20200501046__MANUALREFDATA__arduino__
ArduinoType__FAIL__toronto__d1__ObjectDetection__arduino_uno__
arduino_uno__arduino_mega__98.85__62.44__0.0__90__90__75__0__0__0__
296__34__207__0__306__61__441__216__0__0__0__1__1__0__PASS_FAIL_PASS__
OR__AND__OR__1__1__1__arduino_mega___.jpeg
```

- Users can create new projects and data sets on the fly from Maximo Visual Inspection Mobile.

Planning for and installing

Learn the requirements and steps for installing Maximo Visual Inspection Mobile.

Requirements

The following are requirements for running Maximo Visual Inspection Mobile:

- Any iOS 13 compatible device including iPods, iPhones, and iPads.
- One Maximo Visual Inspection Mobile license for each device.
- Any network connection, including WiFi, LTE and Ethernet, using a 3rd party Lightning to Ethernet adapter. If supported by your organization, you can access internal networks by using VPN. Internet connectivity is only required for sending Twilio text messages.
- IBM Maximo Visual Inspection 1.3.0 (only one license is required)
- A IBM Maximo Visual Inspection project that contains a trained model enabled for Core ML and a data set. Classification and object detection models are supported.
- (Optional. Required for fixed (mounted) devices) - An MQTT broker such as IBM Integration Bus, App Connect Enterprise, HiveMQ, or Mosquitto.

Note: IoT cloud services that support MQTT, such as the IBM IoT Platform, are NOT compatible because they implement only a subset of the MQTT specification.

- (Optional. Required for fixed devices) - Mobile Device Management (MDM) devices must be "supervised" in order to put the device into "single app" mode. That means you must order from Apple Device Enrollment Program directly or use Apple Configurator on Mac to manually prepare (format) devices. Supervision is a stronger form of management than "managed".

Using MDM, you can assign the app to an internal AppStore provided by the MDM vendor. This allows you to preconfigure device settings.

Installing

To install Maximo Visual Inspection Mobile, download the app from the Apple App Store. After install, it is important that the app is kept current by installing any updates as they are made available.

Setting up

Initial setup of IBM Maximo Visual Inspection Mobile involves connecting it to IBM Maximo Visual Inspection and optionally connecting it to an MQTT broker.

Setting up Maximo Visual Inspection Mobile with IBM Maximo Visual Inspection

Follow these steps to set up Maximo Visual Inspection Mobile and get it running in your environment:

1. In the Maximo Visual Inspection Mobile app, optionally select **Explore Maximo Visual Inspection** to access demo mode.
2. Select **I already have an account** to turn off demo mode and start configuring the app. When specifying the IBM Maximo Visual Inspection server, for Base URL, you must include the server API. For example: `https://xxx/visual-inspection/api/`.

Important: The device name must be unique or results will be unpredictable.



3. Tap the icon for **Global Settings**:

On the **Global Settings** page, you can toggle Demo Mode and Handheld Mode, as well as specifying the IBM Maximo Visual Inspection server and MQTT broker details.

Note: These settings cannot be edited remotely.

Demo Mode

The app runs locally using preloaded CoreML models and all other settings are hidden. There is an overlay that indicates to the user that it contains only demonstration data.

Tap the + icon to attempt to create a new inspection. This opens a form that allows a user to request access to a IBM Maximo Visual Inspection server. The requester will be contacted if access has been provisioned.

Handheld Mode

Auto-Capture mode and MQTT settings are disabled.

IBM Maximo Visual Inspection Server

If possible, configure with TLS enabled and a valid CA-issued certificate.

MQTT broker

Used for fixed (mounted) devices. The broker must conform to the full MQTT specification.

Note: This group of fields displays only when **Demo Mode** and **Handheld Mode** are toggled off.

After IBM Maximo Visual Inspection Mobile is initially set up to work with IBM Maximo Visual Inspection, configure the device's settings. See [“Device configuration” on page 145](#).

Device configuration

IBM Maximo Visual Inspection Mobile has many device-level settings that you can configure.

Before you configure these settings, make sure that IBM Maximo Visual Inspection Mobile is set up to work with IBM Maximo Visual Inspection. See [“Setting up ” on page 144](#).



To access the **Configure Device** menu, tap the icon next to the device name:

If the device you want to configure is not listed, tap Change to Another Device. This will search for other devices connected on the same MQTT broker. It may take up to 10 seconds to see all devices. The device in your hand will be prefixed by "Local". Other devices are prefixed by "Remote".

The following settings are available:

- [“Socket Trigger > Trigger Config” on page 145](#)
- [“App Settings” on page 146](#)
- [“Location Details” on page 147](#)
- [“Twilio Settings” on page 147](#)
- [“Socket Notification” on page 147](#)
- [“Auto-Capture” on page 147](#)

Socket Trigger > Trigger Config

An external system can invoke the device using a socket based connection. The device accepts a trigger message that is similar to MQTT trigger format, except that it is comma separated. For example:

```
"date","ref","triggerString"
```

Maximo Visual Inspection Mobile responds with PASS or FAIL over the socket connection, depending on the inspection results.

Note: The response merely signals back to the calling system whether the inspection passed or failed, defined by the configured rules in the inspection. Detailed inspection results are still sent by the configured channels, such as MQTT notification or Socket Notification.

To maintain a persistent socket based connection, the calling system can send a pingreq message, to which the app will respond with PINGOK. This is optional. Any processing errors, such as a nonconforming trigger message results in a NOTOK response instead.

App Settings

Flash Mode

Controls the camera flash mode.

Upload to IBM Maximo Visual Inspection

Saves photos from collections and inspections to the data set defined in the inspection being used.

Upload Failures Only

Uploads images to IBM Maximo Visual Inspection when an inspection fails.

Note: Avoid toggling this option on, since it skews the results on the dashboard. Instead, archive old images from the server by using a script.

Image pixel dimensions

The size of the photo uploaded to the IBM Maximo Visual Inspection data set. It is recommended that you keep the default settings. When you use Visual Trigger, Barcode or OCR Trigger, or Video mode, the maximum image size uploaded is 1080 pixels.

Shutter Time

How much time to give the iOS camera to auto focus.

Use lens position

Retain the focal length used on the first photo taken so that less auto focus time is required for subsequent photos. This might be useful in high volume production lines where the device is mounted and always has the same angle.

Lens reset

The number of images to re-use the focal length for.

Save to Photos

Save the images to the mobile device's photo library.

Note: Toggling this option on can significantly impact local storage.

Enable Continuous Learning

This enables the Use Latest Model inspection setting. When Use Latest Model is turned on, inferences always occur on the latest deployed model in a IBM Maximo Visual Inspection project. In the case of models that are enabled for CoreML, the latest *trained* model is used.

Note: This does not turn on Use Latest Model. It enables the setting so it can be turned on in an inspection. If this is not turned on, you cannot turn on Use Latest Model for any inspections.

Always On Camera

Prevents the inference results screen from displaying. This option can be useful if you take rapid photos (up to one photo per second) when in *Auto-Capture* mode. This option is also useful in video mode, if you set the **Inspect When Label Detected** rule to run every frame.

Notify On Inspection Pass

Sends a notification message via MQTT, even if an inspection passes. By default, only failed inspections send notifications.

Camera Zoom Level

Specifies the camera zoom level to use. Zoom level can also be applied by using a pinch gesture on the camera preview screen. Regardless of zoom method, the setting is retained if the device is mounted or placed in a fixed position.

Dashboard Refresh Interval

Specifies how frequently the app's dashboard is refreshed.

Frame Rate Throttle

The number of frames per second to read from camera when the app is in *Video mode*. The app automatically switches to video mode in the following scenarios:

- When the app is in *Auto-Capture* mode and the inspection is defined to use Visual Trigger.
- When the app is in *Manual Capture* mode and you toggle **Enable Streaming** on during an inspection.

A lower **Frame Rate Throttle** reduces thermal load and energy consumption. Select the maximum value only if your device is equipped with an A13 Bionic chip or higher.

Location Details

Location details that were entered on the Welcome screen can be edited here. If you change these settings, ensure that any systems sending MQTT messages to this device are reconfigured as Maximo Visual Inspection Mobile will resubscribe to the MQTT topics accordingly, with the specified location and device name in the topic path.

Twilio Settings

You must have a Twilio account to enable Twilio notifications. When Twilio notifications are on, the `thresholdMsg` field in the more comprehensive MQTT message payload is sent out as a text message using the Twilio Cloud Messaging platform. You must specify the account information as well as the following:

Note: For the account **User name**, specify the Twilio account SID.

From No.

A phone number set up in the Twilio account, used as the message sender.

Supervisor No

The phone number that text messages will be sent to.

Socket Notification

Writes out to a socket the JSON payload that is identical to the MQTT message for inspection results. This is useful if MQTT brokers are not be available.

Auto-Capture

Send Trigger to Secondary Device

This option is typically used with a Visual Trigger or a Barcode and OCR trigger. The use case for this option involves two devices and two inspections. One device is positioned up close. Another device is mounted further back, with a clear line of sight to the same object. The first device reads the object's bar code or serial number and does a primary inspection. Then, an inspection rule from the first device triggers a secondary inspection, which is done on the second device.

Image Trigger Delay(s)

The number of seconds to wait after a Visual Trigger or a Barcode or OCR Trigger is triggered before the inspection is run. This setting applies to inspections that are triggered on primary or secondary devices.

Pass video frame to Inspection

By default, a new photo is taken in streaming video mode, in accordance with the inspection that is specified in the Visual Trigger setting. By toggling **Pass video frame to Inspection** on, the default behavior is overridden, and the video frame itself is used as the image for the secondary inspection.

Select Inspection

This Visual Trigger setting specifies which inspection to run when video frames are streamed in *Auto-Capture* mode. The inspection that is selected must use a CoreML model and must have at least one label.

Mode

Specifies whether OCR runs in Fast or Accurate mode.

Note: Accurate mode consumes significant resources.

Frames to Reset

After a barcode (or OCR text) is read, this setting specifies the number of frames with the same barcode or text to ignore before the next scan is done.

Add Rule

Defines regular expression rules for reading barcodes or OCR text. When there is a match, you can optionally run an inspection by passing the value that was read as the reference. If you leave **Trigger** empty, then the device sends the value that was read on an MQTT topic called `ibmvi/barcodeocrvalue`.

Creating inspections

An inspection is a group of settings that define specific IBM Maximo Visual Inspection elements to interact with. Inspections are used in both handheld mode and fixed (or mounted) mode. To create an inspection, from the Inspections home screen, tap the circle with three dots, then tap Create new inspection. Specify the following when creating a new inspection:

Image

A reference image for the inspection to help identify this inspection. This image is not uploaded to IBM Maximo Visual Inspection.

Project

The project in IBM Maximo Visual Inspection to use. You can select an existing project or create a new one, which will be added to IBM Maximo Visual Inspection.

Model

The model within the project to use for inference (if this inspection will be used for inference). If this inspection will only be used to collect and upload photos, choose **None** for the model.

Considerations for Core ML models:

- Only Core ML models are downloaded to the Maximo Visual Inspection Mobile device. Therefore, if you want to do local inspections, you must have a GoogLeNet or Tiny YOLO v2, or a YOLO v3 model. Other types of models will not have Core ML assets in IBM Maximo Visual Inspection 1.3.0.
- If the selected model has been trained to support Core ML, that is indicated in the app. For such models, the app will always use the Core ML model if "Core ML support enabled" is turned on. Core ML files are downloaded for use on the device and support offline inference. However, they do not support remote inference.

If you do not want to use the device for local inspections and want to use a GoogLeNet, tiny YOLO v2, or YOLO v3 model, do not turn on the "Core ML support enabled" setting.

- For models that support Core ML, the model only needs to be trained. It does not need to be deployed, since Maximo Visual Inspection Mobile downloads the necessary files to the device.
- You can update the Core ML model later by opening the inspection and tapping **Update CoreML model**. The Core ML assets associated with the latest trained model in the project are downloaded to the device.

Important: If the latest model has labels that do not match those in the model specified in the inspection, an error will result.

Set Thresholds

The results threshold is the required accuracy threshold for each label in the model. Inference results with a confidence lower than the value set for **Ignore Results Below** are not included in the results.

The inspection will be marked as a fail if the confidence returned for any label is above the specified threshold. Conversely, if you set Notify When to "below", then the inspection will be marked as a failure when any label returns a confidence that is less than specified.

For example, if you trained on "bad connector" and an inspection resulted in a bad connector with 75% confidence, then the inspection would be marked as a failure and you would want Notify When to be set to "above".

Alternatively, if you trained on "good connector", you would set Notify When to "below".

In the **Inspection Fails When** section of the **Set Thresholds** screen, you can define one or more rules that describe a failed inspection. When you define two or more rules, you can add AND or OR logic between them. By default, OR logic is applied, so if any label fails, the entire inspection fails. You can group labels by joining them with one or more AND operators. Once grouped, every label within that group must fail in order for that the inspection result to fail.

You can drag-and-drop labels to change the order. To move a label, tap and hold its handle icon until the entry detaches. Then drag the label on top of the label that you want to swap with. Then release the label.

For example, you can rearrange labels and change their AND or OR logic, so the inspection failure is definition is:

```
Label11 AND Label12 OR Label13
```

In this scenario, an inspection is considered a failure if either of the following conditions are met:

- Label1 and Label 2 fail
- Label3 fails

When a label is listed, you can tap its **Advanced > Additional Rules** to define the rule with more granularity:

Expected Count

This option is available only for Object Detection models. The expected count specifies how many of each type of object must be in a scene. Each successfully identified object type is counted. If the number of objects that are found does not match the "expected count" value, the inspection is marked as "fail". Selecting this option returns you to the **Advanced** screen, where you can type a value in the **Expected Count** field.

Inspect When Label Detected

Select this option to automatically run the inspection when one or more labels are detected according to the frame rate timing rules. Select this option to enable an inspection to use *Auto-Capture* model.

Note: You can use this option only if model supports CoreML, ideally YOLO v3.

Selecting this option returns you to the **Advanced** screen, which displays more fields that are specific to this option:

Frames To Consider

This value represents the rolling consideration window for Visual Trigger.

Frames Label Must Be Present

This value represents the number of frames within the consideration window where the label must be present at the label's specified confidence level. The inspection runs after these conditions are met.

Frames Without Label To Reset

This value represents the number of frames where the label must not be present after an inspection is run. When this threshold is met, the next inspection can run.

Obstruction For Label Detection

Select this option if there is a risk that the camera's line of sight might be blocked, say, by a person that walks in front of it. Obscuring the camera might prevent Visual Trigger from firing again. An obstruction in manual capture mode can also have undesirable effects, since the obstruction can pause frame rate counting.

Send Label to Secondary Inspection

Use this option to send a label from one inspection (and its bounding box if it has one) to a secondary inspection. This operation, which can yield better accuracy, is called *pipelining*.

Selecting this option returns you to the **Advanced** screen, where an **Inspection** field displays. Tap it, and select the secondary inspection.

The primary and secondary inspections in a *pipeline* must support CoreML to maximize performance. However, the primary inspection can be an Object Detection model or a Classification model. Another performance consideration is to perform inference on the same device. Although remote inferencing is supported, latency might be an issue if multiple images are uploaded for inference.

The primary inspection fails if the secondary inspection fails according to any of its rules.

Data Set

The data set within the project to upload images to. You can select an existing data set or create a new one, which will be added to IBM Maximo Visual Inspection.

Use Latest Model

Whether to use the latest model. This option is only available if Enable Continuous Learning has been enabled in App Settings, which means that remote inferences occur on the latest deployed model in a project in IBM Maximo Visual Inspection. However, if the labels in the latest model do not match the labels that were being used initially for this inspection, no results will occur.

Considerations for Core ML models:

The Core ML assets associated with the latest trained model in the project are downloaded to the device. However, if the labels in the latest model do not match the labels that were being used initially for this inspection, no results will occur.

Trigger String

A trigger string that identifies this inspection uniquely on the device. When an MQTT or socket based message is received from an external system in auto capture mode, the message's `trigger` field is used to find an inspection that has a matching trigger string defined. When using Visual Trigger, the model must have a label that matches a trigger string. The label must meet the specified threshold and the frame timing rules that are defined in the inspection. A photo is taken only if there is a match. The results of an attempted match are shown in the top right corner of the **Auto-Capture** screen. Trigger strings are used only in *Auto-Capture* mode on a mounted device.

Dashboard view

A dashboard displays images from a single IBM Maximo Visual Inspection data set within a project. Additionally, it displays Pass / Fail metrics for the displayed images.

Multiple inspections can upload to a single data set, however, it is recommended that each dashboard displays data for just one inspection or the metrics might be skewed. The dashboard refreshes on an interval defined in App Settings, which is 60 seconds by default. It also refreshes when a new image is uploaded.

Metrics are calculated over the number of images available, up to the configured number of images to return from IBM Maximo Visual Inspection (which is up to 500 images).

Photo thumbnails display a green check mark if they passed, red if they failed, and no mark if it was only collected. Click the photo to see more details, such as the confidence level. If you do not want to see all the images, you can filter by image type. For example, inspection images only.

Configure a dashboard

From the Dashboard screen, tap the ellipses and fill in the fields presented. When specifying the number of images to return, it is recommended that you do not exceed the default value of 100.

Data Collection vs. Inspection

When you take photos in *collect* mode, they are uploaded to a data set in IBM Maximo Visual Inspection. The uploaded images can then be labeled and used to train a model, or they can be used to validate a model. You can then "upgrade" your inspection to specify the newly trained model, thereby retaining all of the existing settings, such as the trigger string.

You can collect data for training in either handheld or fixed mode.

Note: If no model is specified for an inspection, collect mode is automatically used when taking photos.

Using Maximo Visual Inspection Mobile for labeling and deep learning

If you want to take photos that will be sent to a model for inferencing, use *inspect* mode when taking pictures. This mode is available whether you are taking photos manually (handheld devices) or are using *Auto-Capture* mode (fixed devices).

The photo and inference result are then uploaded to a IBM Maximo Visual Inspection data set (specified in the Inspection) and are also displayed in a Maximo Visual Inspection Mobile dashboard. See [“Dashboard view” on page 150](#) for details.


Demo mode

You can turn on Demo mode from the Global Settings page. The app runs locally using preloaded CoreML models and all other settings are hidden. There is an overlay that indicates to the user that it contains only demonstration data.

Tap the + icon to attempt to create a new inspection. This opens a form that allows a user to request access to a IBM Maximo Visual Inspection server. The requester will be contacted if access has been provisioned.

Handheld mode

If the device is not in a fixed position, it should be used in Handheld mode. When in Handheld mode, you can take photos manually.

To enable Handheld Mode, go to **Global Settings:** 

To take a manual photo, from the Inspection home screen, tap the **Capture** button. The app defaults to *collect mode* if you do not specify a model within an inspection. When you specify a model within an inspection, the app enters *inspect mode*.

The results are displayed in the text box at the bottom of the screen. If it says "No Results" then the model returned no results or the calculated accuracy was below the value set for "Ignore Results Below" in Threshold Configuration. The resulting photo can be shared using iOS share sheet.

- When an inference is performed on an object detection model, results with bounding boxes are stamped on the photo.
- When Maximo Visual Inspection Mobile performs an inference, it fails or passes inspection and the results are sent via MQTT or socket, text (to the number configured for the Supervisor), or both. You must turn off handheld mode before taking a picture if you want the results sent by MQTT or socket. This enables outbound MQTT or TCP/IP socket based communication.

Related concepts

[Fixed \(mounted\) devices for Auto-Capture mode](#)

Using Maximo Visual Inspection Mobile on fixed devices requires the device to be in Single App mode and also requires a connection to an MQTT broker or a TCP/IP based socket server.

Fixed (mounted) devices for Auto-Capture mode

Using Maximo Visual Inspection Mobile on fixed devices requires the device to be in Single App mode and also requires a connection to an MQTT broker or a TCP/IP based socket server.

You can capture photos manually or in *Auto-Capture* mode. The app defaults to *collect mode* if you do not specify a model within an inspection. When you specify a model within an inspection, the app enters *inspect mode*. However, you can still take photos in *inspect mode* by tapping **Capture**.

Note: *Auto-Capture* mode is available only when **Handheld Mode** is turned off in **Global Settings**. See [“Setting up ” on page 144](#).

A single device can automatically switch between inspections based on external triggers when in *Auto-Capture* mode. The device is locked to Maximo Visual Inspection Mobile so the app always remains in the foreground, even though the screen turns off automatically after 30 seconds. Additionally, if the device runs out of power, the app crashes, or the app is terminated by iOS, when it is launched (which will happen automatically if Single App Mode is enabled on the device) the app will immediately go back into the previously engaged mode.

If you use a zoom setting via pinching, it will be maintained in *Auto-Capture* mode.

Accessing a remote device

To access a remote device that is connected on the same MQTT broker, tap the ellipsis icon and tap *Change to Another Device*. It can take up to 10 seconds for all available devices to appear. On the remote device, you can do the following:

- Create, edit, and delete inspections.
- Modify app settings (not Global settings).
- Engage Collect or Inspect modes for *Auto-Capture*. Camera preview is not available.

You cannot configure global settings or the dashboard for a remote device.

Note: While a remote device is downloading CoreML assets, Maximo Visual Inspection Mobile cannot be engaged remotely.

Related concepts

Handheld mode

If the device is not in a fixed position, it should be used in Handheld mode. When in Handheld mode, you can take photos manually.

Visual Trigger

Visual Trigger is a mode that uses AI to run an inspection.

This mode can detect when an object comes into the camera frame, identifies that object, and runs an inspection. The inspection's trigger string matches the label name.

Visual Trigger does not require an external trigger that uses MQTT, although that type of trigger can work in parallel with Visual Trigger.

When Visual Trigger finds a label in the frame that satisfies the frame timing logic, an internal trigger message is sent within the app; not by using MQTT. Through this trigger message, an attempt is made to identify an inspection that has a trigger string with a matching label. Usually a separate inspection is assessed, but it is possible to use the same inspection. However, if you use the same inspection, make sure that the trigger string on the inspection matches the label that is being searched in the frame.

Visual Trigger has the following requirements:

- Although object detection and classification models are supported, a classification model must be well-trained because its confidence levels can fluctuate.
- The inspection must have at least one label with frame timing configuration. You can optionally assign an obstruction label to prevent obstructions, such as a person that walks into the frame.
- The label name or names of the model in the inspection must be trained to match the trigger string that is defined in the inspection.

Note: When you use Visual Trigger, the maximum image size that is uploaded is 1080 pixels.

Visual Trigger can run in parallel with [“Barcode and OCR Trigger” on page 154](#).

Setting up Visual Trigger

Follow these steps to set up Visual Trigger.

Before you begin

Make sure that **Handheld Mode** is turned off in **Global Settings**. See [“Setting up ” on page 144](#).

Procedure

1. Create an inspection by clicking "+" in the **Inspections** pane.
 2. Specify a **Project**, **Model**, and **Data Set**.
- Note:** The model type can be *Tiny YOLO v2* or *YOLO v3*, although *YOLO v3* consumes high CPU cycles. If you select a *Tiny YOLO v2* model, be sure to set a low **Frame Rate Throttle**, which is set in **Configure Device > App Settings**.
3. In the **Model** screen, tap **Set Thresholds**.
 4. In the **Set Thresholds** screen, select the label that you want to detect while using Visual Trigger, then tap **Advanced**.
 5. In the **Advanced** screen, tap **Additional Rules**.
 6. Select **Inspect When Label Detected**.

Note: When this option is selected, Auto-Capture mode opens in streaming video mode, at the frames per second that you define in **Configure Device > App Settings > Frame Rate Throttle**.

7. Tap back to return to the **Advanced** screen.
 8. Tap **Frames to Consider** and set the rolling consideration window for Visual Trigger.
 9. Tap **Frames Label Must Be Present** and set the number of frames within the consideration window where the label must be present. This value represents label's specified confidence level. The inspection runs when this threshold is met.
- Note:** This value is affected by the **Frame Rate Throttle** setting, in **Configure Device > App Settings**. If you change **Frame Rate Throttle**, you might need to readjust the value in this step.
10. Tap **Frames Without Label To Reset** and set the number of frames where the label must not be present after the inspection is run before the inspection is run again.

Note: This value is affected by the **Frame Rate Throttle** setting, in **Configure Device > App Settings**. If you change **Frame Rate Throttle**, you might need to readjust the value in this step.

11. Optional: For another label in the same inspection, you can specify that label as an **Obstruction for Label Detection** from the **Additional Rules** screen.
12. Tap back until you are back at the main Edit screen, then tap **Done**.
13. Tap the icon to open the **Configure Device** screen.
14. Tap **Auto Capture**.
15. In the **Visual Trigger** category, tap **Select Inspection** and select the inspection to use for Visual Trigger.
16. Optional: If you toggle **Pass video frame to inspection** on, the app uses the frame for auto capture mode as the image when triggering the inspection, instead of taking a new photo.
17. Tap **Save** to save your Auto-Capture settings.
18. Tap **Close** to return to the home screen.



19. From the home screen, tap the camera icon.
20. Tap the **Start Auto-Capture** button.

What to do next

If the **Auto-Capture** Settings are specified an inspection, the app automatically uses video streaming mode and lists the objects that are detected by Visual Trigger. The app also lists several states:

- "Waiting" for an object.
- "Resetting" if it is waiting an object to leave the frame.
- "Obstructed" if an obstruction label was specified and an object is fully obstructed.

Barcode and OCR Trigger

IBM Maximo Visual Inspection Mobile can read barcodes or use optical character recognition (OCR) for use cases such as reading serial numbers.

A couple of actions can occur after the barcode or text is read:

- The barcode or text value can be sent over MQTT.
- The value can then be compared to a rule that has a regular expression. If the barcode or text matches the regular expression, an inspection can be run by using the associated rule in the trigger string.

The Barcode and OCR reader supports multiple rules. In other words, triggers can be sent to the same local device, or to a secondary device by sending a trigger message over MQTT.

Note: When you use the Barcode and OCR reader, the maximum image size that is uploaded is 1080 pixels.

The following barcode types are supported:

- Aztec code recognition
- Code 39 barcode recognition
- Code 39 with checksum
- ASCII Code 39
- ASCII Code 39 with checksum
- Code 93 barcode recognition
- Code 93i barcode recognition
- Code 128 barcode recognition
- Data Matrix barcode recognition
- EAN-8 barcode recognition
- EAN-13 barcode recognition
- Interleaved 2 of 5 (ITF) barcode recognition
- Interleaved 2 of 5 (ITF) with checksum
- ITF-14 barcode recognition
- PDF417 barcode recognition
- QR code recognition
- UPC-E barcode recognition

The following OCR formats are supported:

- Fast
- Accurate

Barcode and OCR Trigger can run in parallel with [“Visual Trigger” on page 152](#).

Setting up Barcode or OCR Trigger


Follow these steps to set up barcode or OCR reading or triggering.

Before you begin

- Make sure that **Handheld Mode** is turned off in **Global Settings**. See [“Setting up ” on page 144](#).

- If you are using the barcode or OCR reader to trigger an inspection, decide whether you intend to run the inspection on the local device or a secondary device.

Procedure

1. From the home screen, click the icon that opens **Configure Device**.
2. Tap **Auto-Capture**.
3. If you are reading codes to trigger an inspection on a secondary device, toggle **Send Trigger to Secondary Device** on. Toggling this option displays more fields:
 - a) Set the **Secondary Device Location**.
 - b) Set the **Secondary Device Name**.
 - c) Optional: If you want to delay the image trigger, input the number of seconds in **Image Trigger Delay(s)**.
4. **For OCR only:** In the **BARCODE/OCR TRIGGER** section, change **Mode** to *Accurate*.
5. Tap **Add Rule**. The **Add Rule** screen opens.
6. In the **Add Rule** screen:
 - a) Tap **Type** and select *OCR* or *Barcode*, depending in your use case.
 - b) Tap **Regular Expression** type the regular expression that matches the value that is read. Examples:
 - To match any value: (.*?)
 - To match a vehicle's VIN number: [A-HJ-NPR-Z0-9]{17}
 - c) Optional: If you are using the barcode or OCR value to trigger an inspection, tap **Trigger** and type the trigger string for the inspection you want to run. If left blank, results are sent by MQTT to `ibmvi/barcodeocrvalue`.
 - d) Tap **Save**.
7. To add more rules, repeat the previous two steps.
8. From the home screen, tap the camera icon. 
9. Tap the **Start Auto-Capture** button.

Configuring MQTT

Configure MQTT to allow external systems to interact with Maximo Visual Inspection Mobile.

- [“Topic structure” on page 155](#)
- [“Topics in use” on page 156](#)
- [“Inbound trigger” on page 156](#)
- [“Inbound example” on page 157](#)
- [“Outbound notification” on page 157](#)
- [“Outbound examples” on page 159](#)
- [“Outbound error” on page 160](#)
- [“Configurable notification output” on page 161](#)
- [“Configurable notification output example” on page 161](#)

Topic structure

All topics include the location and device name that is configured within each iOS app instance. Refer to the MQTT Specification for full details.

Notes:

- Do not use a leading forward slash or ending slash, in accordance with MQTT best practices.

- Additional JSON attributes may be included in subsequent releases. Any consuming systems must code defensively to ensure they do not error if additional elements or attributes are present in the JSON.

Topics in use

The following are the topics that Maximo Visual Inspection Mobile leverages. All topics include the location and device name which is configured within each iOS app instance. While the full list of topics are listed for informational purposes, this information focuses only on the inbound trigger message, the outbound notification message and outbound error message, which are the three messages that external systems need to work with in order to interface with the app.

ibmvi/heartbeat/<location>/<device>

Publishes the heartbeat information of all devices connected to the same MQTT broker so that they can be monitored remotely.

ibmvi/error/<location>/<device>

Signals any errors in iOS app instances running in Inspection mode, such as failed inferences.

ibmvi/threshold/<location>/<device>

Publishes the inference results.

ibmvi/systemrequest/<location>/<device>

Sends device configuration updates.

ibmvi/uploadresult/dataset/<datasetID>/<location>/<device>

Refreshes the dashboard when a new image is captured and uploaded to IBM Maximo Visual Inspection server.

ibmvi/takephoto/<location>/<device>

Sends commands to the app to take photos.

ibmvi/notificationtemplate

This topic is used to publish custom templates that are used for output by Maximo Visual Inspection Mobile.

ibmvi/barcodeocrvalue/<location>/<device>

This topic is used to publish the results of reading a barcode or OCR value after the regular expression is evaluated.

Inbound trigger

This MQTT message is used to trigger Maximo Visual Inspection Mobile to take a photo. It requires the message to be sent to a particular location and device using the topic structure:

```
ibmvi/takephoto/<location>/<device>
```

Note: The trigger field must match a defined inspection on the device and the device must be in *Auto-Capture* mode. It can be used for collections or inspections.

YAML Specification

```
---
required:
  - "date"
  - "reference"
  - "trigger"
properties:
  date:
    type: "string"
    example: "20190925232952"
    description: "Date the trigger request was sent in YYYYMMDDHHMMSS format"
  reference:
    type: "string"
    example: "VIN12345"
    description: "External reference used to invoke the inspection on the device"
  trigger:
    type: "string"
    example: "TELE"
```

```
description: "Value used to route the trigger request to a specific Inspection on the device, this must match the Trigger String set in an Inspection"
```

Inbound example

```
{
  "date": "20190925232952",
  "reference": "VIN12345",
  "trigger": "TELE"
}
```

Outbound notification

This is used to send the results of inference via this topic:

```
ibmvi/threshold/<location>/<device>
```

The most important data element is `thresholdMsg`, which contains the human readable content of inferencing. This should be included in the body of any email or text message. Examples:

```
<Label> at <Confidence %> is <Above or Below> specified confidence of <threshold value>
<Label> with a count of <count> is <above or below> specified count of <count value>
```

Further, under the `deviceInfo` element, the `locationName` and `deviceNames` are important to identify the device that produces the outbound notification.

YAML specification

```
---
required:
  - "isPass"
  - "modelType"
  - "scoresAndThresholds"
  - "reference"
  - "datasetId"
  - "thresholdMsg"
  - "fileId"
  - "imageUrl"
  - "deepLink"
  - "deviceInfo"
properties:
  isPass:
    type: "boolean"
    example: false
    description: "Indicates whether this inference was a pass"
  modelType:
    type: "string"
    example: "ObjectDetection"
    description: "Indicates the type of model, either Classification or ObjectDetection"
  scoresAndThresholds:
    type: "array"
    items:
      type: "object"
      properties:
        isBelow:
          type: "boolean"
          example: false
          description: "If true, indicates if the score returned by PAIV should be compared below the specified threshold to indicate a fail. The default (false) is to always check that the returned score is above the threshold."
        score:
          type: "string"
          example: "6.89"
          description: "The confidence score for the label (name) returned by PAIV"
        bbData:
          required:
            - "xmin"
            - "xmax"
            - "ymin"
            - "ymax"
          properties:
            xmin:
              type: "number"
              example: 0
```

```

        description: "Xmin coordinate for bounding box"
      xmax:
        type: "number"
        example: 999
        description: "Xmax coordinate for bounding box"
      ymin:
        type: "number"
        example: 256
        description: "Ymin coordinate for bounding box"
      ymax:
        type: "number"
        example: 810
        description: "Ymax coordinate for bounding box"
    type: "object"
  name:
    type: "string"
    example: "arduino_mega"
    description: "Name of label from PAIV"
  threshold:
    type: "string"
    example: "0.0"
    description: "Configured threshold value from the app"
  reference:
    type: "string"
    example: "MANUALREFDATA"
    description: "The external reference specified by the MQTT trigger message. Will read MANUALREFDATA if photo was taken manually by the iOS app."
  datasetId:
    type: "string"
    example: "1b48624c-1cb0-43f1-9f1c-5f03cb40d5af"
    description: "UUID of the PAIV Dataset the image was uploaded to"
  thresholdMsg:
    type: "string"
    example: "arduino_uno at 99.73% is Above specified confidence of 10.0%"
    description: "Human readable result message intended for emails or text messages"
  fileId:
    type: "string"
    example: "476706a9-8c9e-417b-bffa-515b698edf4b"
    description: "UUID of the file uploaded to PAIV"
  imageURL:
    type: "string"
    example: "https://vision-poc1.aus.stglabs.ibm.com/visual-inspection-v130/uploads/coreml/datasets/1b48624c-1cb0-43f1-9f1c-5f03cb40d5af/files/476706a9-8c9e-417b-bffa-515b698edf4b.jpg"
    description: "Link to IBM Maximo Visual Inspection to view the image that was uploaded to the data set"
  deeplink:
    type: "string"
    example: "ibmvisualinspector://openImage?datasetId=1b48624c-1cb0-43f1-9f1c-5f03cb40d5af&fileId=476706a9-8c9e-417b-bffa-515b698edf4b"
    description: "Deep link to the image in the dashboard in the IBM Maximo Visual Inspection Edge app using iOS URL schema"
  deviceInfo:
    required:
      - "mqttClientId"
      - "locationName"
      - "engagedMode"
      - "lastPhotoTaken"
      - "deviceName"
    properties:
      mqttClientId:
        type: "string"
        example: "658B163D-0C9D-47A9-9169-3B6ECA157788"
        description: "MQTT client ID of the iOS device sending the message"
      locationName:
        type: "string"
        example: "loc1"
        description: "Location name specified in the IBM Maximo Visual Inspection Edge app"
      engagedMode:
        type: "string"
        example: "RUNTIME"
        description: "Whether the device was in TRAINING or RUNTIME mode. Inference results are included only in RUNTIME mode."
      lastPhotoTaken:
        type: "number"
        example: 593964309.603459
        description: "Time the last photo was taken on device, expressed as the interval in seconds since 00:00:00 UTC on 1 January 2001, expressed as a float"
      deviceName:
        type: "string"
        example: "dev123"

```

```
description: "Device name specified in the IBM Maximo Visual Inspection Edge app"
type: "object"
```

Outbound examples

Object detection result example:

```
{
  "isPass": false,
  "modelType": "ObjectDetection",
  "scoresAndThresholds": [{
    "isBelow": false,
    "score": "6.89",
    "bbData": {
      "xmin": 0,
      "xmax": 999,
      "ymin": 256,
      "ymax": 810
    },
    "name": "arduino_mega",
    "threshold": "0.0"
  }, {
    "isBelow": false,
    "score": "99.73",
    "bbData": {
      "xmin": 0,
      "xmax": 950,
      "ymin": 235,
      "ymax": 789
    },
    "name": "arduino_uno",
    "threshold": "10"
  }],
  "reference": "MANUALREFDATA",
  "datasetId": "1b48624c-1cb0-43f1-9f1c-5f03cb40d5af",
  "thresholdMsg": "arduino_uno at 99.73% is Above specified confidence of 10.0%",
  "fileId": "476706a9-8c9e-417b-bffa-515b698edf4b",
  "imageUrl": "https://vision-poc1.aus.stglabs.ibm.com/powerai-vision-v115-daily/uploads/coreml/datasets/1b48624c-1cb0-43f1-9f1c-5f03cb40d5af/files/476706a9-8c9e-417b-bffa-515b698edf4b.jpg",
  "deepLink": "ibmvisualinspector://openImage?datasetId=1b48624c-1cb0-43f1-9f1c-5f03cb40d5af&fileId=476706a9-8c9e-417b-bffa-515b698edf4b",
  "deviceInfo": {
    "mqttClientId": "658B163D-0C9D-47A9-9169-3B6ECA157788",
    "locationName": "loc1",
    "engagedMode": "RUNTIME",
    "lastPhotoTaken": 593964309.603459,
    "deviceName": "dev123"
  }
}
```

Object detection example with a result based on the expected count:

```
{
  "isPass": false,
  "modelType": "ObjectDetection",
  "scoresAndThresholds": [{
    "isBelow": false,
    "score": "2.74",
    "bbData": {
      "xmin": 25,
      "xmax": 770,
      "ymin": 229,
      "ymax": 801
    },
    "name": "arduino_mega",
    "threshold": "0.0"
  }, {
    "expectedCount": 2,
    "isBelow": false,
    "score": "99.77",
    "bbData": {
      "xmin": 0,
      "xmax": 999,
      "ymin": 221,
      "ymax": 761
    },
    "name": "arduino_uno",
  }]
```

```

    "threshold": "10"
  },
  "reference": "MANUALREFDATA",
  "datasetId": "1b48624c-1cb0-43f1-9f1c-5f03cb40d5af",
  "thresholdMsg": "arduino_uno with a count of 1 is not matching specified count of 2",
  "fileId": "fba1a41b-5ac5-4d5f-aba7-599aea726b5a",
  "imageUrl": "https://vision-poc1.aus.stglabs.ibm.com/powerai-vision-v115-daily/uploads/coreml/datasets/1b48624c-1cb0-43f1-9f1c-5f03cb40d5af/files/fba1a41b-5ac5-4d5f-aba7-599aea726b5a.jpg",
  "deepLink": "ibmvisualinspector://openImage?datasetId=1b48624c-1cb0-43f1-9f1c-5f03cb40d5af&fileId=fba1a41b-5ac5-4d5f-aba7-599aea726b5a",
  "deviceInfo": {
    "mqttClientId": "658B163D-0C9D-47A9-9169-3B6ECA157788",
    "locationName": "loc1",
    "engagedMode": "RUNTIME",
    "lastPhotoTaken": 593964572.57300794,
    "deviceName": "dev123"
  }
}

```

Classification model example:

```

{
  "isPass": false,
  "modelType": "Classification",
  "scoresAndThresholds": [{
    "isBelow": true,
    "score": "89.28",
    "name": "ArduinoUno",
    "threshold": "99"
  }],
  "reference": "MANUALREFDATA",
  "datasetId": "bd363f4b-e48c-47c5-9fee-d9904e9a1911",
  "thresholdMsg": "ArduinoUno at 89.28% is Below specified confidence of 99.0%",
  "fileId": "0132d3bd-0b74-41c0-b40b-c7b12d7ecf17",
  "imageUrl": "https://vision-poc1.aus.stglabs.ibm.com/powerai-vision-v115-daily/uploads/coreml/datasets/bd363f4b-e48c-47c5-9fee-d9904e9a1911/files/0132d3bd-0b74-41c0-b40b-c7b12d7ecf17.jpg",
  "deepLink": "ibmvisualinspector://openImage?datasetId=bd363f4b-e48c-47c5-9fee-d9904e9a1911&fileId=0132d3bd-0b74-41c0-b40b-c7b12d7ecf17",
  "deviceInfo": {
    "mqttClientId": "658B163D-0C9D-47A9-9169-3B6ECA157788",
    "locationName": "loc1",
    "engagedMode": "RUNTIME",
    "lastPhotoTaken": 593985750.97822499,
    "deviceName": "dev123"
  }
}

```

Outbound error

This is the error message when inference fails on a device due to misconfiguration or server or device error. These are published via:

```
ibmvi/error/<location>/<device>
```

YAML Specification

```

---
required:
  - "msg"
  - "deviceInfo"
  - "time"
properties:
  msg:
    type: "string"
    example: "Device: dev123, Location: loc1 Unable to update token while making the request https://vision-poc1.aus.stglabs.ibm.com/powerai-vision-v115-daily/api/projects, ErrorCode: -1003, ErrorDesc: Optional(Error Domain=NSURLErrorDomain Code=-1003 \"A server with the specified hostname could not be found.\" UserInfo={NSUnderlyingError=0x2814bb8a0 {Error Domain=kCFErrorDomainCFNetwork Code=-1003 \"(null)\" UserInfo={_kCFStreamErrorCodeKey=8, _kCFStreamErrorDomainKey=12}}, NSErrorFailingURLStringKey=https://vision-poc1.aus.stglabs.ibm.com/powerai-vision-v115-daily/api/tokens, NSErrorFailingURLKey=https://vision-poc1.aus.stglabs.ibm.com/powerai-vision-v115-daily/api/tokens, _kCFStreamErrorDomainKey=12, _kCFStreamErrorCodeKey=8, NSLocalizedDescription=A server with the specified hostname could not be found.})"
  deviceInfo:

```



```

required:
  - "mqttClientId"
  - "locationName"
  - "lastPhotoTaken"
  - "deviceName"
properties:
  mqttClientId:
    type: "string"
    example: "658B163D-0C9D-47A9-9169-3B6ECA157788"
    description: "MQTT client ID of the iOS device sending the message"
  locationName:
    type: "string"
    example: "loc1"
    description: "Location name specified in the IBM Maximo Visual Inspection Mobile app"
  lastPhotoTaken:
    type: "number"
    example: 593974708.779805
    description: "Time last photo taken on the device, expressed as the interval in seconds
since 00:00:00 UTC on 1 January 2001, expressed as a float"
  deviceName:
    type: "string"
    example: "dev123"
    description: "Device name specified in the IBM Maximo Visual Inspection Mobile app"
type: "object"
time:
  type: "number"
  example: 593985487.698025
  description: "Error timestamp, to be precise the interval in seconds since 00:00:00 UTC on
1 January 2001, expressed as a float"

```

Example JSON

```

{
  "msg": "Device: dev123, Location: loc1 Unable to update token while making the request
https://vision-poc1.aus.stglabs.ibm.com/powerai-vision-v115-daily/api/projects, ErrorCode:
-1003, ErrorDesc: Optional(Error Domain=NSURLErrorDomain Code=-1003 \"A server with the
specified hostname could not be found.\" UserInfo={NSUnderlyingError=0x2814bb8a0 {Error
Domain=kCFErrorDomainCFNetwork Code=-1003 \"(null)\" UserInfo={_kCFStreamErrorCodeKey=8,
_kCFStreamErrorDomainKey=12}}, NSErrorFailingURLStringKey=https://vision-
poc1.aus.stglabs.ibm.com/powerai-vision-v115-daily/api/tokens,
NSErrorFailingURLKey=https://vision-poc1.aus.stglabs.ibm.com/powerai-vision-v115-daily/api/
tokens, _kCFStreamErrorDomainKey=12, _kCFStreamErrorCodeKey=8, NSLocalizedDescription=A server
with the specified hostname could not be found.})",
  "deviceInfo": {
    "mqttClientId": "658B163D-0C9D-47A9-9169-3B6ECA157788",
    "locationName": "loc1",
    "lastPhotoTaken": 593974708.77980494,
    "deviceName": "dev123"
  },
  "time": 593985487.69802499
}

```

Configurable notification output

This is used to send the template as an MQTT message in JSON format to the topic `ibmvi/notificationtemplate`.

Make sure that the retained flag is set on the MQTT message. This flag ensures that any new devices that come online and subscribe to the topic will be sent the existing template.

Update the template by using the same technique.

To clear the template and to revert the default template, send an empty MQTT message to the `ibmvi/notificationtemplate` topic. Depending on the MQTT client, you might need to include a space to publish it. You also need to delete the topic with the retained message, to prevent the app from pulling down the retained message the next time the app restarts.

Configurable notification output example

This example is a template for the ISA-95 standard that leverages the "Select Transformation" process (ST.js).

```

{
  "Header": {

```

```

        "MessageTimeStamp": "",
        "MessageType": "PartData",
        "SchemaMajor": 1,
        "SchemaMinor": 3,
        "Plant": "",
        "Area": "",
        "Department": "{{deviceInfo.locationName}}",
        "Line": "",
        "Cell": "",
        "Machine": "",
        "Station": ""
    },
    "PartData": {
        "DataSource": "{{deviceInfo.deviceName}}",
        "CycleStartTimeStamp": "",
        "UnitID": "{{reference}}",
        "Task": [
            {
                "Label": "",
                "Description": "",
                "Image": [
                    {
                        "Label": "",
                        "Description": "",
                        "Format": "jpg",
                        "Status": "{{isPass == true ? 'Pass' : 'Reject'}}",
                        "ImagePath": "{{imageUrl}}",
                        "Image": "IMAGECONVERTEDTOBASE64STRING",
                        "Feature": {
                            "{{#each scoresAndThresholds}}": {
                                "Label": "{{this.name}}",
                                "Description": "",
                                "Status": "",
                                "ResultValue": "{{this.score}}",
                                "Units": "%",
                                "UpperLimit": "{{this.threshold}}",
                                "Xmin": "{{this.bbData.xmin}}",
                                "Xmax": "{{this.bbData.xmax}}",
                                "Ymin": "{{this.bbData.ymin}}",
                                "Ymax": "{{this.bbData.ymax}}"
                            }
                        }
                    }
                ]
            }
        ]
    }
}

```

The following is sample output after applying the template.

```

{
  "Header": {
    "MessageTimeStamp": "",
    "MessageType": "PartData",
    "SchemaMajor": 1,
    "SchemaMinor": 3,
    "Plant": "",
    "Area": "",
    "Department": "loc1",
    "Line": "",
    "Cell": "",
    "Machine": "",
    "Station": ""
  },
  "PartData": {
    "DataSource": "dev123",
    "CycleStartTimeStamp": "",
    "UnitID": "MANUALREFDATA123",
    "Task": [
      {
        "Label": "",
        "Description": "",
        "Image": [
          {
            "Label": "",
            "Description": "",
            "Format": "jpg",
            "Status": "Pass",
            "ImagePath": "https://vision-poc1.aus.stglabs.ibm.com/powerai-vision-v115-daily/uploads/coreml/datasets/1b48624c-1cb0-43f1-9f1c-5f03cb40d5af/files/476706a9-8c9e-417b-bffa-515b698edf4b.jpg",
            "Image": "IMAGECONVERTEDTOBASE64STRING",

```

```
"Feature": [
  {
    "Label": "arduino_mega",
    "Description": "",
    "Status": "",
    "ResultValue": "6.89",
    "Units": "%",
    "UpperLimit": "0.0",
    "Xmin": 0,
    "Xmax": 999,
    "Ymin": 256,
    "Ymax": 810
  },
  {
    "Label": "arduino_uno",
    "Description": "",
    "Status": "",
    "ResultValue": "99.73",
    "Units": "%",
    "UpperLimit": "10",
    "Xmin": 0,
    "Xmax": 950,
    "Ymin": 235,
    "Ymax": 789
  }
]
}
```

MQTT streaming

MQTT streaming is a video streaming mode that sends MQTT messages without uploading images to IBM Maximo Visual Inspection.

More specifically, MQTT streaming has all of the following characteristics:

- Streams video in Auto-Capture mode.
- Constantly inspects frames against a local CoreML model, checking for one or more labels.
- When labels are encountered, an MQTT message that contains the results is sent. The MQTT message is sent whether the results are pass or fail.
- Related images are not uploaded to IBM Maximo Visual Inspection.

Not uploading images to IBM Maximo Visual Inspection during video streaming is a key characteristic of MQTT streaming. Otherwise, this network activity would overwhelm the app, the network, and the IBM Maximo Visual Inspection server.

A typical use case for MQTT streaming is to facilitate additional logic of the streamed images. For example, a downstream software component can subscribe to the MQTT messages and run trigonometry calculations. These calculations can to determine where objects are in the frame, the distance between, running counts of objects, and so on.

Note: If a custom MQTT template is used with streaming, the device might overheat due to thermal load as CPU use increases.

Setting up MQTT streaming

Follow these steps to set up MQTT streaming.

Before you begin

Make sure that **Handheld Mode** is turned off in **Global Settings**. See “Setting up ” on page 144.

Procedure

1. Create an inspection by clicking "+" in the **Inspections** pane.

2. Specify a **Project**, **Model**, and **Data Set**.

Note: The model type must be *CoreML*. The model that you select must include the labels that you want to monitor.

3. In the **Model** screen, tap **Set Thresholds**.
4. In the **Set Thresholds** screen, tap **Advanced** for each of the labels that you want to monitor in the scene.
5. In the **Advanced** screen, tap **Additional Rules**.
6. Select **Inspect When Label Detected**. The app returns to the **Advanced** screen.
7. In the **Advanced** screen, set **Frames To Consider** to 1. This value means that every frame is checked.
8. In the **Advanced** screen, set **Frames Label Must Be Present** to 1. This value forces an inspection on each frame.
9. In the **Advanced** screen, set **Frames Without Label To Reset** to 0. This value forces a reset after an inspection is done on each frame.

Note: You can set the frames per second rate in **Configure Device > App Settings > Frame Rate Throttle**.

10. Repeat steps “4” on page 164 to “9” on page 164 for each label in the model.

Important: Avoid using a threshold whose **Additional Rules** is set to "Obstruction for Label Detection". This setting can cause the video stream to pause.

11. Tap back until you are back at the main Edit screen, then tap **Done**.
12. Tap the icon to open the **Configure Device** screen.
13. Tap **App Settings** and set the following values:

- Toggle **Upload to IBM Maximo Visual Inspection** off.

Important: Make sure that this option is not on. Otherwise, every image is uploaded to the server, which overloads the app, the network, and the IBM Maximo Visual Inspection server.

- Toggle **Upload Failures Only** off.
- Toggle **Always On Camera** on.
- Toggle **Notify on Inspection Pass** on.

14. Tap **Save** to return to the **Configure Device** screen.

15. From the **Configure Device** screen, tap **Auto-Capture**, then set the following values:

- Toggle **Pass Video Frame To Inspection** on.
- In the **Visual Trigger** category, tap **Select Inspection** and select the inspection to use for MQTT streaming.

16. Tap **Save** to save your Auto-Capture settings.

17. Tap **Close** to return to the home screen.



18. From the home screen, tap the camera icon.

19. Tap the **Start Auto-Capture** button.

Troubleshooting

If something is not working right with Maximo Visual Inspection Mobile, there are several places to look for errors, depending on where they occurred. It is recommended that you enable the ability to share crash logs with developers.

- [“Finding error messages” on page 165](#)
- [“Crash logs” on page 165](#)

Finding error messages

- Application errors, such as being unable to reach the IBM Maximo Visual Inspection server, are displayed on the Maximo Visual Inspection Mobile user interface. They are also posted to an MQTT topic with a payload similar to this JSON:

```
ibmvi/error/<location>/<device>

{
  "msg": "Device: my phone, Location: office Unable to update token while making the request",
  "deviceInfo": {
    "mqttClientID": "99C9827-FC33-46AB-917A-1081510151D",
    "locationName": "office",
    "lastPhotoTaken": "595289659.985447",
    "deviceName": "my phone"
  },
  "time": "5936155.207686"
}
```

- Any API invocation that fails, for example, the IBM Maximo Visual Inspection server is misconfigured or invalid login credentials will be posted on the Maximo Visual Inspection Mobile user interface and on the MQTT error queue.
- Errors that occur during *Auto-Capture* mode are only posted on the MQTT error queue. Therefore, observing the error queue is particularly important during *Auto-Capture* mode, where the device screen is unlikely to be viewed by a user.

Crash logs

Support can use AppStore Connect Crash Logs, if the option is enabled. These reports are anonymous and do not link to individual devices. To leverage this capability, from the Privacy page, open the Analytics page and enable "Share With App Developers".

Chapter 16. Troubleshooting and contacting support

To isolate and resolve problems with your IBM products, you can use the following troubleshooting and support information. This information contains instructions for using the problem-determination resources that are provided with your IBM products, including IBM Maximo Visual Inspection.

Troubleshooting known issues - IBM Maximo Visual Inspection standard install

Following are some problems you might encounter when using IBM Maximo Visual Inspection, along with steps to fix them.

- [“Action detection training fails, video inference does not process full video, or auto-capture does not capture frames in full video” on page 168](#)
- [“Action detection training fails some instances with error Internal Server Error - Generic exception thrown” on page 168](#)
- [“When importing a DICOM format file, the Waiting for import... notification does not go away” on page 169](#)
- [“IBM Maximo Visual Inspection seems to be connected to IBM Watson IoT Platform , but no data is showing up” on page 169](#)
- [“The IBM Maximo Visual Inspection user interface does not work” on page 170](#)
- [“Resource pages are not being populated in the user interface” on page 170](#)
- [“Unexpected / old pages displayed when accessing the user interface” on page 170](#)
- [“IBM Maximo Visual Inspection does not play video” on page 170](#)
- [“IBM Maximo Visual Inspection cannot train or deploy models after reboot” on page 171](#)
- [“A Tiny YOLO V2 model fails to train with a small data set” on page 171](#)
- [“A Tiny YOLO v2 model fails to train using a data set with many similar bounding boxes” on page 171](#)
- [“Tiny YOLO v2 models do not train, but other models train” on page 171](#)
- [“Changing the port for the IBM Maximo Visual Inspection user interface” on page 172](#)
- [“Out of space error from load_images.sh” on page 172](#)
- [“GPUs are not available for training or inference” on page 173](#)
- [“I forgot my user name or password” on page 173](#)
- [“IBM Maximo Visual Inspection cannot train a model” on page 174](#)
- [“Thumbnails do not load or images previews are missing” on page 175](#)
- [“Training or deployment hangs - Kubernetes pod cleanup” on page 175](#)
- [“Training fails with error indicating You must retrain the model.” on page 175](#)
- [“Training or deployment of models fails - sometimes inconsistently” on page 176](#)
- [“Model import generates an error alert” on page 176](#)
- [“Model training and inference fails” on page 175](#)
- [“Model accuracy value is unexpected” on page 176](#)
- [“Deployed models stuck in Starting” on page 176](#)
- [“Auto labeling of a data set returns Auto Label Error” on page 177](#)
- [“IBM Maximo Visual Inspection does not start” on page 177](#)
- [“IBM Maximo Visual Inspection does not start with non-default Docker root directory” on page 178](#)
- [“IBM Maximo Visual Inspection fails on first startup and Kubernetes services do not start” on page 177](#)
- [“IBM Maximo Visual Inspection fails to start - Kubernetes connection issue” on page 179](#)

- [“IBM Maximo Visual Inspection startup hangs - helm issue” on page 180](#)
- [“Helm status errors when starting IBM Maximo Visual Inspection” on page 181](#)
- [“Uploading a large file fails” on page 182](#)
- [“Some IBM Maximo Visual Inspection functions don't work” on page 182](#)

Action detection training fails, video inference does not process full video, or auto-capture does not capture frames in full video

Problem

Operations on a video (auto-capture, frame capture, or video inference) do not process the full video, or training of an action detection model fails.

IBM Maximo Visual Inspection uses video processing utilities to read frames from the videos and some videos cannot be fully processed.

If an action detection training failed, the **video-service** log can be checked for specific errors that indicate this failure. For example, the following command shows the ERROR for a failed training because only 613 of the 741 frames in the video could be read:

```
# kubectl logs `kubectl get pods -o custom-columns=NAME:.metadata.name | grep vision-service`
| grep -A2 -C2 ERROR
root      : INFO      processing 000500/000741 ...
root      : INFO      processing 000600/000741 ...
root      : ERROR      Could not read frame 614.
root      : INFO      Extract video as RGB frame is completed 614
root      : INFO      complete extracting video /opt/ibm/vision/data/admin/datasets/
fd1d7222-2800-4585-b711-000120592811/training/fc575915-5be5-42c8-8d8b-125d5c7a85e2/96c9b3d2-
da24-4e1d-b624-d8ce3141be97.mp4
```

Solution

Try one of these options to solve this problem:

- **Recreate the video** - Use a video processing tool to recreate the video in a standardized format. Such tools can regenerate the video using commonly used and supported video codecs.
- **Avoid labeling past failing frame** - After identifying the problematic point in the video by using product logs or attempting to capture at different frames in the video, ensure that there are no labeled actions that start or end past the problematic point in the video.

Action detection training fails some instances with error "Internal Server Error - Generic exception thrown"

Problem

Multiple action detection labels are selected and renamed. When training is attempted, it fails with Generic exception thrown.

Solution

Rename each label individually, then train the model again.

Postgres Kubernetes pod fails to start

Problem

The IBM Maximo Visual Inspection application does not start, and the Kubernetes node status indicate the postgres pod is in CrashLoopBackOff state. For example:

vision-elasticsearch-59b9b89b56-8h9bt	1/1	Running	0	2m41s
vision-fpga-device-plugin-q9p7b	1/1	Running	0	2m41s
vision-keycloak-98d6cf9db-jfvgl	0/1	Init:0/1	0	2m41s
vision-logstash-7778f58977-b2bqg	1/1	Running	0	2m41s
vision-mongodb-5c9956d784-ws8h4	1/1	Running	0	2m41s
vision-postgres-769698d5c4-j5wtm	0/1	CrashLoopBackOff	4	2m41s
vision-service-6c48b5688b-lmcs2	1/1	Running	0	2m41s
vision-taskanaly-6c8bbb9868-t4xxr	1/1	Running	0	2m41s
vision-ui-589dbd466-sk9tk	1/1	Running	0	2m41s
vision-video-microservice-5678fbdcbb-kfn85	1/1	Running	0	2m41s

Solution

The problem may be related to system configuration that prevents the postgres pod from running successfully. When the log is captured successfully for the pod it shows:

```
./kubectl logs vision-postgres-769698d5c4-sbpm2 -p
...
fixing permissions on existing directory /var/lib/postgresql/data ... ok
creating subdirectories ... ok
selecting default max_connections ... 10
selecting default shared_buffers ... 400kB
selecting dynamic shared memory implementation ... posix
creating configuration files ... ok
Bus error (core dumped)
child process exited with exit code 135
initdb: removing contents of data directory "/var/lib/postgresql/data"
running bootstrap script ...
```

The exit code 135 can occur when huge pages is enabled in the system configuration (reference <https://github.com/docker-library/postgres/issues/451>).

The solution can be to set `vm.nr_hugepages = 0` in `/etc/sysctl.conf` if it was set to non-zero, then reboot the system to have the new configuration take effect.

When importing a DICOM format file, the "Waiting for import..." notification does not go away

Problem

When using the IBM Maximo Visual Inspection user interface on a Windows platform to import a DICOM file, the "Waiting for import..." notification does not automatically close.

Solution

There is no actual impact to the import of the image, and it should be visible in the data set view. The notification can safely be closed or deleted.

IBM Maximo Visual Inspection seems to be connected to IBM Watson IoT Platform , but no data is showing up

After configuration, IBM Watson IoT Platform displays a green dot and a "Connected" to indicate that an IBM Maximo Visual Inspection instance is connected.

Problem

IBM Maximo Visual Inspection appears to be connected to IBM Watson IoT Platform , but no inference data is showing up.

Solution

There could be two solutions to this problem:

- The inference operation produced classified data. If the inference results did not result in any classification data (image classification or object detection), no data is published to IBM Watson IoT Platform .
- The connection was not fully established with IBM Watson IoT Platform . To fix this problem, run the following command:

For a standalone installation:

```
sudo /opt/ibm/vision/bin/kubectl delete pod --selector=run=vision-edge-connector-dp
```

For a cloud installation:

```
kubectl delete pod --selector=run=vision-<namespace>-edge-connector-dp
```

When connection is re-established, any pending inference results are published to IBM Watson IoT Platform . The connection should be fully established and all new inference results are published to IBM Watson IoT Platform .

The IBM Maximo Visual Inspection user interface does not work**Problem**

You cannot label objects, view training charts, or create categories.

Solution

Verify that you are using a supported web browser. The following web browsers are supported:

- Google Chrome Version 60, or later
- Firefox Quantum 59.0, or later

Resource pages are not being populated in the user interface**Problem**

Resource pages, such as data sets and models, are not being populated. Notifications indicate that there is an error obtaining the resource. For example, "Error obtaining data sets."

Solution

Check the status of the `vision-service` pod. This pod provides the data to the user interface, and until it is ready (1/1) with a status of Running, these errors will occur. See [“Checking Kubernetes node status”](#) on page 39 for instructions.

If the application is restarting, there is an expected delay before all services are available and fully functioning. Otherwise, this may indicate an unexpected termination (error) of the `vision-service` pod. If that happens, follow these instructions: [“Gather IBM Maximo Visual Inspection logs and contact support”](#) on page 185.

Unexpected / old pages displayed when accessing the user interface**Problem**

After updating, reinstalling, or restarting IBM Maximo Visual Inspection, the browser presents pages that are from the previous version or are stale.

Solution

This problem is typically caused by the browser using a cached version of the page. To solve the problem, try one of these methods:

- Use a Firefox Private Window to access the user interface.
- Use a Chrome Incognito Window to access the user interface.
- Bypass the browser cache:
 - In most Windows and Linux browsers: Hold down **Ctrl** and press **F5**.
 - In Chrome and Firefox for Mac: Hold down **⌘** **Cmd** and **⇧** **Shift** and press **R**.

IBM Maximo Visual Inspection does not play video**Problem**

You cannot upload a video, or after the video is uploaded the video does not play.

Solution

Verify that your video is a supported type:

- Ogg Vorbis (.ogg)
- VP8 or VP9 (.webm)
- H.264 encoded videos with MP4 format (.mp4)

If your video is not in a supported format, transcode your video by using a conversion utility. Such utilities are available under various free and paid licenses.

IBM Maximo Visual Inspection cannot train or deploy models after reboot

Problem

On RHEL 7.6 systems with CUDA 10.1, the SELinux context of NVIDIA GPU files is lost at boot time. SELinux then prevents IBM Maximo Visual Inspection from using the GPUs for training and deployment.

Solution

Restart IBM Maximo Visual Inspection by running **vision-stop.sh / vision-start.sh**. This resets the problematic SELinux contexts if they are incorrect, restoring the ability to access GPUs for training and inference.

A Tiny YOLO V2 model fails to train with a small data set

Problem

When using a small data set with fewer than 15 labeled images, training a Tiny YOLO v2 model sometimes fails.

The vision-service log shows exceptions because of the ratio of images:

```
# kubectl logs `kubectl get pods -o custom-columns=NAME:.metadata.name | grep vision-  
service` | grep Exception | grep ratio  
root      : INFO      Exception: Based on the ratio of 0.8, the data set was split into 13  
training images and 0 test images. At least one test image is required. Please set a lower  
ratio or add more images to the data set.
```

Solution

Try the training again, or follow the guidance in the message and increase the number of labeled images in the data set. See [“Data set considerations”](#) on page 59 for information.

A Tiny YOLO v2 model fails to train using a data set with many similar bounding boxes

Problem

When using a data set with many images and similar bounding boxes, training a Tiny YOLO v2 model might fail. The model requires unique bounding box anchors to be successfully trained.

The vision-service log shows an error because of the ratio of images:

```
# kubectl logs `kubectl get pods -o custom-columns=NAME:.metadata.name | grep vision-service`  
| grep Error | grep anchor  
root      : INFO      root      : ERROR      Requested to create 5 initial anchors but only  
found 4 unique bounding box sizes in the data set. Please label more objects or make sure  
there are at least 5 uniquely sized boxes.
```

Solution

Modify existing bounding boxes or use data augmentation to create new images with different bounding box anchors, then try the training again. See [“Augmenting the data set”](#) on page 87 for instructions.

Tiny YOLO v2 models do not train, but other models train

Problem

An object detection model cannot be trained using Tiny YOLO v2, but other object detection models, such as SSD, FR-CNN are training successfully.

Solution

Verify that the NVIDIA GPUs are not configured to run in "exclusive" mode. The `nvidia-smi` command can be used to ensure the GPUs are in "default" mode:

```
nvidia-smi -c 0
```

Compare the following `nvidia-smi` output to the standard `nvidia-smi` output by following these instructions: “Checking system GPU status” on [page 48](#). It should show E. Process instead of Default for Compute M. in the final column:

```
+-----+
| NVIDIA-SMI 418.87.00      Driver Version: 418.87.00      CUDA Version: 10.1 |
+-----+-----+-----+
| GPU Name Persistence-M   | Bus-Id Dis.A   |      Volatile Uncorr. ECC      |
| Fan  Temp  Perf  Pwr:Usage/Cap|  Memory-Usage  | GPU-Util  Compute M. |
+-----+-----+-----+
| 0 Tesla P100-SXM2...  On | 00000002:01:00.0 Off |   0          |
| N/A  30C  P0   31W / 300W | 0MiB / 16280MiB |   0% E. Process  |
+-----+-----+-----+
```

Changing the port for the IBM Maximo Visual Inspection user interface

Problem

By default, the IBM Maximo Visual Inspection user interface uses port 443, forwarded from port 80.

Solution

If you need to use either port for something else, follow these steps to change the IBM Maximo Visual Inspection port.

1. If IBM Maximo Visual Inspection is running, use the following command to stop it:

```
$ /opt/ibm/vision/bin/vision-stop.sh
```

2. Change `/opt/ibm/vision/bin/port.sh`.

- Update this line with the appropriate port: `POWERAI_VISION_EXTERNAL_HTTP_PORT=80`.
- Update this line with the appropriate port: `POWERAI_VISION_EXTERNAL_HTTPS_PORT=443`.

3. Make sure the new port is open in your operating system's firewall by running the following command:

```
$ /opt/ibm/vision/sbin/firewall.sh
```

4. Restart IBM Maximo Visual Inspection by running the following command:

```
$ /opt/ibm/vision/bin/vision-start.sh
```

Out of space error from `load_images.sh`

Problem

When installing the product, the `load_images.sh` script is used to load the IBM Maximo Visual Inspection Docker images. The script might terminate with errors, the most frequent issue being insufficient disk space for loading the Docker images.

For example, the `/var/lib/docker` file system can run out of space, resulting in a message indicating that an image was not fully loaded. The following output shows that the Docker image `vision-dnn` was not able to be fully loaded because of insufficient file system space:

```
# df --output -BG "/var/lib/docker/"
Filesystem      Type Inodes  IUsed  IFree IUse% 1G-blocks  Used Avail Use% File
Mounted on
/dev/vda2       ext4 8208384 595697 7612687    8%      124G    81G   37G  70% /var/lib/docker/ /
#

*****
892d6f64ce41: Loading layer [=====>] 21.26MB/
21.26MB
785af1d0c551: Loading layer [=====>] 1.692MB/
1.692MB
dc102f4a3565: Loading layer [=====>] 747.9MB/
747.9MB
aac4b03de02a: Loading layer [=====>] 344.1MB/
344.1MB
d0ea7f5f6aab: Loading layer [=====>] 2.689MB/
2.689MB
62d3d10c6cc2: Loading layer [=====>] 9.291MB/
9.291MB
240c4d86e5c7: Loading layer [=====>] 778MB/
778MB
889cd0648a86: Loading layer [=====>] 2.775MB/
2.775MB
56bbb2f20054: Loading layer [=====>] 3.584kB/
3.584kB
3d3c7acb72e2: Loading layer [=====>] 2.117GB/
3.242GB
Error processing tar file(exit status 1): write /usr/bin/grops: no space left on device

[ FAIL ] Some images failed to load
[ FAIL ] Failure info:
          Loading the PowerAI Vision docker images...
#
```

This situation can also be noted in the output from `/opt/ibm/vision/bin/kubectl get pods`. This command is described in Chapter 6, “Checking the application and environment,” on page 35, which shows images that could not be loaded with a status of `ErrImagePull` or `ImagePullBackOff`.

Solution

The file system space for `/var/lib/docker` needs to be increased, even if the file system is not completely full. There might still be space in the file system where `/var/lib/docker` is located, but insufficient space for the IBM Maximo Visual Inspection Docker images. There are operating system mechanisms to do this, including moving or mounting `/var/lib/docker` to a file system partition with more space.

After the error situation has been addressed by increasing or cleaning up disk space on the `/var/lib/docker/` file system, re-run the `load_images.sh` script to continue loading the images. No clean up of the previous run of `load_images.sh` is required.

I forgot my user name or password

Problem

You forgot your user name or password and cannot log in to the IBM Maximo Visual Inspection GUI.

Solution

IBM Maximo Visual Inspection uses an internally managed users account database. To change your user name or password, see Chapter 7, “Logging in to IBM Maximo Visual Inspection,” on page 51.

GPUs are not available for training or inference

Problem

If IBM Maximo Visual Inspection cannot perform training or inference operations, check the following:

- Verify that the `nvidia-smi` output shows all relevant information about the GPU devices. For example, the following output shows Unknown error messages indicating that the GPUs are not in the proper state:

```
Mon Dec  3 15:43:07 2018
+-----+
| NVIDIA-SMI 410.72          Driver Version: 410.72          CUDA Version: 10.0   |
+-----+-----+-----+
| GPU  Name                Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap|  Memory-Usage | GPU-Util  Compute M. |
|=====+=====+=====+
|    0   Tesla V100-SXM2...    Off      | 000000004:04:00.0 Off |             0         |
| N/A   31C    P0           49W / 300W | Unknown Error  |          0%      Default |
+-----+-----+-----+
...
```

- Verify that the **nvidia-persistenced** service is enabled and running (active) by using the command `sudo systemctl status nvidia-persistenced`:

```
# systemctl status nvidia-persistenced
* nvidia-persistenced.service - NVIDIA Persistence Daemon
   Loaded: loaded (/etc/systemd/system/nvidia-persistenced.service; enabled; vendor
   preset: disabled)
   Active: active (running) since Tue 2018-11-13 08:41:22 CST; 2 weeks 6 days ago
   ...
```

Solution

- If the GPU status indicates errors and the **nvidia-persistenced** service is not enabled and active, enable and start the service:

1. Enable the service:

```
sudo systemctl enable nvidia-persistenced
```

2. Start the service:

```
sudo systemctl start nvidia-persistenced
```

- If the **nvidia-persistenced** service is enabled but the Persistence-M state still shows Off, verify that the udev rules have been set correctly if the system is a RHEL server. See this topic for details: [“NVIDIA Components: IBM POWER9 specific udev rules \(Red Hat only\)”](#) on page 23.

IBM Maximo Visual Inspection cannot train a model

Problem

The model training process might fail if your system does not have enough GPU resources or if there is not enough information defined in the model.

Note: Starting with version 1.2.0.1, training jobs are added to a queue, so training jobs will not fail due to lack of GPU resources.

Solution

- If you are training a data set for image classification, verify that at least two image categories are defined, and that each category has a minimum of five images.
- If you are training a data set for object detection, verify that at least one object label is used. You must also verify that each object is labeled in a minimum of five images.
- Check the status and availability of the GPUs on the system. For information about checking the GPU status, see [Checking system GPU status](#). For information about checking the GPU availability, see [Checking GPU availability](#).

Thumbnails do not load or images previews are missing

Problem

IBM Maximo Visual Inspection data set or model thumbnails do not load, or images are not visible when labeling or previewing an image.

Solution

Disable ad blockers, or exempt the IBM Maximo Visual Inspection user interface from ad blocking.

Training or deployment hangs - Kubernetes pod cleanup

Problem

You submit a job for training or deployment, but it never completes. When doing training or deployments, sometimes some pods that are running previous jobs are not terminated correctly by the Kubernetes services. In turn, they hold GPUs so no new training or deployment jobs can complete. They will be in the Scheduled state forever.

To verify that this is the problem, run `kubectl get pods` and review the output. The last column shows the age of the pod. If it is older than a few minutes, use the information in the Solution section to solve the problem.

Example:

```
kubectl get pods
vision-infer-ic-06767722-47df-4ec1-bd58-91299255f6hxxzk 1/1 Running 0 22m
vision-infer-ic-35884119-87b6-4d1e-a263-8fb645f0addqd2z 1/1 Running 0 22m
vision-infer-ic-7e03c8f3-908a-4b52-b5d1-6d2befec69ggqw5 1/1 Running 0 5h
vision-infer-od-c1c16515-5955-4ec2-8f23-bd21d394128b6k4 1/1 Running 0 3h
```

Solution

Follow these steps to manually delete the deployments that are hanging.

1. Determine the running deployments and look for those that have been running longer than a few minutes:

```
kubectl get deployments
```

2. Delete the deployments that were identified as hanging in the previous step.

```
kubectl delete deployment deployment_id
```

3. You can now try the training or deploy again, assuming there are available GPUs.

Note: When a deployment is manually deleted, vision-service might try to recreate it when it is restarted. The only way to force Kubernetes to permanently delete it is to remove the failing model from IBM Maximo Visual Inspection.

Training fails with error indicating "You must retrain the model".

Problem

Very long label names can result in training failures. Label or class names used in the data set are longer than 64 characters, and/or international characters that have multi-byte representation are used.

Solution

Label and class names should be 64 characters or less. Longer label names are supported but using international characters or very long label names can cause an internal metadata error, resulting in a training failure.

Model training and inference fails

Problem

The NVIDIA GPU device is not accessible by the IBM Maximo Visual Inspection Docker containers. To confirm this, run `kubectl logs -f _vision-service-ID_` and then check `pod_vision-`

`service-ID_vision-service.log` for an error indicating `error == cudaSuccess (30 vs. 0)`:

```
F0731 20:34:05.334903    35 common.cpp:159] Check failed: error == cudaSuccess (30 vs. 0)
unknown error
*** Check failure stack trace: ***
/opt/py-faster-rcnn/FRCNN/bin/train_frcnn.sh: line 24:    35 Aborted                  (core
dumped) _train_frcnn.sh
```

Solution

Use `sudo` to alter SELINUX permissions for all of the NVIDIA devices so they are accessible via the IBM Maximo Visual Inspection Docker containers.

```
sudo chcon -t container_file_t /dev/nvidia*
```

Training or deployment of models fails - sometimes inconsistently

Problem

There is a known race issue on RHEL systems that can prevent Kubernetes from successfully initializing and using one or more GPUs. This issue is typically indicated by `cudaSuccess 3 vs. 0` errors that can be found in the log of the vision-service pod. To confirm this, follow these steps:

1. Run the following command:

```
sudo /opt/ibm/vision/bin/kubect1 logs `sudo /opt/ibm/vision/bin/kubect1 get pods
-o custom-columns=NAME:.metadata.name | grep vision-service` | grep cudaSuccess
```

2. Check the output for a message such as the following: `error == cudaSuccess (3 vs. 0)`:

```
F0731 20:34:05.334903    35 common.cpp:159]
Check failed: error == cudaSuccess (3 vs. 0)  initialization error
*** Check failure stack trace: ***
```

Solution

See the blog entry on this issue: [What to do with “cudaSuccess \(3 vs. 0\) initialization error” on a POWER9 system?](#).

Model import generates an error alert

Problem

When you import a model, you get the error "The model was not imported. You can only import .zip files that were exported from an IBM Maximo Visual Inspection model."

Solution

This can occur when models are imported from older versions of IBM Maximo Visual Inspection that do not include model metadata in the exported file, such as the model name or thumbnail image. The model should still be imported successfully and will be available on the model details page for deployment.

Model accuracy value is unexpected

Problem

A trained model has an unexpected value for accuracy, such as 0%, 100%, or "Unknown". This happens when there is not enough data for training to work properly.

Solution

Ensure that there are enough images in the data set for each category or object label. For details, see [“Data set considerations” on page 59](#).

Deployed models stuck in "Starting"

Problem

IBM Maximo Visual Inspection models remain in "Starting" state and do not become available for inference operations.

Solution

Delete and redeploy the models. One possible cause is that the IBM Maximo Visual Inspection models were deployed in a prior version of the product that is not compatible with the currently installed version. For example, this can happen after upgrading.

Auto labeling of a data set returns "Auto Label Error"

Problem

Auto labeling cannot be performed on a data set that does not have unlabeled images, unless some of the images were previously labeled by the auto label function.

Solution

Ensure that the **Objects** section of the data set side bar shows there are objects that are "Unlabeled". If there are none, that is, if "Unlabeled (0)" is displayed in the side bar, add new images that are unlabeled or remove labels from some images, then run auto label again.

IBM Maximo Visual Inspection does not start

Problem

When you enter the URL for IBM Maximo Visual Inspection from a supported web browser, nothing is displayed. You see a 404 error or Connection Refused message.

Solution

Complete the following steps to solve this problem:

1. Verify that IP version 4 (IPv4) port forwarding is enabled by running the **/sbin/sysctl net.ipv4.conf.all.forwarding** command and verifying that the value for `net.ipv4.conf.all.forwarding` is set to 1.

If IPv4 port forwarding is not enabled, run the **/sbin/sysctl -w net.ipv4.conf.all.forwarding=1** command. For more information about port forwarding with Docker, see [UCP requires IPv4 IP Forwarding in the Docker success center](#).

2. If IPv4 port forwarding is enabled and the `docker0` interface is a member of the trusted zone, follow the steps in ["Checking application deployment" on page 43](#) to see if there are issues with the startup of application pods.
3. If the `docker0` interface is a member of a trusted zone and all IBM Maximo Visual Inspection components are available, verify that the firewall is configured to allow communication through port 443 (used to connect to IBM Maximo Visual Inspection) by running this command:

```
sudo firewall-cmd --permanent --zone=public --add-port=443/tcp
```

IBM Maximo Visual Inspection fails on first startup and Kubernetes services do not start

Problem

On the first start of the application, the Kubernetes services fail to start with the following output:

```
# sudo /opt/ibm/vision/bin/vision-start.sh
...
Checking kubernetes cluster status...
Probing cluster status #1:
Probing cluster status #2:
Probing cluster status #3:
Probing cluster status #4:
Probing cluster status #5:
Probing cluster status #6:
Probing cluster status #7:
Probing cluster status #8:
Probing cluster status #9:
Probing cluster status #10:
Probing cluster status #11:
[ FAIL ] Retry timeout. ...
```

This problem is characterized further by these symptoms:

- The steps in [Checking Kubernetes services status](#) indicate that the Kubernetes pods are not starting.
- The SSL key files are 0-length and no CSR or CRT files are created:

```
$ ls -l /opt/ibm/vision/config/
total 16
-rw-----. 1 root root  0 Aug 24 09:39 ca.key
-rw-----. 1 root root 620 Aug 24 09:39 csr.conf
-rw-r--r--. 1 root root 186 Aug 24 09:39 kube.config
-rw-r--r--. 1 root root 776 Aug 24 09:40 kubelet.config
-rw-r--r--. 1 root root 158 Aug 24 09:39 kubeproxy.config
-rw-----. 1 root root  0 Aug 24 09:39 server.key
```

- The `/opt/ibm/vision/log/start_k8s.log` file has errors:

```
Generating RSA private key, 2048 bit long modulus (2 primes)
140736267111472:error:2406C06E:random number generator:RAND_DRBG_instantiate:error
retrieving entropy:crypto/rand/drbg_lib.c:342:
140736267111472:error:2406C06E:random number generator:RAND_DRBG_instantiate:error
retrieving entropy:crypto/rand/drbg_lib.c:342:
140736267111472:error:2406B072:random number generator:RAND_DRBG_generate:in error
state:crypto/rand/drbg_lib.c:589:
140736267111472:error:2406C06E:random number generator:RAND_DRBG_instantiate:error
retrieving entropy:crypto/rand/drbg_lib.c:342:
140736267111472:error:2406C06E:random number generator:RAND_DRBG_instantiate:error
retrieving entropy:crypto/rand/drbg_lib.c:342:
140736267111472:error:2406B072:random number generator:RAND_DRBG_generate:in error
state:crypto/rand/drbg_lib.c:589:
140736267111472:error:2406C06E:random number generator:RAND_DRBG_instantiate:error
retrieving entropy:crypto/rand/drbg_lib.c:342:
140736267111472:error:2406B072:random number generator:RAND_DRBG_generate:in error
state:crypto/rand/drbg_lib.c:589:
140736267111472:error:04081003:rsa routines:rsa_builtin_keygen:BN lib:crypto/rsa/
rsa_gen.c:387:
unable to load Private Key
```

Solution

This problem can occur when the `openssl` package of Anaconda3 is installed and takes precedence over the system `openssl` in the environment.

The `/opt/ibm/vision/bin/k8s_start.sh` file can be modified to explicitly use the system `openssl` by following these steps:

1. Create a backup of the original file:

```
$ sudo cp /opt/ibm/vision/bin/k8s_start.sh /opt/ibm/vision/bin/k8s_start.sh.orig
```

2. Use `sed` to change `openssl` calls to use the `/usr/bin` location:

```
$ sudo sed -i "s/openssl/\usr/bin/openssl/g" /opt/ibm/vision/bin/k8s_start.sh
```

IBM Maximo Visual Inspection does not start with non-default Docker root directory

Problem

The IBM Maximo Visual Inspection application fails to start. [“Checking Kubernetes node status” on page 39](#) shows the `vision-ui` pod failing to start, and dependent pods (`vision-service` and `vision-keycloak`) in `Init` state:

NAME	AGE	IP	NODE	NOMINATED	NODE	READY	STATUS	RESTARTS
vision-elasticsearch-7598c68579-cvvc6					1/1	Running		23h
172.17.0.9	127.0.0.1	<none>	<none>				0	23h
vision-fpga-device-plugin-4tk6f					1/1	Running		23h
172.17.0.5	127.0.0.1	<none>	<none>				0	23h
vision-keycloak-7884d55485-ngv1b					0/1	Init:0/1		23h
172.17.0.10	127.0.0.1	<none>	<none>				0	23h
vision-logstash-59db969bd7-g7l79					1/1	Running		23h
172.17.0.11	127.0.0.1	<none>	<none>				0	23h
vision-mongodb-5997fcfd57-flzzm					1/1	Running		23h
172.17.0.12	127.0.0.1	<none>	<none>				0	23h
vision-postgres-cd4fdf548-gpkjp					1/1	Running		23h
172.17.0.13	127.0.0.1	<none>	<none>				0	23h
vision-service-b6b8567b6-ckdr4					0/1	Init:1/2		23h
172.17.0.14	127.0.0.1	<none>	<none>				0	23h
vision-taskanalyzer-84d4f88c48-tqs2c					1/1	Running		23h
172.17.0.6	127.0.0.1	<none>	<none>				280	23h
vision-ui-66c7c4b8b8-wsjs2					0/1	CrashLoopBackOff		23h
172.17.0.7	127.0.0.1	<none>	<none>				0	23h
vision-video-microservice-b595d75b-1lw55					1/1	Running		23h
172.17.0.8	127.0.0.1	<none>	<none>					

The “`kubectl describe pods`” on page 44 output for the `vision-ui` shows events indicating an issue with accessing the `resolv.conf` file, where `/mount/space` is a file path used for Docker data. For example:

```
$ kubectl describe pod vision-ui-66c7c4b8b8-wsjs2
Name:          vision-ui-66c7c4b8b8-wsjs2
...
Events:
  Type      Reason              Age             From              Message
  ----      -
  Normal    Scheduled           4m24s          default-scheduler Successfully
assigned default/vision-ui-589dbd466-bjmkb to 127.0.0.1
  Warning   FailedCreatePodSandbox 4m6s          kubelet, 127.0.0.1 Failed create
pod sandbox: rpc error: code = Unknown desc = rewrite resolv.conf failed for pod "vision-
ui-66c7c4b8b8-wsjs2": ResolvConfPath "/mount/space/docker/containers/
4a26a137c352177bef6b2d5b99a9d16668c3f80fa88b0c45def1da90bb7d063c/resolv.conf" does not exist
  Normal    Started              3m14s (x4 over 4m5s) kubelet, 127.0.0.1 Started container
...
```

Solution

If `/var/lib/docker` is a symlink to a different file system space, this can cause issues with starting Kubernetes containers, as reported in <https://github.com/kubernetes/kubernetes/issues/52655>. The usage of the symlink must be avoided, alternatives include:

- Reallocating file system space so `/var/lib/docker` file system has sufficient space for the Docker containers and runtime data.
- Removing the symlink and using `mount --bind`.
- Configuring Docker to use the non-default root directory.

IBM Maximo Visual Inspection fails to start - Kubernetes connection issue

Problem

There are different problems that can cause this issue:

- If the host system does not have a default route defined in the networking configuration the Kubernetes cluster will fail to start with connection issues. For example:

```
$ sudo /opt/ibm/vision/bin/vision_start.sh
[ INFO ] Starting kubernetes...
        Copying kubect1 out of k8s image...
        Copying helm executable to /opt/ibm/vision/bin
Using /run/systemd/resolve/resolve.conf for name service configuration.
        Checking kubernetes cluster status...
        Probing cluster status #1: NotReady
        Probing cluster status #2: Ready
        Booting up ingress controller...
        Initializing helm...
        Initializing GPU device plugin...
```

- You might see this problem if /opt does not have enough space. For example:

```
[ INFO ] Starting kubernetes...
        Copying kubect1 out of k8s image...
        Copying helm executable to /opt/ibm/vision/bin
Using /etc/resolve.conf for name service configuration.
        Checking kubernetes cluster status...
        Probing cluster status #1:
        Probing cluster status #2: NotReady
        Probing cluster status #3: NotReady
        Probing cluster status #4: NotReady
        Probing cluster status #5: NotReady
        Probing cluster status #6: NotReady
        Probing cluster status #7: NotReady
        Probing cluster status #8: NotReady
        Probing cluster status #9: NotReady
        Probing cluster status #10: NotReady
        Probing cluster status #11: NotReady

[ FAIL ] Retry timeout. Error in starting kubernetes cluster check logs
```

Solution

There are multiple possible solutions to this problem, depending on the underlying cause:

- Define a default route in the networking configuration.
 - For instructions to do this on Ubuntu, refer to the [IP addressing](#) section in the [Ubuntu Network Configuration](#). Search for the steps to configure and verify the default gateway.
 - For instructions to do this on Red Hat Enterprise Linux (RHEL), refer to [2.2.4 Static Routes and the Default Gateway](#) in the [Red Hat Customer Portal](#).
- Confirm that at least five GB of free space exists in the file system that is hosting /opt/ibm/vision/run. Refer to [Chapter 3, “Planning for IBM Maximo Visual Inspection,”](#) on page 15 for details on space requirements.

IBM Maximo Visual Inspection startup hangs - helm issue

Problem

IBM Maximo Visual Inspection startup hangs with the message "Unable to start helm within 30 seconds - trying again." For example:

```

root> sudo /opt/ibm/vision/bin/vision-start.sh
Checking ports usage...
Checking ports completed, no conflict port usage detected.
[ INFO ] Setting up the GPU...
        Init cuda devices...
        Devices init completed!
        Persistence mode is already Enabled for GPU 00000004:04:00.0.
        Persistence mode is already Enabled for GPU 00000004:05:00.0.
        Persistence mode is already Enabled for GPU 00000035:03:00.0.
        Persistence mode is already Enabled for GPU 00000035:04:00.0.
        All done.
[ INFO ] Starting kubernetes...
        Checking kubernetes cluster status...
        Probing cluster status #1: NotReady
        Probing cluster status #2: NotReady
        Probing cluster status #3: NotReady
        Probing cluster status #4: Ready
        Booting up ingress controller...
        Initializing helm...
[ WARN ] Unable to start helm within 30 seconds - trying again. If this continues,
contact support.
[ WARN ] Unable to start helm within 30 seconds - trying again. If this continues,
contact support.
[ WARN ] Unable to start helm within 30 seconds - trying again. If this continues,
contact support.
[ WARN ] Unable to start helm within 30 seconds - trying again. If this continues,
contact support.

```

Solution

To solve this problem, you must follow these steps exactly as written:

1. Cancel IBM Maximo Visual Inspection startup by pressing `ctrl+c`.
2. Stop IBM Maximo Visual Inspection by running this command:

```
sudo /opt/ibm/vision/bin/vision-stop.sh
```

3. Modify the RHEL settings as follows:

```

sudo nmcli device set docker0 managed yes
sudo nmcli connection modify docker0 connection.zone trusted
sudo systemctl stop NetworkManager.service
sudo firewall-cmd --permanent --zone=trusted --change-interface=docker0
sudo systemctl start NetworkManager.service
sudo nmcli connection modify docker0 connection.zone trusted
sudo systemctl restart docker.service

```

4. Start IBM Maximo Visual Inspection again:

```
sudo /opt/ibm/vision/bin/vision-start.sh
```

If the above commands do not fix the startup issue, check for a cgroup leak that can impact Docker. A Kubernetes/Docker issue can cause this situation, and after fixing the firewall issue the start up can still fail if there was cgroup leakage.

One symptom of this situation is that the `df` command is slow to respond. To check for excessive cgroup mounts, run the mount command:

```
$ mount | grep cgroup | wc -l
```

If the cgroup count is in thousands, reboot the system to clear up the cgroups.

Helm status errors when starting IBM Maximo Visual Inspection

Problem

There is an issue in some RHEL releases that causes the startup of IBM Maximo Visual Inspection to fail after restarting the host system. When this is the problem, the system tries to initialize Helm at 30

second intervals but never succeeds. Therefore, the startup never succeeds. You can verify this status by running the Helm status vision command:

```
# /opt/ibm/vision/bin/helm status vision
```

Result:

```
Error: getting deployed release "vision": Get https://10.10.0.1:443/api/v1/namespaces/kube-system/configmaps[...]: dial tcp 10.10.0.1:443: getsockopt: no route to host
```

Solution

To solve this problem, you must follow these steps exactly as written:

1. Cancel IBM Maximo Visual Inspection startup by pressing `ctrl+c`.
2. Stop IBM Maximo Visual Inspection by running this command:

```
sudo /opt/ibm/vision/bin/vision-stop.sh
```

3. Modify the RHEL settings as follows:

```
sudo nmcli device set docker0 managed yes
sudo nmcli connection modify docker0 connection.zone trusted
sudo systemctl stop NetworkManager.service
sudo firewall-cmd --permanent --zone=trusted --change-interface=docker0
sudo systemctl start NetworkManager.service
sudo nmcli connection modify docker0 connection.zone trusted
sudo systemctl restart docker.service
```

4. Start IBM Maximo Visual Inspection again:

```
sudo /opt/ibm/vision/bin/vision-start.sh
```

If the above commands do not fix the startup issue, check for a cgroup leak that can impact Docker. A Kubernetes/Docker issue can cause this situation, and after fixing the firewall issue the start up can still fail if there was cgroup leakage.

One symptom of this situation is that the `df` command is slow to respond. To check for excessive cgroup mounts, run the mount command:

```
$ mount | grep cgroup | wc -l
```

If the cgroup count is in thousands, reboot the system to clear up the cgroups.

Uploading a large file fails

When uploading files into a data set, there is a 24 GB size limit per upload session. This limit applies to a single .zip file or a set of files. When you upload a large file that is under 24 GB, you might see the upload start (showing a progress bar) but then you get an error message in the user interface. This error happens due to a Nginx timeout, where the file upload is taking longer than the defined 5 minute Nginx timeout.

Despite the notification error, the large file has been uploaded. Refreshing the page shows the uploaded files in the data set.

Some IBM Maximo Visual Inspection functions don't work

Problem

IBM Maximo Visual Inspection seems to start correctly, but some functions, like automatic labeling or automatic frame capture, do not function.

To verify that this is the problem, run `/opt/ibm/vision/bin/kubect1.sh get pods` and verify that one or more pods are in state `CrashLoopBackOff`. For example:

```
kubectl get pods
NAME                                READY   STATUS
RESTARTS   AGE
...
vision-video-rabmq-5d5d786f9f-7jfk9 0/1     CrashLoopBackOff
2                                     54s
```

Solution

IBM Maximo Visual Inspection requires IPv6. Enable IPv6 on the system.

Troubleshooting known issues - IBM Maximo Visual Inspection Edge

Following are some problems you might encounter when using IBM Maximo Visual Inspection, along with steps to fix them.

- [“Problems installing an rpm on a RHEL system with Docker CE” on page 183](#)
- [“When deploying a model, you get an error that the /decrypt container name is already in use” on page 183](#)
- [“Model fails to deploy on RHEL system with SE Linux” on page 183](#)
- [“Model fails to deploy to GPU on RHEL with a CUDA runtime error” on page 184](#)
- [“Model fails to deploy to GPU on RHEL - No GPU found” on page 184](#)
- [“Model fails to deploy with time out message” on page 184](#)

Problems installing an rpm on a RHEL system with Docker CE

Problem

When installing an rpm on a RHEL system with Docker CE, you see this error: Error: Failed dependencies: docker is needed by <file_name.rpm>. For example:

Solution

To install an rpm on a system with Docker CE instead of Docker, force install the rpm by the following command

```
rpm --nodeps -i file_name.rpm
```

When deploying a model, you get an error that the /decrypt container name is already in use

Problem

When deploying a model, you get a docker error such as the following:

```
docker: Error response from daemon: Conflict. The container name "/decrypt" is already in use by container "b9deb17c4651162aaf609cb97835098b69f6f9ac5c5558041a0ff52e8d0777". You have to remove (or rename) that container to be able to reuse that name. See 'docker run --help'.
```

This error can occur if a previous model deployment/decryption was terminated or failed unexpectedly during the deployment/decryption process.

Solution

Remove the Docker image by running the following commands:

```
docker stop decrypt; docker rm decrypt
```

Model fails to deploy on RHEL system with SE Linux

Problem

A model deployment fails, and the log of the container indicates the "model.zip was not mounted correctly:Deployment failed." The last few lines of the log should be displayed, and include the line "please mount...".

[illegible]

Check the SELinux context of the model file specified. If it does not have `container_file_t`, the model file cannot be successfully read in the container. Use `chcon` to correct the context, for example:

```
chcon unconfined_u:object_r:container_file_t:s0 <modelfile.zip>
```

When deploying a model to a GPU on a RHEL system, it fails with an error indicating an issue with the NVIDIA CUDA driver:

```
F1113 18:46:17.689370 17 syncedmem.cpp:518] Check failed: error == cudaSuccess (35 vs. 0)
      CUDA driver version is insufficient for CUDA runtime version
```

Ensure that SELinux is enabled. If it is enabled, verify that the security context of the `/usr/lib64/libcudax*` libraries includes `textrel_shlib_t`. If it does not, use `chcon` to set the context correctly:

```
# getenforce
Enforcing

# # sudo chcon -t textrel_shlib_t /usr/lib64/libcuda.so.*
```

When deploying a model to a GPU on a RHEL system, it fails with error indicating that the GPU could not be found. For example, when attempting to deploy to a GPU

[illegible]

Ensure that SELinux is enabled. If it is enabled, verify that the security context of the `/dev/nvidia` devices includes `container_file_t`. If it does not, use `chcon` to set the context correctly:

```
# getenforce
Enforcing
# ls -lZ /dev/nvidia*crw-rw-rw-.
    root root system_u:object_r:xserver_misc_device_t:s0 /dev/nvidia0crw-rw-rw-.
    root root system_u:object_r:xserver_misc_device_t:s0 /dev/nvidia1
# sudo chcon -t container_file_t /dev/nvidia*
# ls -lZ /dev/nvidia*crw-rw-rw-.
    root root system_u:object_r:container_file_t:s0 /dev/nvidia0crw-rw-rw-.
    root root system_u:object_r:container_file_t:s0 /dev/nvidia1
```

When using `deploy_zip_model.sh` to deploy a IBM Maximo Visual Inspection model, the action fails with a message "Deployment timed out at 180 seconds".

Solution

This can occur if the GPU specified for the deployment no longer has available memory to deploy the model. Check the GPU usage by using the `nvidia-smi` command as described in this topic: [“Checking system GPU status”](#) on page 48. For example, if the model failed to deploy to GPU 1, run `nvidia-smi -i 1` to check the usage of GPU 1. If there are limited memory resources, stop and delete some of the models currently deployed to the GPU by running these commands:

```
docker stop <model-name>
docker rm <model-name>
```

The following output demonstrates a situation where the GPU does not have sufficient memory to deploy the model:

```
# /opt/ibm/vision-edge/dnn-deploy-service/bin/deploy_zip_model.sh -m
8193_cars_custom_COD_model -p 7005 -g 1 /root/inference-only-testing/new_models/cars-tf-
cod.zip
chcon: can't apply partial context to unlabeled file '/root/inference-only-testing/
new_models/cars-tf-cod.zip'
WARNING: This might cause model file permission issue inside container
chcon: can't apply partial context to unlabeled file '/tmp/aivision_inference'
WARNING: This might cause permission issue inside container
```

```
Deployment timed out at 180 seconds
[root@dldev4 ~]# nvidia-smi -i 1
Tue Apr 30 14:13:39 2019
```

NVIDIA-SMI 418.29				Driver Version: 418.29		CUDA Version: 10.1	
GPU	Name	Persistence-M	Bus-Id	Disp.A	Volatile	Uncorr.	ECC
Fan	Temp	Perf	Pwr:Usage/Cap	Memory-Usage	GPU-Util	Compute M.	
=====							
1	Tesla	P100-PCIE...	Off	00000000:81:00.0	Off		0
N/A	32C	P0	31W / 250W	15919MiB / 16280MiB	0%		Default
+-----+							

Processes:					GPU Memory
GPU	PID	Type	Process name	Usage	
1	13691	C	python	2133MiB	
1	17165	C	python	2065MiB	
1	17955	C	python	958MiB	
1	18832	C	python	742MiB	
1	19545	C	python	10009MiB	

Gather IBM Maximo Visual Inspection logs and contact support

Sometimes you cannot solve a problem by troubleshooting the symptoms. In such cases, you must collect diagnostic data and contact support.

About this task

Collecting and inspecting data before you open a problem management record (PMR) can help you to answer the following questions:

- Do the symptoms match any known problems? If so, has a fix or workaround been published?
- Can the problem be identified and resolved without a code fix?
- When does the problem occur?

To gather logs for support, follow these steps:

Procedure

1. Collect logs from the IBM Maximo Visual Inspection application.

- **Standalone installation:**

- **Collect the vision-service log:** The most useful logs to debug an issue with the application are the **vision-service** logs. Run this command to collect the logs from the vision-service pod, and output them to a log file that includes a timestamp in the file name for reference:

```
sudo /opt/ibm/vision/bin/kubect1 logs `sudo /opt/ibm/vision/bin/kubect1 get pods -o
custom-columns=NAME:.metadata.name | grep vision-service` > ./vision-service-`date +%d
%m%Y-%H%M%S`.log
```

- **Collect all logs:**

Run the **sudo /opt/ibm/vision/bin/collect_logs.sh** script. The directory where the log file is saved is listed in the **INFO: FFDC Collected** section, as shown in the following example:

```
[ INFO ] Collecting Visual Inspection Logs in Parallel...
[ INFO ] Collecting Visual Inspection Application Logs...
[ INFO ] Collecting Visual Inspection Infrastructure Logs...
/opt/ibm/vision /var/log/vision/vision.logs.18_20_22_Feb_27_2020
/var/log/vision/vision.logs.18_20_22_Feb_27_2020
[ INFO ] Collecting configuration information...
[ INFO ] Collecting System Details...
[ INFO ] Collecting Platform Logs...
[ INFO ] FFDC Collected below:
-rw-r--r--. 1 root root 5895480 Feb 27 18:20 /var/log/vision/
vision.logs.18_20_22_Feb_27_2020.tgz
```

The log files to provide are generated here: /var/log/vision.

2. Optionally, you can obtain the logs for a single pod of the application.
 - a. Use the `kubect1 get pods` command to view the running pods for the application. See [“kubect1.sh get pods” on page 43](#). For example:

```
$ /opt/ibm/vision/bin/kubect1.sh get pods
```

NAME	RESTARTS	AGE	READY	STATUS
vision-cod-infer-ce5c3be1-491a-486c-81ae-2c49bdf17242-86cdtmv8q	0	14h	1/1	Running
vision-edge-connector-7db99dbb7f-vbtpg	0	38h	1/1	Running
vision-edge-mqttbroker-79cb65b865-rhr7w	0	38h	1/1	Running
vision-elasticsearch-6b779bb5f5-9mnck	0	38h	1/1	Running
vision-event-service-86c8c8cd6d-2wq5r	0	38h	1/1	Running
vision-fpga-device-plugin-86htq	0	38h	1/1	Running
vision-keycloak-58f7566896-nwph6	0	38h	1/1	Running
vision-logstash-565d995764-hnnxg	0	38h	1/1	Running
vision-mongodb-b59b56645-wlxgn	0	38h	1/1	Running
vision-postgres-6c57856875-zlknm	0	38h	1/1	Running
vision-service-687b85b97f-spg8n	1	38h	1/1	Running
vision-taskanaly-6dc659c45c-t9rb2	0	38h	1/1	Running
vision-ui-7d885944fc-x8tfc	0	38h	1/1	Running
vision-video-microservice-8457664dc6-dcb2g	0	38h	1/1	Running

- b. Run the following command, where `<pod-name>` is obtained from the `kubectl.sh get pods` command:

```
kubectl.sh logs <pod-name> > <outputfile>
```

For example, using the above output, to collect logs in the file `vision-service.log`, run the command:

```
$ kubectl.sh logs vision-service-5588ffdfc-cnq8h > vision-service.log
```

3. Submit the problem to IBM Support in one of the following ways:
- Online through the IBM Support Portal: <http://www.ibm.com/mysupport/>: You can open, update, and view all of your service requests from the Service Request portlet on the Service Request web page.
 - By phone: For the phone number to call in your region, see the Directory of worldwide contacts web page: <http://www.ibm.com/planetwide/>.

Getting fixes from Fix Central

You can use Fix Central to find the fixes that are recommended by IBM Support for various products, including IBM Maximo Visual Inspection. With Fix Central, you can search, select, order, and download fixes for your system with a choice of delivery options. A IBM Maximo Visual Inspection product fix might be available to resolve your problem.

About this task

Procedure

To find and install fixes:

1. Obtain the tools that are required to get the fix. If it is not installed, obtain your product update installer. You can download the installer from Fix Central: <http://www.ibm.com/support/fixcentral>. This site provides download, installation, and configuration instructions for the update installer.
- Note:** For more information about how to obtain software fixes, from the Fix Central page, click **Getting started with Fix Central**, then click the Software tab.
2. Under **Find product**, type "IBM Maximo Visual Inspection" in the **Product selector** field.
3. Select IBM Maximo Visual Inspection. For **Installed version**, select **All**. For **Platform**, select the appropriate platform or select **All**, then click **Continue**.
4. Identify and select the fix that is required, then click **Continue**.
5. Download the fix. When you download the file, ensure that the name of the maintenance file is not changed, either intentionally or by the web browser or download utility.
6. Stop IBM Maximo Visual Inspection by using this script:

```
sudo /opt/ibm/vision/bin/vision-stop.sh
```

7. Install the RPM that was downloaded by running this command:

```
sudo yum install ./<fixpack-rpmfile>.rpm
```

8. Log in as root or with sudo privileges, then load the images provided in the TAR file that was downloaded by running this script:

```
sudo /opt/ibm/vision/bin/load_images.sh ./<fixpack-tarfile>.tar
```

9. Start IBM Maximo Visual Inspection by running the following script. You must read and accept the license agreement that is displayed before you can use IBM Maximo Visual Inspection.

```
sudo /opt/ibm/vision/bin/vision-start.sh
```

Contacting IBM Support

IBM Support provides assistance with product defects, answers FAQs, and helps users resolve problems with the product.

Before you begin

After trying to find your answer or solution by using other self-help options such as technotes, you can contact IBM Support. Before contacting IBM Support, your company or organization must have an active IBM software maintenance agreement (SWMA), and you must be authorized to submit problems to IBM. For information about the types of available software support, see the [Support portfolio](#) topic in the *"Software Support Handbook"*.

To determine what versions of the product are supported, refer to the [Software lifecycle page](#).

Procedure

To contact IBM Support about a problem:

1. Define the problem, gather background information, and determine the severity of the problem.
For software support information, see the [Getting IBM support](#) topic in the *Software Support Handbook*.
2. Gather diagnostic information.
3. Submit the problem to IBM Support in one of the following ways:
 - Using IBM Support Assistant (ISA):
 - Online through the [IBM Support Portal](#): You can open, update, and view all of your service requests on the Service Request page.
 - By phone: For the phone number to call in your region, see the [Directory of worldwide contacts](#) web page.

Results

If the problem that you submit is for a software defect or for missing or inaccurate documentation, IBM Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Support provides a workaround that you can implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM Support website daily, so that other users who experience the same problem can benefit from the same resolution.

What to do next

Chapter 17. Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year).

Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. _enter the year or years_.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

The registered trademark Linux is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, Windows, and Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Red Hat®, OpenShift®, Fedora®, and Gluster® are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Other product and service names might be trademarks of IBM or other companies.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

This Software Offering does not use cookies or other technologies to collect personally identifiable information.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at <http://www.ibm.com/privacy> and IBM's Online Privacy Statement at <https://www.ibm.com/privacy/details/us/en/> in the section entitled "Cookies, Web Beacons and Other Technologies".

Chapter 18. IBM Maximo Visual Inspection 1.3.0

Release Notes

Requirements

For hardware and software requirements, see the [Chapter 3, “Planning for IBM Maximo Visual Inspection,” on page 15](#) topic.

Installing

For information about installing IBM Maximo Visual Inspection, see [Chapter 5, “Installing, upgrading, and uninstalling IBM Maximo Visual Inspection,” on page 21](#).

Limitations

Following are some limitations for IBM Maximo Visual Inspection 1.3.0:

- IBM Maximo Visual Inspection uses an entire GPU when you are training a dataset. Multiple GoogleNet or Faster R-CNN models can be deployed to a single GPU. Other types of models take an entire GPU when deployed. For details about other differences between model types, see [“Model functionality” on page 11](#).

The number of active GPU tasks (model training and deployment) that you can run at the same time depends on the number of GPUs on your server. You must verify that there are enough available GPUs on the system for the desired workload. The number of available GPUs is displayed on the user interface.

- You cannot install IBM Maximo Visual Inspection stand-alone on a system that already has any of these products installed:
 - IBM Data Science Experience (DSX)
 - IBM Cloud Private
 - IBM Watson Studio Local Edition
 - Any other Kubernetes based applications

Notices

This information was developed for products and services offered in the US.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data and client examples cited are presented for illustrative purposes only. Actual performance results may vary depending on specific configurations and operating conditions.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year).

Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. _enter the year or years_.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at [Copyright and trademark information](http://www.ibm.com/legal/copytrade.shtml) at www.ibm.com/legal/copytrade.shtml.

The registered trademark Linux is used pursuant to a sublicense from the Linux Foundation, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Red Hat, OpenShift, Fedora, and Gluster are trademarks or registered trademarks of Red Hat, Inc. or its subsidiaries in the United States and other countries.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Other product and service names might be trademarks of IBM or other companies.

