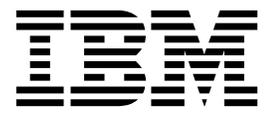


IBM PowerAI Vision

Version 1.1.4

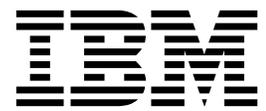
PowerAI Vision Guide



IBM PowerAI Vision

Version 1.1.4

PowerAI Vision Guide



Note

Before using this information and the product it supports, read the information in "Notices" on page 147.

This edition applies to IBM PowerAI Vision Version 1.1.4 and to all subsequent releases and modifications until otherwise indicated in new editions.

© **Copyright IBM Corporation 2018, 2019.**

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

Contents

About this document	v	Training a model	66
Highlighting	v	Working with custom models	69
ISO 9000.	v	Base models included with PowerAI Vision	77
IBM PowerAI Vision overview	1	Testing a model	77
Use cases	2	Refining a model	79
What's new.	7	Automatically labeling objects	79
PowerAI Vision editions	7	Augmenting the data set	81
IBM PowerAI Vision Trial	8	Deploying a trained model	82
PowerAI Vision Non-production edition.	11	Importing and exporting PowerAI Vision information	83
Model functionality.	11	PowerAI Vision REST APIs	85
PowerAI Vision concepts	13	Understanding metrics	85
Planning for PowerAI Vision	15	Using PowerAI Vision	89
License Management in IBM License Metric Tool	19	Example: Detecting objects in images	89
Installing, upgrading, and uninstalling PowerAI Vision	21	Example: Detecting objects in a video.	92
Prerequisites for installing PowerAI Vision	21	Example: Classifying images.	97
Installing PowerAI Vision stand-alone	25	Example: Detecting segmented objects in images	98
Installing PowerAI Vision with IBM Cloud Private	29	Example: Detecting actions in a video	100
Upgrading PowerAI Vision	30	Administering PowerAI Vision	103
Uninstalling PowerAI Vision stand-alone	32	Managing users	103
Checking the application and environment	35	Installing a new SSL certificate in PowerAI Vision stand-alone	105
Checking the application Docker images in standalone installation.	35	Monitoring usage metrics	106
Checking the application status in an ICP installation	36	PowerAI Vision utilities	106
Checking Kubernetes services status	37	PowerAI Vision Inference Server	111
Checking Kubernetes node status	39	Inference on embedded edge devices	117
Checking Kubernetes storage status	43	PowerAI Vision REST APIs	121
Checking application deployment	44	Troubleshooting and contacting support	123
Checking system GPU status	49	Troubleshooting known issues - PowerAI Vision standard install.	123
Logging in to PowerAI Vision	51	Troubleshooting known issues - PowerAI Vision Inference Server	134
Working with the user interface	53	Troubleshooting known issues - IBM Cloud Private install	136
Training and working with models	57	Gather PowerAI Vision logs and contact support	137
Creating and working with data sets	57	Getting fixes from Fix Central	139
Data set considerations	58	Contacting IBM Support.	140
Importing images with COCO annotations	60	Notices	141
Labeling objects	61	Trademarks	142
Labeling actions	64	Terms and conditions for product documentation	143
		Notices	147
		Trademarks	149

About this document

This document provides you with information about installing and using IBM® PowerAI Vision 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.
Monospace	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.

IBM PowerAI Vision overview

The IBM PowerAI Vision 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.

PowerAI Vision includes tools and interfaces for anyone with limited skills in deep learning technologies. You can use PowerAI Vision 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 PowerAI Vision:

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 PowerAI Vision consists of hardware, resource management, deep learning computation, service management, and application service layers. Each layer is built around industry-standard technologies.

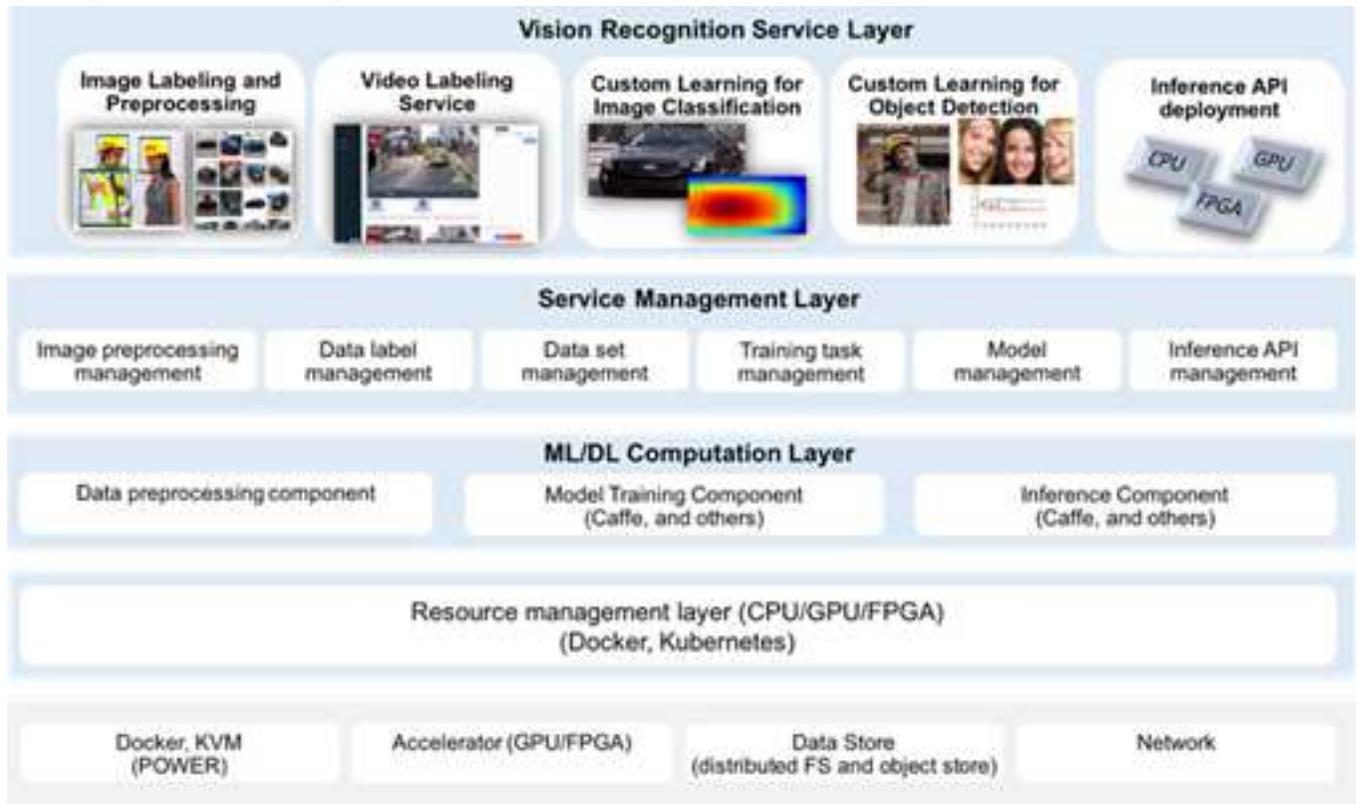


Table 1. Overview of the architecture layers

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

Use cases

IBM PowerAI Vision 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

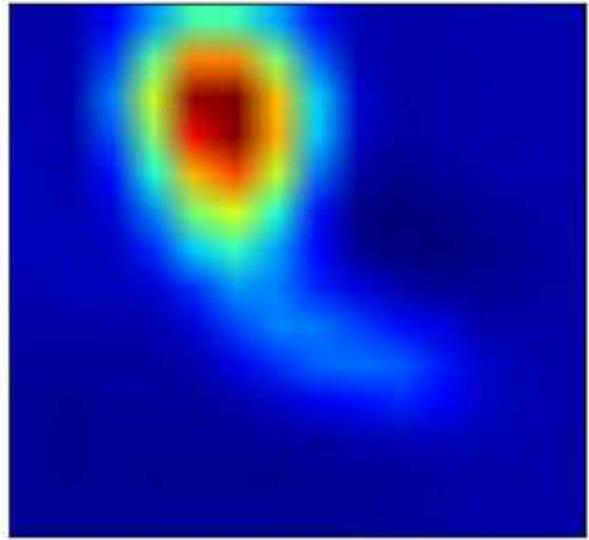


Figure 1. Detecting the overall contents of an image, based on custom training data

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.

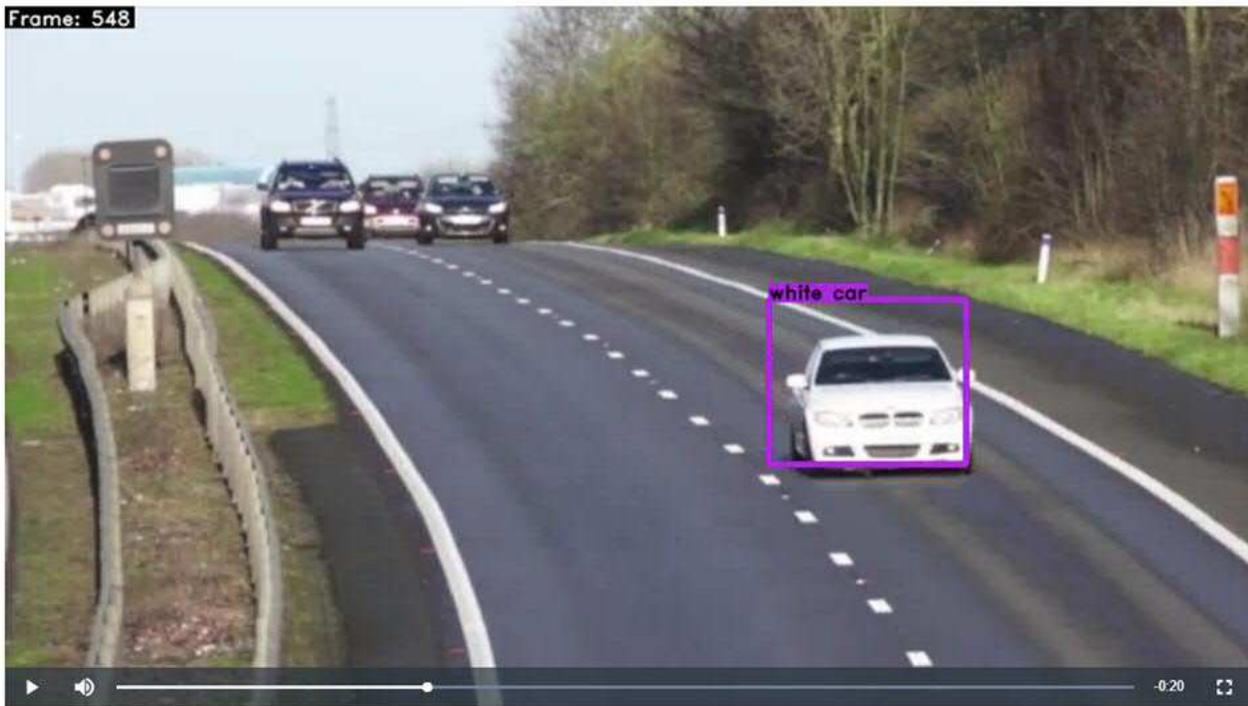


Figure 3. Detecting and labeling instances of objects within captured video frames based on custom training data

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.

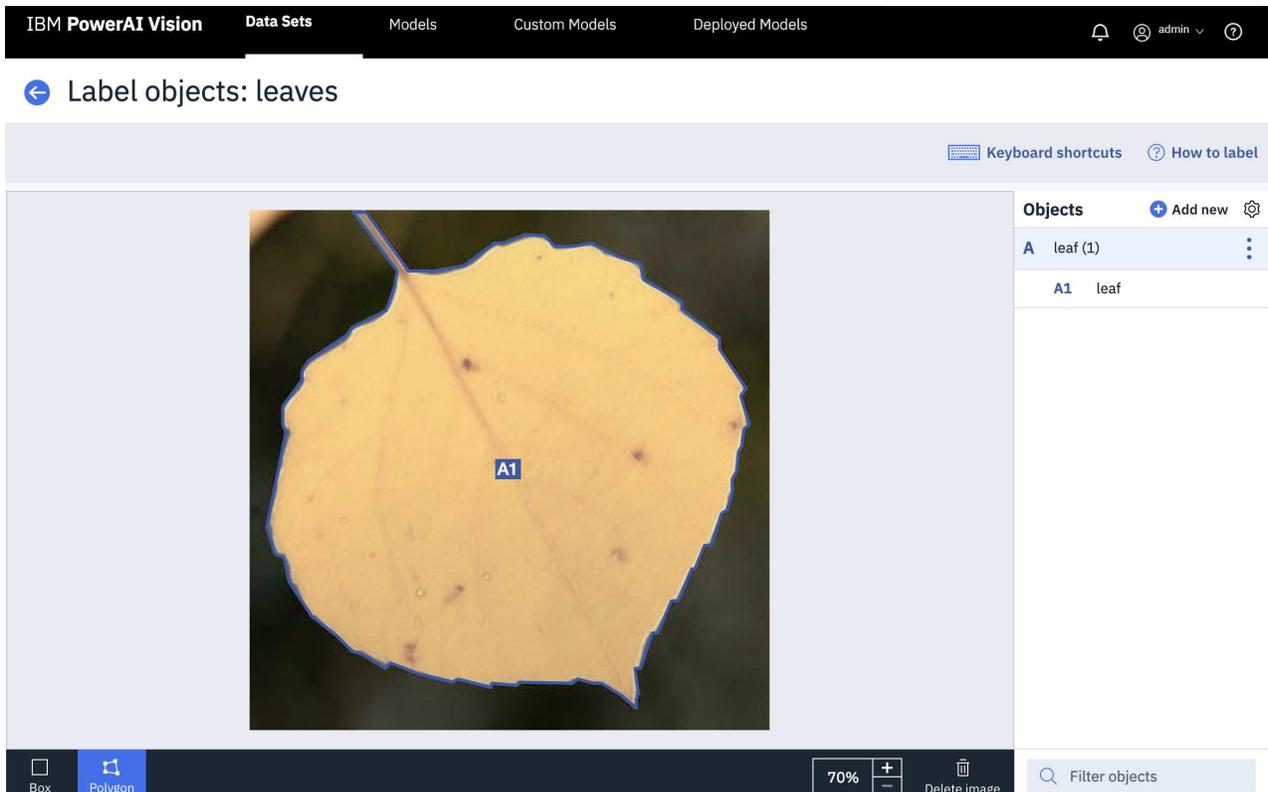


Figure 4. Detecting and labeling the precise edges of an object within an image based on custom training data

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 improves the model's accuracy by quickly adding more data to the data set.

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

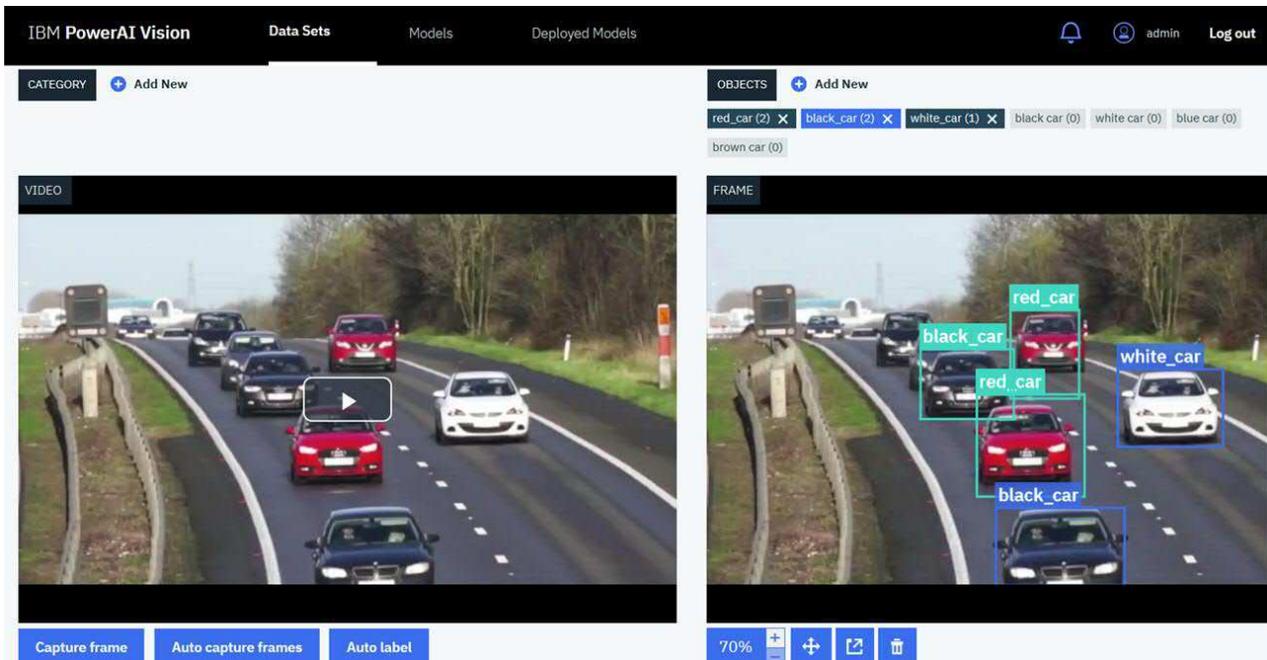


Figure 5. Auto labeled video

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.

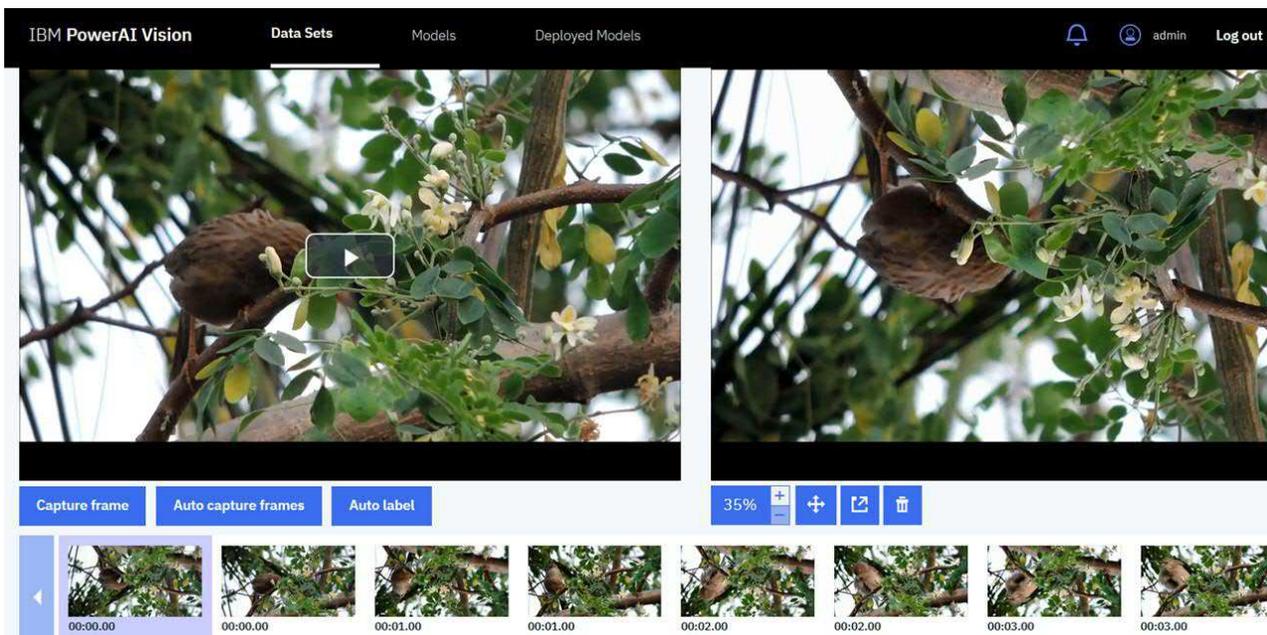


Figure 6. Augmented video

What's new

The following functions, features, and support have been added for PowerAI Vision Version 1.1.4:

Action identification

You can train models to identify actions in a video. To get started, see “Labeling actions” on page 64.

PowerAI Vision Non-production edition update

This edition has updated functionality. The number of deployed and trained models is limited, but you can now export models. See “PowerAI Vision Non-production edition” on page 11 for details.

Track usage metrics

Administrators can track how many inferences have been run in PowerAI Vision. For details, see “Monitoring usage metrics” on page 106.

PowerAI Vision trial version

The functionality of the trial version has been updated. The number of deployed and trained models is limited, but you can now export models. See “IBM PowerAI Vision Trial” on page 8 for details.

Inferencing for videos with Detectron models

Detectron models now accept videos for inferencing; not just images.

View video inferencing results for Detectron models

After a video inferencing completes, the annotated video plays. For details, see “Testing a model” on page 77.

Updates to the user interface

The following changes have been made to the user interface to improve your experience:

- The user name drop down menu in the navigation bar now contains the following:
 - The log out action
 - Edition information (time trial and Non-production versions only)
 - Product version
- A refresh button has been added to the GPU usage information.
- If you have external GPUs, they are now included in the GPU usage list. External GPUs are those that are used for processes outside of PowerAI Vision. For IBM Cloud™ Private installations, all GPUs are listed as External.
- When auto capturing frames, you can now specify values as small as .001.
- Notifications will be sent that show progress on active trainings and video inferences.
- Columns for Original file name, Actions, and Augment were added to the Data Set Details grid. The original file name will not be accurate, or will be missing for older data sets that are imported. Additionally, the original file name is not available for captured video frames or augmented files.
- An “Auto labeled” filter has been added to the Data Set Details side bar, which can be used to hide or show images or video frames that have auto labels.

PowerAI Vision editions

Each edition of PowerAI Vision has different functionality.

Table 2. PowerAI Vision editions

Edition	Can be installed on IBM Cloud Private	Can be installed as standalone	Model limit	Watermarked images	Can create additional users	Time limit	Renewable
Trial	Y	Y	5	Y		90 days	No - must reinstall
Non-production	Y	Y	5	Y		1 year	Y
Full	Y	Y	N/A		Y		N/A

IBM PowerAI Vision Trial

PowerAI Vision 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”
- “Installing the trial version”
- “What happens when the trial expires?” on page 9
- “Upgrading to the full version of PowerAI Vision” on page 10

Limitations

This edition of PowerAI Vision 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 and exporting PowerAI Vision information” on page 83 for instructions.
- Only five models can be deployed at any time. To delete a deployed model, click **Deployed Models**. Next, select the model that you want to delete and click **Delete**. The trained model is not deleted from PowerAI Vision.
- 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 PowerAI Vision stand-alone on the same system that has the following software installed:

- IBM Data Science Experience (DSX)
- IBM Cloud Private
- Any other Kubernetes based applications

1. You must complete the following installation prerequisites steps before you install PowerAI Vision.
 - a. Complete all steps in the “Prerequisites for installing PowerAI Vision” 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.
2. Go to PowerAI Vision Trial download site. Download the .tar file and the .rpm files as instructed.
3. Decompress 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
```

4. From the directory that contains the downloaded tar file, run the appropriate script as root or with sudo privileges:

```
sudo /opt/powerai-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 PowerAI Vision by running this script:

```
sudo /opt/powerai-vision/sbin/firewall.sh
```

6. After the installation is complete, you can start PowerAI Vision by running this script:

```
sudo /opt/powerai-vision/bin/powerai_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 103.

You must read and accept the license agreement that is displayed before you can use PowerAI Vision.

It can take several minutes to start PowerAI Vision. To check the status of the startup process, run this script:

```
sudo /opt/powerai-vision/bin/helm.sh status vision
```

In the output from the **helm.sh status vision** script, you can verify which PowerAI Vision 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 **helm.sh status vision** script that shows all components are available:

```
==> v1beta1/Deployment
NAME                                DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
powerai-vision-mongodb              1         1         1             1         3h
powerai-vision-portal                1         1         1             1         3h
powerai-vision-postgres              1         1         1             1         3h
powerai-vision-taskanalyzer          1         1         1             1         3h
powerai-vision-ui                    1         1         1             1         3h
powerai-vision-video-nginx           1         1         1             1         3h
powerai-vision-video-portal           1         1         1             1         3h
powerai-vision-video-rabmq           1         1         1             1         3h
powerai-vision-video-redis           1         1         1             1         3h
powerai-vision-video-test-nginx      1         1         1             1         3h
powerai-vision-video-test-portal     1         1         1             1         3h
powerai-vision-video-test-rabmq      1         1         1             1         3h
powerai-vision-video-test-redis      1         1         1             1         3h
```

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 PowerAI Vision, follow the instructions in “Upgrading to the full version of PowerAI Vision” on page 10.

If the trial expires and you do not decide to purchase PowerAI Vision, follow these steps:

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

```
sudo /opt/powerai-vision/bin/purge_image.sh 1.1.4.0
```

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

```
sudo /opt/powerai-vision/bin/purge_data.sh
```

2. Remove PowerAI Vision by running the following command:

- For RHEL:

```
sudo yum remove powerai-vision
```

- For Ubuntu:

```
sudo dpkg --remove powerai-vision
```

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

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

Upgrading to the full version of PowerAI Vision

When you are ready to purchase PowerAI Vision, you can buy a license from PowerAI Vision 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 PowerAI Vision by running the following script:

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

2. Obtain and install PowerAI Vision:

Install PowerAI Vision from IBM Passport Advantage

- a. Download the product tar file from the IBM Passport Advantage website.
- b. Decompress 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
```

- c. From the directory that contains the downloaded tar file, run the appropriate script as root or with sudo privileges:

```
sudo /opt/powerai-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 PowerAI Vision by running this script:

```
sudo /opt/powerai-vision/bin/powerai_vision_start.sh
```

Install PowerAI Vision from AAS

- a. Download the product tar.gz file from Advanced Administration System (AAS). This system is also called Entitled Software Support (ESS).
- b. Unzip and untar the tar.gz file by running this command. The install files are extracted to powerai-vision-aas-1.1.4.1/.

```
gunzip -c file_name.tar.gz | tar -xvf
```

- c. Decompress 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
```

- d. From the directory that contains the extracted tar file, run this script as root or with sudo privileges:

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

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

- e. After the installation is complete, you can start PowerAI Vision by running this script:

```
sudo /opt/powerai-vision/bin/powerai_vision_start.sh
```

Related concepts:

“Uninstalling PowerAI Vision stand-alone” on page 32

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

PowerAI Vision Non-production edition

This edition of PowerAI Vision lets you try out the product for one year, with limited functionality.

PowerAI Vision Non-production edition functionality

This edition of PowerAI Vision 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 and exporting PowerAI Vision information” on page 83 for instructions.
- Only five models can be deployed at any time. To delete a deployed model, click **Deployed Models**. Next, select the model that you want to delete and click **Delete**. The trained model is not deleted from PowerAI Vision.
- 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

Follow the “Install PowerAI Vision from IBM Passport Advantage” instructions in “Installing PowerAI Vision stand-alone” on page 25.

What happens after the license expires?

When your license expires, your trained models and labeled data sets are preserved. To continue working with PowerAI Vision, follow these steps:

1. Contact your IBM sales representative to obtain an extended license key.
2. Input the licence key:
 - From the user interface, click the arrow by your user name and select **Edition information**. Paste the license extension token in the text box by **Extend license** and click **Submit**.
 - From the API, run PUT /system/trial-ext, JSON Payload { "token" : "token_value" } For example:

```
curl -X PUT -H "Content-Type: application/json" \
-H "X-Auth-Token:<paiv-token>" \
-d '{"token": "<token-from-ibm>"}' \
https://<paiv-host>:<paiv-port>/powerai-vision[-release-name]/api/system/trial-ext
```

Model functionality

You can train these types of models in PowerAI Vision.

Table 3. Types of models and model functionality

Model type	GoogLeNet	Faster R-CNN	tiny YOLO V2	Detectron	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 to deploy to.	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 video action detection	Imported model used for training.
Image classification	Yes	No	No	No	No	Yes
Object detection	No	Yes	Yes	Yes	No	Yes
Action detection	No	No	No	No	Yes	No
Model processing						
Import / Export model	Yes	Yes	Yes	Yes	Yes	Yes
Deploy to CPU (Standard install)	No	No	Yes	No	No	No
Deploy multiple models to one GPU	Yes	Yes	No	No	No	No
Use for transfer learning	Yes	Yes	No	No	No	No
Supported on Inference Server	Yes (default)	Yes (default)	Yes	Yes	No	Yes - TensorFlow and Keras
Deploy to CPU (Inference Server)	Yes	Yes	Yes	No	No	No
Model type	GoogLeNet	Faster R-CNN	tiny YOLO V2	Detectron	Structured segment network (SSN)	Custom model

PowerAI Vision concepts

PowerAI Vision 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 PowerAI Vision:

Data set

A data set is a collection of images and videos that you uploaded to PowerAI Vision. 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).

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.

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.

Planning for PowerAI Vision

You must meet the software and hardware requirements and understand the supported file types before you can install PowerAI Vision.

- “Hardware requirements”
- “Software requirements”
- “Networking requirements”
- “Disk space requirements” on page 16
- “Supported web browsers” on page 16
- “Image support” on page 16
- “Supported video types” on page 17
- “Deep learning frameworks” on page 17
- “Limitations” on page 17

Hardware requirements

PowerAI Vision requires the following hardware:

- POWER8 S822LC (8335-GTB) or POWER9 AC922 with at least one NVIDIA NVLink capable GPU
- 128 GB of memory
- Ethernet network interface
- 40 GB of storage. See “Disk space requirements” on page 16 for details.
- If **Optimized for speed (tiny YOLO v2)** is selected when training the model, 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 PowerAI Vision:

Linux

- Red Hat Enterprise Linux (RHEL) 7.6 (little endian).
- Ubuntu 18.04 or later.

NVIDIA CUDA

10.1 Update 1 or later drivers. For information, see the NVIDIA CUDA Toolkit website.

Docker

- Docker must be installed. The recommended version is 1.13.1 or later. Version 1.13.1 is installed with RHEL 7.6.
- 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, Docker 0 must be in a trusted firewall zone. If it is not in a trusted firewall zone, modify the RHEL settings as follows:

```
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

PowerAI Vision has the following storage requirements for the initial product installation and for the data sets that will be managed by the product.

Standalone installation

- `/var` - The product installation requires at least 25 Gb of space in the `/var` file system for the product Docker images. PowerAI Vision also generates log information in this file system.

Recommendation: If you want to minimize the root (`/`) file system, make sure that `/var` has its own volume. The `/var` file system should have at least 50 Gb of space, more if additional applications are being run on the system that use this file system for log data and so on.

- `/opt` - PowerAI Vision data sets are stored in this file system. The storage needs will vary depending on the data sets and the contents - i.e., 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.

IBM Cloud Private installation

The PowerAI Vision product will use the configured persistent storage for the deployment, the requirements are documented in [Installing PowerAI Vision with IBM Cloud Private](#).

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

- Images with COCO annotations are supported. For details, see “Importing images with COCO annotations” on page 60.
- PowerAI Vision 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 PowerAI Vision can import the annotations successfully.

- The models used by PowerAI Vision 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.
 - High resolution images will be scaled to a maximum of 1000 x 600 pixels.
 - For image classification, images are scaled to 224 x 224 pixels.
 - For object detection with Detectron, all images are scaled to 1333 x 800 pixels. However, the original aspect ratio is maintained. If necessary, black bands are added to the image to make it fit.
 - For object detection with tiny YOLO V2, all images are scaled to 416 x 416. The longest edge is scaled to 416 pixels and, if necessary, black bands are added to the shorter side to make it 416 pixels.
 - For object detection with FR-CNN and video action detection, anything over 1000 x 600 pixels is down-sampled so that the longest edge will fit. However, the original aspect ratio is maintained. If necessary, black bands are added to the image to make it fit.
 - 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.

Supported video types

The following video formats are supported:

Can be played in the PowerAI Vision 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 PowerAI Vision.

Table 4. Included frameworks

Framework	Version	Python 2.7 support	Python3.6 support	Notes
Caffe 2	1.0.0	Yes	No	Supported for Detectron models
IBM Caffe	1.0	Yes	No	Supported for GoogLeNet, Faster R-CNN, and tiny YOLO V2 models
Keras	2.2.4	Yes	Yes	Supported for custom models
TensorFlow	1.14	Yes	Yes	Supported for custom models

Limitations

Following are some limitations for IBM PowerAI Vision 1.1.4:

- PowerAI Vision 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 Power[®] System 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 PowerAI Vision stand-alone on a system that already has any of these products installed:
 - IBM Data Science Experience (DSX)
 - IBM Cloud Private
 - Any other Kubernetes based applications

License Management in IBM License Metric Tool

The IBM PowerAI Vision 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 PowerAI Vision Training and Inference* or *IBM PowerAI Vision Inference for Servers*

- **PersistentId**

Unique identifier of the software identity. For IBM PowerAI Vision 1.1.4, the assigned **PersistentId** is:

- **IBM PowerAI Vision Training and Inference** - ebb8d2e1bd62488c8c196f568857ae38
- **IBM PowerAI Vision Inference for Servers** - 297aaa94baa441e0ad91a609b24083b7

- **InstanceId**

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

Metrics information

IBM PowerAI Vision 1.1.4 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.

Installing, upgrading, and uninstalling PowerAI Vision

Use the information in these topics to work with the product installation. You can install PowerAI Vision by using the command line (stand-alone) or by using IBM Cloud Private.

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

After installing PowerAI Vision, you can optionally change the SSL certificate by following the steps in this topic: “Installing a new SSL certificate in PowerAI Vision stand-alone” on page 105.

Prerequisites for installing PowerAI Vision

Before you can install either PowerAI Vision stand-alone or PowerAI Vision with IBM Cloud Private, 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 PowerAI nor IBM Watson[®] Machine Learning Accelerator are required for running PowerAI Vision.

- “Red Hat Enterprise Linux operating system and repository setup”
- “Ubuntu operating system and repository setup” on page 22
- “NVIDIA Components: IBM POWER9 specific udev rules (Red Hat only)” on page 22
- “Install the GPU driver (Red Hat)” on page 23
- “Installing the GPU driver” on page 24
- “Verify the GPU driver” on page 24
- “Installing docker, nvidia-docker2 on Ubuntu ” on page 25

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 (Inference Server only):

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

2. Install packages needed for the installation.

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

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

```
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:


```
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 containers can use NVIDIA GPUs. For instructions, see Using nvidia-docker 2.0 with RHEL 7.

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. Load the latest kernel


```
sudo apt-get install linux-headers-$(uname -r)
sudo reboot
```
3. Or 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)'  
rpm -qa | egrep '(cuda|nvidia).*(396|410)\.'
```

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
```

5. Reboot the system to unload the GPU driver

```
sudo shutdown -r now
```

Install the GPU driver (Red Hat)

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 [NVIDIA Driver Download](#).
- Select Product Type: **Tesla**.
- Select Product Series: **P-Series** (for Tesla P100) or **V-Series** (for Tesla V100).
- Select Product: **Tesla P100** or **Tesla V100**.
- Select Operating System: **Linux POWER LE RHEL 7** for POWER or **Linux 64-bit RHEL7** for x86 (Inference Server only), depending on your cluster architecture. Click **Show all Operating Systems** if your version is not available.
- Select CUDA Toolkit: **10.1**
- Click **SEARCH** to go to the download link.
- Click **Download** to download the driver.

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.

```
sudo rpm -ivh nvidia-driver-local-repo-rhel7-418.*.rpm  
sudo yum install nvidia-driver-latest-dkms
```

3. Set nvidia-persistenced to start at boot

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

4. Restart to activate the driver.

Installing the GPU driver

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 NVIDIA Driver Download.
 - Select Product Type: **Tesla**
 - Select Product Series: **P-Series** (for Tesla P100) or **V-Series** (for Tesla V100).
 - Select Product: **Tesla P100** or **Tesla V100**.
 - Select Operating System: **Linux POWER LE Ubuntu 18.04** for POWER or **Linux 64-bit Ubuntu 18.04** for x86, depending on your cluster architecture. Click **Show all Operating Systems** if your version is not available.
 - Select CUDA Toolkit: **10.1**
 - Click **SEARCH** to go to the download link.
 - Click **Download** to download the driver.
2. The driver file name is `NVIDIA-Linux-ppc64le-418.87.01.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. Ensure the kernel headers are installed and match the running kernel. Compare the outputs of:

```
dpkg -l | grep linux-headers kernel-devel 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 downlevel packages.
 - Reboot the system if the packages are newer than the active kernel.
4. Install the GPU driver repository and cuda-drivers:

```
sudo dpkg -i nvidia-driver-local-repo-ubuntu1804-418.*.deb
sudo apt-get update
sudo apt-get install cuda-drivers
```
 5. Set `nvidia-persistenced` to start at boot

```
sudo systemctl enable nvidia-persistenced
```
 6. 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    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      |             43%      Default      |
| N/A   50C    P0   109W / 300W | 2618MiB / 16280MiB |             |             |
+-----+-----+-----+
|  1   Tesla P100-SXM2...  On      | 00000003:01:00:0  Off      |             0%      Default      |
| N/A   34C    P0    34W / 300W |    0MiB / 16280MiB |             |             |
+-----+-----+-----+
|  2   Tesla P100-SXM2...  On      | 0000000A:01:00:0  Off      |             0%      Default      |
| N/A   48C    P0    44W / 300W | 5007MiB / 16280MiB |             |             |
+-----+-----+-----+
|  3   Tesla P100-SXM2...  On      | 0000000B:01:00:0  Off      |             0%      Default      |
| N/A   36C    P0    33W / 300W |    0MiB / 16280MiB |             |             |
+-----+-----+-----+

Processes:
+-----+-----+-----+
| GPU  PID  Type  Process name                      GPU Memory Usage |
+-----+-----+-----+
|  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 49.

Installing docker, nvidia-docker2 on Ubuntu

Use these steps in 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 (Inference server only)

```
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
```

Note:

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

Installing PowerAI Vision stand-alone

You use the command line to install PowerAI Vision stand-alone.

PowerAI Vision stand-alone installation prerequisites

You must complete the following installation prerequisites steps before you install PowerAI Vision.

1. Complete all steps in the “Prerequisites for installing PowerAI Vision” 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.

Attention: You cannot install PowerAI Vision stand-alone on the same system that has the following software installed:

- IBM Data Science Experience (DSX)
- IBM Cloud Private
- Any other Kubernetes based applications
- “Install PowerAI Vision from IBM Passport Advantage”
- “Install PowerAI Vision from AAS ” on page 27
- “Install PowerAI Vision trial mode” on page 28

Install PowerAI Vision from IBM Passport Advantage

To install PowerAI Vision stand-alone, complete the following steps:

1. Download the product tar file from the IBM Passport Advantage website.
2. Decompress 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
```

3. From the directory that contains the downloaded tar file, run the appropriate script as root or with sudo privileges:

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

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

4. (RHEL only) Open ports for the firewall to access PowerAI Vision by running this script:

```
sudo /opt/powerai-vision/sbin/firewall.sh
```

5. After the installation is complete, you can start PowerAI Vision by running this script:

```
sudo /opt/powerai-vision/bin/powerai_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 103.

You must read and accept the license agreement that is displayed before you can use PowerAI Vision.

It can take several minutes to start PowerAI Vision. To check the status of the startup process, run this script:

```
sudo /opt/powerai-vision/bin/helm.sh status vision
```

In the output from the **helm.sh status vision** script, you can verify which PowerAI Vision 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 **helm.sh status vision** script that shows all components are available:

```

==> v1beta1/Deployment
NAME                DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
powerai-vision-mongodb      1        1        1            1          3h
powerai-vision-portal       1        1        1            1          3h
powerai-vision-postgres     1        1        1            1          3h
powerai-vision-taskanalyzer 1        1        1            1          3h
powerai-vision-ui           1        1        1            1          3h
powerai-vision-video-nginx  1        1        1            1          3h
powerai-vision-video-portal 1        1        1            1          3h
powerai-vision-video-rabmq  1        1        1            1          3h
powerai-vision-video-redis  1        1        1            1          3h
powerai-vision-video-test-nginx 1        1        1            1          3h
powerai-vision-video-test-portal 1        1        1            1          3h
powerai-vision-video-test-rabmq 1        1        1            1          3h
powerai-vision-video-test-redis 1        1        1            1          3h

```

6. Install any available fix packs. For instructions see “Getting fixes from Fix Central” on page 139.

Install PowerAI Vision from AAS

1. Download the product tar.gz file from Advanced Administration System (AAS). This system is also called Entitled Software Support (ESS).
2. Unzip and untar the tar.gz file by running this command. The install files are extracted to `powerai-vision-aas-1.1.4.1/`.

```
gunzip -c file_name.tar.gz | tar -xvf
```
3. Decompress 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
```

4. From the directory that contains the extracted tar file, run this script as root or with sudo privileges:

```
sudo /opt/powerai-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 PowerAI Vision by running this script:

```
sudo /opt/powerai-vision/sbin/firewall.sh
```
6. After the installation is complete, you can start PowerAI Vision by running this script:

```
sudo /opt/powerai-vision/bin/powerai_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 103.

You must read and accept the license agreement that is displayed before you can use PowerAI Vision. It can take several minutes to start PowerAI Vision. To check the status of the startup process, run this script:

```
sudo /opt/powerai-vision/bin/helm.sh status vision
```

In the output from the `helm.sh status vision` script, you can verify which PowerAI Vision 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 `helm.sh status vision` script that shows all components are available:

```

==> v1beta1/Deployment
NAME                DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
powerai-vision-mongodb      1        1        1            1          3h
powerai-vision-portal       1        1        1            1          3h
powerai-vision-postgres     1        1        1            1          3h
powerai-vision-taskanalyzer 1        1        1            1          3h
powerai-vision-ui           1        1        1            1          3h
powerai-vision-video-nginx  1        1        1            1          3h
powerai-vision-video-portal 1        1        1            1          3h
powerai-vision-video-rabmq  1        1        1            1          3h
powerai-vision-video-redis  1        1        1            1          3h
powerai-vision-video-test-nginx 1        1        1            1          3h
powerai-vision-video-test-portal 1        1        1            1          3h
powerai-vision-video-test-rabmq 1        1        1            1          3h
powerai-vision-video-test-redis 1        1        1            1          3h

```

7. Install any available fix packs. For instructions see “Getting fixes from Fix Central” on page 139.

Install PowerAI Vision trial mode

1. Go to PowerAI Vision Trial download site. Download the .tar file and the .rpm files as instructed.
2. Decompress 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
```

3. From the directory that contains the downloaded tar file, run the appropriate script as root or with sudo privileges:

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

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

4. (RHEL only) Open ports for the firewall to access PowerAI Vision by running this script:

```
sudo /opt/powerai-vision/sbin/firewall.sh
```

5. After the installation is complete, you can start PowerAI Vision by running this script:

```
sudo /opt/powerai-vision/bin/powerai_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 103.

You must read and accept the license agreement that is displayed before you can use PowerAI Vision.

It can take several minutes to start PowerAI Vision. To check the status of the startup process, run this script:

```
sudo /opt/powerai-vision/bin/helm.sh status vision
```

In the output from the **helm.sh status vision** script, you can verify which PowerAI Vision 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 **helm.sh status vision** script that shows all components are available:

```

==> v1beta1/Deployment
NAME                                DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
powerai-vision-mongodb              1        1        1            1          3h
powerai-vision-portal                1        1        1            1          3h
powerai-vision-postgres              1        1        1            1          3h
powerai-vision-taskanalyzer          1        1        1            1          3h
powerai-vision-ui                    1        1        1            1          3h
powerai-vision-video-nginx           1        1        1            1          3h
powerai-vision-video-portal          1        1        1            1          3h
powerai-vision-video-rabmq           1        1        1            1          3h
powerai-vision-video-redis           1        1        1            1          3h
powerai-vision-video-test-nginx      1        1        1            1          3h
powerai-vision-video-test-portal     1        1        1            1          3h
powerai-vision-video-test-rabmq      1        1        1            1          3h
powerai-vision-video-test-redis      1        1        1            1          3h

```

Related concepts:

“Logging in to PowerAI Vision” on page 51
 Follow these steps to log in to PowerAI Vision.

Installing PowerAI Vision with IBM Cloud Private

If you have more than one IBM Power Systems server available, you can use IBM Cloud Private 3.1.0 to install a single instance of PowerAI Vision that has access to all the Power Systems GPUs across the entire cluster.

If you have only a single IBM Power Systems server and do not have an existing IBM Cloud Private environment, you should use the PowerAI Vision stand-alone process. For more information, see the “Installing PowerAI Vision stand-alone” on page 25 topic.

To install PowerAI Vision with IBM Cloud Private, complete the following steps:

Notes:

- If IBM Cloud Private is already installed and configured in your environment, you can go to step 4.
 - The links to IBM Cloud Private go to the 3.1.0 Knowledge Center. To go to a different version, click the link, then click **Change version**.
1. Install IBM Cloud Private. For more information, see the Installing IBM Cloud Private topic.
 2. Install the IBM Cloud CLI. For more information, see the Install IBM Cloud CLI topic.
 3. Authenticate to your master node in your IBM Cloud Private environment. For more information, see the Configuring authentication for the Docker CLI topic.

To log in to the IBM Cloud Private cluster, run the following command:

```
cloudctl login -a https://<cluster-domain-name>:8443/ --skip-ssl-validation
```

4. Set up your system to install your IBM Cloud Private deployment into a non-default namespace. It is recommended that you do not install into the default namespace for security reasons.

Important: Install each distinct deployment of IBM Cloud Private into a unique namespace.

- a. Create an appropriate ClusterRoleBinding to enable PowerAI Vision to query Kubernetes. To create this, copy the below text into a crb.yaml file, where *CustomNamespace* is your custom namespace name:

```

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: CustomNamespace-crb
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin

```

```
subjects:
-
  kind: ServiceAccount
  name: default
  namespace: CustomNamespace
```

b. Run the following command:

```
kubectl create -f crb.yaml
```

5. Download the appropriate tar file from IBM Passport Advantage.
6. Untar the `powerai-vision-ppa-1.1.4.0.tar` tar file. It contains install packages for the standalone product, as well as the tar file with the containers that must be loaded for the IBM Cloud Private installation.
7. To make PowerAI Vision available in IBM Cloud Private catalog, run the following command:
`cloudctl catalog load-archive --archive file_name.tar --registry <icp full host name>:8500/<namespace>`

Where:

--registry <value>

Lets you specify the docker registry that the images will be pushed to.

Example:

```
mycluster-icp:8500/<namespace>
```

--clustername <cluster_CA_domain>

Lets you specify the certificate authority (CA) domain. If you did not specify a CA domain, the default value is `mycluster.icp`.

8. Review the Chart README for PowerAI Vision carefully. It documents prerequisites, requirements, and limitations of PowerAI Vision in IBM Cloud Private.
9. Verify that you have a minimum of 40 GB of persistent storage. If your IBM Cloud Private installation has dynamic provisioned storage, you can use it for your 40 GB of persistent storage. To manually create persistent volumes in IBM Cloud Private, see the [Creating a Persistent Volume](#) topic. After you create the persistent volume, you must make the volume sharable across all nodes in the cluster.

Note: Do not use `HostPath` for the persistent storage unless you have only one node in your cluster. See [Creating a Persistent Volume](#) in the IBM Cloud Private documentation for details.

10. To install PowerAI Vision from the IBM Cloud Private catalog, from the navigation menu select **Catalog > Helm Charts**.
11. In the search box, enter `vision` and click **powerai-vision**. Review the information.
12. Click **Configure** and enter information for the **Release name** and the **Namespace** fields. The default user name is `admin` and the default password is `passwd0rd`. For instructions to change these values, see [“Managing users”](#) on page 103. For information about namespaces, see [Namespaces](#) in the IBM Cloud Private Knowledge Center.
13. Click **Install**.
14. For information about accessing PowerAI Vision, see [Logging into PowerAI Vision](#).

Important: NFS volumes should have the `“no_root_squash”` flag set in `/etc/exports`:

```
/var/nfs *(rw,no_root_squash,no_subtree_check)
```

Upgrading PowerAI Vision

When upgrading to the latest version of PowerAI Vision, 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 will need to delete and redeploy any deployed models after upgrading.

- Upgrade the stand-alone version
- Upgrade PowerAI Vision with IBM Cloud Private

Upgrade the stand-alone version

1. Stop the current instance of PowerAI Vision by running the following script:

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

2. Obtain and install PowerAI Vision:

Install PowerAI Vision from IBM Passport Advantage

- a. Download the product tar file from the IBM Passport Advantage website.
- b. Decompress 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.

- c. From the directory that contains the downloaded tar file, run the appropriate script as root or with sudo privileges:

```
sudo /opt/powerai-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 PowerAI Vision by running this script:

```
sudo /opt/powerai-vision/bin/powerai_vision_start.sh
```

A user named admin is created with a password of passw0rd. Users will be preserved from the previous installation on upgrade. For instructions to manage existing users, and to learn how to create new users, see “Managing users” on page 103.

- e. Install any available fix packs. For instructions see “Getting fixes from Fix Central” on page 139.

Install PowerAI Vision from AAS

- a. Download the product tar.gz file from Advanced Administration System (AAS). This system is also called Entitled Software Support (ESS).
- b. Unzip and untar the tar.gz file by running this command. The install files are extracted to powerai-vision-aas-1.1.4.1/.

```
gunzip -c file_name.tar.gz | tar -xvf
```

- c. Decompress 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
```

- d. From the directory that contains the extracted tar file, run this script as root or with sudo privileges:

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

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

- e. After the installation is complete, you can start PowerAI Vision by running this script:

```
sudo /opt/powerai-vision/bin/powerai_vision_start.sh
```

A user named `admin` is created with a password of `passwd`. Users will be preserved from the previous installation on upgrade. For instructions to manage existing users, and to learn how to create new users, see “Managing users” on page 103.

- f. Install any available fix packs. For instructions see “Getting fixes from Fix Central” on page 139.
3. Delete any deployed GoogLeNet (Classification) and FR-CNN (object detection) models deployed in the previous version of the product to allow redeployment and GPU resource sharing. Until they are deleted, the models will attempt to deploy and will fail. To delete a deployed model, click **Deployed Models**. Next, select the model that you want to delete and click **Delete**. The trained model is not deleted from PowerAI Vision.
4. 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.

Upgrade PowerAI Vision with IBM Cloud Private

1. Download the product tar file from the IBM Passport Advantage website.
2. To make PowerAI Vision available in IBM Cloud Private catalog, run the following command:
`cloudctl catalog load-archive --archive file_name.tar --registry <icp full host name>:8500/<namespace>`

Where:

--registry <value>

Lets you specify the docker registry that the images will be pushed to.

Example:

`mycluster-icp:8500/<namespace>`

--clustername <cluster_CA_domain>

Lets you specify the certificate authority (CA) domain. If you did not specify a CA domain, the default value is `mycluster.icp`.

3. Navigate to your Helm Release. Click **Upgrade** and the upgrade to the new PowerAI Vision images starts.

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

4. As part of the upgrade process, PowerAI Vision is restarted and a user named `admin` is created with a password of `passwd`. Users will be preserved from the previous installation on upgrade. For instructions to manage existing users, and to learn how to create new users, see “Managing users” on page 103.

Uninstalling PowerAI Vision stand-alone

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

To uninstall PowerAI Vision, complete the following steps:

Note: 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 PowerAI Vision by running the following script:
`sudo /opt/powerai-vision/bin/powerai_vision_stop.sh`
2. Remove previously installed images by running the following script:
`sudo /opt/powerai-vision/bin/purge_image.sh 1.1.4.0`
Optionally remove all product data by running the following script. This will remove data sets, models, and so on:
`sudo /opt/powerai-vision/bin/purge_data.sh`
3. Remove PowerAI Vision by running the following command:
 - For RHEL:
`sudo yum remove powerai-vision`
 - For Ubuntu:
`sudo dpkg --remove powerai-vision`
4. Delete the data directory by running the following command:
`sudo rm -rf /opt/powerai-vision/`
5. Verify that PowerAI Vision was uninstalled by running the following command:
 - For RHEL:
`rpm -q powerai-vision`
 - For Ubuntu:
`dpkg -l powerai-vision`

Checking the application and environment

After installation of PowerAI Vision, you can check the status of the application and environment by using commands documented in these topics. The Kubernetes commands `helm.sh` and `kubectl.sh` are installed in the `bin` directory of the product install path. (default: `/opt/powerai-vision`).

Checking the application Docker images in standalone installation

Space limitations or Kubernetes garbage collection activities can result in PowerAI Vision Docker images not being available in the Docker repository on a system.

- “Using docker images to validate PowerAI Vision Docker image availability”
- “Loading missing images” on page 36

Using docker images to validate PowerAI Vision Docker image availability

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

```
# /opt/powerai-vision/bin/load_images.sh -f ./powerai-vision-1.1.4.0-ppa/powerai-vision-images-1.1.4.0.tar
[ INFO ] Waiting for docker loads to complete. This will take some time...
Loaded image: gcr.io/google_containers/hyperkube:v1.13.0
Loaded image: quay.io/kubernetes-ingress-controller/nginx-ingress-controller-ppc64le:0.20.0
Loaded image: gcr.io/google_containers/etcd:3.3.10
Loaded image: nvidia/k8s-device-plugin:1.11
Loaded image: coredns/coredns:1.2.6
Loaded image: gcr.io/google_containers/pause:3.1
Loaded image: powerai-vision-tiller:2.12.0
Loaded image: powerai-vision-service:1.1.4.0
Loaded image: powerai-vision-video-rabbitmq:1.1.4.0
Loaded image: powerai-vision-ui:1.1.4.0
Loaded image: postgres:9.6.8
Loaded image: powerai-vision-mongodb:1.1.4.0
Loaded image: powerai-vision-dnn-detectron:1.1.4.0
Loaded image: powerai-vision-dnn-actiondetect:1.1.4.0
Loaded image: powerai-vision-video-redis:1.1.4.0
Loaded image: powerai-vision-models:1.1.4.0
Loaded image: powerai-vision-keycloak:1.1.4.0
Loaded image: powerai-vision-logstash:1.1.4.0
Loaded image: powerai-vision-dnn-custom:1.1.4.0
Loaded image: powerai-vision-video-portal:1.1.4.0
Loaded image: powerai-vision-video-nginx:1.1.4.0
Loaded image: powerai-vision-video-test-nginx:1.1.4.0
Loaded image: powerai-vision-fpga-device-plugin:1.1.4.0
Loaded image: powerai-vision-elasticsearch:1.1.4.0
Loaded image: powerai-vision-taskanalay:1.1.4.0
Loaded image: powerai-vision-dnn-edge:1.1.4.0
Loaded image: powerai-vision-dnn-microservices:1.1.4.0
Loaded image: powerai-vision-preprocessing:1.1.4.0
Loaded image: powerai-vision-video-test-portal:1.1.4.0
Loaded image: powerai-vision-usermgt:1.1.4.0

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

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

```
$ sudo docker images
[sudo] password for ubuntu:
REPOSITORY                                TAG                                IMAGE ID                                CREATED
powerai-vision-video-nginx                 1.1.4.0                            a837f9375131                           5 days ago
powerai-vision-video-portal                 1.1.4.0                            e73d34f5d57e                           5 days ago
powerai-vision-video-test-nginx            1.1.4.0                            25ed23a446ca                           5 days ago
```

powerai-vision-video-test-portal	1.1.4.0	ee208f689e4e	5 days a
powerai-vision-service	1.1.4.0	95a23655b40c	5 days a
powerai-vision-mongodb	1.1.4.0	040c95712d72	5 days a
powerai-vision-models	1.1.4.0	3380a4fa7c0b	5 days a
powerai-vision-dnn-edge	1.1.4.0	0550b5fe0634	5 days a
powerai-vision-taskanally	1.1.4.0	6575b7c0c7cf	5 days a
powerai-vision-dnn-custom	1.1.4.0	0b9379cdf6e	5 days a
powerai-vision-preprocessing	1.1.4.0	d08fd56f605f	5 days a
powerai-vision-dnn-microservices	1.1.4.0	e3b1dcbe955e	5 days a
powerai-vision-dnn-actiondetect	1.1.4.0	abaf81d4fe3f	5 days a
powerai-vision-dnn-detectron	1.1.4.0	235dc16b1caa	5 days a
powerai-vision-ui	1.1.4.0	d385852cb385	7 days a
powerai-vision-keycloak	1.1.4.0	8385768601b6	7 days a
powerai-vision-logstash	1.1.4.0	4b22b8369519	8 days a
powerai-vision-fpga-device-plugin	1.1.4.0	aacc3fbd9987	8 days a
powerai-vision-elasticsearch	1.1.4.0	01209c4c42d7	8 days a
powerai-vision-usermgt	1.1.4.0	a9779d215b4f	8 days a
powerai-vision-video-rabbitmq	1.1.4.0	048be6e8ee7f	8 days a
powerai-vision-video-redis	1.1.4.0	23a2ae7f8efd	8 days a
powerai-vision-tiller	2.12.0	8a925bb46988	6 months
gcr.io/google_containers/hyperkube	v1.13.0	02fc0a6f63cc	6 months
gcr.io/google_containers/etcd	3.3.10	af050d2caadf	6 months
coredns/coredns	1.2.6	9dda08c8b15f	7 months
quay.io/kubernetes-ingress-controller/nginx-ingress-controller-ppc64le	0.20.0	9aad57947cb	8 months
nvdiia/k8s-device-plugin	1.11	5c02dafa13ad	10 month
postgres	9.6.8	7bf8f906163b	13 month
gcr.io/google_containers/pause	3.1	1652adb11bd5	18 month
k8s.gcr.io/pause	3.1	1652adb11bd5	18 month

Loading missing images

If any of the PowerAI Vision 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 the application status in an ICP installation

Before you can use the `kubectl` commands to check the application status, you must log in to the IBM Cloud Private cluster.

```
cloudctl login -a https://<cluster-domain-name>:8443/ --skip-ssl-validation
```

Example

In the following example, `cloudctl` is used to log in to the IBM Cloud Private cluster `icp1` with master node `icp1.domain.com` as the user `admin`, to access the default namespace where the PowerAI Vision application is installed:

```
# cloudctl login -a https://icp1.domain.com:8443 --skip-ssl-validation -u admin
```

```
Password>
Authenticating...
OK
```

```
Targeted account icp1 Account (id-icp1-account)
```

```
Select a namespace:
1. cert-manager
2. default
3. ibmcom
4. istio-system
5. kube-public
6. kube-system
7. platform
8. services
9. vision
```

```

Enter a number> 2
Targeted namespace default

Configuring kubectl ...
Property "clusters.icpl" unset.
Property "users.icpl-user" unset.
Property "contexts.icpl-context" unset.
Cluster "icpl" set.
User "icpl-user" set.
Context "icpl-context" created.
Switched to context "icpl-context".
OK

Configuring helm: /root/.helm
OK
#

```

Now the `kubectl` commands can be used to similar to the way they are used in the standalone environment.

Checking Kubernetes services status

The Kubernetes infrastructure is used to run the PowerAI Vision 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”
- “Using `kubectl describe pods` to check kube-system”

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 PowerAI Vision application.

Example output

```

# /opt/powerai-vision/bin/kubectl.sh get pods --namespace kube-system
NAME                                READY   STATUS    RESTARTS   AGE
coredns-76f484447b-9sqwz            1/1    Running   0           3d4h
nginx-ingress-lb-ppc641e-hmtg5      1/1    Running   0           3d4h
nvidia-device-plugin-daemonset-wd1k1 1/1    Running   0           3d4h
tiller-deploy-7f65888dc8-kcglg      1/1    Running   0           3d4h

```

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 `kubectl describe pods` to check kube-system

The `kubectl 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 `kubectl 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:

```

# /opt/powerai-vision/bin/kubect1.sh describe pods --namespace kube-system
Name:          coredns-76f484447b-9sqwz
Namespace:    kube-system
Node:         127.0.0.1/127.0.0.1
Start Time:   Tue, 12 Mar 2019 07:44:34 -0500
Labels:       k8s-app=kube-dns
              pod-template-hash=76f484447b
Annotations:  <none>
Status:       Running
IP:           172.17.0.2
Controlled By: ReplicaSet/coredns-76f484447b
Containers:
  coredns:
    Container ID:  docker://e94399e73b84c4fe55f54807c9bdfdcacdafcab27fa2f746421bfd5ba9443e175
    Image:         coredns/coredns:1.2.6
    Image ID:     docker-pullable://coredns/coredns@sha256:81936728011c0df9404cb70b95c17bbc8af922ec9a70d0561a5d01fefaf6ffa51
    Ports:        53/UDP, 53/TCP, 9153/TCP
    Host Ports:   0/UDP, 0/TCP, 0/TCP
    Args:         -conf /etc/coredns/Corefile
    State:        Running
      Started:    Tue, 12 Mar 2019 07:44:44 -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
    Environment:  <none>
    Mounts:
      /etc/coredns from config-volume (ro) /var/run/secrets/kubernetes.io/serviceaccount from default-token-wgppqf (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-wgppqf:
    Type:          Secret (a volume populated by a Secret)
    SecretName:    default-token-wgppqf
    Optional:      false
QoS Class:       Burstable
Node-Selectors:  beta.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.sh get pods”
- “kubectl describe nodes command”
- “kubectl describe pods command” on page 41

kubectl.sh get pods

The `kubectl` command is used to show the detailed status of the Kubernetes pods deployed to run the PowerAI Vision application.

Example output

```
# /opt/powerai-vision/bin/kubectl.sh get pods
NAME                                READY  STATUS   RESTARTS  AGE
powerai-vision-dnn-infer-bb06a6f3-75d0-4614-96fb-a3d69f79dvhxjh  1/1    Running  0          11h
powerai-vision-elasticsearch-855648d6d4-7jwct                    1/1    Running  0          5d2h
powerai-vision-fpga-device-plugin-qjhsd                          1/1    Running  0          5d2h
powerai-vision-keycloak-84cfc79796-z15qk                        1/1    Running  0          5d2h
powerai-vision-logstash-7b48f978d7-pwscv                         1/1    Running  0          5d2h
powerai-vision-mongodb-57bcd46cdd-vqs2b                          1/1    Running  0          5d2h
powerai-vision-postgres-795987c56c-w6x62                       1/1    Running  0          5d2h
powerai-vision-service-5588ffdfc-cnq8h                          1/1    Running  0          5d2h
powerai-vision-taskanaly-6448c56f89-918mf                       1/1    Running  0          5d2h
powerai-vision-ui-5f78b785cf-xxfm6                               1/1    Running  0          5d2h
powerai-vision-video-nginx-799bc6d9f6-6c27r                     1/1    Running  0          5d2h
powerai-vision-video-portal-c9b6cd996-qrj68                     1/1    Running  0          5d2h
powerai-vision-video-rabmq-6b8dcfd69b-w5wqb                     1/1    Running  0          5d2h
powerai-vision-video-redis-6d45598b4-5688j                      1/1    Running  0          5d2h
powerai-vision-video-test-nginx-7bbc9ddcfc-6xhwm                1/1    Running  0          5d2h
powerai-vision-video-test-portal-84f7b7f55b-n7jxv              1/1    Running  0          5d2h
powerai-vision-video-test-rabmq-59f59cc9b-4rbfh                 1/1    Running  0          5d2h
powerai-vision-video-test-redis-5c78bb7f48-ffg9q                1/1    Running  0          5d2h
```

Interpreting the output

- When the application is running correctly, each of the pods should have:
 - A value of 1/1 in the READY column
 - A value of Running in the STATUS column
- In the above example output, pods with `infer` in the name are created when a model is deployed. These will only appear if there are models deployed in the instance of the application running on the system.
- A STATUS value other than Running indicates an issue with the pod.
- A non-0 and increasing value in the RESTARTS column indicates an issue with that pod.

If there are indications of issues with pods, see “Troubleshooting known issues - PowerAI Vision standard install” on page 123.

kubectl describe nodes command

The `kubectl describe nodes` command provides status information regarding the Kubernetes environment used to run the PowerAI Vision application.

Example output

```

/opt/powerai-vision/bin/kubect1.sh describe nodes
Name: 127.0.0.1
Roles: <none>
Labels: beta.kubernetes.io/arch=ppc64le
        beta.kubernetes.io/os=linux
        kubernetes.io/hostname=127.0.0.1
Annotations: node.alpha.kubernetes.io/ttl: 0
              volumes.kubernetes.io/controller-managed-attach-detach: true
CreationTimestamp: Tue, 12 Mar 2019 07:44:29 -0500
Taints: <none>
Unschedulable: false
Conditions:

```

Type	Status	LastHeartbeatTime	LastTransitionTime	Reason	Message
MemoryPressure	False	Fri, 15 Mar 2019 12:08:05 -0500	Tue, 12 Mar 2019 07:44:28 -0500	KubeletHasSufficientMemory	kubelet has s
DiskPressure	False	Fri, 15 Mar 2019 12:08:05 -0500	Tue, 12 Mar 2019 07:44:28 -0500	KubeletHasNoDiskPressure	kubelet has m
PIDPressure	False	Fri, 15 Mar 2019 12:08:05 -0500	Tue, 12 Mar 2019 07:44:28 -0500	KubeletHasSufficientPID	kubelet has s
Ready	True	Fri, 15 Mar 2019 12:08:05 -0500	Tue, 12 Mar 2019 07:44:29 -0500	KubeletReady	kubelet is po

```

Addresses:
  InternalIP: 127.0.0.1
  Hostname: 127.0.0.1

```

```

Capacity:
  cpu: 160
  ephemeral-storage: 922396572Ki
  hugepages-16Gi: 0
  hugepages-16Mi: 0
  memory: 133784000Ki
  nvidia.com/gpu: 4
  pods: 500

```

```

Allocatable:
  cpu: 160
  ephemeral-storage: 850080679348
  hugepages-16Gi: 0
  hugepages-16Mi: 0
  memory: 133681600Ki
  nvidia.com/gpu: 4
  pods: 500

```

```

System Info:
  Machine ID: 3d6ff2f75c7d3ae927580249a28e7e05
  System UUID: 2101CAA
  Boot ID: a4952737-d779-43d8-ae75-8432ab041c00
  Kernel Version: 4.15.0-36-generic
  OS Image: Debian GNU/Linux 9 (stretch)
  Operating System: linux
  Architecture: ppc64le
  Container Runtime Version: docker://18.6.1
  Kubelet Version: v1.13.0
  Kube-Proxy Version: v1.13.0

```

```

Non-terminated Pods: (22 in total)

```

Namespace	Name	CPU Requests	CPU Limits	Memory Requests
default	powerai-vision-cod-infer-33f53f4e-b6d4-4476-bb19-c16c0e4c0sbtv6	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-cod-infer-b4d1e503-2f43-4652-9679-650b3ae1b4nkhp	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-dnn-infer-f5d2182a-2aae-496c-9688-3d1e7e3977pxr9	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-fpga-device-plugin-bg69p	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-keycloak-7df657794b-6v4pb	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-mongodb-6cdc4b654b-c7g99	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-portal-7fb5d5d66-6tk45	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-postgres-54d6dbdcf4-zp27c	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-taskanaly-54bf4f658f-b2hzw	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-ui-85494f77f7-9wg68	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-video-nginx-84f4dd84f6-k4tf2	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-video-portal-59678d77fb-f4qxv	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-video-rabmq-bb8f588c6-k9sps	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-video-redis-5dcf7f4b74-q6v86	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-video-test-nginx-7fb6ff6dd9-b7vz1	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-video-test-portal-5988b6d66-vpvvk	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-video-test-rabmq-7c55648476-d7154	0 (0%)	0 (0%)	0 (0%)
default	powerai-vision-video-test-redis-f64c589f8-rkzf7	0 (0%)	0 (0%)	0 (0%)
kube-system	coredns-76f484447b-9sqwz	100m (0%)	0 (0%)	70Mi (0%)
kube-system	nginx-ingress-lb-ppc64le-hmtg5	0 (0%)	0 (0%)	0 (0%)
kube-system	nvidia-device-plugin-daemonset-wd1kl	0 (0%)	0 (0%)	0 (0%)
kube-system	tiller-deploy-7f6588dc8-kcglg	0 (0%)	0 (0%)	0 (0%)

```

Allocated resources:

```

(Total limits may be over 100 percent, i.e., overcommitted.)

Resource	Requests	Limits
cpu	100m (0%)	0 (0%)
memory	70Mi (0%)	170Mi (0%)
ephemeral-storage	0 (0%)	0 (0%)
nvidia.com/gpu	3	3
Events:	<none>	

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 PowerAI Vision. 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	Message
OutOfDisk	False	[...]	[...]	KubeletHasSufficientDisk	kubelet has su
MemoryPressure	False	[...]	[...]	KubeletHasSufficientMemory	kubelet has su
DiskPressure	True	[...]	[...]	KubeletHasDiskPressure	kubelet has di
Ready	True	[...]	[...]	KubeletReady	kubelet is pos

- 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	

kubectl describe pods command

The `kubectl.sh describe pods` command provides detailed information about each of the pods used by the PowerAI Vision application. If the output from a specific pod is desired, the command `kubectl.sh describe pod podname`. To determine the values for `podname` look at the output from `kubectl.sh get pods`.

Example output

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

```

Name:          powerai-vision-ui-5f78b785cf-xxfm6
Namespace:    default
Node:         127.0.0.1/127.0.0.1
Start Time:   Thu, 13 Jun 2019 16:33:49 -0500
Labels:       app=powerai-vision
              chart=ibm-powerai-vision-prod-1.4.0
              component=powerai-vision-ui
              heritage=Tiller
              pod-template-hash=5f78b785cf
              release=vision
Annotations:  run=powerai-vision-ui-deployment-pod
              checksum/config: 5c015ae70c4e9169a83675f61c7068009abd3827c19f9f9eb0f677e8060a19e6
              productID: 5737-H10
              productName: IBM PowerAI Vision
              productVersion: 1.1.4.0
Status:       Running
IP:           172.17.0.18
Controlled By: ReplicaSet/powerai-vision-ui-5f78b785cf
Containers:
  powerai-vision-ui:
    Container ID:  docker://6c0be0c9ce6b499f776e861a2cbb3ede8b289a6ab0b5b7a2417ef20ea41ff946
    Image:          powerai-vision-ui:1.1.4.0
    Image ID:      docker://sha256:d385852cb38502584428725f98bcd3ab6e088c8c817bcf5e9b5047743ebabde0
    Port:          80/TCP
    Host Port:     0/TCP
    State:         Running
      Started:     Thu, 13 Jun 2019 16:34:08 -0500
    Ready:         True
    Restart Count: 0
    Liveness:      http-get http://:http/powerai-vision/index.html delay=240s timeout=5s period=10s #success=1 #failure=3
    Readiness:     http-get http://:http/powerai-vision/index.html delay=5s timeout=1s period=10s #success=1 #failure=3
    Environment:
      CONTEXT_ROOT:      <set to the key 'CONTEXT_ROOT' of config map 'powerai-vision-config'>      Optional: false
      DLAAS_API_SERVER:  <set to the key 'DLAAS_API_SERVER' of config map 'powerai-vision-config'>      Optional: false
      SERVER_HOST_VIDEO_TEST: <set to the key 'SERVER_HOST_VIDEO_TEST' of config map 'powerai-vision-config'> Optional: false
      SERVICE_PORT_VIDEO_TEST: <set to the key 'SERVICE_PORT_VIDEO_TEST' of config map 'powerai-vision-config'> Optional: false
      WEBROOT_VIDEO_TEST: <set to the key 'WEBROOT_VIDEO_TEST' of config map 'powerai-vision-config'> Optional: false
    Mounts:
      /opt/powerai-vision/data from data-mount (rw)
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-wn58h (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: powerai-vision-data-pvc
    ReadOnly:  false
  default-token-wn58h:
    Type:      Secret (a volume populated by a Secret)
    SecretName: default-token-wn58h
    Optional:  false
QoS Class:     BestEffort
Node-Selectors: beta.kubernetes.io/arch=ppc64le
Tolerations:   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 application pods include:

- Information about the product name and version are given in **productName** and **productVersion**.
- The **Status** 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.

Checking Kubernetes storage status

The PowerAI Vision 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”
- “Using the `kubectl describe pv` command”
- “Using the `kubectl describe pvc` command” on page 44

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/powerai-vision/bin/kubectl.sh get pv
NAME                CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM                                STORAGECLASS  REASON
powerai-vision-data  40Gi      RWX           Retain          Bound   default/powerai-vision-data-pvc
# /opt/powerai-vision/bin/kubectl.sh get pvc
NAME                STATUS    VOLUME          CAPACITY  ACCESS MODES  STORAGECLASS  AGE
powerai-vision-data-pvc  Bound    powerai-vision-data  40Gi      RWX           default/powerai-vision-data-pvc  48d
```

Interpreting the output

The above output shows information about the Persistent Volume and Persistent Volume Claim for PowerAI Vision. 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 `kubectl describe pv` command

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

Example output

```
# /opt/powerai-vision/bin/kubectl.sh describe pv
NAME                CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM                                STORAGECLASS  REASON
powerai-vision-data  40Gi      RWX           Retain          Bound   default/powerai-vision-data-pvc
[root@d1f01 ~]# /opt/powerai-vision/bin/kubectl.sh describe pv
Name:                powerai-vision-data
Labels:               assign-to=powerai-vision-data
                    type=local
Annotations:         pv.kubernetes.io/bound-by-controller=yes
StorageClass:
Status:              Bound
Claim:               default/powerai-vision-data-pvc
Reclaim Policy:     Retain
Access Modes:       RWX
Capacity:           40Gi
Message:
Source:
  Type:              HostPath (bare host directory volume)
  Path:              /opt/powerai-vision/volume/
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 kubectl describe pvc command

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

Example output

```
[root@d1f01 ~]# /opt/powerai-vision/bin/kubectl.sh describe pvc
Name:          powerai-vision-data-pvc
Namespace:     default
StorageClass:
Status:     Bound
Volume:    powerai-vision-data
Labels:        app=powerai-vision
                chart=ibm-powerai-vision-prod-1.1.0
                heritage=Tiller
                release=vision
Annotations:   pv.kubernetes.io/bind-completed=yes
                pv.kubernetes.io/bound-by-controller=yes
Capacity:     40Gi
Access Modes: RWX
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

PowerAI Vision processes require a Kubernetes environment. Use these commands to verify that the Kubernetes environment was deployed correctly and that all nodes are configured appropriately.

- “helm.sh”
- “kubectl get deployment” on page 46
- “kubectl describe deployment” on page 47

helm.sh

The `helm.sh` command shows the status of the full Kubernetes environment of the PowerAI Vision application.

Example output

```

/opt/powerai-vision/bin/helm.sh status vision
LAST DEPLOYED: Thu Jun 13 16:33:47 2019
NAMESPACE: default
STATUS: DEPLOYED

```

RESOURCES:

==> v1/PersistentVolumeClaim

NAME	STATUS	VOLUME	CAPACITY	ACCESS MODES	STORAGECLASS	AGE
powerai-vision-data-pvc	Bound	powerai-vision-data	40Gi	RWX	92d	

==> v1/Service

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
powerai-vision-elasticsearch	ClusterIP	10.10.0.170	<none>	9200/TCP,9300/TCP	19d
powerai-vision-keycloak	ClusterIP	10.10.0.50	<none>	8080/TCP,8443/TCP	92d
powerai-vision-logstash	ClusterIP	10.10.0.59	<none>	9600/TCP	19d
powerai-vision-mongodb	ClusterIP	10.10.0.111	<none>	27017/TCP	92d
powerai-vision-postgres	ClusterIP	10.10.0.220	<none>	5432/TCP	92d
powerai-vision-service	ClusterIP	10.10.0.58	<none>	9080/TCP	19d
powerai-vision-taskanally	ClusterIP	10.10.0.145	<none>	5000/TCP	92d
powerai-vision-ui	ClusterIP	10.10.0.39	<none>	80/TCP	92d
powerai-vision-video-nginx	ClusterIP	10.10.0.245	<none>	8081/TCP	92d
powerai-vision-video-portal	ClusterIP	10.10.0.152	<none>	8080/TCP,8081/TCP	92d
powerai-vision-video-rabmq	ClusterIP	10.10.0.2	<none>	5672/TCP	92d
powerai-vision-video-redis	ClusterIP	10.10.0.135	<none>	6379/TCP	92d
powerai-vision-video-test-nginx	ClusterIP	10.10.0.186	<none>	8083/TCP	92d
powerai-vision-video-test-portal	ClusterIP	10.10.0.66	<none>	8080/TCP,8081/TCP	92d
powerai-vision-video-test-rabmq	ClusterIP	10.10.0.187	<none>	5672/TCP	92d
powerai-vision-video-test-redis	ClusterIP	10.10.0.105	<none>	6379/TCP	92d

==> v1beta1/DaemonSet

NAME	DESIRED	CURRENT	READY	UP-TO-DATE	AVAILABLE	NODE SELECTOR	AGE
powerai-vision-fpga-device-plugin	1	1	1	1	1	<none>	5d2h

==> v1/Deployment

NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
powerai-vision-elasticsearch	1	1	1	1	5d2h
powerai-vision-keycloak	1	1	1	1	5d2h
powerai-vision-logstash	1	1	1	1	5d2h
powerai-vision-mongodb	1	1	1	1	5d2h
powerai-vision-postgres	1	1	1	1	5d2h
powerai-vision-service	1	1	1	1	5d2h
powerai-vision-taskanally	1	1	1	1	5d2h
powerai-vision-ui	1	1	1	1	5d2h
powerai-vision-video-nginx	1	1	1	1	5d2h
powerai-vision-video-portal	1	1	1	1	5d2h
powerai-vision-video-rabmq	1	1	1	1	5d2h
powerai-vision-video-redis	1	1	1	1	5d2h
powerai-vision-video-test-nginx	1	1	1	1	5d2h
powerai-vision-video-test-portal	1	1	1	1	5d2h
powerai-vision-video-test-rabmq	1	1	1	1	5d2h
powerai-vision-video-test-redis	1	1	1	1	5d2h

==> v1beta1/Ingress

NAME	HOSTS	ADDRESS	PORTS	AGE
powerai-vision-ing	*	80	92d	

==> v1/Pod(related)

NAME	READY	STATUS	RESTARTS	AGE
powerai-vision-elasticsearch-855648d6d4-7jwct	1/1	Running	0	5d2h
powerai-vision-fpga-device-plugin-qjhsd	1/1	Running	0	5d2h
powerai-vision-keycloak-84cfc79796-zl5qk	1/1	Running	0	5d2h
powerai-vision-logstash-7b48f978d7-pwscv	1/1	Running	0	5d2h
powerai-vision-mongodb-57bcd46cdd-vqs2b	1/1	Running	0	5d2h
powerai-vision-postgres-795987c56c-w6x62	1/1	Running	0	5d2h
powerai-vision-service-5588ffdfc-cnq8h	1/1	Running	0	5d2h
powerai-vision-taskanally-6448c56f89-918mf	1/1	Running	0	5d2h
powerai-vision-ui-5f78b785cf-xxfm6	1/1	Running	0	5d2h
powerai-vision-video-nginx-799bc6d9f6-6c27r	1/1	Running	0	5d2h

```

powerai-vision-video-portal-c9b6cd996-qrj68      1/1   Running 0      5d2h
powerai-vision-video-rabmq-6b8dcfd69b-w5wqb     1/1   Running 0      5d2h
powerai-vision-video-redis-6d455598b4-5688j     1/1   Running 0      5d2h
powerai-vision-video-test-nginx-7bbc9ddcfc-6xhwm 1/1   Running 0      5d2h
powerai-vision-video-test-portal-84f7b7f55b-n7jxv 1/1   Running 0      5d2h
powerai-vision-video-test-rabmq-59f59cc9b-4rbfh  1/1   Running 0      5d2h
powerai-vision-video-test-redis-5c78bb7f48-ffg9q 1/1   Running 0      5d2h

==> v1/Secret
NAME                TYPE      DATA  AGE
powerai-vision-secrets Opaque   7      92d

==> v1/ConfigMap
NAME                DATA  AGE
powerai-vision-config 58     92d

NOTES:

Find the PowerAI Vision UI URL by running the following commands:

export NODE_IP=$(kubectl get ing powerai-vision-ing --namespace default -o jsonpath="{.status.loadBalancer.ingress[0].ip}")
echo https://${NODE_IP}/powerai-vision/

```

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
...
powerai-vision-portal      1        1        1            1          14d
...

```

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/powerai-vision/bin/kubect1.sh get deployment
NAME                                READY  UP-TO-DATE  AVAILABLE  AGE
powerai-vision-dnn-infer-bb06a6f3-75d0-4614-96fb-a3d69f79d17f  1/1    1            1          11h
powerai-vision-elasticsearch        1/1    1            1          5d2h
powerai-vision-keycloak              1/1    1            1          5d2h
powerai-vision-logstash             1/1    1            1          5d2h
powerai-vision-mongodb              1/1    1            1          5d2h
powerai-vision-postgres             1/1    1            1          5d2h
powerai-vision-service              1/1    1            1          5d2h
powerai-vision-taskanalyzer         1/1    1            1          5d2h
powerai-vision-ui                   1/1    1            1          5d2h
powerai-vision-video-nginx          1/1    1            1          5d2h
powerai-vision-video-portal         1/1    1            1          5d2h
powerai-vision-video-rabmq         1/1    1            1          5d2h
powerai-vision-video-redis          1/1    1            1          5d2h
powerai-vision-video-test-nginx     1/1    1            1          5d2h
powerai-vision-video-test-portal    1/1    1            1          5d2h
powerai-vision-video-test-rabmq     1/1    1            1          5d2h
powerai-vision-video-test-redis     1/1    1            1          5d2h
```

kubect1 describe deployment

The `kubect1 describe deployment` command provides verbose status information about each of the deployed nodes in the Kubernetes environment that is being used to run PowerAI Vision.

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/powerai-vision/bin/kubect1.sh describe deployment
...
Name:                powerai-vision-ui
Namespace:           default
CreationTimestamp:   Thu, 13 Jun 2019 16:33:48 -0500
Labels:              app=powerai-vision
                    chart=ibm-powerai-vision-prod-1.4.0
                    heritage=Tiller
                    release=vision
                    run=powerai-vision-ui-deployment
Annotations:         deployment.kubernetes.io/revision: 1
Selector:            run=powerai-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=powerai-vision
                    chart=ibm-powerai-vision-prod-1.4.0
                    component=powerai-vision-ui
                    heritage=Tiller
                    release=vision
                    run=powerai-vision-ui-deployment-pod
  Annotations:       checksum/config: 5c015ae70c4e9169a83675f61c7068009abd3827c19f9f9eb0f677e8060a19e6
                    productID: 5737-H10
                    productName: IBM PowerAI Vision
                    productVersion: 1.1.4.0
Containers:
  powerai-vision-ui:
    Image:            powerai-vision-ui:1.1.4.0
    Port:             80/TCP
    Host Port:        0/TCP
    Liveness:         http-get http://:http/powerai-vision/index.html delay=240s timeout=5s period=10s #success=1 #failure=3
    Readiness:        http-get http://:http/powerai-vision/index.html delay=5s timeout=1s period=10s #success=1 #failure=3
    Environment:
      CONTEXT_ROOT:   <set to the key 'CONTEXT_ROOT' of config map 'powerai-vision-config'>      Optional: false
      DLAAS_API_SERVER: <set to the key 'DLAAS_API_SERVER' of config map 'powerai-vision-config'>      Optional: false
      SERVER_HOST_VIDEO_TEST: <set to the key 'SERVER_HOST_VIDEO_TEST' of config map 'powerai-vision-config'>  Optional: false
      SERVICE_PORT_VIDEO_TEST: <set to the key 'SERVICE_PORT_VIDEO_TEST' of config map 'powerai-vision-config'>  Optional: false
      WEBROOT_VIDEO_TEST: <set to the key 'WEBROOT_VIDEO_TEST' of config map 'powerai-vision-config'>  Optional: false
    Mounts:
      /opt/powerai-vision/data from data-mount (rw)
  Volumes:
    data-mount:
      Type:            PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
      ClaimName:       powerai-vision-data-pvc
      ReadOnly:        false
Conditions:
  Type            Status  Reason
  ----            -
  Available       True   MinimumReplicasAvailable
  Progressing     True   NewReplicaSetAvailable
OldReplicaSets:  <none>
NewReplicaSet:   powerai-vision-ui-5f78b785cf (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 PowerAI Vision installed:
productVersion: 1.1.4.0
- The **Image** value provides information about the Docker container:
Image: powerai-vision-ui:1.1.4.0

- 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 PowerAI Vision, 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 PowerAI Vision” 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  Memory-Usage  GPU-Util  Compute M. |
+-----+-----+-----+-----+-----+-----+
|  0   Tesla P100-SXM2...  On          | 00000002:01:00:0 Off      |         0          0          |
| N/A   50C    P0      109W / 300W | 2618MiB / 16280MiB |         43%         Default  |
+-----+-----+-----+-----+-----+-----+
|  1   Tesla P100-SXM2...  On          | 00000003:01:00:0 Off      |         0          0          |
| N/A   34C    P0       34W / 300W |    0MiB / 16280MiB |          0%         Default  |
+-----+-----+-----+-----+-----+-----+
|  2   Tesla P100-SXM2...  On          | 0000000A:01:00:0 Off      |         0          0          |
| N/A   48C    P0       44W / 300W | 5007MiB / 16280MiB |          0%         Default  |
+-----+-----+-----+-----+-----+-----+
|  3   Tesla P100-SXM2...  On          | 0000000B:01:00:0 Off      |         0          0          |
| N/A   36C    P0       33W / 300W |    0MiB / 16280MiB |          0%         Default  |
+-----+-----+-----+-----+-----+-----+
+-----+
| Processes:                                     GPU Memory |
| GPU      PID   Type   Process name                                     Usage      |
+-----+-----+-----+-----+-----+-----+
|    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 PowerAI Vision 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 126 for more information.

Logging in to PowerAI Vision

Follow these steps to log in to PowerAI Vision.

Note: PowerAI Vision is supported on these browsers:

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

1. Enter the appropriate PowerAI Vision URL in a supported browser:

PowerAI Vision stand-alone URL

`https://hostname/powerai-vision/`, where *hostname* is the system on which you installed PowerAI Vision.

PowerAI Vision with IBM Cloud Private URL

`https://proxyhost/powerai-vision-RELEASE/`, where *proxyhost* is the host name of your IBM Cloud Private proxy server, and *RELEASE* is the name you specified in the Release name field when you deployed the Helm chart.

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 103.

Related concepts:

“Managing users” on page 103

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

Working with the user interface

The PowerAI Vision 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:

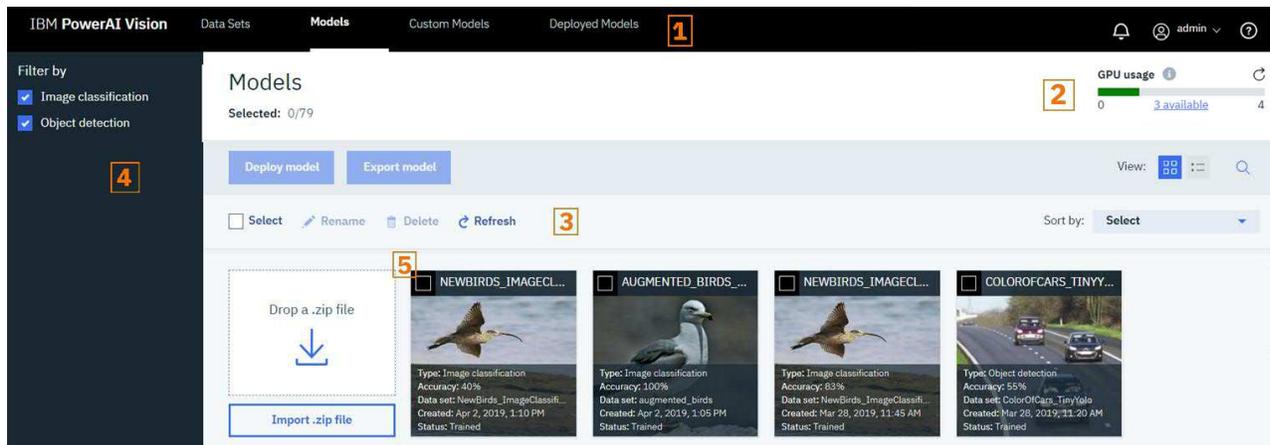


Figure 7. PowerAI Vision user interface

1: The navigation bar

The navigation bar lets you choose a work space, access the notification area, or work with your profile.

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 PowerAI Vision.

Deployed Models

GPUs currently used for deployed models by PowerAI Vision.

External

External GPUs are those that are used for processes outside of PowerAI Vision. For IBM Cloud Private installations, all GPUs are listed as External.

Note: If the output shows "Unknown", then GPUs are in use, but not for PowerAI Vision 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 **Delete**, 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 **Delete**.

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.

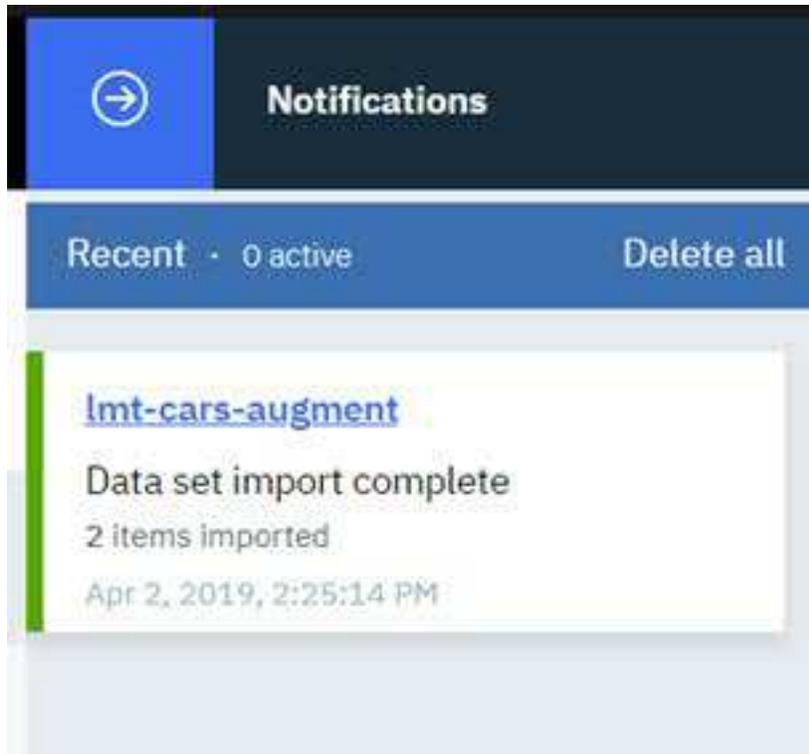


Figure 8. Notification area

Related concepts:

“Training and working with models” on page 57

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

“Example: Detecting objects in a video” on page 92

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” on page 66

After the data set has all of the object labels added, you can train your deep learning model. Trained models can then be deployed for use.

“Creating and working with data sets” on page 57

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” on page 82

Deploy a trained model to get it ready to use within PowerAI Vision or a different program, such as IBM PowerAI. Deploying a model creates a unique API endpoint based on that model for inference

operations.

“Example: Classifying images” on page 97

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.

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.

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

1. Log in to PowerAI Vision.
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 **Duplicate**.

File considerations:

• Videos

- You can play only the following video types in the PowerAI Vision 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

- PowerAI Vision 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 PowerAI Vision can import the annotations successfully.
- PowerAI Vision 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, PowerAI Vision only keeps the information it will use, as follows:
 - PowerAI Vision 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 RLE is not saved. Only the bounding box is used.

Note: Images without tags *are* saved.

3. 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 at the top of the page 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 PowerAI Vision page or refresh until the upload completes. You can navigate to different pages within PowerAI Vision 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.

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.

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.

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” refers to both individual images and captured video frames.

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

What are the limitations on uploaded files?

- The following image formats are supported:
 - JPEG
 - PNG
- You can play only the following video types in PowerAI Vision:
 - Ogg Vorbis (.ogg)

- VP8 or VP9 (.webm)
- H.264 encoded videos with MP4 format (.mp4)
- The models used by PowerAI Vision 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.
 - High resolution images will be scaled to a maximum of 1000 x 600 pixels.
 - For image classification, images are scaled to 224 x 224 pixels.
 - For object detection with Detectron, all images are scaled to 1333 x 800 pixels. However, the original aspect ratio is maintained. If necessary, black bands are added to the image to make it fit.
 - For object detection with tiny YOLO V2, all images are scaled to 416 x 416. The longest edge is scaled to 416 pixels and, if necessary, black bands are added to the shorter side to make it 416 pixels.
 - For object detection with FR-CNN and video action detection, anything over 1000 x 600 pixels is down-sampled so that the longest edge will fit. However, the original aspect ratio is maintained. If necessary, black bands are added to the image to make it fit.
 - 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.
- Images with COCO annotations are supported. For details, see “Importing images with COCO annotations” on page 60.

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.

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 PowerAI Vision.

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 PowerAI Vision. You can import (upload) these images into an existing PowerAI Vision 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, PowerAI Vision only keeps the information it will use, as follows:

- PowerAI Vision 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 RLE is not saved. Only the bounding box is used.

Note: Images without tags *are* saved.

To import images with COCO annotations into PowerAI Vision, 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`, PowerAI Vision 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 PowerAI Vision 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 PowerAI Vision 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.

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.

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”
- “Labeling images” on page 63

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** - PowerAI Vision 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.
- When performing multiple auto label operations on the same video, it is possible to get multiple frames with the same time offset. This situation can occur when the intervals overlap and labels have been edited on the frames at the overlap points.

For example, labeling at a 10 second interval, editing some of the labels on those frames, and then labeling again at a 5 second interval has an overlap every 10 seconds. There might be duplicate images at each of the 10 second intervals with edited labels.

- **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 77.

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.

- **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.

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.

- **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.

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” on page 79

After deploying a model for object detection, you can improve its accuracy by using the Auto label function. This function improves the model's accuracy by quickly adding more data to the data set.

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”
- “Preparing videos for import” on page 65
- “Steps to label actions” on page 65
- “Working with action labels” on page 65
- “Tips” on page 65

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.
- 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 all of the object labels added, you can train your deep learning model. Trained models can then be deployed for use.

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.

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.

Action detection

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

Model selection

Select the model that you want to use:

System default (GoogLeNet)

Models trained with this model can only be run on a GPU. This option is only available when training for image classification.

Accuracy (Faster R-CNN)

Models optimized for accuracy can only be run on a GPU. This option is only available when training for object detection.

Speed (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. This option is only available when training for object detection.

You will choose the accelerator to deploy to when deploying the model. You can choose GPU, CPU, or Xilinx FPGA - 16 bit (technology preview).

Segmentation (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 PowerAI Vision will use rectangles instead of polygons. The actual images are not modified, so you can train with segmentation later.

Custom model

Select an imported model to use for training.

Structured segment network (SSN)

Used for action detection models only.

Advanced options

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.

When you specify a base model, PowerAI Vision 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 PowerAI Vision, or you can choose your own model that you previously trained or imported. For models that were trained in 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 be Faster R-CNN (for object detection) or GoogLeNet (for image classification). Only viable models are listed in the Base model table.

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

PowerAI Vision 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 PowerAI Vision” on page 77.

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.

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 PowerAI Vision 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.

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 small can result in a sub-optimal trained model arriving at a local minimum of the loss function. 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.

Notes:

- If a training job appears to be hanging, it might be waiting for another training job to complete, or there might not be a GPU available to run it. For information to fix this, see “PowerAI Vision cannot train a model” on page 127.
 - When training an action detection model, it can take several minutes before the status changes from Scheduled to Training, depending on the data set size.
3. (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. 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 PowerAI Vision 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.

Loss VS Iteration

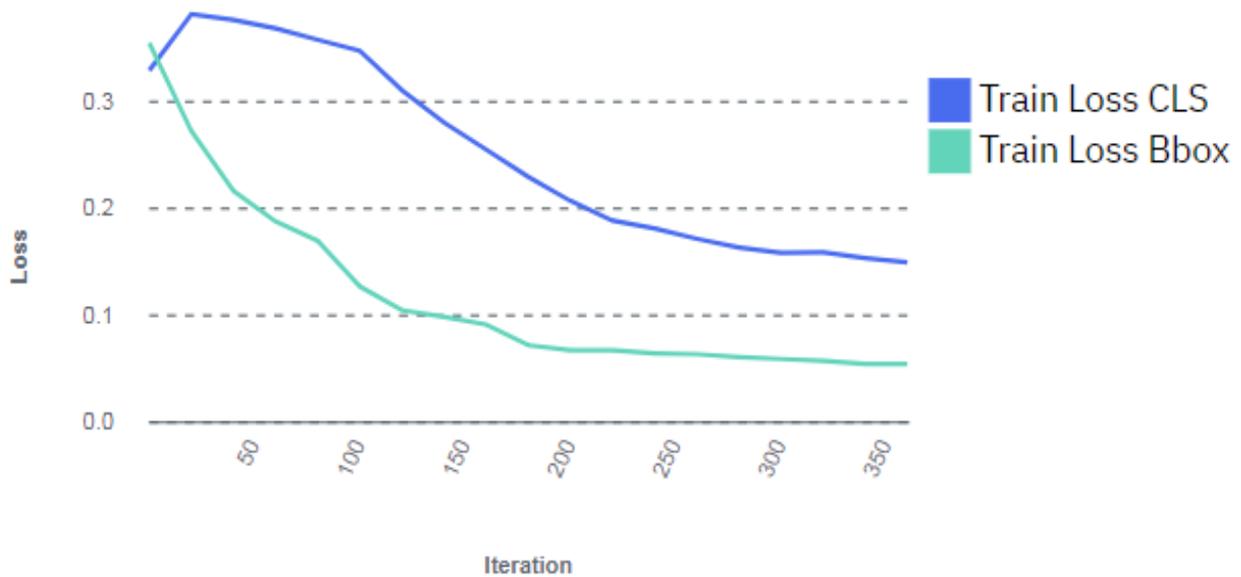


Figure 9. Model training graph

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.

Related concepts:

“Understanding metrics” on page 85

PowerAI Vision 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 PowerAI Vision. In general, custom models work the same as any other model in PowerAI Vision. 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 PowerAI Vision and generate a PowerAI Vision trained model.

Note: Custom models cannot be used to import pre-trained models into PowerAI Vision. Additionally, transfer learning is not supported with custom models.

Use the information in this topic to prepare a PowerAI Vision trained model by using a custom TensorFlow model: “Preparing a model that will be used to train data sets in PowerAI Vision” on page 70.

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

Preparing a model that will be used to train data sets in PowerAI Vision

If your custom model will be used to train data sets in the PowerAI Vision framework, your custom model must meet the following requirements.

After the model is properly prepared, upload it to PowerAI Vision 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 66.

Custom model requirements:

- It must be TensorFlow based.
- 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
```

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

class MyTrain(TrainCallback):

Use the MyTrain API to prepare a TensorFlow model that will be used to train data sets with PowerAI Vision.

- “Template”
- “def onPreprocessing(self, labels, images, workspace_path, params)” on page 71
- “def onTraining(self, monitor_handler)” on page 72
- “def onCompleted(self, model_path)” on page 72
- “def onFailed(self, train_status, e, tb_message):” on page 72
- “Monitoring and reporting statistics” on page 72

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/Acridotherees/001.jpg': 'Acridotherees', '/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, 'train_test_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 PowerAI Vision. 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 73
- “`def updateTestMetrics(mAP)`” on page 73

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 PowerAI Vision 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 PowerAI Vision 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)”
- “def updateTestMetrics(current_iter, accuracy, loss, epoch)” on page 74

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 PowerAI Vision 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 PowerAI Vision 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 PowerAI Vision

If your custom model will be deployed in the PowerAI Vision framework, your custom model must meet the following requirements.

After the model is properly prepared, import it to PowerAI Vision 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 based.
- 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 training 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 model that will be deployed in PowerAI Vision.

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.

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
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 PowerAI Vision

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 PowerAI Vision.

Table 5. Base models included with PowerAI Vision

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/cbocm5pvtuadoaypdwl3jaypets1hel
General	ImageNet dataset		ilsvrc12 http://image-net.org/download
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

Testing a model

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

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 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.

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 overlaid, 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 (Detectron models only)

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, with the name displayed next to the dot, 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.

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 79.

Related concepts:

“Importing and exporting PowerAI Vision information” on page 83

You can import and export PowerAI Vision models and data sets. This allows you to save them for archiving then use them later, use them on a different PowerAI Vision 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.

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

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. For instructions, see "Augmenting the data set" on page 81.
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" for instructions.

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 improves the model's accuracy by quickly adding more data to the data set.

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 labels have been added through auto label, 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 performing multiple auto label operations on the same video, it is possible to get multiple frames with the same time offset. This situation can occur when the intervals overlap and labels have been edited on the frames at the overlap points.
For example, labeling at a 10 second interval, editing some of the labels on those frames, and then labeling again at a 5 second interval has an overlap every 10 seconds. There might be duplicate images at each of the 10 second intervals with edited labels.
- You can manually add labels to images and frames that have been auto labeled, and you can manipulate (move, resize) the labels that were automatically generated. If an auto labeled frame or image is edited, either by modifying an automatically generated label, or by manually adding a new label, **all** auto label annotations in the frame or image are converted to manually labeled annotations. If you then run **Auto label** again, those images and frames are skipped because the frame was manually edited.

When you automatically label objects, an existing trained model is used to generate new labels in the data set.

1. Open the data set that you want to add more data to and select **Auto label**.
2. Select the trained model to use, then click **Auto label**.
3. Labels are added to existing images or videos 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.

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.

Notes:

- All images and frames that were not manually labeled are processed. Therefore, objects that contain only labels that were added by using the auto label function 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.

Follow these steps to generate new labels in the data set.

1. Open the data set that you want to add more data to and select **Auto label**.
2. Select the trained model to use, then click **Auto label**.
3. Labels are added to existing images or videos have not been manually labeled. By default, the automatically added labels are light red and manually added labels are blue.

You can manually add labels to images and frames that have been auto labeled, and you can manipulate (move, resize) the labels that were automatically generated. If an auto labeled frame or image is edited, either by modifying an automatically generated label, or by manually adding a new label, **all** auto label annotations in the frame or image are converted to manually labeled annotations. If you then run **Auto label** again, those images and frames are skipped because the frame was manually edited.

Automatically labeling videos

When using the auto label function on a data set, only frames and images processed. Videos are ignored. However, you can run the auto label function on an individual video.

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.

1. Open the data set that contains the video.
2. Select the video and click **Label objects**.
3. Click **Auto label** then choose the time interval to capture frames and the trained model to use for labeling, then click **Auto label**.
4. Frames are captured at the specified interval and labels are added by using the specified trained model. By default, the automatically added labels are light red.

After processing, you can manually add labels to the frames that have been auto labeled and you can manipulate (move, resize) the labels that were automatically generated. If a frame with automatically generated labels is edited, all labels on the frame are converted to manual labels.

When performing multiple auto label operations on the same video, it is possible to get multiple frames with the same time offset. This situation can occur when the intervals overlap and labels have been edited on the frames at the overlap points.

For example, labeling at a 10 second interval, editing some of the labels on those frames, and then labeling again at a 5 second interval has an overlap every 10 seconds. There might be duplicate images at each of the 10 second intervals with edited labels.

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.

To augment a data set, follow these steps:

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 66.

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 PowerAI Vision 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.

Deploying a trained model

Deploy a trained model to get it ready to use within PowerAI Vision or a different program, such as IBM PowerAI. Deploying a model creates a unique API endpoint based on that model for inference operations.

To deploy the trained model, follow these steps:

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).

Note: Deploying a model to a Xilinx FPGA requires the Xilinx Alveo U200 Accelerator card. GPUs are used as follows:

- Each Tiny YOLO V2, Detectron, or custom deployed model takes one GPU. The GPU group is listed as '-', which indicates that this model uses a full GPU and does not share the resource with any other deployed models.
- Multiple Faster R-CNN and GoogLeNet models are deployed to a single GPU. PowerAI Vision 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: PowerAI Vision 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 Vision Service 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. To delete a deployed model, click **Deployed Models**. Next, select the model that you want to delete and click **Delete**. The trained model is not deleted from PowerAI Vision.

Related concepts:

“Working with the user interface” on page 53

The PowerAI Vision 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” on page 85

PowerAI Vision provides several metrics to help you measure how effectively your model has been trained.

Related information:

 [Vision Service API documentation](#)

Importing and exporting PowerAI Vision information

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

- “Exporting”
- “Importing” on page 84

Exporting

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 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 custom trained model (a model that was trained by using a custom model), the generated zip file is not encrypted or password protected, unlike with other models exported from PowerAI Vision.

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 can only be used with PowerAI Vision. It can be imported into the Inference Server product and deployed with the Inference Server product.

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 (Detectron only) 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.

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 PowerAI Vision tab or refresh the page. Doing so stops the upload.

3. After the upload completes, the data set has its original name.

Notes:

- In 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 PowerAI Vision to train a new model, you can *import* a model that was previously trained with PowerAI Vision, 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 PowerAI Vision. Use the

information in this topic to prepare a custom model that will be deployed in PowerAI Vision: “Preparing a model that will be deployed in PowerAI Vision” on page 74.

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 PowerAI Vision tab or refresh the page. Doing so stops the upload.

3. After the upload completes, the model has its original name.

PowerAI Vision REST APIs

You can use REST APIs to work with PowerAI Vision 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 PowerAI Vision processes or solutions.

For information about using the API see Vision Service API documentation. There are also examples of using the APIs for different actions, published here.

Understanding metrics

PowerAI Vision 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 PowerAI Vision 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 PowerAI Vision 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 PowerAI Vision 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 PowerAI Vision 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)”
- “Metrics for object detection (Optimized for accuracy)” on page 86
- “Metrics for object detection using the Tiny Yolo model (Optimized for speed)” on page 87
- “Metrics for custom models” on page 88
- “Metrics for action detection models” on page 88

Metrics for image classification (Optimized for accuracy)

Accuracy

Measures the percentage of correctly classified images. It is calculated by (true positives + true negatives) / (true positives + true negatives + false positives+ 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 true positives / (true positives + 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 true positives / (true positives + 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 PowerAI Vision 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 (Optimized for accuracy)

Accuracy

Measures the percentage of correct image classifications. It is calculated by (true positives + true negatives) / all cases.

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 PowerAI Vision.

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 PowerAI Vision

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 PowerAI Vision 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 true positives/(true positives + 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 (true positives + true negatives) / (true positives + true negatives + false positives + false negatives).

Metrics for action detection models

Accuracy

Measures the percentage of correctly detected actions. It is calculated by (true positives + true negatives) / (true positives + true negatives + false positives + 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 true positives / (true positives + 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 true positives/(true positives + 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 PowerAI Vision 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.

Using PowerAI Vision

These fictional examples give step-by-step instructions of how to use PowerAI Vision 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"
2. "Labeling objects in an image"
3. "Training a model" on page 90
4. "Deploying a trained model" on page 91

Import images and create a data set

First, create a data set and add images to it.

1. Log in to PowerAI Vision.
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 PowerAI Vision page or refresh until the upload completes. You can navigate to different pages within PowerAI Vision 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. 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.

Loss VS Iteration

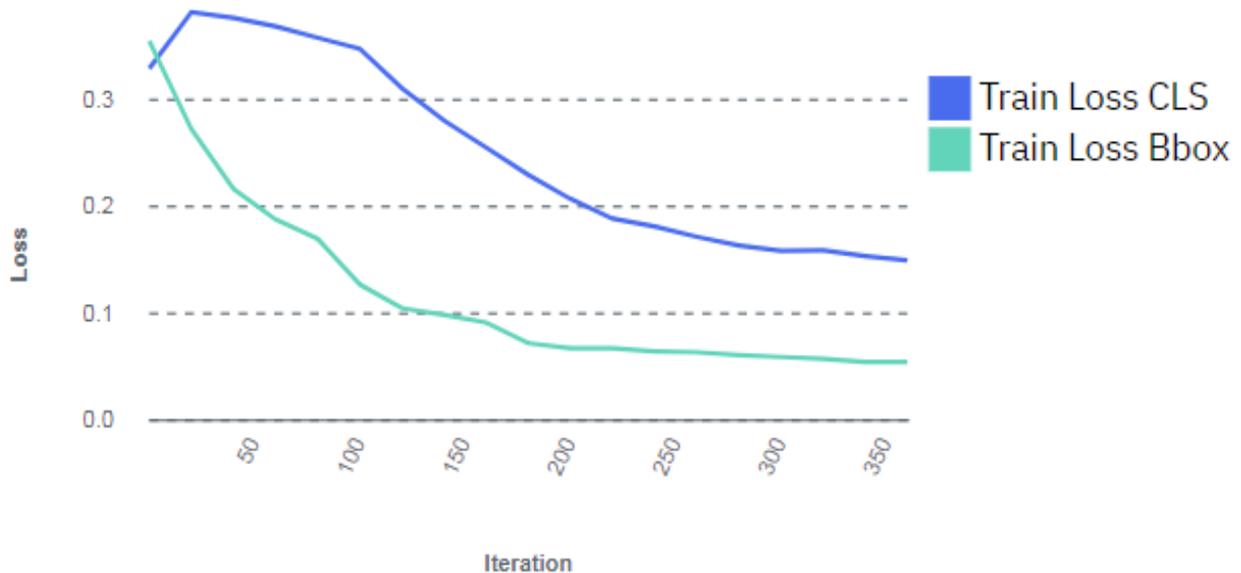


Figure 10. Model training graph

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:

- Each Tiny YOLO V2, Detectron, or custom deployed model takes one GPU. The GPU group is listed as '-', which indicates that this model uses a full GPU and does not share the resource with any other deployed models.
- Multiple Faster R-CNN and GoogLeNet models are deployed to a single GPU. PowerAI Vision 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: PowerAI Vision 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 Vision Service 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 PowerAI Vision.
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 PowerAI Vision page or refresh until the upload completes. You can navigate to different pages within PowerAI Vision 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**.
 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.

- **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.

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 11. Labeling objects in PowerAI Vision

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. 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.

Loss VS Iteration

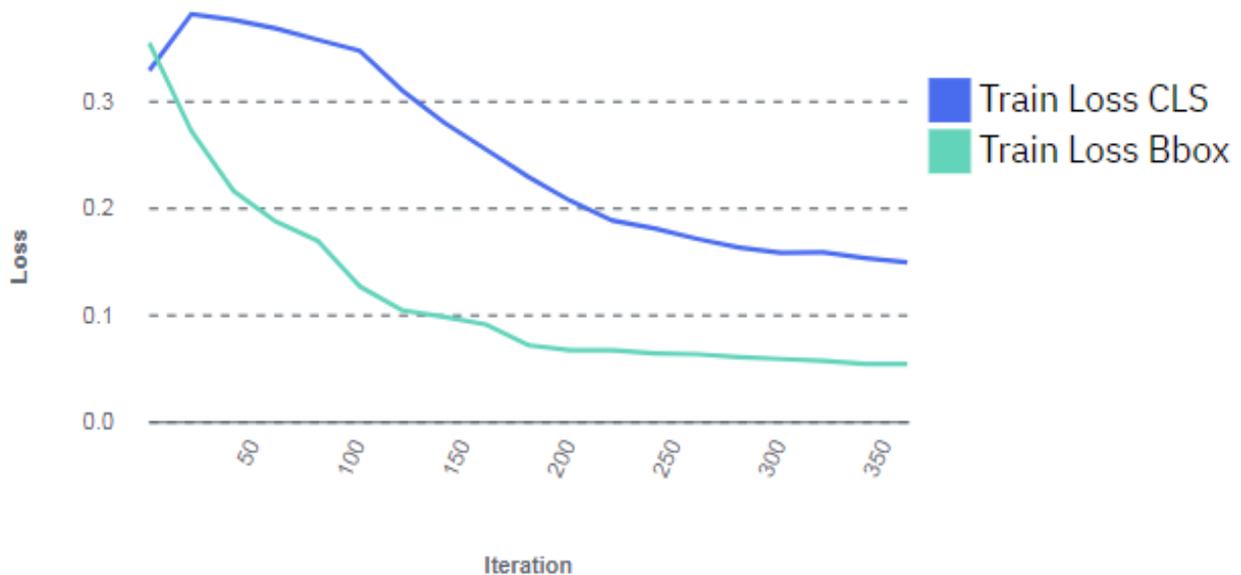


Figure 12. Model training graph

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:

- Each Tiny YOLO V2, Detectron, or custom deployed model takes one GPU. The GPU group is listed as '-', which indicates that this model uses a full GPU and does not share the resource with any other deployed models.
- Multiple Faster R-CNN and GoogLeNet models are deployed to a single GPU. PowerAI Vision 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: PowerAI Vision 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 Vision Service 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

After processing, you can manually add labels to the frames that have been auto labeled and you can manipulate (move, resize) the labels that were automatically generated. If a frame with automatically generated labels is edited, all labels on the frame are converted to manual labels.

When performing multiple auto label operations on the same video, it is possible to get multiple frames with the same time offset. This situation can occur when the intervals overlap and labels have been edited on the frames at the overlap points.

For example, labeling at a 10 second interval, editing some of the labels on those frames, and then labeling again at a 5 second interval has an overlap every 10 seconds. There might be duplicate images at each of the 10 second intervals with edited labels.

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” on page 53

The PowerAI Vision 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” on page 85

PowerAI Vision 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.

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
2. Create a data set. Log in to the PowerAI Vision 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 overlaid, 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 79. Otherwise, the model is ready to be used in production.

Related concepts:

“Working with the user interface” on page 53

The PowerAI Vision 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” on page 85

PowerAI Vision 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”
2. “Training a model” on page 99
3. “Deploying a trained model” on page 100

Import images and create a data set

First, create a data set and add images to it.

1. Log in to PowerAI Vision.
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 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 PowerAI Vision page or refresh until the upload completes. You can navigate to different pages within PowerAI Vision 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. 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.

Loss VS Iteration

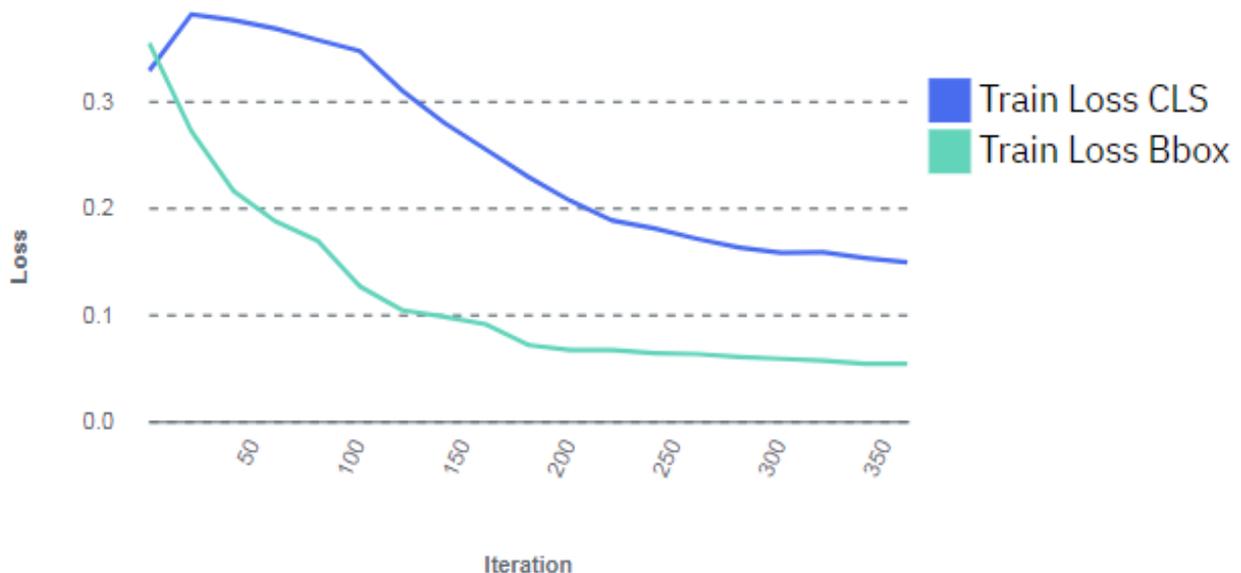


Figure 13. Model training graph

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, 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 Vision Service 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"
3. "Labeling actions in a video" on page 101
4. "Training the model" on page 101

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 PowerAI Vision.
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 PowerAI Vision page or refresh until the upload completes. You can navigate to different pages within PowerAI Vision 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 64:

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. 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.

Loss VS Iteration

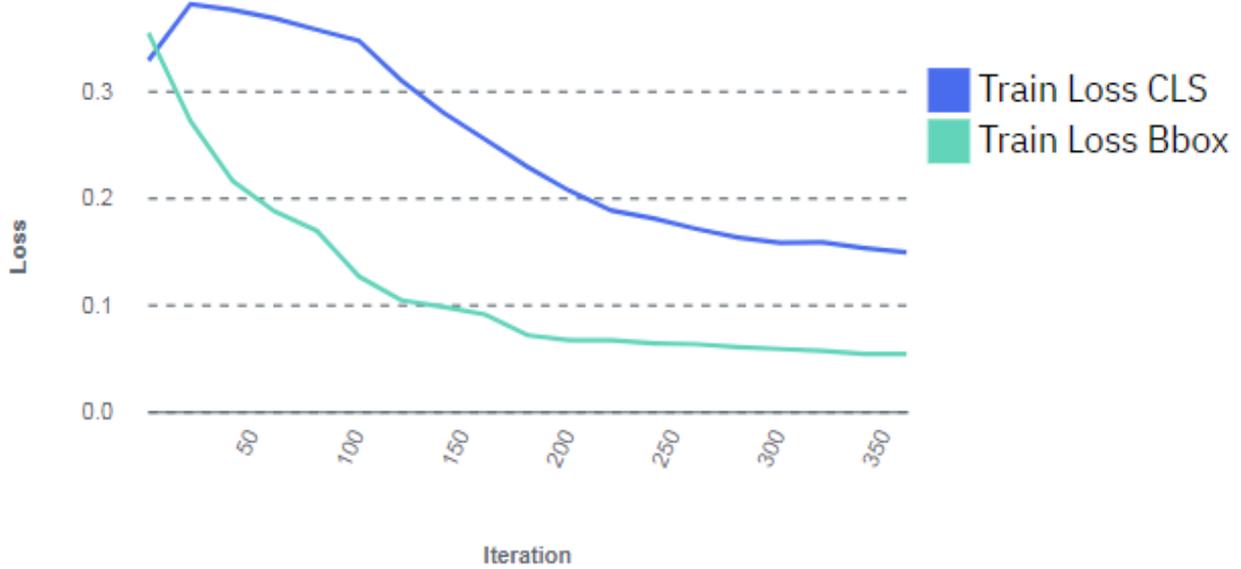


Figure 14. Model training graph

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.

Administering PowerAI Vision

Use this information to administer PowerAI Vision, such as stopping, starting, and determining the status of the pods.

Start or stop PowerAI Vision

There are several situations when you might need to stop and start PowerAI Vision. 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 PowerAI Vision, as appropriate:

```
powerai-vision-stop.sh  
powerai-vision-start.sh
```

Determine the status of PowerAI Vision pods

When troubleshooting a problem with PowerAI Vision, you might need to check the status of the Docker pods that are part of PowerAI Vision. For example, if the product does not start, if it is returning errors, or if actions are not completing. Run `kubect1 get pods` to see the status. For example:

```
$ /opt/powerai-vision/bin/kubect1 get pods  
NAME                                READY   STATUS    RESTARTS   AGE  
powerai-vision-mongodb-764f99fcf6-12nzd    1/1     Running   0           12h  
powerai-vision-portal-76fbc7db68-7rr47     1/1     Running   0           12h  
powerai-vision-postgres-55c6f7fcf6-42fht   1/1     Running   0           12h  
powerai-vision-taskanaly-55bfb587d4-cvz1n   1/1     Running   0           12h  
powerai-vision-ui-845d8c8d8-bmf7          1/1     Running   0           12h  
powerai-vision-video-nginx-8474f7c44c-qmxm4 1/1     Running   0           12h  
powerai-vision-video-portal-5b76558784-8mb8d 1/1     Running   0           12h  
powerai-vision-video-rabmq-5d5d786f9f-nz7pn 1/1     Running   0           12h  
powerai-vision-video-redis-59c557b69-hf8pg 1/1     Running   0           12h  
powerai-vision-video-test-nginx-5dc6887666-19tb8 1/1     Running   0           12h  
powerai-vision-video-test-portal-54d85ff65b-945gp 1/1     Running   0           12h  
powerai-vision-video-test-rabmq-6858cc749-grhgm 1/1     Running   0           12h  
powerai-vision-video-test-redis-75977cdd8f-1bljb 1/1     Running   0           12h
```

If one or more pods is not running, try stopping and restarting PowerAI Vision.

Managing users

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

PowerAI Vision 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 `kubect1` command.

- “Types of users”
- “PowerAI Vision installed as stand-alone” on page 104
- “PowerAI Vision installed with IBM Cloud Private” on page 105

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 PowerAI Vision 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 **kubect**l 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.

PowerAI Vision installed as stand-alone

If you installed PowerAI Vision stand-alone, you can use the `powerai_vision_users.sh` script in the `/opt/powerai-vision/bin/` directory to create, delete, modify, and list users.

Usage

```
powerai_vision_users.sh [command] [ --user name ] [ --password password ]
```

Command

Specifies the action to take.

create

Create a user in the PowerAI Vision 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 PowerAI Vision instance. The user argument is required for this operation.

list

List the currently created users for a specified PowerAI Vision 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`.

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.

PowerAI Vision installed with IBM Cloud Private

1. Authenticate to the cluster, so that you can run kubectl commands. For example:

- In an IBM Cloud Private 2.1.0 environment, run:

```
bx pr login -a https://<cluster-domain-name>:8443/ --skip-ssl-validation
```
- In an IBM Cloud Private 3.1.0 environment, run:

```
cloudctl login -a https://<cluster-domain-name>:8443/ --skip-ssl-validation
```

2. Note your release name. In the example below, this is aivision.

3. To manage users, run the following command:

```
kubectl run --rm -i --restart=Never usermgt --image=cluster-domain-name:8443/powerai-vision-usermgt:version -- action  
--user newusername --password password --release release
```

The above command has the following variables:

- *action* can be one of these values: create, delete, modify, or list.
- *version* is the release number of the PowerAI Vision product. For example, 1.1.4.0. To find the correct value, view the configmap. For example:

```
$ kubectl get cm  
NAME DATA AGE  
powerai-vision-v1.1.4-config 52 56d
```

The password argument is optional. You can set the password in one of these ways:

- The --password argument in powerai-vision-usermgt.
- The --env option for kubectl with the VISION_USER_PASSWORD environment variable. For example, add --env="VISION_USER_PASSWORD=\${MY_PASS}" to the kubectl run command.

Example: To create customusername with password custompassw0rd1234 on release aivision, run:

```
$ kubectl run --rm -i --restart=Never usermgt --image=myicpcluster.com:8443/powerai-vision-usermgt:1.1.4.0  
-- create --user customusername --password custompassw0rd1234 --release aivision  
Created user: customusername
```

Example: To list users in the PowerAI Vision 1.1.4 deployment, run:

```
$ kubectl run --rm -i --restart=Never usermgt --image=powerai-vision-usermgt:1.1.4.0 -- list --release v111  
If you don't see a command prompt, try pressing enter.  
admin  
testuser1  
testuser2
```

Notes:

- If running in the non-default namespace, make sure to specify the --namespace option.
- The version tag on the container should match image.releaseTag in the values.yaml file.
- The argument release should match the release name you assigned when deploying the chart.
- There is not a typo with the spacing of the "--" before create. It should be --<SPACE>create<SPACE> --user *username*.... This is intentional and an artifact of how the commands are passed into the user management tool.

Installing a new SSL certificate in PowerAI Vision stand-alone

PowerAI Vision ships with a self-signed certificate that is used by default, but this can be replaced with a certificate generated for PowerAI Vision for secure communications. If you want to use your own certificate, follow these steps to update the PowerAI Vision configuration.

1. Shut down PowerAI Vision:

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

2. Edit `/opt/powerai-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 PowerAI Vision.
3. Start PowerAI Vision:

```
$ sudo /opt/powerai-vision/bin/powerai-vision-start.sh
```

Monitoring usage metrics

Administrators can view the PowerAI Vision 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.

Note: YOLO and Faster R-CNN models call the inference API once every second of the video. For example, if you use a YOLO model for inferencing on a 31 second video, this results in 31 API calls. If you use an action detection model for inferencing on a 31 second video, the API is called only once. To access the Usage metrics page, click your user name then click **Usage metrics**.

All metrics are kept from the time that PowerAI Vision 1.1.4 was installed, but you can use the date range selector to filter the data you want displayed.

Note: PowerAI Vision cannot track metrics for models deployed prior 1.1.4. If models were deployed in a previous version, they must be redeployed if you want their metrics tracked.

PowerAI Vision utilities

PowerAI Vision includes these utilities for working with the product.

- “Administer”
- “Troubleshooting” on page 109
- “Cleanup and uninstall” on page 109

Administer

accept-powerai-vision-license

Usage

```
# accept-powerai-vision-license.sh
```

Description

The `accept-powerai-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, "3" to print, "4" Read non-IBM terms.

Note: This is called by the startup script `powerai_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-powerai-vision-license

Usage

```
# check-powerai-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 PowerAI Vision stand-alone” on page 105 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 `powerai_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 PowerAI Vision deployment. See “Checking application deployment” on page 44 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 PowerAI Vision deployment. See these topics for details:

- “Checking Kubernetes services status” on page 37
- “Checking Kubernetes node status” on page 39
- “Checking application deployment” on page 44

For information about the `kubect1` utility, see Overview of `kubect1`.

Requirements

None.

load_images.sh

Usage

```
# load_images.sh -f [ <powerai-vision-images-release>.tar ]
```

Description

Utility to load the PowerAI Vision Docker images, which are provided with the product installation package in the `<powerai-vision-images-release>.tar` file. The `load_images.sh` utility requires approximately 30 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 PowerAI Vision user interface will use. The default port is 443.

Requirements

None - not an executable.

powerai_vision_start.sh

Usage

```
# powerai_vision_start.sh [ -nD ]
```

Description

Used to start the PowerAI Vision 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.

powerai_vision_stop.sh

Usage

```
# powerai_vision_stop.sh
```

Description

Used to stop the PowerAI Vision 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 PowerAI Vision 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 PowerAI Vision 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 PowerAI Vision Docker images matching the tag will be removed from the Docker repository. This script can be used to clean up images from a prior PowerAI Vision installation after an upgrade, or to remove PowerAI Vision images when uninstalling the product.

For example, to remove Docker images for the 1.1.4.0 release from the Docker repository, run this command:

```
# purge_images 1.1.4.0
```

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.

PowerAI Vision Inference Server

With a PowerAI Vision Inference server, you can quickly and easily deploy multiple trained models to a single server. 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.

- “Hardware requirements”
- “Platform requirements” on page 112
- “Software requirements” on page 112
- “Installing” on page 112
- “Deploying a trained model” on page 113
- “Deployment output” on page 113
- “Inference” on page 114
- “Inference output” on page 116
- “Stopping a deployed model” on page 116
- “Decrypting a trained model” on page 117

Hardware requirements

Disk space requirements

- Installation - The Inference Server install package contains Docker containers for deployment on all supported platforms and requires 25 Gb to download. Only the images needed for the platform will be installed by the `load_images.sh` operation, but this requires at least 15 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 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 GoogLeNet, Faster R-CNN, Tiny Yolo v2, or Detectron 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 | 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   36C    P0     39W / 300W | 1853MiB / 16280MiB |    0%      Default     |
+-----+-----+-----+-----+-----+-----+
|   1   Tesla P100-SXM2...    On          | 00000003:01:00:0 | Off      |          0          |
| N/A   38C    P0     42W / 300W | 4179MiB / 16280MiB |    0%      Default     |
+-----+-----+-----+-----+-----+-----+
|   2   Tesla P100-SXM2...    On          | 0000000A:01:00:0 | Off      |          0          |
| N/A   63C    P0    243W / 300W | 3351MiB / 16280MiB |   73%      Default     |
+-----+-----+-----+-----+-----+-----+
|   3   Tesla P100-SXM2...    On          | 0000000B:01:00:0 | Off      |          0          |
| N/A   35C    P0     31W / 300W | 10MiB / 16280MiB  |    0%      Default     |
+-----+-----+-----+-----+-----+-----+
```

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 2GB memory.

Platform requirements

- The Inference Server can be deployed on x86 and IBM Power Systems™ platforms.
- Detectron models require Nvidia GPUs. Other models can be deployed in CPU only environments.

Software requirements

Linux

- Red Hat Enterprise Linux (RHEL) 7.6 (little endian).
- Ubuntu 18.04 or later.

NVIDIA CUDA

- x86 - 10.1 or later drivers. For information, see the NVIDIA CUDA Toolkit website.
- ppc64le - 10.1 Update 1 or later drivers. For information, see the NVIDIA CUDA Toolkit website.

Docker

- Docker must be installed. The recommended version is 1.13.1 or later. Version 1.13.1 is installed with RHEL 7.6.
- 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

1. Download the install files by using one of these methods:
 - Download the product tar file from the IBM Passport Advantage website.
 - Download the product tar.gz file from Advanced Administration System (AAS). This system is also called Entitled Software Support (ESS).
2. Run the appropriate commands to install the product, depending on the platform you are installing on. There are RPM files for installation on RHEL (x86 and ppc64le) and DEB files for installation on Ubuntu (amd64 and ppc64le).

RHEL `rpm -i file_name.rpm`

Ubuntu

`dpkg -i file_name.deb`

Load the product Docker images with the appropriate container's tar file. The file name has this format: `powerai-vision-inference-<arch>-containers-<release>.tar`, where `<arch>` is x86 or ppc64le, and `<release>` is the product version being installed.

`/opt/powerai-vision/dnn-deploy-service/bin/load_images.sh -f <tar_file>`

PowerAI Vision Inference Server will be installed at `/opt/powerai-vision/dnn-deploy-service`.

Deploying a trained model

The following types of models can be deployed:

- Object detection using Faster R-CNN (default), tiny-YOLO V2, Detectron, custom TensorFlow models, and Keras models.
- Image classification using GoogLeNet (default) and custom TensorFlow models.

To deploy a model, run this command:

```
/opt/powerai-vision/dnn-deploy-service/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: Detectron 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 PowerAI Vision. It can be an image classification model or an object detection model, but must be in zip format.

Examples:

```
/opt/powerai-vision/dnn-deploy-service/bin/deploy_zip_model.sh --model dog --port 6001 --gpu 1 ./dog_classification.zip  
/opt/powerai-vision/dnn-deploy-service/bin/deploy_zip_model.sh --m car -p 6002 -g -1 /home/user/mydata/car.zip  
/opt/powerai-vision/dnn-deploy-service/bin/deploy_zip_model.sh -m coco -p 6001 -g 1 /home/ycheng/model/new_models/cdb-co
```

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/powerai-vision/dnn-deploy-service/bin/deploy_zip_model.sh -m coco -p 6001 -g 1 /home/ycheng/model/new_model
```

```
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 - PowerAI Vision Inference Server" on page 134 for details about dealing with errors.

- Ran out of GPU memory

- ```

root@d1dev4 ~]# /opt/powerai-vision/dnn-deploy-service/bin/deploy_zip_model.sh -m durga_detectron_cars8 -p 7018 -
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/co
root : INFO Callback message: {'msgId': '6ef7e371-1209-47b3-94c3-940640324ac8', 'msgReturnCode': 'ErrM
root : INFO Wait 5s for messaging completed...
[root@d1dev4 ~]#

```
- Invalid GPU ID specified

```

[root@d1dev4 ~]# /opt/powerai-vision/dnn-deploy-service/bin/deploy_zip_model.sh -m durga_detectron_cars8 -p 7018
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@d1dev4 ~]#

```
  - Processing was interrupted:

```

/usr/bin/docker-current: Error response from daemon: Conflict. The container name "/decrypt" is already in use by
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@d1dev4 ~]# /opt/powerai-vision/dnn-deploy-service/bin/deploy_zip_model.sh -m durga_detectron_cars8 -p 7018
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@d1dev4 ~]#

```
  - Deployment times out:

```

[root@d1dev4 ~]# /opt/powerai-vision/dnn-deploy-service/bin/deploy_zip_model.sh -t 15 -m durga_custom_cars3 -p 70
Deployment timed out at 15 seconds

If the deployment times out, increase the time limit by using the -t option.

```

## Inference

Inference can be done by using the deployed model with a local image file or a URL to an uploaded image file.

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/i98xa4dfpff6jwv01xmcu41ybr8b5kxj.jpg&confthre=0.7&containPoly
```

#### **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/ycheng/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/ycheng/model/new_models/pics/cars.jpg" -F "confthre=0.98" -F "containRle=true" -F "containPoly
```

## Inference output

The PowerAI Vision Inference Server 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": 1
```

#### Example 5 output - success

The output includes a rectangle and rle.

```
{"classified": [{"confidence": 0.9874554872512817, "ymax": 244, "rle": "RXb3h0e;e0^02nDcN1:b101000200000001000101N2
```

## 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
```

## Decrypting a trained model

You can decrypt a model that was trained by using PowerAI Vision by running `decrypt_zip_model`. This will allow data scientists to understand the weights and networks configured by PowerAI Vision and possibly use that information to further train the model. Alternatively, you might want to port these models to edge devices not supported by PowerAI Vision. A decrypted model cannot be used with PowerAI Vision.

**Usage:** `/opt/powerai-vision/dnn-deploy-service/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 PowerAI Vision.

### Example:

```
/opt/powerai-vision/dnn-deploy-service/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.

---

## 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 PowerAI Vision 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” 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 118.

## Inference on DeepRED

DeepRED is an embedded artificial intelligence development system that supports PowerAI Vision. 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 PowerAI Vision host operating system, run the appropriate command.
    - For a standard install:

```
$ /opt/powerai-vision/bin/kubect1.sh edit configmap powerai-vision-config
```
    - For PowerAI Vision installed on IBM Cloud Private:

```
$ kubect1 edit configmap powerai-vision-release_name-config
```
  - b. Find the row beginning `EMB_COD_IMAGE` in the configuration file and replace `ZC706` with `DEEPRED`:

```
"EMB_COD_IMAGE": ["DEEPRED,DEEPRED,powerai-vision-dnn-edge:1.1.4.0"],
```

c. Save and exit.

3. Restart PowerAI Vision by running the appropriate command. The deleted pods will automatically restart.

- For a standard install:

```
$ /opt/powerai-vision/bin/kubect1.sh delete pod -l app=powerai-vision
```

- For PowerAI Vision installed on IBM Cloud Private:

```
$ kubect1 delete pod -l app=powerai-vision-release_name
```

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/powerai-vision/volume/data/trained-models.

- For PowerAI Vision installed on IBM Cloud Private, it is stored in your Persistent Volume under `<PATH_TO_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, PowerAI Vision 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 PowerAI Vision host operating system, run the appropriate command.

- For a standard install:

```
$ /opt/powerai-vision/bin/kubect1.sh edit configmap powerai-vision-config
```

- For PowerAI Vision installed on IBM Cloud Private:

```
$ kubect1 edit configmap powerai-vision-release_name-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,powerai-vision-dnn-edge:1.1.4.0"]`, as shown here:

```
"EMB_COD_IMAGE": ["ZC706,DSP_NUM=700:RAM18E_NUM=800:DDR_BANDWIDTH=80000.0:
DDR_DATA_WIDTH=512:FPGA_TYPE=xcvu9p12fsgd2104e,powervision-dnn-edge:1.1.4.0"],
```

- c. Save and exit.
2. Restart PowerAI Vision by running the appropriate command. The deleted pods will automatically restart.
  - For a standard install:  
\$ /opt/powerai-vision/bin/kubect1.sh delete pod -l app=powervision
  - For PowerAI Vision installed on IBM Cloud Private:  
\$ kubect1 delete pod -l app=powervision-release\_name
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/powerai-vision/volume/data/trained-models.
  - For PowerAI Vision installed on IBM Cloud Private, it is stored in your Persistent Volume under `<PATH_TO_VOLUME>/data/trained-models`.



---

## PowerAI Vision REST APIs

You can use REST APIs to work with PowerAI Vision 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 PowerAI Vision processes or solutions.

For information about using the API see Vision Service API documentation. There are also examples of using the APIs for different actions, published here.



---

## 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 PowerAI Vision.

---

### Troubleshooting known issues - PowerAI Vision standard install

Following are some problems you might encounter when using PowerAI Vision, along with steps to fix them.

- “The PowerAI Vision user interface does not work”
- “Resource pages are not being populated in the user interface” on page 124
- “Unexpected / old pages displayed when accessing the user interface” on page 124
- “PowerAI Vision does not play video” on page 124
- “PowerAI Vision cannot train or deploy models after reboot” on page 124
- “Changing the port for the PowerAI Vision user interface” on page 125
- “Auto detection video does not play in Firefox browser” on page 125
- “Out of space error from load\_images.sh” on page 125
- “GPUs are not available for training or inference” on page 126
- “I forgot my user name or password” on page 126
- “PowerAI Vision cannot train a model” on page 127
- “Training or deployment hangs - Kubernetes pod cleanup” on page 128
- “Training fails with error indicating "You must retrain the model."” on page 128
- “Model fails to deploy with time out message” on page 128
- “Model training and inference fails” on page 129
- “Unexpected inference result using image with EXIF Orientation” on page 129
- “Model accuracy value is unexpected” on page 130
- “Deployed models stuck in "Starting"” on page 130
- “Auto labeling of a data set returns "Auto Label Error"” on page 130
- “PowerAI Vision does not start” on page 130
- “PowerAI Vision fails to start - Kubernetes connection issue” on page 131
- “PowerAI Vision startup hangs - helm issue” on page 132
- “Helm status errors when starting PowerAI Vision” on page 132
- “Uploading a large file fails” on page 133
- “Some PowerAI Vision functions don't work ” on page 133

### The PowerAI Vision 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 `powerai-vision-portal` 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 `powerai-vision-portal` pod. If that happens, follow these instructions: "Gather PowerAI Vision logs and contact support" on page 137.

## Unexpected / old pages displayed when accessing the user interface

### Problem

After updating, reinstalling, or restarting PowerAI Vision, 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`.

## PowerAI Vision 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.

## PowerAI Vision 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 PowerAI Vision from using the GPUs for training and deployment.

### Solution

Restart PowerAI Vision by running `powerai_vision_stop.sh` / `powerai_vision_start.sh`. This resets the problematic SELinux contexts if they are incorrect, restoring the ability to access GPUs for training and inference.

## Changing the port for the PowerAI Vision user interface

### Problem

By default, the PowerAI Vision 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 PowerAI Vision port.

1. If PowerAI Vision is running, use the following command to stop it:  

```
$ /opt/powerai-vision/bin/powerai_vision_stop.sh
```
2. Change `/opt/powerai-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/powerai-vision/sbin/firewall.sh
```
4. Restart PowerAI Vision by running the following command:  

```
$ /opt/powerai-vision/bin/powerai_vision_start.sh
```

## Auto detection video does not play in Firefox browser

### Problem

The Firefox browser reports “The media playback was aborted due to a corruption problem or because the media used features your browser did not support”. This happens in versions of the Firefox browser that do not support YUV444 chroma subsampling, which prevents the video from being played successfully.

### Solution

Use a version of Firefox that supports YUV444 chroma subsampling or use a different browser (such as Chrome) that does support it.

## Out of space error from load\_images.sh

### Problem

When installing the product, the `load_images.sh` script is used to load the PowerAI Vision 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 `powerai-vision-dnn` was not able to be fully loaded because of insufficient file system space:

```

root@kottos-vm1:~# 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/ /
root@kottos-vm1:~#

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...
root@kottos-vm1:~#

```

This situation can also be noted in the output from `/opt/powerai-vision/bin/kubect1 get pods`. This command is described in “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 PowerAI Vision 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 PowerAI Vision GUI.

### Solution

PowerAI Vision uses an internally managed users account database. To change your user name or password, see “Logging in to PowerAI Vision” on page 51.

## GPUs are not available for training or inference

### Problem

If PowerAI Vision 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 | 00000004:04:00.0 | Off           | 0       |
| N/A | 31C                | P0  | 49W / 300W       | Unknown Error | 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 22.

## PowerAI Vision cannot train a model

### Problem

The model training process might fail if your system does not have enough 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.
- Ensure that enough GPUs are available. GPUs are assigned as follows:
  - Each active training job takes one GPU.
  - Each Tiny YOLO V2, Detectron, or custom deployed model takes one GPU. The GPU group is listed as '-', which indicates that this model uses a full GPU and does not share the resource with any other deployed models.
  - Multiple Faster R-CNN and GoogLeNet models are deployed to a single GPU. PowerAI Vision 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:** PowerAI Vision leaves a 500MB buffer on the GPU.

If a training job appears to be hanging, it might be waiting for another training job to complete, or there might not be a GPU available to run it.

To determine how many GPUs are available on the system, view the GPU usage on the Models or Trained Models page in the user interface.

If all the systems GPUs are in use, you can either delete the group of deployed models that are using a GPU (making the models unavailable for inference) or you can stop model that is being trained. The deployed models that share a GPU have the same group number. To free up a GPU, all deployed models in one group must be deleted.

- To delete a deployed model, click **Deployed Models**. Next, select the model that you want to delete and click **Delete**. The trained model is not deleted from PowerAI Vision. You can redeploy the model later when more GPUs are available.
- To stop a training model that is running, click **Models**. Next, select the model that has a status of **Training in Progress** and click **Stop Training**.

## 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
powerai-vision-infer-ic-06767722-47df-4ec1-bd58-91299255f6hxxzk 1/1 Running 0 22m
powerai-vision-infer-ic-35884119-87b6-4d1e-a263-8fb645f0addqd2z 1/1 Running 0 22m
powerai-vision-infer-ic-7e03c8f3-908a-4b52-b5d1-6d2befec69ggqw5 1/1 Running 0 5h
powerai-vision-infer-od-clc16515-5955-4ec2-8f23-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 PowerAI Vision.

## 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 fails to deploy with time out message

### Problem

When using `deploy_zip_model.sh` to deploy a PowerAI Vision model, the action fails with a message of "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 of the GPU using the `nvidia-smi` command as described

in this topic: “Checking system GPU status” on page 49. For example, if the model failed to deploy to GPU 1, use `nvidia-smi -i 1` to check the usage of GPU 0 and stop or delete some of the models currently deployed to the GPU if there are limited memory resources. The following output demonstrates a situation where the GPU does not have sufficient memory to deploy the model:

```
/opt/powerai-vision/dnn-deploy-service/bin/deploy_zip_model.sh -m 8193_cars_custom_COD_model -p 7005 -g 1 /root
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 Usage |
|------------|-------|------|--------------|--|------------------|
| GPU        | PID   | Type | Process name |  |                  |
| 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         |

## Model training and inference fails

### Problem

The NVIDIA GPU device is not accessible by the PowerAI Vision Docker containers. To confirm this, run `kubectl logs -f _powerai-vision-portal-ID_` and then check `pod_powerai-vision-portal-ID_powerai-vision-portal.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.
```

### Solution

Use `sudo` to alter SELINUX permissions for all of the NVIDIA devices so they are accessible via the PowerAI Vision Docker containers.

```
sudo chcon -t container_file_t /dev/nvidia*
```

## Unexpected inference result using image with EXIF Orientation

### Problem

An unexpected result is received when using the REST API to perform an inference operation when using an image with EXIF Orientation specified. The PowerAI Vision deployed model does not use the EXIF Orientation to perform rotations of the provided image, which may cause an unexpected inference result.

### Solution

Rotate the image prior to providing to the REST API for inference. For example, the Linux tool `exiftran` can be used to rotate the image. Then, pass the rotated image to the REST API for inference.

## 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 58.

## Deployed models stuck in "Starting"

### Problem

PowerAI Vision 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 PowerAI Vision 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.

## PowerAI Vision does not start

### Problem

When you enter the URL for PowerAI Vision 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, check the Helm chart status by running this script:

```
sudo /opt/powerai-vision/bin/helm.sh status vision
```

In the script output, verify that the PowerAI Vision 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 **helm.sh status vision** script that shows all components are available:

```

RESOURCES:
==> v1beta1/Deployment
NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE
powerai-vision-mongodb 1 1 1 1 4d
powerai-vision-portal 1 1 1 1 4d
powerai-vision-postgres 1 1 1 1 4d
powerai-vision-taskanalyzer 1 1 1 1 4d
powerai-vision-ui 1 1 1 1 4d
powerai-vision-video-nginx 1 1 1 1 4d
powerai-vision-video-portal 1 1 1 1 4d
powerai-vision-video-rabbitmq 1 1 1 1 4d
powerai-vision-video-redis 1 1 1 1 4d
powerai-vision-video-test-nginx 1 1 1 1 4d
powerai-vision-video-test-portal 1 1 1 1 4d
powerai-vision-video-test-rabbitmq 1 1 1 1 4d
powerai-vision-video-test-redis 1 1 1 1 4d

```

If you recently started PowerAI Vision and some components are not available, wait a few minutes for these components to become available. If any components remain unavailable, gather the logs and contact IBM Support, as described in this topic: “Gather PowerAI Vision logs and contact support” on page 137.

3. If the docker0 interface is a member of a trusted zone and all PowerAI Vision components are available, verify that the firewall is configured to allow communication through port 443 (used to connect to PowerAI Vision) by running this command:

```
sudo firewall-cmd --permanent --zone=public --add-port=443/tcp
```

## PowerAI Vision fails to start - Kubernetes connection issue

### Problem

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/powerai-vision/bin/powerai_vision_start.sh
INFO: Setting up GPU...
[...]
Checking kubernetes cluster status...
The connection to the server 127.0.0.1:8080 was refused - did you specify the right host or port?
INFO: Probing cluster status #1:
The connection to the server 127.0.0.1:8080 was refused - did you specify the right host or port?
INFO: Probing cluster status #2:
The connection to the server 127.0.0.1:8080 was refused - did you specify the right host or port?
INFO: Probing cluster status #3:
The connection to the server 127.0.0.1:8080 was refused - did you specify the right host or port?
INFO: Probing cluster status #4:
The connection to the server 127.0.0.1:8080 was refused - did you specify the right host or port?
INFO: Probing cluster status #5:
The connection to the server 127.0.0.1:8080 was refused - did you specify the right host or port?
INFO: Probing cluster status #6:
The connection to the server 127.0.0.1:8080 was refused - did you specify the right host or port?
INFO: Probing cluster status #7:
The connection to the server 127.0.0.1:8080 was refused - did you specify the right host or port?
INFO: Probing cluster status #8:
The connection to the server 127.0.0.1:8080 was refused - did you specify the right host or port?
INFO: Probing cluster status #9:
The connection to the server 127.0.0.1:8080 was refused - did you specify the right host or port?
INFO: Probing cluster status #10:
The connection to the server 127.0.0.1:8080 was refused - did you specify the right host or port?
INFO: Probing cluster status #11:
ERROR: Retry timeout. Error in starting kubernetes cluster, please check /opt/powerai-vision/log/kubernetes for logs.

```

### Solution

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.

## PowerAI Vision startup hangs - helm issue

### Problem

PowerAI Vision startup hangs with the message "Unable to start helm within 30 seconds - trying again." For example:

```
root> sudo /opt/powerai-vision/bin/powerai_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 PowerAI Vision startup by pressing `ctrl+c`.

2. Stop PowerAI Vision by running this command:

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

3. Modify the RHEL settings as follows:

```
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 PowerAI Vision again:

```
sudo /opt/powerai-vision/bin/powerai_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 PowerAI Vision

### Problem

There is an issue in some RHEL releases that causes the startup of PowerAI Vision 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/powerai-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 1
```

## Solution

To solve this problem, you must follow these steps exactly as written:

1. Cancel PowerAI Vision startup by pressing `ctrl+c`.

2. Stop PowerAI Vision by running this command:

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

3. Modify the RHEL settings as follows:

```
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 PowerAI Vision again:

```
sudo /opt/powerai-vision/bin/powerai_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 will show the uploaded files in the data set.

## Some PowerAI Vision functions don't work

### Problem

PowerAI Vision 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/powerai-vision/bin/kubect1.sh get pods` and verify that one or more pods are in state `CrashLoopBackOff`. For example:

```
kubect1 get pods
NAME READY STATUS RESTARTS AGE
...
powerai-vision-video-rabmq-5d5d786f9f-7jfk9 0/1 CrashLoopBackOff 2 54s
```

## Solution

PowerAI Vision requires IPv6. Enable IPv6 on the system.

---

## Troubleshooting known issues - PowerAI Vision Inference Server

Following are some problems you might encounter when using PowerAI Vision, along with steps to fix them.

- “Problems installing an rpm on a RHEL system with Docker CE”
- “When deploying a model, you get an error that the /decrypt container name is already in use”
- “Unexpected inference result using image with EXIF Orientation”
- “Model fails to deploy on RHEL system with SE Linux” on page 135
- “Model fails to deploy with time out message” on page 135

### 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 "b9deb17c4651162aaf60". 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
```

### Unexpected inference result using image with EXIF Orientation

#### Problem

An unexpected result is received when using the REST API to perform an inference operation when using an image with EXIF Orientation specified. The PowerAI Vision deployed model does not use the EXIF Orientation to perform rotations of the provided image, which may cause an unexpected inference result.

#### Solution

Rotate the image prior to providing to the REST API for inference. For example, the Linux tool `exiftran` can be used to rotate the image. Then, pass the rotated image to the REST API for inference.



## Troubleshooting known issues - IBM Cloud Private install

Following are some problems you might encounter when using PowerAI Vision in an IBM Cloud Private (ICP) environment, along with steps to fix them.

### PowerAI Vision pods do not start - ICP installation

#### Problem

In an IBM Cloud Private installation, checking the status of the PowerAI Vision pods shows many in ContainerCreating and Init state.

Example:

```
kubectl get pods -o wide
NAME READY STATUS RESTARTS AGE IP
powerai-vision-icp-keycloak-7bff8db8b-8thfm 0/1 Init:0/1 0 7m 10.1.44.2
powerai-vision-icp-mongodb-5599969957-pgwwc 0/1 ContainerCreating 0 7m <none>
powerai-vision-icp-portal-6dcc65cfd9-n4fnr 0/1 Init:0/2 0 7m <none>
powerai-vision-icp-postgres-6ffc46dd59-hj6sq 0/1 Running 0 7m 10.1.44.2
powerai-vision-icp-taskanaly-97dff698-p5qcg 0/1 ContainerCreating 0 7m <none>
powerai-vision-icp-ui-5d64c856d6-mtzh9 0/1 ContainerCreating 0 7m <none>
powerai-vision-icp-video-nginx-77945f4cc9-9wjjq 0/1 ContainerCreating 0 7m <none>
powerai-vision-icp-video-portal-774c5b799d-sgntt 0/1 Init:0/1 0 7m <none>
powerai-vision-icp-video-rabmq-65bfb5799-c5knd 0/1 ContainerCreating 0 7m <none>
powerai-vision-icp-video-redis-fb67bb445-5r7s2 1/1 Running 0 7m 10.1.44.2
powerai-vision-icp-video-test-nginx-8675b6fd4d-rsf4b 0/1 Running 0 7m 10.1.44.2
powerai-vision-icp-video-test-portal-ccb4c4f8-dxhns 0/1 Init:0/1 0 7m <none>
powerai-vision-icp-video-test-rabmq-7bb766c575-2l4qm 0/1 ContainerCreating 0 7m <none>
powerai-vision-icp-video-test-redis-d5ffd75f7-8jjcr 1/1 Running 0 7m 10.1.44.2
```

The pod describe output for the pods that are not starting will also show events indicating problems with the underlying storage. For example:

```
Volumes:
 run-mount:
 Type: PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
 ClaimName: powerai-vision-icp-data-pvc
 ReadOnly: false
 default-token-hz9c4:
 Type: Secret (a volume populated by a Secret)
 SecretName: default-token-hz9c4
 Optional: false
QoS Class: BestEffort
Node-Selectors: beta.kubernetes.io/arch=ppc64le
Tolerations: node.kubernetes.io/not-ready:NoExecute for 300s
 node.kubernetes.io/unreachable:NoExecute for 300s

Events:
 Type Reason Age From Message
 ---- -
 Warning FailedMount 24m (x28 over 103m) kubelet, 10.10.10.4 Unable to mount volumes for pod "powerai-vision-icp-video-test-portal-ccb4c4f8-dxhns": MountVolume.SetUp failed for volume "powerai-vision-icp-data-pvc" with error: mount.nfs: mounting 10.10.10.1:/data/nfs/icp failed, reason given by server: No such file or directory
 Warning FailedMount 5m42s (x59 over 103m) kubelet, 10.10.10.4 (combined from similar events): MountVolume.SetUp failed for volume "powerai-vision-icp-data-pvc" with error: mount.nfs: mounting 10.10.10.1:/data/nfs/icp failed, reason given by server: No such file or directory

Mounting command: systemd-run
Mounting arguments: --description=Kubernetes transient mount for /var/lib/kubelet/pods/dbe0edd0-8375-11e9-b64b-1463
Output: Running scope as unit run-9236.scope.
mount.nfs: mounting 10.10.10.1:/data/nfs/icp failed, reason given by server: No such file or directory
```

#### Solution

The problem is likely that the persistent volume claim is not being bound to a valid persistent volume.

1. Log in to the ICP environment. See “Checking the application status in an ICP installation” on page 36 for instructions.
2. Check the status of the storage and ensure that the state is “Bound:” by following the steps in this topic: “Checking Kubernetes storage status” on page 43.
3. If the storage is not correctly bound, fix the problem then redeploy the application.

## PowerAI Vision Training and Deployed Model pods cannot access GPUs

### Problem

When trying to train or deploy a model, the operation fails. Logs gathered may indicate issues with GPU initialization, noted by non-zero cudaSuccess values. For example:

```
root : INFO F0510 14:51:21.103844 23 common.cpp:159] Check failed: error == cudaSuccess (3 vs. 0)
root : INFO *** Check failure stack trace: ***
root : INFO APPMSG: {'status': 'aborted', 'type': 'status_msg'}
```

### Solution

Ensure that GPUs are visible in the ICP dashboard. If they are, then ensure that the `nvidia-container-runtime-hook` is not installed on the system:

```
rpm -qa | grep nvidia-container-runtime-hook
nvidia-container-runtime-hook-1.4.0-2.ppc64le
```

The ICP environment provides a GPU plug-in container, and the `nvidia-container-runtime-hook` must be uninstalled.

After uninstalling the `nvidia-container-runtime-hook`, restart the ICP services on the node by following these instructions: .

**Important:** Before stopping the `kubelet` and `docker` service on the node, mark the node as `unschedulable`. Run the following command:

```
kubectl cordon 9.111.255.122
```

**Note:** Marking the node as `unschedulable` disables scheduling new pods on the node.

1. Shut down the system by stopping the `kubelet` on the target node by running the following command:

```
sudo systemctl stop kubelet
```

Allow the **kubelet** services time to quiesce - this can take up to one minute.

2. Stop the `docker` containers or the `docker` runtime by running the following command:

```
sudo systemctl stop docker
```

3. Restart the `Docker` by running the following command:

```
sudo systemctl start docker
```

4. Restart the `kubelet` and ensure that it is running successfully by running the following command:

```
sudo systemctl start kubelet
sudo systemctl status kubelet
```

5. If the `kubelet` service is unsuccessful, view the logs for the `kubelet` by running the following command:

```
sudo journalctl -e -u kubelet
```

6. Exit maintenance by running the following command:

```
kubectl uncordon 9.111.255.122
```

---

## Gather PowerAI Vision 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.

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:

1. Collect logs from the PowerAI Vision 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/powerai-vision/bin/kubectl logs `sudo /opt/powerai-vision/bin/kubectl get pods -o custom-columns=NAME:.m
```

- **Collect all logs:**

Run the **sudo /opt/powerai-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 PowerAI Vision Application Logs...
INFO: Collecting PowerAI Infrastructure Logs...
INFO: Collecting configuration information...
INFO: Collecting System Details...
INFO: Collecting Platform Logs...
INFO: FFDC Collected below:
-rw-r--r--. 1 root root 95477342 May 22 18:15 /var/log/powerai-vision/powerai-vision.logs.18_15_11_May_22_2019.tgz
```

The log files to provide are generated here: **/tmp/kubectl\_logs\***.

- **IBM Cloud Private installation:**

- a. Enable the **kubectl** command. For instructions, see this topic in the IBM Cloud Private Knowledge Center: [Accessing your IBM® Cloud private cluster by using the kubectl CLI](#).
- b. **Collect all logs** from the PowerAI Vision pods using the `<release_name>` specified when installing/deploying:

```
kubectl get pods > /tmp/kubectl_pod_status.txt
for i in $(kubectl get pods -o name | grep <release_name>); do
 kubectl logs $i > /tmp/kubectl_logs_${i}.txt
 kubectl describe pods $i > /tmp/kubectl_describe_pods_${i}.txt
done
kubectl describe deploy > /tmp/kubectl_deploy.txt
kubectl describe configmap > /tmp/kubectl_cm.txt
```

The log files to provide are generated here: **/tmp/kubectl\_logs\***.

- c. **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/powerai-vision/bin/kubectl logs `sudo /opt/powerai-vision/bin/kubectl get pods -o custom-columns=NAME:.m
```

2. Optionally, you can obtain the logs for a single pod of the application.

- a. Use the `kubectl get pods` command to view the running pods for the application. See “`kubectl.sh get pods`” on page 39. For example:

```
/opt/powerai-vision/bin/kubect1.sh get pods
NAME READY STATUS RESTARTS AGE
powerai-vision-dnn-infer-bb06a6f3-75d0-4614-96fb-a3d69f79dvxjh 1/1 Running 0 11h
powerai-vision-elasticsearch-855648d6d4-7jwct 1/1 Running 0 5d2h
powerai-vision-fpga-device-plugin-qjhsd 1/1 Running 0 5d2h
powerai-vision-keycloak-84cfc79796-z15qk 1/1 Running 0 5d2h
powerai-vision-logstash-7b48f978d7-pwscv 1/1 Running 0 5d2h
powerai-vision-mongodb-57bcd46cdd-vqs2b 1/1 Running 0 5d2h
powerai-vision-postgres-795987c56c-w6x62 1/1 Running 0 5d2h
powerai-vision-service-5588ffdfcc-cnq8h 1/1 Running 0 5d2h
powerai-vision-taskanalyzer-6448c56f89-918mf 1/1 Running 0 5d2h
powerai-vision-ui-5f78b785cf-xxfm6 1/1 Running 0 5d2h
powerai-vision-video-nginx-799bc6d9f6-6c27r 1/1 Running 0 5d2h
powerai-vision-video-portal-c9b6cd996-qrj68 1/1 Running 0 5d2h
powerai-vision-video-rabmq-6b8dcfd69b-w5wqb 1/1 Running 0 5d2h
powerai-vision-video-redis-6d455598b4-5688j 1/1 Running 0 5d2h
powerai-vision-video-test-nginx-7bbc9ddcfc-6xhwm 1/1 Running 0 5d2h
powerai-vision-video-test-portal-84f7b7f55b-n7jxv 1/1 Running 0 5d2h
powerai-vision-video-test-rabmq-59f59cc9b-4rbfh 1/1 Running 0 5d2h
powerai-vision-video-test-redis-5c78bb7f48-ffg9q 1/1 Running 0 5d2h
```

- b. Run the following command, where *<pod-name>* is obtained from the `kubect1.sh get pods` command:

```
kubect1.sh logs <pod-name> > <outputfile>
```

For example, using the above output, to collect logs in the file `vision-service.log`, run the command:

```
$ kubect1.sh logs powerai-vision-service-5588ffdfcc-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/software/support/>: 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 PowerAI Vision. With Fix Central, you can search, select, order, and download fixes for your system with a choice of delivery options. A PowerAI Vision product fix might be available to resolve your problem.

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 "PowerAI Vision" in the **Product selector** field.
3. Select PowerAI Vision. 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 PowerAI Vision by using this script:

```
sudo /opt/powerai-vision/bin/powerai_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/powerai-vision/bin/load_images.sh ./<fixpack-tarfile>.tar
```
9. Start PowerAI Vision by running the following script. You must read and accept the license agreement that is displayed before you can use PowerAI Vision.  

```
sudo /opt/powerai-vision/bin/powerai_vision_start.sh
```

---

## Contacting IBM Support

IBM Support provides assistance with product defects, answers FAQs, and helps users resolve problems with the product.

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.

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.

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.

---

## 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](http://www.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](http://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.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.



---

## 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 PowerAI Vision 1.1.4 Release Notes®

## Requirements

For hardware and software requirements, see the “Planning for PowerAI Vision” on page 15 topic.

## Installing

You can install PowerAI Vision stand-alone or PowerAI Vision with IBM Cloud Private. For more information, see the “Installing, upgrading, and uninstalling PowerAI Vision” on page 21 topic.

## Limitations

Following are some limitations for IBM PowerAI Vision 1.1.4:

- PowerAI Vision 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 Power System 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 PowerAI Vision stand-alone on a system that already has any of these products installed:
  - IBM Data Science Experience (DSX)
  - IBM Cloud Private
  - 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](http://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](http://www.ibm.com/legal/copytrade.shtml).

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

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..







Printed in USA