ALIBABA CLOUD

Alibaba Cloud

Machine Learning Platform for AI

Modelhub-Model Repository

Document Version: 20220610

Alibaba Cloud

Legal disclaimer

Alibaba Cloud reminds you to carefully read and fully understand the terms and conditions of this legal disclaimer before you read or use this document. If you have read or used this document, it shall be deemed as your total acceptance of this legal disclaimer.

1. You shall download and obtain this document from the Alibaba Cloud website or other Alibaba Cloud-authorized channels, and use this document for your own legal business activities only. The content of this document is considered confidential information of Alibaba Cloud. You shall strictly abide by the confidentiality obligations. No part of this document shall be disclosed or provided to any third party for use without the prior written consent of Alibaba Cloud.

2. No part of this document shall be excerpted, translated, reproduced, transmitted, or disseminated by any organization, company or individual in any form or by any means without the prior written consent of Alibaba Cloud.

3. The content of this document may be changed because of product version upgrade, adjustment, or other reasons. Alibaba Cloud reserves the right to modify the content of this document without notice and an updated version of this document will be released through Alibaba Cloud-authorized channels from time to time. You should pay attention to the version changes of this document as they occur and download and obtain the most up-to-date version of this document from Alibaba Cloud-authorized channels.

4. This document serves only as a reference guide for your use of Alibaba Cloud products and services. Alibaba Cloud provides this document based on the "status quo", "being defective", and "existing functions" of its products and services. Alibaba Cloud makes every effort to provide relevant operational guidance based on existing technologies. However, Alibaba Cloud hereby makes a clear statement that it in no way guarantees the accuracy, integrity, applicability, and reliability of the content of this document, either explicitly or implicitly. Alibaba Cloud shall not take legal responsibility for any errors or lost profits incurred by any organization, company, or individual arising from download, use, or trust in this document. Alibaba Cloud shall not, under any circumstances, take responsibility for any indirect, consequential, punitive, contingent, special, or punitive damages, including lost profits arising from the use or trust in this document (even if Alibaba Cloud has been notified of the possibility of such a loss).

5. By law, all the contents in Alibaba Cloud documents, including but not limited to pictures, architecture design, page layout, and text description, are intellectual property of Alibaba Cloud and/or its affiliates. This intellectual property includes, but is not limited to, trademark rights, patent rights, copyrights, and trade secrets. No part of this document shall be used, modified, reproduced, publicly transmitted, changed, disseminated, distributed, or published without the prior written consent of Alibaba Cloud and/or its affiliates. The names owned by Alibaba Cloud shall not be used, published, or reproduced for marketing, advertising, promotion, or other purposes without the prior written consent of Alibaba Cloud. The names owned by Alibaba Cloud include, but are not limited to, "Alibaba Cloud", "Aliyun", "HiChina", and other brands of Alibaba Cloud and/or its affiliates, which appear separately or in combination, as well as the auxiliary signs and patterns of the preceding brands, or anything similar to the company names, trade names, trademarks, product or service names, domain names, patterns, logos, marks, signs, or special descriptions that third parties identify as Alibaba Cloud and/or its affiliates.

6. Please directly contact Alibaba Cloud for any errors of this document.

Document conventions

Style	Description	Example
Danger	A danger notice indicates a situation that will cause major system changes, faults, physical injuries, and other adverse results.	**Danger:** Resetting will result in the loss of user configuration data.
Warning	A warning notice indicates a situation that may cause major system changes, faults, physical injuries, and other adverse results.	**Warning:** Restarting will cause business interruption. About 10 minutes are required to restart an instance.
Notice	A caution notice indicates warning information, supplementary instructions, and other content that the user must understand.	**Notice:** If the weight is set to 0, the server no longer receives new requests.
Note	A note indicates supplemental instructions, best practices, tips, and other content.	**Note:** You can use Ctrl + A to select all files.
>	Closing angle brackets are used to indicate a multi-level menu cascade.	Click **Settings**> **Network**> **Set network type**.
Bold	Bold formatting is used for buttons , menus, page names, and other UI elements.	Click **OK**.
`Courier font`	Courier font is used for commands	Run the `cd /d C:/window` command to enter the Windows system folder.
Italic	Italic formatting is used for parameters and variables.	`bae log list --instanceid` *Instance_ID*
[] or [a|b]	This format is used for an optional value, where only one item can be selected.	`ipconfig [-all|-t]`
{} or {a|b}	This format is used for a required value, where only one item can be selected.	`switch {active|stand}`

Table of Contents

1.Intelligent video processing models

Model Hub of Machine Learning Platform for AI (PAI) provides a wide range of intelligent video processing models. This topic describes the features, input formats, and output formats of these models and provides examples.

Background information

The following table describes the intelligent video processing models that are provided by Model Hub of PAI.

Model	Description
General video classification model	Uses the UCF101 dataset for model training based on the ResNet3D framework.
Video highlight generation model	Generates a five-second clip of video highlights from a video.
Super-resolution model	Uses super-resolution technologies to improve video resolution.

Go to Model Hub

To go to Model Hub, perform the following steps:

1. Log on to the Machine Learning Platform for AI console.
2. In the left-side navigation pane, click **Workspaces**. On the Workspace list page, click the name of the workspace that you want to manage.
3. In the left-side navigation pane, choose **AI Computing Asset Management > Models**.
4. On the **Model Management** page, click the **Model Hub** tab.

General video classification model

- Overview

  The general video classification model uses the UCF101 dataset for model training based on the ResNet3D framework. For more information, see Can Spatiotemporal 3D CNNs Retrace the History of 2D CNNs and ImageNet?

- Input format

  The input data must be in the JSON format. It contains the url and play_duration fields. The value of the url field is the URL of the input video. The value of the play_duration field is the first several microseconds of the input video to be processed for classification. If the play_duration field is not specified, the total length of the input video is processed. The following code provides an example of the input data:

```
{
  "input": {
    "url": "URL of the input video",
    "play_duration": "Length of the input video clip to be processed"
  }
}
```

- Output format

  The output data is key-value pairs (KVPs) in the JSON format. The following table describes the fields in the output data.

Field	Description	Dimension	Type
class	The name of the category.	[]	STRING

- Example

  The following code provides an example of the input data of the model:

```
{"input" : {"url": "http://pai-vision-data-sh.oss-cn-shanghai-internal.aliyuncs.com/tmp/50005632219.mp4"}}
```

  PAI displays information that is similar to the following output:

```
null
```

Video highlight generation model

- Overview

  This model generates a five-second clip of video highlights from a video.

- Input format

  The input data must be in the JSON format. It contains the url and play_duration fields. The value of the url field is the URL of the input video. The value of the play_duration field is the first several microseconds of the input video to be processed to generate the clip. If the play_duration field is not specified, the total length of the input video is processed. The following code provides an example of the input data:

```
{
  "input": {
    "url": "URL of the input video",
    "play_duration": "Length of the input video clip to be processed"
  }
}
```

- Output format

  The output data is KVPs in the JSON format. The following table describes the fields in the output data.

Field	Description	Dimension	Type

Field	Description	Dimension	Type
oss_path	The Object Storage Service (OSS) path of the generated clip.	[]	STRING
result	The result field.	[]	STRING
success	Indicates whether the clip is generated. Valid values: ○ true: The clip is generated. ○ false: The clip failed to be generated.	[]	BOOL

- Example

  The following code provides an example of the input data of the model:

```
{"input" : {"url": "http://pai-vision-data-sh.oss-cn-shanghai-internal.aliyuncs.com/tmp/50005632219.mp4"}}
```

  PAI displays information that is similar to the following output:

```
{
   "oss_path": "oss://experience-ai/video_5s/50005632219.mp4",
   "result": "",
   "success": true
}
```

Super-resolution model

- Overview

  The super-resolution model uses super-resolution technologies to improve video resolution.

- Input format

  The input data is must be in the JSON format. The following table describes the fields in the input data.

Field	Required	Description	Type
url	Yes	The URL of the input video.	STRING
play_duration	No	The first several microseconds of the input video from which a clip with higher resolution to be generated. If the play_duration field is not specified, the entire input video is processed.	STRING

  The following code provides an example of the input data:

```
{
  "input": {
    "url": "URL of the input video",
    "play_duration": "Length of the input video clip to be processed"
  }
}
```

- Output format

Field	Description	Dimension	Type
oss_path	The OSS path of the generated clip.	[]	STRING
success	Indicates whether a clip with higher resolution is obtained after super resolution. Valid values: ○ true: A clip with higher resolution is obtained after super resolution. ○ false: A clip with higher resolution failed to be obtained after super resolution.	[]	BOOL
result	The result field.	[]	STRING

- Example

  The following code provides an example of the input data of the model:

```
{
    "input" : {
        "url": "http://pai-vision-data-sh.oss-cn-shanghai-internal.aliyuncs.com/tmp/50005
63****.mp4"
    }
}
```

  PAI displays information that is similar to the following output:

```
{
    "oss_path": "oss://experience-ai/video_sr/5000563****.mp4",
    "result": "",
    "success": true
}
```

2.Intelligent image processing models

Model Hub of Machine Learning Platform for AI (PAI) provides a variety of trained intelligent image processing models for you to use. This topic describes the input format and output format of these models and provides examples.

The following table describes the trained intelligent image processing models that are provided by Model Hub of PAI. These models help you conduct business in a convenient way.

Model	Feature
General image classification model	Returns the categories of recognized objects in images.
General image recognition model	Returns the categories and positions of recognized objects in images.
Semantic segmentation model	Returns the categories of segmented objects and the information about segmentation masks.
Instance segmentation model	Returns the categories and locations of segmented objects, and the information about segmentation masks.
General OCR model	Detects and recognizes text.
Foreground segmentation model	Segments human figures from short videos and live streams.
Scenario classification model	Recognizes various indoor and outdoor scenarios, such as skies, beaches, blue skies, kitchens, and concert halls, in images.
Inventory counting model	Returns the categories of recognized commodities in images, the coordinates of the bounding boxes that mark recognized commodities, and the number of recognized commodities in each category.
Image similarity comparison model	Returns the similarity between two images. This model can be used for image comparison and retrieval.
Image coloring model	Returns the colored image.

Go to Model Hub

To go to Model Hub, perform the following steps:

1. Log on to the Machine Learning Platform for AI console.
2. In the left-side navigation pane, click **Workspaces**. On the Workspace list page, click the name of the workspace that you want to manage.
3. In the left-side navigation pane, choose **AI Computing Asset Management > Models**.
4. On the **Model Management** page, click the **Model Hub** tab.

General image classification model

- Overview

  The general image classification model is trained by using the ImageNet dataset. This model returns the categories of recognized objects. The general image classification model uses the ResNet framework. For more information, see Deep Residual Learning for Image Recognition.

- Input format

  The input data must be in the JSON format. It contains the image field. The value of this field is the image content that is encoded in the Base64 format.

```
{
  "image": "Base64-encoded image content"
}
```

- Output format

  The output data is in the JSON format. The following table describes the fields in the output data.

Parameter	Description	Shape	Type
class	The ID of the category.	[]	INT32
class_name	The name of the category.	[]	STRING
class_probs	The probabilities for all categories.	[num_classes]	Dict[STRING, FLOAT]
request_id	The unique ID of the request.	[]	STRING
success	Indicates whether the request was successful.	[]	BOOL
error_code	The error code that is returned if the request failed.	[]	INT
error_msg	The error message that is returned if the request failed.	[]	STRING

  The following code provides an example of the output data:

```
{
"class": 3,
"class_name": "coho4",
"class_probs": {"coho1": 4.028851974258174e-10,
          "coho2": 0.48115724325180054,
          "coho3": 5.116515922054532e-07,
          "coho4": 0.5188422446937221},
 "request_id": "9ac294a4-f387-4c48-b640-d2c6d41f****",
 "success": true
}
```

- Test data

  Download the data to test the general image classification model.

General image recognition model

- Overview

  The general image recognition model uses the Faster R-CNN framework. This model returns the categories and positions of recognized objects in images. For more information, see Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. The general image recognition model is trained by using the COCO dataset.

- Input format

  The input data must be in the JSON format. It contains the image field. The value of this field is the image content that is encoded in the Base64 format.

```
{
  "image": "Base64-encoded image content"
}
```

- Output format

  The output data is in the JSON format. The following table describes the fields in the output data.

Parameter	Description	Shape	Type
detection_boxes	The bounding boxes that mark the recognized objects. The coordinates of each bounding box [y1, x1, y2, x2] are specified in the [top, left, bottom, right] order.	[num_detections, 4]	FLOAT
detection_scores	The probabilities that the objects are recognized.	num_detections	FLOAT
detection_classes	The IDs of the categories to which the objects belong.	num_detections	INT
detection_class_names	The names of the categories to which the objects belong.	num_detections	STRING
request_id	The unique ID of the request.	[]	STRING
success	Indicates whether the request was successful.	[]	BOOL
error_code	The error code that is returned if the request failed.	[]	INT
error_msg	The error message that is returned if the request failed.	[]	STRING

The following code provides an example of the output data:

```
{
  "detection_boxes": [[243.5308074951172, 197.69570922851562, 385.59625244140625, 247.724
7772216797], [292.1929931640625, 114.28043365478516, 571.2748413085938, 165.0977172851562
5]],
  "detection_scores": [0.9942291975021362, 0.9940272569656372],
  "detection_classes": [1, 1],
  "detection_classe_names": ["text", "text"],
  "request_id": "9ac294a4-f387-4c48-b640-d2c6d41f****",
  "success": true
 }
```

Semantic segmentation model

- Overview

  The semantic segmentation model uses the DeepLab V3 framework. This model returns the categories of segmented objects and the information about segmentation masks. For more information, see Rethinking Atrous Convolution for Semantic Image Segmentation. The semantic segmentation model is trained by using the Pascal_Voc dataset.

- Input format

  The input data must be in the JSON format. It contains the image field. The value of this field is the image content that is encoded in the Base64 format.

  ```
  {
    "image": "Base64-encoded image content"
  }
  ```

- Output format

  The output data is in the JSON format. The following table describes the fields in the output data.

Parameter	Description	Shape	Type
probs	The probabilities that the pixels obtained after segmentation belong to specific categories.	[output_height, output_width]	FLOAT
preds	The IDs of the categories to which the pixels obtained after segmentation belong.	[output_height, output_widths]	INT
request_id	The unique ID of the request.	[]	STRING
success	Indicates whether the request was successful.	[]	BOOL

Parameter	Description	Shape	Type
error_code	The error code that is returned if the request failed.	[]	INT
error_msg	The error message that is returned if the request failed.	[]	STRING

The following code provides an example of the output data:

```
{
  "probs" : [[[0.8, 0.8], [0.6, 0.7]],[[0.8, 0.5], [0.4, 0.3]]],
  "preds" : [[1,1], [0, 0]],
   "request_id": "9ac294a4-f387-4c48-b640-d2c6d41f****",
   "success": true
}
```

- Test data

  Download the data to test the semantic segmentation model.

Instance segmentation model

- Overview

  The instance segmentation model uses the Mask R-CNN framework. This model returns the categories and positions of recognized objects, and the information about segmentation masks. For more information, see Mask R-CNN. The instance segmentation model is trained by using the COCO dataset.

- Input format

  The input data must be in the JSON format. It contains the image field. The value of this field is the image content that is encoded in the Base64 format.

```
{
  "image": "Base64-encoded image content"
}
```

- Output format

  The output data is in the JSON format. The following table describes the fields in the output data.

Parameter	Description	Shape	Type
detection_boxes	The bounding boxes that mark the recognized objects. The coordinates of each bounding box [y1, x1, y2, x2] are specified in the [top, left, bottom, right] order.	[num_detections, 4]	FLOAT

Parameter	Description	Shape	Type
detection_scores	The probabilities that the objects are recognized.	num_detections	FLOAT
detection_classes	The IDs of the categories to which the objects belong.	num_detections	INT
detection_class_names	The names of the categories to which the objects belong.	num_detections	STRING
detection_masks	The segmentation masks for the objects. The value of this field is the segmentation mask data that is encoded in the run-length encoding (RLE) format. Each segmentation mask contains the following two properties: ◦ size: the height and width of the mask image. ◦ counts: the RLE data of the mask. An odd term indicates the counts of consecutive False values. An even term indicates the counts of consecutive True values. For example, the RLE-encoded data of [True False, False, True] is [0,1,2,1]. After RLE data is decoded, you can reshape the mask data to a two-dimensional mask based on the size property.	[num_detections]	DICT
request_id	The unique ID of the request.	[]	STRING

Parameter	Description	Shape	Type
success	Indicates whether the request was successful.	[]	BOOL
error_code	The error code that is returned if the request failed.	[]	INT
error_msg	The error message that is returned if the request failed.	[]	STRING

The following code provides an example of the output data:

```
{
  "detection_boxes": [[243.5308074951172, 197.69570922851562, 385.59625244140625, 247.724
7772216797], [292.1929931640625, 114.28043365478516, 571.2748413085938, 165.0977172851562
5]],
  "detection_scores": [0.9942291975021362, 0.9940272569656372],
  "detection_classes": [1, 1],
  "detection_classe_names": ["text", "text"],
  "detection_masks": [{"counts": [398408, 11, 671, 30, 652, 44, 636], "size":[640, 480]},
                      {"counts": [398408, 11, 671, 30, 652, 44, 636], "size":[640, 480]}]
,
   "request_id": "9ac294a4-f387-4c48-b640-d2c6d41f****",
   "success": true
 }
```

General OCR model

- Overview

  The general optical character recognition (OCR) model uses the end-to-end OCR model that is developed by PAI. This model can detect and recognize text.

- Input format

  The input data must be in the JSON format. It contains the image field. The value of this field is the image content that is encoded in the Base64 format.

```
{
  "image": "Base64-encoded image content"
}
```

- Output format

  The output data is in the JSON format. The following table describes the fields in the output data.

Parameter	Description	Shape	Type

Parameter	Description	Shape	Type
detection_boxes	The bounding boxes that mark the recognized text areas. The coordinates of each bounding box are specified in the [top, left, bottom, right] order.	[num_detections, 4]	FLOAT
detection_scores	The probabilities that the text areas are detected.	num_detections	FLOAT
detection_classes	The IDs of the categories to which the text areas belong.	num_detections	INT
detection_class_names	The names of the categories to which the text areas belong.	num_detections	STRING
detection_keypoints	The four vertices of each text area that is detected. The coordinates of each vertex are specified in the (y, x) format.	[num_detections, 4, 2]	float
detection_texts_ids	The ID of the category to which a single line of the recognized text belongs.	[num_detections, max_text_length]	INT
detection_texts	The recognition result of each single-line text.	[num_detections]	STRING
detection_texts_scores	The probability that each single-line text is recognized.	[num_detections]	FLOAT
request_id	The unique ID of the request.	[]	STRING
success	Indicates whether the request was successful.	[]	BOOL
error_code	The error code that is returned if the request failed.	[]	INT

Parameter	Description	Shape	Type
error_msg	The error message that is returned if the request failed.	[]	STRING

The following code provides an example of the output data:

```
 {
  "detection_keypoints": [[[243.57516479492188, 198.84210205078125], [243.91038513183594, 247.62425231933594], [385.5513916015625, 246.61660766601562], [385.2197570800781, 197.79345703125]], [[292.2718200683594, 114.44700622558594], [292.2237243652344, 164.684814453125], [571.1962890625, 164.931640625], [571.2444458007812, 114.674331665039063]]],
  "detection_boxes": [[243.5308074951172, 197.69570922851562, 385.59625244140625, 247.7247772216797], [292.1929931640625, 114.28043365478516, 571.2748413085938, 165.09771728515625]],
  "detection_scores": [0.9942291975021362, 0.9940272569656372],
  "detection_classes": [1, 1],
  "detection_classe_names": ["text", "text"],
  "detection_texts_ids" : [[1,2,2008,12], [1,2,2008,12]],
  "detection_texts": ["This is an example", "This is an example"],
  "detection_texts_scores" : [0.88, 0.88],
  "request_id": "9ac294a4-f387-4c48-b640-d2c6d41f****",
  "success": true
 }
```

- Test data

  Download the data to test the general OCR model.

Foreground segmentation model

- Overview

  The foreground segmentation model uses the MobileNet framework. This model can segment human figures from short videos and live streams.

- Input format

  The input data must be in the JSON format. It contains the url field that specifies the URL of the image. The following code provides an example of the input data:

```
{
  "input": {
    "url": "Image URL"
  }
}
```

- Output format

  The output data is in the JSON format. The following table describes the fields in the output data.

Parameter	Description	Shape	Type

Parameter	Description	Shape	Type
human_ratio	The ratio of the pixels of the foreground to the pixels of the image.	[]	STRING
mask	The segmentation mask for the foreground.	[h,w]	LIST

- Example

  The following code provides an example of the input data of the model:

```
{"input" : {"url": "http://yq****.oss-cn-hangzhou-zmf.aliyuncs.com/tb_quality.png"}}
```

  For more information about the test result, see Sample result.

Scenario classification model

- Overview

  The scenario classification model uses the ResNet framework. For more information, see Deep Residual Learning for Image Recognition. This model can recognize various indoor and outdoor scenarios, such as skies, beaches, blue skies, kitchens, and concert halls, in images.

- Input format

  The input data must be in the JSON format. It contains the image field. The value of this field is the image content that is encoded in the Base64 format.

```
{
  "image": "Base64-encoded image content"
}
```

- Output format

  The output data is in the JSON format. The following table describes the fields in the output data.

Parameter	Description	Shape	Type
class	The IDs of the top five categories.	5	INT32
class_name	The names of the top five categories.	5	STRING
class_probs	The probabilities for all categories.	[num_classes]	Dict[STRING, FLOAT]
request_id	The unique ID of the request.	[]	STRING

Parameter	Description	Shape	Type
success	Indicates whether the request was successful. Valid values: ○ true: indicates that the request was successful. ○ false: indicates that the request failed.	[]	BOOL
error_code	The error code that is returned if the request failed.	[]	INT
error_msg	The error message that is returned if the request failed.	[]	STRING

The following code provides an example of the output data:

```
{
    "request_id": "a72304d8-cf84-479e-b29e-5e341e3d****",
    "success": true,
    "class": [266, 57, 159,  260,  243],
    "class_name": [
        "pier",
        "boardwalk",
        "gazebo-exterior",
        "pavilion",
        "ocean"
    ],
    "class_probs": {
        "airfield": 2.6841306066671677e-8,
        "airplane_cabin": 2.4176902702066627e-9,
        "airport_terminal": 1.3229835360561992e-7,
        "alcove": 1.5998873337252917e-8,
        "alley": 7.053529316181084e-8,
        "amphitheater": 2.0278820400676523e-8,
        "amusement_arcade": 1.5128257757623942e-8,
        "amusement_park": 6.29929459705636e-8,
        "apartment_building-outdoor": 1.846876926947516e-7,
        "aquarium": 6.034031940771456e-8,
        "aqueduct": 1.6192875307297072e-7,
        "arcade": 3.719276833180629e-7,
        "arch": 0.000001615617293282412,
        "archaelogical_excavation": 1.9157377906253714e-9,
        "archive": 1.915566905097421e-8
    }
}
```

Inventory counting model

- Overview

  The inventory counting model integrates the YOLOv5 model and fine-grained classification model in two different phases. This model returns the categories of recognized commodities in images, the coordinates of the bounding boxes that mark recognized commodities, and the number of recognized commodities in each category. This model supports 171 regular categories of bottled drinks, each category identified by a unique stock keeping unit (SKU). The following information shows the complete list.

```
CLASSES = ['189', '772', '773', '307', '306', '305', '304', '303', '757', '342', '343', '
344', '5', '346', '347', '348', '349', '438', '439', '440', '441', '341', '460', '457', '
764', '459', '469', '329', '331', '332', '333', '334', '762', '763', '337', '338', '340',
'471', '470', '336', '335', '767', '770', '769', '758', '455', '456', '454', '446', '775'
, '761', '778', '777', '779', '789', '780', '453', '452', '451', '450', '449', '448', '44
4', '447', '445', '472', '468', '190', '759', '195', '196', '301', '300', '188', '186', '
185', '184', '183', '182', '181', '308', '309', '310', '311', '734', '312', '313', '355',
'339', '774', '791', '180', '187', '198', '197', '368', '369', '299', '298', '374', '372'
, '373', '371', '370', '350', '351', '352', '353', '755', '754', '429', '432', '431', '75
3', '433', '434', '435', '436', '437', '718', '717', '716', '715', '714', '712', '430', '
354', '200', '6', '476', '477', '478', '479', '713', '474', '473', '768', '443', '442', '
321', '315', '318', '316', '322', '317', '319', '179', '320', '324', '323', '325', '326',
'327', '328', '330', '458', '776', '765', '766', '4', '461', '199', '462', '464', '466',
'467', '756', '465', '463', '719', '4345']
```

- Input format

  The input data must be in the JSON format. It contains the image field. The value of this field is the image content that is encoded in the Base64 format.

```
{
  "image": "Base64-encoded image content"
}
```

- Output format

  The output data is in the JSON format. The following table describes the fields in the output data.

Parameter	Description	Shape	Type
detection_boxes	The bounding boxes that mark the recognized objects. The coordinates of each bounding box [y1, x1, y2, x2] are specified in the [top, left, bottom, right] order.	[num_detections, 4]	FLOAT
detection_scores	The probabilities that the commodities are recognized.	num_detections	FLOAT
detection_classes	The IDs of the categories to which the commodities belong.	num_detections	INT

Parameter	Description	Shape	Type
detection_class_names	The names of the categories to which the commodities belong.	num_detections	STRING
product_count	The number of recognized commodities in each category.	num_classes	DICT /INT
request_id	The unique ID of the request.	[]	STRING
success	Indicates whether the request was successful.	[]	BOOL
error_code	The error code that is returned if the request failed.	[]	INT
error_msg	The error message that is returned if the request failed.	[]	STRING

The following code provides an example of the output data:


```
{
    "request_id": "c3e8572d-95fc-479c-8fa9-bfe5b9be****",
    "success": true,
    "ori_img_shape": [1280, 1706],
    "detection_boxes": [[1630.1121826171875, 548.7086791992188, 1702.461181640625, 763.65
09399414062], [1620.415283203125, 94.60894775390625, 1682.6427001953125, 233.499008178710
94], [1553.8291015625, 98.07244110107422, 1617.0072021484375, 235.3670196533203], [1172.0
789794921875, 777.32861328125, 1226.4949951171875, 959.086669921875], [772.0833129882812,
758.675048828125, 825.0681762695312, 913.5953369140625], [828.8756713867188, 760.62567138
67188, 882.7286376953125, 920.3554077148438], [987.754150390625, 1031.87841796875, 1044.6
632080078125, 1207.215576171875], [1111.7740478515625, 772.5474243164062, 1167.0968017578
125, 951.8721923828125], [886.6332397460938, 765.0888061523438, 938.5805053710938, 927.51
57470703125], [1345.31689453125, 1060.8140869140625, 1406.5404052734375, 1269.99304199218
75], [1371.146728515625, 112.70679473876953, 1427.5396728515625, 245.45986938476562], [15
60.9510498046875, 545.3655395507812, 1626.23876953125, 761.7933349609375], [1197.18005371
09375, 123.90782165527344, 1249.66357421875, 253.5529022216797], [1310.7371826171875, 115
.76203918457031, 1367.01904296875, 248.2571563720703], [1107.890380859375, 1023.406677246
0938, 1164.3438720703125, 1228.36865234375], [942.4257202148438, 771.8615112304688, 998.5
577392578125, 934.6956176757812], [1002.333984375, 777.1546020507812, 1060.7852783203125,
941.6340942382812], [1493.2554931640625, 544.4757080078125, 1557.4088134765625, 657.53100
5859375], [631.6664428710938, 322.0820007324219, 678.197265625, 486.94320678710941, [1407
.86865234375, 363.3863220214844, 1468.095458984375, 498.18130493164061, [653.595458984375
, 970.340087890625, 703.8126220703125, 1146.232666015625], [1173.056884765625, 538.817199
7070312, 1228.7896728515625, 644.2966918945312], [585.7047119140625, 321.9823913574219, 6
28.947021484375, 487.156982421875], [891.7557983398438, 336.12506103515625, 947.544738769
5312, 494.72930908203125], [1179.0208740234375, 368.91839599609375, 1231.1116943359375, 4
97.69680786132811, [730.7046508789062, 319.1811218261719, 779.738037109375, 489.429199218
```

75], [791.6107177734375, 540.0254516601562, 838.8424072265625, 710.7826538085938], [1289.136962890625, 367.61407470703125, 1343.65478515625, 497.8343505859375], [1049.073974609375, 1039.80419921875, 1103.576904296875, 1216.981689453125], [1072.4429931640625, 100.25904083251953, 1130.01171875, 186.9414825439453], [1026.8106689453125, 9.758943557739258, 1084.0970458984375, 96.64080810546875], [457.7298278808594, 948.81640625, 507.786376953125, 1111.3463134765625], [1234.4410400390625, 368.9875793457031, 1285.9132080078125, 496.5082092285156], [1133.8148193359375, 95.66622924804688, 1192.635986328125, 183.78764343261727], [1229.1578369140625, 1036.1744384765625, 1292.0968017578125, 1250.7900390625], [390.2349548339844, 719.7738647460938, 439.1220397949219, 872.43115234375], [1347.3271484375, 365.58258056640625, 1403.486572265625, 497.9493103027344], [622.7703857421875, 745.7404785156625, 667.4136352539062, 899.0189819335938], [811.9891967773438, 1007.6695556640625, 869.5931396484375, 1174.7613525390625], [1473.03759765625, 359.28070068359375, 1535.2049560546875, 498.5074462890625], [757.7261962890625, 998.3402099609375, 807.4312744140625, 1163.7625732421875], [1253.5140380859375, 120.7535171508789, 1306.639892578125, 249.9414825439453], [1087.5078125, 5.977339744567871, 1146.104736328125, 93.57987213134766], [1423.3592529296875, 543.2384033203125, 1487.9222412109375, 655.9434204101562], [842.3613891601562, 543.4869384765625, 894.4482421875, 713.8154296875], [443.619140625, 730.7120361328125, 485.4419860839844, 877.8041381835938], [539.8949584960938, 323.5038146972656, 583.0478515625, 486.302978515625], [706.948974609375, 976.8787841796875, 754.7720336914062, 1153.9981689453125], [929.3824462890625, 1025.2862548828125, 983.93212890625, 1199.8162841796875], [1012.2584228515625, 105.00666046142578, 1068.331787109375, 189.37049865722656], [1269.915771484375, 21.24080467224121, 1322.0224609375, 115.1187973022461], [577.4571533203125, 742.8727416992188, 619.8193969726562, 894.5945434570312], [541.6632080078125, 527.994140625, 582.3711547851562, 689.1560668945312], [1232.9144287109375, 540.1036376953125, 1291.3621826171875, 647.2518920898438], [692.6676635742188, 541.361083984375, 738.5997314453125, 703.142578125], [1353.6865234375, 788.33740234375, 1403.29052734375, 957.48583984375], [952.47900390625, 334.1702880859375, 1007.0637817382812, 494.97747802734375], [430.1739196777344, 535.9578857421875, 484.27825927734375, 686.00732421875], [1360.4755859375, 541.090576171875, 1417.861328125, 651.1341552734375], [946.75244140625, 545.697998046875, 995.1062622070312, 721.5091552734375], [969.4580078125, 13.584962844848633, 1023.1282958984375, 98.95903015136719], [670.2022705078125, 748.6325073242188, 714.4813842773438, 903.7423095703125], [1229.794189453125, 779.18212890625, 1278.884033203125, 948.625], [1488.021972656625, 661.7587280273438, 1551.5296630859375, 756.62371826171887], [1590.016357421875, 806.0292358398438, 1655.314453125, 994.7868041992188], [1477.2318115234375, 1076.368896484375, 1536.288818359375, 1277.616943359375], [1411.1800537109375, 1069.7974853515625, 1473.112548828125, 1275.5064697265625], [1621.7462158203125, 2.3413498401641846, 1680.2470703125, 90.46394348144531], [1325.664794921875, 18.60450553894043, 1378.037353515625, 111.772811889648447], [1214.715087890625, 25.299074172973633, 1264.5130615234375, 117.627014160156257], [1124.7440185546875, 369.00946044921875, 1175.7701416015625, 496.77911376953125], [741.9351196289062, 539.4319458007812, 789.4860229492188, 706.5347900390625], [1010.4663085937519, 319.15960693359375, 1061.8902587890625, 494.5950012207031], [348.3062744140625, 324.97149658203125, 393.9333190917969, 480.3936767578125], [510.96856689453125, 953.662109375, 555.7396240234375, 1118.6304931640625], [1066.5125732421875, 319.3757629394531, 1120.20947265625, 494.90948486328125], [1055.790771484375, 549.7044677734375, 1109.9053955078125, 727.7008666992188], [1172.795654296875, 648.4998779296875, 1227.1788330078125, 738.3115844726562], [585.5626831054688, 528.1153564453125, 631.716796875, 691.9830932617188], [559.0197143554688, 958.7254638671875, 609.9586791992188, 1125.0369873046875], [98.209953308105547, 328.4491882324219, 139.44583129882812, 475.39495849609375], [717.43896484375, 755.4843139648438, 768.76318359375, 908.827880859375], [1296.5169677734375, 540.7620849609375, 1355.987548828125, 650.4367065429688], [1231.087646484375, 650.8126220703125, 1289.8023681640625, 742.1888427734375], [319.880615234375, 935.620849609375, 359.2186584472656, 1089.6805419921875], [498.1044921875, 526.2054443359375, 539.1546630859375, 686.7112426757812], [1529.4716796875, 801.7390747070312, 1585.5760498046875, 965.0985717773438], [141.8803253173828, 326.2720031738281, 178.31863403320312, 476.84689331054691], [180.48562622070312

, 325.25469970703125, 218.10015869140625, 477.2813720703125], [1356.2423095703125, 655.66
92504882812, 1414.081787109375, 748.3955688476562], [644.6302490234375, 539.3516845703125
, 690.34033203125, 699.429443359375], [1132.5467529296875, 189.2131805419922, 1190.796875
, 256.9765625], [1150.775390625, 2.947110891342163, 1208.0450439453125, 89.4374008178711]
, [1558.8004150390625, 4.572126865386963, 1617.202392578125, 93.98892974853516], [999.533
203125, 546.2440185546875, 1051.2882080078125, 725.1801147460938], [1658.9302978515625, 8
09.345458984375, 1706.0, 987.3961791992188], [301.7475280761719, 322.6171875, 345.1357727
050781, 479.57415771484375], [1441.016845703125, 10.674417495727539, 1494.4871826171875,
104.4920883178711], [0.0, 332.31707763671875, 27.537397384643555, 473.57806396484375], [1
168.52880859375, 1028.2525634765625, 1224.62646484375, 1242.695556640625], [385.035552978
5156, 527.996337890625, 426.7388000488281, 679.4779663085938], [1290.9697265625, 785.0053
7109375, 1349.5552978515625, 969.3139038085938], [1115.1236572265625, 636.8223876953125,
1168.280029296875, 733.9985961914062], [220.47109985351562, 323.8011169433594, 257.402313
2324219, 478.64166259765625], [264.0448303222656, 714.7813110351562, 302.7651672363281, 8
59.0498046875], [345.0574645996094, 716.671630859375, 387.1029052734375, 867.166870117187
5], [206.6976776123047, 922.9601440429688, 246.54168701171875, 1076.590087890625], [1.044
507384300232, 528.8065795898438, 35.633365631103516, 654.75244140625], [724.311279296875,
127.85202026367188, 779.6589965820312, 272.2434387207031], [1431.4530029296875, 110.81505
584716797, 1488.5677490234375, 242.13816833496094], [361.1867370605469, 938.4345092773438
, 398.9381103515625, 1095.2716064453125], [532.401611328125, 739.7827758789062, 574.71990
96679688, 889.4303588867188], [214.19235229492188, 712.5242919921875, 260.76123046875, 85
4.987060546875], [1419.115234375, 659.1714477539062, 1482.155029296875, 752.891357421875]
, [1294.6837158203125, 653.4657592773438, 1351.66552734375, 745.649169921875], [1541.1834
716796875, 359.8212890625, 1604.64599609375, 497.11737060546875], [1382.46533203125, 14.3
16915512084961, 1436.901123046875, 108.40576171875], [302.00823974609375, 531.26849365234
38, 340.7933654785156, 673.8898315429688], [674.8233642578125, 127.94266510009766, 720.96
71630859375, 273.1181335449219], [1462.7388916015625, 798.2828979492188, 1524.70617675781
25, 960.7492065429688], [496.38641357421875, 326.00018310546875, 536.8085327148438, 484.9
5260620117191], [30.211156845092773, 329.74224853515625, 81.44339752197266, 475.3193969726
5625], [37.6212043762207, 529.85986328125, 71.42378997802734, 656.5762939453125], [1542.1
700439453125, 1102.0360107421875, 1598.62451171875, 1278.4517822265625], [900.51782226562
5, 537.5365600585938, 944.1212158203125, 713.9387817382812], [259.7227783203125, 323.3601
379394531, 298.8307800292969, 478.821044921875], [1070.843994140625, 191.7174530029297, 1
128.8150634765625, 259.4518737792969], [487.79217529296875, 735.4498291015625, 529.316711
4257812, 882.5300903320312], [61.2783088684082, 904.1649169921875, 96.44532775878906, 105
0.5137939453125], [260.006103515625, 532.0455932617188, 299.47967529296875, 672.373901367
1875], [470.2472839355469, 194.9483642578125, 519.4772338867188, 278.3583984375], [1407.2
518310546875, 791.810791015625, 1457.4464111328125, 943.2254638671875], [680.863647460937
5, 321.4189758300781, 727.7527465820312, 487.68475341796875], [495.5222473144531, 112.642
05932617188, 546.762451171875, 189.70358276367188], [956.587890625, 107.38723754882812, 1
008.4374389648438, 190.206436157226561, [140.90228271484375, 535.46337890625, 175.3621215
8203125, 663.3697509765625], [522.7005615234375, 193.42730712890625, 575.5490112304688, 2
78.0523681640625], [305.0035705566406, 717.711181640625, 342.8013610839844, 862.485168457
0312], [414.94366455078125, 198.54217529296875, 467.16632080078125, 281.40344238281251, [
281.9747314453125, 929.93896484375, 317.1320495605469, 1086.060791015625], [167.658187866
21094, 918.7197265625, 204.38609313964844, 1066.5509033203125], [343.3868103027344, 526.7
223510742188, 382.8097229003906, 675.83349609375], [1603.56494140625, 1117.916748046875,
1662.6053466796875, 1278.85546875], [74.37649536132812, 702.6388549804688, 126.4326095581
0547, 841.2471313476562], [21.8240909576416, 900.139892578125, 59.19005584716797, 1042.68
15185546875], [106.59941864013672, 534.1705932617188, 139.18856811523438, 660.86474609375
], [1012.179931640625, 193.49234008789062, 1066.67333984375, 260.82550048828125], [397.41
51611328125, 324.3984375, 448.72412109375, 482.9453125], [73.04920196533203, 529.44970703
125, 104.75665283203125, 658.6680908203125], [248.28958129882812, 925.177001953125, 280.4
014587402344, 1080.201416015625], [178.4268035888672, 535.2265014648438, 230.941680908203

12, 666.9403076171875], [884.0213623046875, 987.2916259765625, 927.0027465820312, 1156.64
501953125], [549.9115600585938, 106.3712387084961, 596.8367919921875, 188.51582336425781],
[443.46502685546875, 114.68463897705078, 491.8631896972656, 192.1271209716797], [1.471573
9488601685, 693.3464965820312, 36.453121185302734, 830.9559936523438], [38.31742858886719
, 695.1664428710938, 72.37419891357422, 835.30615234375], [954.9435424804688, 195.2787628
173828, 1008.845458984375, 259.7560729980469], [1667.5626220703125, 1134.794677734375, 17
06.0, 1279.4964599609375], [1686.12548828125, 93.76277923583984, 1705.9976806640625, 229.
68470764160156], [600.6934204101562, 101.86792755126953, 639.33349609375, 186.12889099121
094], [579.7816772460938, 191.2084503173828, 622.8585815429688, 285.9966125488281], [838.
6114501953125, 133.8973846435547, 893.9561767578125, 266.54144287109375], [173.0609741210
9375, 93.7769546508789, 222.9745330810547, 227.4434814453125], [267.94830322265625, 194.8
8739013671875, 329.9585876464844, 281.4122619628906], [625.7830810546875, 188.23483276367
188, 669.029296875, 284.2396545410156], [1500.2259521484375, 5.001504421234131, 1549.6607
666015625, 157.59373474121094], [897.1384887695312, 141.02505493164062, 951.2171020507812
, 265.0691223144531], [131.01351928710938, 909.6174926757812, 165.9655303955078, 1059.858
3984375], [5.650207042694092, 171.43228149414062, 41.43198013305664, 293.5045471191406],
[401.110595703125, 943.1052856445312, 446.6176452636719, 1101.0853271484375], [642.322509
765625, 98.64411163330078, 679.0467529296875, 185.11129760742188], [783.5684204101562, 32
1.5312194824219, 840.6314086914062, 491.3663635253906], [388.7322692871094, 110.369026184
08203, 437.2371520996094, 189.37811279296875], [904.7755126953125, 24.372352600097656, 95
6.6429443359375, 140.1663055419922], [42.415218353271484, 169.89064025878906, 84.30032348
632812, 293.67529296875], [1123.1492919921875, 536.31298828125, 1168.6253662109375, 629.7
305908203125], [1066.952880859375, 766.4168090820312, 1109.4813232421875, 931.38116455078
12], [97.95085906982422, 902.5506591796875, 129.5143280029297, 1048.4857177734375], [847.
4807739257812, 27.722585678100586, 900.123291015625, 139.66189575195312], [127.7292480468
75, 96.03638458251953, 169.45960998535156, 230.73397827148438], [340.8103942871094, 118.0
4727935791016, 384.4753112792969, 187.30514526367188], [139.41033935546875, 696.455078125
, 177.34764099121094, 846.3213500976562], [792.2659301757812, 31.95219612121582, 841.6213
37890625, 170.00433349609375], [469.0047302246094, 324.67108154296875, 496.4168701171875,
482.17764282226562], [0.0, 884.8282470703125, 17.672161102294922, 1034.2427978515625], [13
04.42431640625, 1016.9224853515625, 1344.2850341796875, 1127.891845703125], [853.44274902
34375, 318.38800048828125, 890.311279296875, 489.5318603515625], [94.36766815185547, 103.
63837432861328, 127.6654281616211, 235.8852081298828], [178.01722717285156, 706.8359375,
211.0641326904297, 849.4093017578125], [236.95358276367188, 92.60077667236328, 276.406951
9042969, 214.38238525390625], [739.52880859375, 39.07585525512695, 788.1685180664062, 122
.85972595214844], [670.7278442382812, 758.4542236328125, 767.2702026367188, 904.948730468
75], [687.3861083984375, 42.37852096557617, 735.3460693359375, 126.87435150146484], [586.
9550170898438, 325.20208740234375, 676.0425415039062, 484.8310546875], [1111.216064453125
, 1029.0745849609375, 1223.0106201171875, 1231.901123046875], [742.7843017578125, 540.910
5224609375, 837.2607421875, 707.0625610351562], [773.1411743164062, 761.5508422851562, 88
1.4009399414062, 914.9718017578125], [954.6842651367188, 331.9721984863281, 1059.85778808
59375, 492.9162902832031], [794.1738891601562, 544.6355590820312, 890.7616577148438, 710.
3450927734375], [694.3463134765625, 543.0559692382812, 787.919921875, 702.7500610351562],
[1113.4049072265625, 777.3311157226562, 1224.6505126953125, 953.4342041015625], [646.5301
513671875, 542.5607299804688, 736.1397705078125, 698.578857421875], [1196.8214111328125,
123.65190124511719, 1305.749755859375, 250.38565063476562], [990.5540161132812, 1039.5032
958984375, 1101.88330078125, 1210.2777099609375], [1129.6866455078125, 648.4544677734375,
1226.5853271484375, 735.9266357421875], [1413.683349609375, 1075.1392822265625, 1533.6282
958984375, 1273.2591552734375], [707.50732421875, 992.4437255859375, 804.9892578125, 1156
.294921875], [542.0517578125, 325.77276611328125, 628.0956420898438, 484.30596923828125],
[829.8920288085938, 763.3922729492188, 935.835693359375, 919.3780517578125], [1254.757080
078125, 120.52075958251953, 1363.234130859375, 246.87265014648438], [623.5628051757812, 7
49.8399047851562, 713.0081176757812, 898.132080078125], [949.4491577148438, 548.726013183
5938, 1048.52001953125, 719.8856811523438], [578.9968872070312, 747.5648803710938, 666.24

69482421875, 895.8831176757812], [1126.2138671875, 370.9186096191406, 1229.4268798828125, 496.34356689453125], [1291.05615234375, 368.99163818359375, 1401.2298583984375, 497.38067626953125], [444.12347412109375, 735.2342529296875, 528.4783325195312, 878.1942138671875], [1073.3040771484375, 98.97442626953125, 1190.6715087890625, 183.95008850097656], [1372.2496337890625, 114.16034698486328, 1486.437744140625, 242.55491638183594], [534.5315551757812, 744.227783203125, 619.9736938476562, 891.4317626953125], [735.7371215820312, 323.04827880859375, 836.4581298828125, 488.6044006347656], [1215.888916015625, 23.8293724060058, 1319.39111328125, 115.29741668701172], [500.6796569824219, 326.7883605957031, 582.5845947265625, 484.3184814453125], [897.6112670898438, 336.0537109375, 1006.1414184570312, 493.06027221679691], [1027.85888671875, 9.108338356018066, 1143.273681640625, 94.31218719482422], [1326.4664306640625, 17.73615264892578, 1434.5263671875, 108.57911682128906], [958.501220703125, 106.51896667480469, 1065.1185302734375, 188.28852844238281], [1383.61376953125, 15.228241920471191, 1493.1195068359375, 106.08690643310547], [498.2950439453125, 529.564453125, 581.901123046875, 685.283447265625], [1292.169921875, 788.9209594726562, 1401.341552734375, 963.022705078125], [1071.1756591796875, 191.64903259277344, 1190.23583984375, 256.8297119140625], [955.3148803710938, 195.2711181640625, 1065.1444091796875, 259.44232177734375], [1233.7081298828125, 541.7613525390625, 1353.7147216796875, 646.1815185546875], [1407.967529296875, 365.1348876953125, 1532.2718505859375, 497.6174011230469], [110.55758666992188, 537.4360961914062, 173.7180633544922, 661.8848266601562], [443.7590637207031, 114.85882568359375, 545.1619873046875, 187.41236877441406], [580.0490112304688, 189.79864501953125, 668.0438842773438, 283.16888427734375], [472.4727783203125, 195.1262664794922, 574.6144409179688, 276.64080810546875], [1528.407958984375, 802.7838134765625, 1626.8011474609375, 976.7135620117188], [355.8107604980469, 327.94268798828125, 447.65167236328125, 480.6085205078125], [1666.717529296875, 385.0246887207031, 1706.0, 494.94146728515625], [391.5310974121094, 114.95945739746094, 491.9383850097656, 189.06500244140625], [522.8659057617188, 194.9291229248047, 621.0183715820312, 277.847412109375], [1152.6348876953125, 15.886126518249512, 1263.7197265625, 102.2232437133789], [263.6202087402344, 718.0106811523438, 333.0270690917969, 858.93658447265621],
    "detection_scores": [0.9273692965507507, 0.9999246597290039, 0.99989914894104, 0.999121367931366, 0.9996966123580933, 0.9995707869529724, 0.7289692163467407, 0.9995418787002563, 0.999920129776001, 0.9995354413986206, 0.999833345413208, 0.4871937930583954, 0.9999123811721802, 0.9997228980064392, 0.9958129525184631, 0.9996127486228943, 0.9998365640640259, 0.9967293739318848, 0.9999990463256836, 0.9984652996063232, 0.9996250867843628, 0.9539254903793335, 0.9534531831741333, 0.9996869564056396, 0.9999657869338989, 0.9999988079071045, 0.9999300241470337, 0.9960856437683105, 0.8885335326194763, 0.9997417330741882, 0.9997662901878357, 0.9998348951339722, 0.9999892711639404, 0.9999392032623291, 0.9971627593040466, 0.9996781349182129, 0.9997746348381042, 0.9998767375946045, 0.6801815032958984, 0.9951504468917847, 0.824370265007019, 0.9998219609260559, 0.9999841451644897, 0.9936991930007935, 0.999853253364563, 0.9992411136627197, 0.906742513179779, 0.9998705387115479, 0.9993428587913513, 0.9999606609344482, 0.9989314675331116, 0.9999186992645264, 0.990565299987793, 0.9949457049369812, 0.997605562210083, 0.5342898368835449, 0.9998645782470703, 0.9976692795753479, 0.994135856628418, 0.9992128610610962, 0.9999184608459473, 0.9668468832969666, 0.9996273517608643, 0.9305729269981384, 0.9998874664306641, 0.9996445178985596, 0.9993064403533936, 0.8758603930473328, 0.9945607781410217, 0.974999725818634, 0.9999758005142212, 0.9992790818214417, 0.9977745413780212, 0.9999972581863403, 0.9997275471687317, 0.999474823474884, 0.9999274015426636, 0.9965685606002808, 0.9923534989356995, 0.9994809031486511, 0.997495174407959, 0.9993118047714233, 0.9959633350372314, 0.9980132579803467, 0.9996987581253052, 0.9977188110351562, 0.9403359293937683, 0.9999799728393555, 0.9999897480010986, 0.990454375743866, 0.9986190795898438, 0.999550998210907, 0.9999936819076538, 0.9405861496925354, 0.9998120665550232, 0.9998401403427124, 0.9733796119689941, 0.9978960752487183, 0.9465305209159851, 0.9913039207458496, 0.9987467527389526, 0.9085096120834351, 0.9899015426635742, 0.9660815596580505, 0.9999738931655884, 0.9993909597396851, 0.9980925917625427, 0.9810968041419983, 0.9997773766517639, 0.9986379742622375, 0.9985313415527344, 0.9997103810310364, 0.9996278285980225, 0.9991834759712219, 0.9953756332397461, 0.996

5437054634094, 0.9949002861976624, 0.9998812675476074, 0.9982750415802002, 0.9519234895706177, 0.83475261926651, 0.9988154172897339, 0.6579822897911072, 0.996841549873352, 0.998006284236908, 0.9875984787940979, 0.9999675750732422, 0.9993166923522949, 0.9999188184738159, 0.9570266008377075, 0.8224517107009888, 0.8004858493804932, 0.9999958276748657, 0.9868762493133545, 0.9981579184532166, 0.9995274543762207, 0.9172521233558655, 0.9999903440475464, 0.6098853349685669, 0.9996042847633362, 0.9990062117576599, 0.9985187649726868, 0.9991430044174194, 0.9985539317131042, 0.9730827808380127, 0.9935812950134277, 0.9999825954437256, 0.9999949932098389, 0.618529200553894, 0.9977074861526489, 0.9997407793998718, 0.9862242937088013, 0.4937046766281128, 0.8809532523155212, 0.9930036664009094, 0.9994494318962097, 0.9934224486351013, 0.6491010785102844, 0.8800499439239502, 0.9985900521278381, 0.9998074173927307, 0.9997654557228088, 0.9815234541893005, 0.46235981583595276, 0.999646782875061, 0.9666797518730164, 0.9996416568756104, 0.9943087697029114, 0.9969773292541504, 0.9998121857643127, 0.9933486580848694, 0.9860081076622009, 0.9596437811851501, 0.7654311060905457, 0.9637073278427124, 0.9902470111846924, 0.9970638155937195, 0.9652664065361023, 0.9454036355018616, 0.9894844889640808, 0.6968677043914795, 0.891010582447052, 0.9997339844703674, 0.99956876039505, 0.9065569639205933, 0.7655655741691589, 0.9942012429237366, 0.7919678092002869, 0.9972113966941833, 0.7805525064468384, 0.9977450370788574, 0.9495887160301208, 0.9901830554008484, 0.5698038339614868, 0.9958011507987976, 0.9998944997787476, 0.7959381341934204, 0.7920934557914734, 0.8259226679801941, 0.6261268854141235, 0.9991492033004761, 0.9997618794441223, 0.9999324083328247, 0.6555682420730591, 0.860374391078949, 0.999653697013855, 0.6417619585990906, 0.9392773509025574, 0.9997584223747253, 0.9999518394470215, 0.8577653765678406, 0.9999479055404663, 0.9998664855957031, 0.9996888637542725, 0.9999748468399048, 0.9979487061500549, 0.9995716214179993, 0.9998250603675842, 0.9998193383216858, 0.8090611100196838, 0.9891413450241089, 0.8536688685417175, 0.9995394945144653, 0.9999127388000488, 0.9870069026947021, 0.9995442032814026, 0.9964131712913513, 0.997424840927124, 0.9299638867378235, 0.9971292614936829, 0.9993711113929749, 0.9787277579307556, 0.9989132881164551, 0.9998195767402649, 0.5174880623817444, 0.9998970031738281, 0.9618293046951294, 0.41249313950538635, 0.9999809265136719, 0.9881607294082642, 0.7479101419448853, 0.9768029451370239, 0.9154120683670044, 0.9999865293502808],
    "detection_classes": [110, 161, 161, 114, 117, 116, 124, 114, 116, 122, 95, 103, 95, 95, 123, 115, 115, 126, 71, 100, 132, 108, 8, 97, 99, 71, 104, 127, 124, 159, 159, 130, 99, 159, 123, 19, 127, 17, 133, 100, 133, 95, 159, 126, 105, 18, 8, 132, 124, 159, 95, 17, 15, 109, 16, 111, 97, 14, 126, 106, 159, 118, 114, 126, 112, 122, 122, 161, 95, 95, 99, 104, 98, 7, 130, 98, 106, 108, 15, 131, 5, 117, 109, 109, 129, 15, 111, 5, 5, 126, 16, 159, 159, 161, 106, 112, 6, 95, 4, 123, 13, 113, 107, 6, 20, 19, 128, 9, 70, 95, 129, 17, 20, 126, 109, 101, 95, 12, 70, 111, 8, 4, 10, 121, 105, 6, 159, 18, 135, 12, 79, 111, 71, 86, 159, 11, 79, 20, 159, 128, 128, 13, 121, 137, 135, 11, 159, 7, 10, 128, 11, 124, 86, 86, 136, 136, 159, 120, 161, 69, 69, 93, 67, 68, 69, 161, 94, 134, 2, 129, 69, 72, 20, 93, 1, 107, 115, 134, 93, 1, 11, 138, 70, 7, 65, 5, 72, 1, 138, 67, 70, 118, 70, 8, 123, 104, 116, 97, 105, 104, 114, 16, 95, 124, 108, 122, 132, 8, 116, 95, 17, 106, 17, 99, 127, 18, 159, 95, 17, 71, 95, 8, 97, 159, 95, 159, 95, 15, 113, 159, 159, 109, 100, 11, 86, 69, 93, 111, 7, 102, 20, 69, 159, 20],
    "detection_class_names": ["429", "199", "199", "433", "436", "435", "712", "433", "435", "715", "368", "370", "368", "368", "714", "434", "434", "354", "301", "372", "479", "755", "757", "299", "374", "301", "350", "200", "712", "4", "4", "477", "374", "4", "714", "440", "200", "438", "713", "372", "713", "368", "4", "354", "351", "439", "757", "479", "712", "4", "368", "438", "348", "754", "349", "432", "299", "347", "354", "352", "4", "437", "433", "354", "431", "715", "715", "199", "368", "368", "374", "350", "298", "303", "477", "298", "352", "755", "348", "478", "305", "436", "754", "754", "476", "348", "432", "305", "305", "354", "349", "4", "4", "199", "352", "431", "304", "368", "306", "714", "346", "753", "353", "304", "441", "440", "6", "342", "196", "368", "476", "438", "441", "354", "754", "373", "368", "5", "196", "432", "757", "306", "343", "716", "351", "304", "4", "439", "473", "5", "181", "432", "301", "313", "4", "344", "181", "441", "4", "6", "6", "346", "716", "443", "473", "344", "4", "303", "343", "6", "344", "712", "313", "313", "768", "768", "4", "717", "199", "195", "195", "198", "199", "759", "195", "199", "197

```
", "474", "773", "476", "195", "300", "441", "198", "772", "353", "434", "474", "198", "772", "344", "442", "196", "303", "472", "305", "300", "772", "442", "190", "196", "437", "196", "757", "714", "350", "435", "299", "351", "350", "433", "349", "368", "712", "755", "715", "479", "757", "435", "368", "438", "352", "438", "374", "200", "439", "4", "368", "438", "301", "368", "757", "299", "4", "368", "4", "368", "348", "753", "4", "4", "754", "372", "344", "313", "195", "198", "432", "303", "371", "441", "195", "4", "441"],
    "product_count": {"429": 1, "199": 6, "433": 4, "436": 2, "435": 4, "712": 5, "715": 4, "368": 16, "370": 1, "714": 4, "434": 3, "354": 6, "301": 4, "372": 3, "479": 3, "755": 3, "757": 6, "299": 4, "374": 4, "350": 4, "200": 3, "4": 19, "477": 2, "440": 2, "438": 6, "713": 2, "351": 3, "439": 3, "348": 4, "754": 5, "349": 3, "432": 5, "347": 1, "352": 4, "437": 2, "431": 2, "298": 2, "303": 4, "478": 1, "305": 4, "476": 3, "304": 3, "306": 2, "346": 2, "753": 2, "353": 2, "441": 6, "6": 4, "342": 1, "196": 5, "373": 1, "5": 2, "343": 2, "716": 2, "473": 2, "181": 2, "313": 4, "344": 5, "443": 1, "768": 2, "717": 1, "195": 6, "198": 4, "190": 2, "759": 1, "197": 1, "474": 2, "773": 1, "300": 2, "772": 3, "442": 2, "472": 1, "371": 1}
}
```


Image similarity comparison model

- Overview

  The image similarity comparison model is developed based on the ResNet50 model and returns the similarity between two images. This model can be used for image comparison and retrieval.

- Input format

  The input data must be in the JSON format. It contains the imagea and imageb fields. The value of each field is the image content that is encoded in the Base64 format.

```
{
  "imagea": "Base64-encoded image content",
  "imageb": "Base64-encoded image content"
}
```

- Output format

  The output data is in the JSON format. The following table describes the fields in the output data.

Parameter	Description	Shape	Type
similarity	The similarity between the two images identified by the imagea and imageb fields. A value of 100 indicates that the two images are the same. A value smaller than 80 indicates that the two images are different.	[]	FLOAT

Parameter	Description	Shape	Type
l2_distance	The distance between the feature vectors of the two images. The larger the value is, the less similarity the two images share.	[]	FLOAT
request_id	The unique ID of the request.	[]	STRING
success	Indicates whether the request was successful. Valid values: ○ true: indicates that the request was successful. ○ false: indicates that the request failed.	[]	BOOL
error_code	The error code that is returned if the request failed.	[]	INT
error_msg	The error message that is returned if the request failed.	[]	STRING

The following code provides an example of the output data:

```
{
  "request_id": "d4e4348a-6101-43d1-9203-dbe8f531****",
  "success": true,
  "similarity": [1.0],
  "l2_distance":[0.0]
 }
```

- Test data
  - Image A
  - Image B

Image coloring model

- Overview

  The image coloring model is developed based on the NoGAN model and returns the colored image.

- Input format

  The input data must be in the JSON format. It contains the image field. The value of this field is the image content that is encoded in the Base64 format.

```
{
  "image": "Base64-encoded image content"
}
```

- Output format

  The output data is in the JSON format. The following table describes the fields in the output data.

Parameter	Description	Shape	Type
out_image	The image to be colored.	[]	BASE 64
request_id	The unique ID of the request.	[]	STRING
success	Indicates whether the request was successful.	[]	BOOL
error_code	The error code that is returned if the request failed.	[]	INT
error_msg	The error message that is returned if the request failed.	[]	STRING

  The following code provides an example of the output data:

```
{
   "out_image": "Base64-encoded image content"
   "request_id": "9ac294a4-f387-4c48-b640-d2c6d41f****",
   "success": true
}
```

3.ASR models

Model Hub of Machine Learning Platform for AI (PAI) provides a trained automatic speech recognition (ASR) model. You can deploy the model to make service calls online. This topic describes the input format and output format for ASR models and provides a test example.

Background information

One of the most important goals of AI is to enable machines to understand human language. To achieve this goal, the first important step is to transcribe human language to text. ASR is an important technology that integrates the disciplines of AI, linguistics, and acoustics. This technology automatically transcribes the audio input of human language to text.

On the basis of ASR, you can perform speech understanding, where AI technologies are used to analyze audio features for deep understanding of the input speech. PAI allows you to deploy ASR and speech understanding models. It provides the speech understanding models described in the following table for online use.

Model	Description
General Chinese speech recognition model	Automatically recognizes Chinese speech from the input audio or video in common scenarios.
Chinese speech recognition model for E-commerce live streaming	Automatically recognizes Chinese speech from the input audio or video in Chinese E-commerce live streaming scenarios.
Chinese speech vectorization model	Recognizes Chinese speech from the input audio or video, uses the self-supervised learning technology to vectorize the speech data, and then export the vectorized results.
English speech vectorization model	Recognizes English speech from the input audio or video, uses the self-supervised learning technology to vectorize the speech data, and then export the vectorized results.
Classification model for Chinese speakers based on speech attributes	Classifies speakers based on the speech attributes that are recognized in Chinese audio or video clips.
Chinese speech detection model	Detects whether the input audio or video contains Chinese speech.
Background music detection model	Detects whether the input audio or video contains background music.

Go to Model Hub

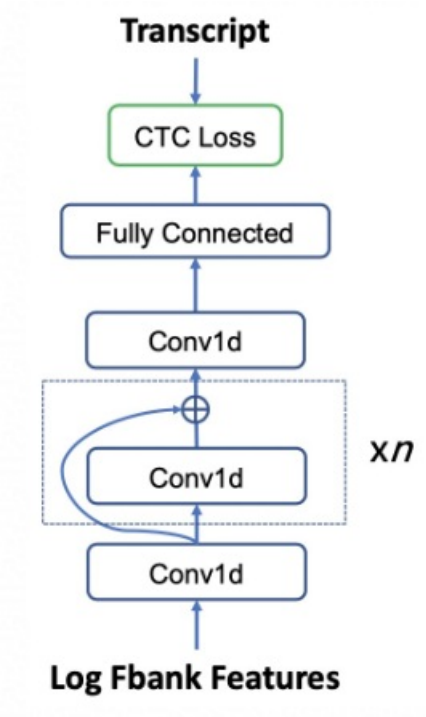
To go to Model Hub, perform the following steps:

1. Log on to the Machine Learning Platform for AI console.
2. In the left-side navigation pane, click **Workspaces**. On the Workspace list page, click the name of the workspace that you want to manage.
3. In the left-side navigation pane, choose **AI Computing Asset Management > Models**.
4. On the **Model Management** page, click the **Model Hub** tab.

General Chinese speech recognition model

- Overview

  The general Chinese speech recognition model is an end-to-end Wav2Letter model provided by PAI. This model can automatically recognize Chinese speech from the input audio or video in common scenarios. The following figure shows the structure of the model.



- Input format

  The input data must be in the JSON format. It contains the url and play_duration fields. The value of the url field is the URL of the input audio or video. The value of the play_duration field is the first several microseconds of the input audio or video to be processed. If the play_duration field is not specified, the total length of the input audio or video is processed. The following code provides an example of the input data:

```
{
  "input": {
    "url": "URL of the input audio or video",
    "play_duration": "Length of the input audio or video to be processed"
  }
}
```

- Output format

  The output data consists of key-value pairs in the JSON format. Each key indicates the start timestamp, in microseconds, of the input audio or video clip that is processed. Each value indicates the output text that is transcribed by the ASR model. The model supports about 4,000 common Chinese characters in the output text. If the result of the model contains a Chinese character that is not included in the supported Chinese character list, the character is replaced by an asterisk (*). Short sentences are separated by semicolons (;). The following code provides an example of the output data:

```
{
  "0": "Text 1 obtained after transcription",
  "500000000": "Text 2 obtained after transcription",
  "1000000000": "Text 3 obtained after transcription",
}
```

- Example

  The following code provides an example of the input data of the model:

```
{
  "input": {
    "url": "http://pai-vision-data-sh.oss-cn-shanghai-internal.aliyuncs.com/tmp/5000563****.mp4",
    "play_duration": "39000000"
  }
}
```

  PAI displays information similar to the following output:

```
{
  "0": ";\u5206\u6d3b\u7845\u85fb\u571f\u9020\u8131;\u6709\u5b54\u901f\u5ea6\u5927\u5438\u6536\u6027\u5f3a\u51c0\u5316\u7a7a\u6c14\u7684\u7279\u70b9;\u53ef\u653e\u7f6e\u624b\u5de5\u6d01\u9762\u9020\u5f62\u517d\u9020\u4e0d\u6613\u9020\u7b49;\u80a5\u7682\u653e\u7f6e\u540e\u51e0\u79d2\u5185\u5c31\u80fd\u77ac\u95f4\u5438\u6536",
  "20031996": "\u7531\u4e8e\u80a5\u7682\u4f7f\u7528\u540e\u6709\u53d8\u8f6f\u7684\u7279\u8d28;\u6240\u4ee5\u7845\u85fb\u4e3b\u9020\u79d1\u80fd\u5b8c\u6574\u4fdd\u62a4\u80a5\u7682;\u4e14\u4e0d\u7528\u62c5\u5fc3\u7682\u6c34\u5916\u6d41;\u666e\u901a\u6ca5\u6c34\u9020\u51fa\u5e95\u90e8\u7684\u683c\u81ea\u4f1a\u5bfc\u81f4\u80a5\u7682\u53d8\u5f62\u53d8\u5c0f;\u7682\u6c34\u8fd8\u4f1a\u7559\u7684\u5012\u4f4f\u6ce5"
}
```

  The output Unicode data can be decoded into Chinese characters in downstream applications.

Chinese speech recognition model for E-commerce live streaming

- Overview

  The Chinese speech recognition model for E-commerce live streaming is an end-to-end Wav2Letter model provided by PAI. This model can automatically recognize Chinese speech from the input audio or video in E-commerce live streaming scenarios. This model differs from the general Chinese speech recognition model. This model is dedicated to Chinese E-commerce live streaming scenarios. However, this model shares the same structure as the general Chinese speech recognition model.

- Input format

  The input data must be in the JSON format. It contains the url and play_duration fields. The value of the url field is the URL of the input audio or video. The value of the play_duration field is the first several microseconds of the input audio or video to be processed. If the play_duration field is not specified, the total length of the input audio or video is processed. The following code provides an example of the input data:

```
{
  "input": {
    "url": "URL of the input audio or video",
    "play_duration": "Length of the input audio or video to be processed"
  }
}
```

- Output format

  The output data consists of key-value pairs in the JSON format. Each key indicates the start timestamp, in microseconds, of the input audio or video clip that is processed. Each value indicates the output text that is transcribed by the ASR model. The model supports about 6,000 common Chinese characters in the output text, which is greater than that supported by the general Chinese speech recognition model. If the result of the model contains a Chinese character that is not included in the supported Chinese character list, the character is replaced by an asterisk (*). Short sentences are separated by semicolons (;). The following code provides an example of the output data:

  ```
  {
    "0": "Text 1 obtained after transcription",
    "500000000": "Text 2 obtained after transcription",
    "1000000000": "Text 3 obtained after transcription",
  }
  ```

- Example

  The following code provides an example of the input data of the model:

  ```
  {
    "input": {
      "url": "https://pai-vision-data-sh.oss-cn-shanghai-internal.aliyuncs.com/chengyu.wcy/tblive_sample/example1.wav",
    }
  }
  ```

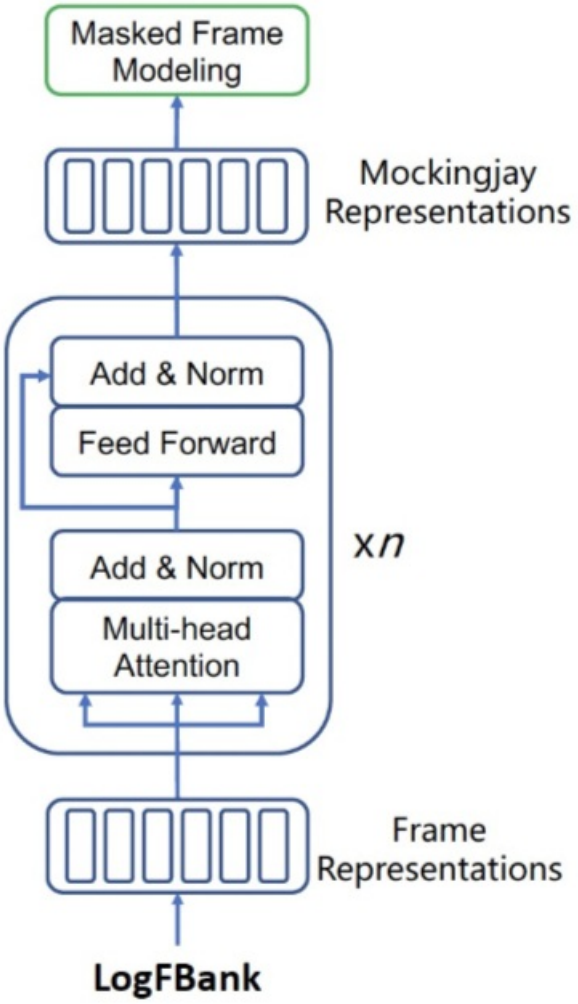
  PAI displays information similar to the following output:

  ```
  {
    "0": "\u800c\u4e14\u8fdb\u4e00\u6b65\u5f3a\u5316\u4e86\u4ea7\u54c1\u7684\u4e00\u4e2a\u4fee\u590d\u7279\u6548;\u4fee\u590d\u529f\u6548\u4f1a\u66f4\u597d;\u800c\u4e14\u5b83\u6bd4\u91d1\u80f6\u7684\u8bdd\u662f\u66f4\u52a0\u6e29\u548c;\u76ae\u80a4\u4e0d\u8010\u53d7\u79ef\u7387\u63a5\u8fd1\u4e3a\u96f6\u4e5f\u5c31\u662f\u8bf4;\u554a\u4eca\u5929\u665a\u4eca\u5929\u51cc\u6668\u53d1\u8d27"
  }
  ```

Chinese speech vectorization model

- Overview

Chinese speech vectorization model is an end-to-end Mockingjay model provided by PAI. This model uses the self-supervised learning technology to analyze Chinese speech from the input audio to meet personalized requirements. This model recognizes Chinese speech from the input audio or video, uses the self-supervised learning technology to vectorize the speech data, and then export the vectorize results. The following figure shows the structure of the model.



- Input format

  The input data must be in the JSON format. It contains only the url field. The value of the url field is the URL of the input audio or video. The following code provides an example of the input data:

```
{
  "input": {
    "url": "URL of the input audio or video",
  }
}
```

- Output format

  The output data is a string that consists of vector features. The features are separated by commas (,). The following code provides an example of the output data:

```
"Vector feature 1, Vector feature 2, Vector feature 3, ..., Vector feature N"
```

- Example

  The following code provides an example of the input data of the model:

```
{
  "input": {
    "url": "http://pai-vision-data-sh.oss-cn-shanghai-internal.aliyuncs.com/tmp/5000563**
**.mp4",
  }
}
```

  PAI displays information similar to the following output:

```
"0.5291504,-0.47187772,-0.7588605,...,-0.48115134,1.7070293"
```

English speech vectorization model

- Overview

  English speech vectorization model is an end-to-end Mockingjay model provided by PAI. This model uses the self-supervised learning technology to analyze English speech from the input audio to meet personalized requirements. This model recognizes English speech from the input audio or video, uses the self-supervised learning technology to vectorize the speech data, and then export the vectorized results. This model shares the same structure as the Chinese speech vectorization model. For more information, see Chinese speech vectorization model.

- Input format

  The input data must be in the JSON format. It contains only the url field. The value of the url field is the URL of the input audio or video. The following code provides an example of the input data:

```
{
  "input": {
    "url": "URL of the input audio or video",
  }
}
```

- Output format

  The output data is a string that consists of vector features. The features are separated by commas (,). The following code provides an example of the output data:

```
"Vector feature 1, Vector feature 2, Vector feature 3, ..., Vector feature N"
```

- Example

  The following code provides an example of the input data of the model:

```
{
  "input": {
    "url": "http://pai-vision-data-sh.oss-cn-shanghai-internal.aliyuncs.com/tmp/5000563**
**.mp4",
  }
}
```

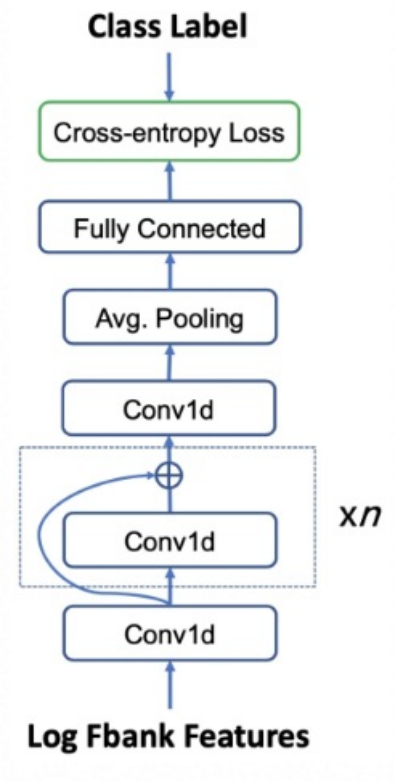
  PAI displays information similar to the following output:

```
"0.29688737,0.78769636,0.4556097,...,0.8212023,0.5032284"
```

Classification model for Chinese speakers based on speech attributes

- Overview

  The classification model for Chinese speakers based on speech attributes is an end-to-end time delay neural network (TDNN) model provided by PAI. This model can classify Chinese speakers based on the speech attributes that are recognized in Chinese audio or video clips. The following figure shows the structure of the model.



- Input format

  The input data must be in the JSON format. It contains the url and play_duration fields. The value of the url field is the URL of the input audio or video. The value of the play_duration field is the first several microseconds of the input audio or video to be processed. If the play_duration field is not specified, the total length of the input audio or video is processed. The following code provides an example of the input data:

```
{
  "input": {
    "url": "URL of the input audio or video",
    "play_duration": "Length of the input audio or video to be processed"
  }
}
```

- Output format

The output data consists of key-value pairs in the JSON format. Each key indicates the start timestamp, in microseconds, of the input audio or video clip that is processed. Each value indicates a classification result, including the speaker label predicted by the model. The following code provides an example of the output data:

```
{
  "0": "{\"class\":\"Predicted label\"}\n",
  "500000000": "{\"class\":\"Predicted label\"}\n",
  "1000000000": "{\"class\":\"Predicted label\"}\n",
}
```

- Example

  The following code provides an example of the input data of the model:

```
{
  "input": {
    "url": "http://pai-vision-data-sh.oss-cn-shanghai-internal.aliyuncs.com/tmp/5000563****.mp4",
    "play_duration": "39000000"
  }
}
```

  PAI displays information similar to the following output:

```
{
  "0": "{\"class\":\"Predicted label\"}\n",
  "20031996": "{\"class\":\"Predicted label\"}\n",
}
```

  The predicted label in the classification result is displayed in the Unicode format. The output Unicode data can be decoded into Chinese characters in downstream applications.

Chinese speech detection model

- Overview

  The Chinese speech detection model is an end-to-end TDNN model provided by PAI. This model can detect whether the input audio or video contains Chinese speech. This model shares the same structure as the classification model for Chinese speakers based on speech attributes.

- Input format

  The input data must be in the JSON format. It contains the url and play_duration fields. The value of the url field is the URL of the input audio or video. The value of the play_duration field is the first several microseconds of the input audio or video to be processed. If the play_duration field is not specified, the total length of the input audio or video is processed. The following code provides an example of the input data:

```
{
  "input": {
    "url": "URL of the input audio or video",
    "play_duration": "Length of the input audio or video to be processed"
  }
}
```

- Output format

The output data consists of key-value pairs in the JSON format. Each key indicates the start timestamp, in microseconds, of the input audio or video clip that is processed. Each value indicates a detection result. The predicted label in the detection result can be Yes or No. The value Yes indicates that the audio or video clip contains Chinese speech. The following code provides an example of the output data:

```
{
  "0": "{\"class\":\"Predicted label\"}\n",
  "500000000": "{\"class\":\"Predicted label\"}\n",
  "1000000000": "{\"class\":\"Predicted label\"}\n",
}
```

- Example

  The following code provides an example of the input data of the model:

```
{
  "input": {
    "url": "http://pai-vision-data-sh.oss-cn-shanghai-internal.aliyuncs.com/tmp/5000563****.mp4",
  }
}
```

  PAI displays information similar to the following output:

```
{
  "0": "{\"class\":\"u662f\"}\n",
  "20031996": "{\"class\":\"u662f\"}\n",
  "40063991": "{\"class\":\"\u5426\"}\n"
}
```

  The predicted label in the detection result is displayed in the Unicode format. The output Unicode data can be decoded into Chinese characters in downstream applications.

Background music detection model

- Overview

  The background music detection model is an end-to-end TDNN model provided by PAI. This model can detect whether the input audio or video contains background music. This model shares the same structure as the classification model for Chinese speakers based on speech attributes.

- Input format

  The input data must be in the JSON format. It contains the url and play_duration fields. The value of the url field is the URL of the input audio or video. The value of the play_duration field is the first several microseconds of the input audio or video to be processed. If the play_duration field is not specified, the total length of the input audio or video is processed. The following code provides an example of the input data:

```
{
  "input": {
    "url": "URL of the input audio or video",
    "play_duration": "Length of the input audio or video to be processed"
  }
}
```

- Output format

  The output data consists of key-value pairs in the JSON format. Each key indicates the start timestamp, in microseconds, of the input audio or video clip that is processed. Each value indicates a detection result. The predicted label in the detection result can be Yes or No. The value Yes indicates that the audio or video clip contains background music. The following code provides an example of the output data:

```
{
  "0": "{\"class\":\"Predicted label\"}\n",
  "500000000": "{\"class\":\"Predicted label\"}\n",
  "1000000000": "{\"class\":\"Predicted label\"}\n",
}
```

- Example

  The following code provides an example of the input data of the model:

```
{
  "input": {
    "url": "http://pai-vision-data-sh.oss-cn-shanghai-internal.aliyuncs.com/tmp/5000563**
**.mp4",
  }
}
```

  PAI displays information similar to the following output:

```
{
  "0": "{\"class\":\"u662f\"}\n",
  "20031996": "{\"class\":\"u662f\"}\n",
  "40063991": "{\"class\":\"\u662f\"}\n"
}
```

  The predicted label in the detection result is displayed in the Unicode format. The output Unicode data can be decoded into Chinese characters in downstream applications.

4.NLP models

Machine Learning Platform for AI (PAI) provides you with a variety of trained natural language processing (NLP) models, such as the Bidirectional Encoder Representations from Transformers (BERT)-based text vectorization model.

NLP is a sub-branch of artificial intelligence (AI) and linguistics. You can use NLP to extract information from natural language text. NLP applies to the following scenarios:

- Text classification: news labeling, sentiment analytics, text anti-spam, and classification of commodity reviews.
- Text matching: Q&A matching, similarity matching for sentences, natural language inferences, and conversation retrieval.
- Sequence labeling: named entity recognition (NER) and sentiment word extraction.
- Feature extraction: The extracted text features can be used to process text or applied to other fields, such as computer vision.

Model Hub of PAI allows you to deploy the preceding services. It also provides the BERT-based feature extraction model.

Go to Model Hub

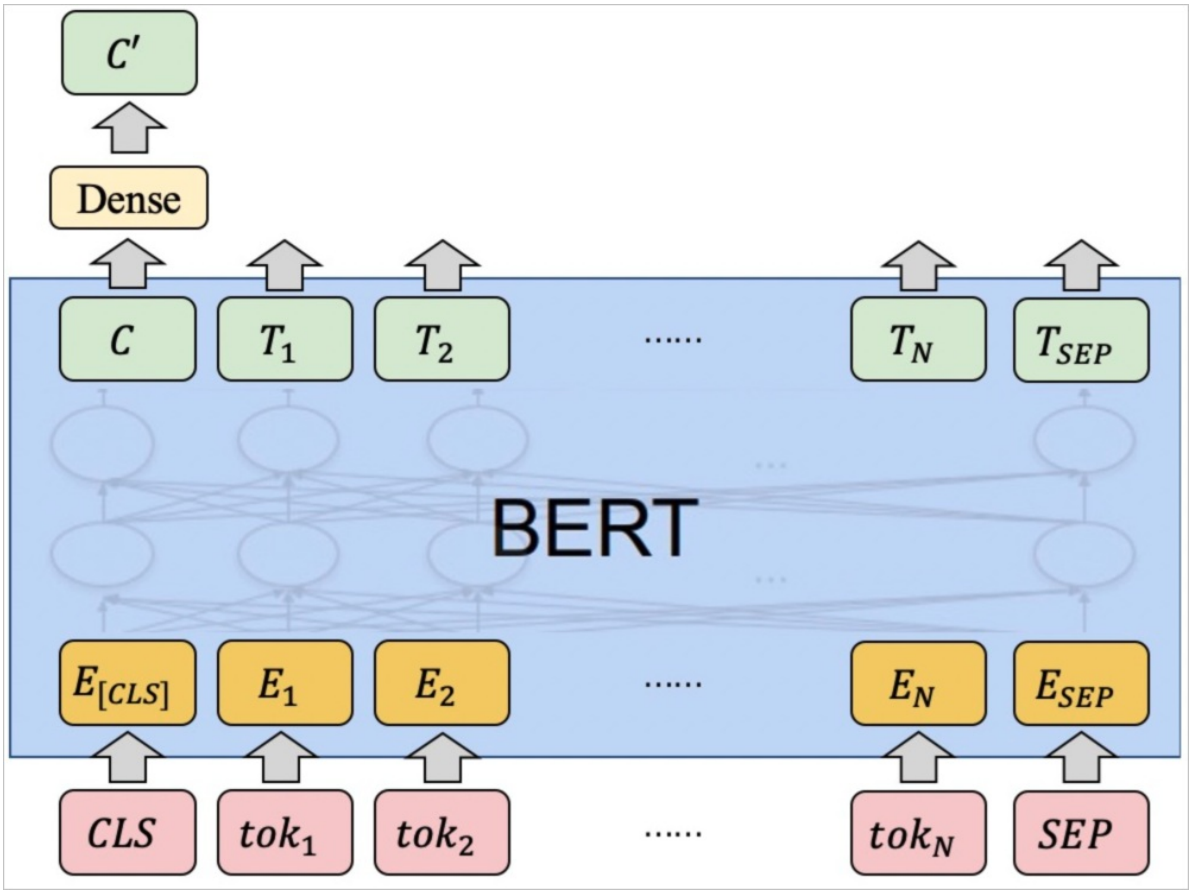
To go to Model Hub, perform the following steps:

1. Log on to the Machine Learning Platform for AI console.
2. In the left-side navigation pane, click **Workspaces**. On the Workspace list page, click the name of the workspace that you want to manage.
3. In the left-side navigation pane, choose **AI Computing Asset Management > Models**.
4. On the **Model Management** page, click the **Model Hub** tab.

BERT-based text vectorization model

- Overview

You can fine-tune the trained model that uses BERT. The vectors that are generated by BERT show great value. For example, when you use BERT to extract features, you can enter a text sequence. Then, a vector sequence is returned. After the CLS vector is processed in the Dense layer, the generated vector can be used as the vector of the whole sentence.



When you enter a sentence, characters in the sentence are automatically vectorized and displayed in the Subtoken format: [CLS, tok1, tok2, ..., tokN, SEP]. The following vector types can be returned:

- pool_output: the vectors of the encoded sentence. The vectors correspond to C' in the figure.
- first_token_output: The vectors correspond to C in the figure.
- all_hidden_outputs: The vectors correspond to [C, T1, T2, ..., TN, TSEP] in the figure.

- Input format

  The input data must be in the JSON format. It contains the following fields:

  - id: the ID of the text.
  - first_sequence: The value of the field is the first text string.
  - second_sequence: The value of the field is the second text string. The value can be empty.
  - sequence_length: the length of the text strings, which cannot exceed 512.

```
{
    "id": "The ID of the text",
    "first_sequence": "The first text string",
    "second_sequence": "The second text string, which can be empty",
    "sequence_length": 128
}
```

- Output format

  The output format is JSON. The output data contains the following fields.

Field	Description	Shape	Type
pool_output	The 768-dimensional vectors that are separated by commas (,). The vectors represent the encoded sentence and correspond to C' in the figure.	[]	STRING
first_token_output	The 768-dimensional vectors that are separated by commas (,). The vectors correspond to C in the figure.	[]	STRING
all_hidden_outputs	The 768-dimensional vectors of the sequence_length type. The vectors are separated by commas (,) and the sequences are separated by semicolons (;). The vectors correspond to [C, T1, T2, ..., TN, TSEP] in the figure.	[]	STRING

- Test data

```
// Input.
{
    "id": "1667",
    "first_sequence": "How can I increase the credit limit of Ant Credit Pay to be used i
n Double 11?",
    "second_sequence": "",
    "sequence_length": 128
}
// Output.
{
    "id": "1667",
    "pool_output": "0.999340713024,...,0.836870908737",
    "first_token_output": "0.789340713024,...,0.536870908737",
    "all_hidden_outputs": "0.999340713024,...,0.836870908737;... ;0.899340713024,...,0.93
6870908737"
}
```

5.Face models

5.1. Face similarity comparison model

This topic describes the details of the face similarity comparison model, including the features, input format, output format, and test data.

- Overview

  The face similarity comparison model uses the ResNet50 framework. For more information, see Deep Residual Learning for Image Recognition.

- Input format

  The input data must be in the JSON format. It contains the imagea and imageb fields. The value of each field is the image content that is encoded in the Base64 format. An error is returned if an image contains more than one face.

```
{
  "imagea": "Base64-encoded image content"
  "imageb": "Base64-encoded image content"
}
```

- Output format

  The output data must be in the JSON format. The following table describes the fields in the output data.

Field	Description	Shape	Type
similarity	The similarity between the two faces provided by the imagea and imageb fields. A value of 100 indicates that the two faces are of the same person. A value smaller than 80 indicates that the two faces are not of the same person.	[]	INT
request_id	The unique ID of the request.	[]	STRING
success	Indicates whether the request was successful.	[]	BOOL
error_code	The request error code.	[]	INT
error_msg	The request error message.	[]	STRING

  The following code provides an example of the output data:

```
{
  "request_id": "d4e4348a-6101-43d1-9203-dbe8f531****",
  "success": true,
  "similarity": 99
 }
```

- Test data
  - Face images of the same person: Positive sample A and Positive sample B
  - Face image of another person: Negative sample

5.2. Facial attributes model

This topic describes the detailed information about the facial attributes model, including the features, input format, and output format. This topic also provides a set of test data.

- Overview

  The facial attributes model uses the Residual Networks 50 (ResNet50) framework. The model involves multiple facial attributes and generates a facial expression, an age, and a detection frame for each detected face.

- Input format

  The input data must be in the JSON format. It contains the image parameter. The value of the parameter is the image content that is encoded in the Base64 format.

```
{
  "image": "Base64-encoded image content"
}
```

- Output format

  The output format is JSON. The following table describes the parameters in the output data.

Parameter	Description	Type
request_id	The unique ID of the request.	STRING
success	Indicates whether the request is successful. Valid values: ◦ true: The request is successful. ◦ false: The request fails.	BOOL
face_age	The estimated ages of the faces detected in the image. The values are stored in a list in sequence.	LIST
face_gender	The estimated genders of the faces detected in the image. The values are stored in a list in sequence.	LIST
face_emo	The estimated emotions of the faces detected in the image. The values are stored in a list in sequence.	LIST

Parameter	Description	Type
face_bbox	The coordinates of the faces detected in the image. The values are stored in a list in sequence.	LIST
error_code	The error code returned if the request fails.	INT
error_msg	The error message returned if the request fails.	STRING

The following code provides an example of the output data:

```
{
"request_id": "597a4180-6fcd-4dec-b1e6-9864b812****",
"success": true,
"face_age": [19.693389892578125, 20.624303817749023],
"face_gender": ["female", "female"],
"face_emo": ["Happiness,", "Happiness,"],
"face_bbox": [[452.0916877910495, 61.250034026801586, 643.2265069484711, 321.386978432536
1, 0.9999947547912598], [87.99690818786621, 81.17595142126083, 260.9599985778332, 312.292
5915122032, 0.9999878406524658]]
 }
```

- Test data
  - A single face image of multiple persons: Sample
  - Different face images of multiple persons: Sample A, Sample B, and Sample C

5.3. Face enhancement model

The face enhancement model can enhance the faces that are detected in the input image and return the enhanced face image. This topic describes the input and output formats of the model.

Overview

The face enhancement model can detect the faces in the input image, enhance the blurry part of the faces, and repair the face details so that the faces in the image become clearer.

Input format

The input data must be in the JSON format. It contains the image field. The value of this field is the image content that is encoded in the Base64 format.

```
{
  "image": "Base64-encoded image content"
}
```

Output format

The output data is in the JSON format. The following table describes the fields in the output data.

Field	Description	Shape	Value type
out_image	The content of the enhanced imaged.	[]	BASE64
request_id	The unique ID of the request.	[]	STRING
success	Indicates whether the request was successful. Valid values: • true: The request was successful. • false: The request failed.	[]	BOOL
error_code	The error code that is returned if the request failed.	[]	INT
error_msg	The error message that is returned if the request failed.	[]	STRING

The following code provides an example of the output data:

```
{
   "out_image": "Base64-encoded image content"
   "request_id": "9ac294a4-f387-4c48-b640-d2c6d41f****",
   "success": true
}
```

6.Product model

6.1. Product comparison model

This topic describes the details of the product comparison model, including the model features, input format, output format, and test data.

- Overview

  The product comparison model adopts the ResNet50 backbone that integrates with advanced metric learning technology for images. This model can be used to compare and analyze various products, such as shoes, bags, and dresses. If you want to compare other categories of products,submit a ticket.

- Input format

  The input data must be in the JSON format. It contains the following fields:

```
{
    "function_name": "match",
    "function_params": {
        "imagea": "Base64-encoded image content",
        "imageb": "Base64-encoded image content",
        "metric":  "L2", #, or "Cosine"
        },
 }
```

Field		Required	Description	Type
function_name		Yes	The name of the function. The function name of the product comparison model is match.	STRING
function_params	imagea	Yes	The Base64-encoded string of the image.	STRING
	imageb	Yes	The Base64-encoded string of the image.	STRING
	metric	Yes	The method that is used to calculate the similarity between images. Valid values: ○ L2: the L2 norm distance. ○ Cosine: the cosine distance.	STRING

- Output format

  The output data is in the JSON format. The following table describes the fields in the output data.

Field	Description	Shape	Type
request_id	The unique ID of the request.	[]	STRING
success	Indicates whether the request is successful. Valid values: ◦ true: indicates that the request succeeded. ◦ false: indicates that the request failed.	[]	BOOL
l2_distance	The L2 norm distance between images. This field is returned when the metric field is set to L2. A value of 0 indicates that the two images are the same. The greater the value is, the more different the two images are.	[]	FLOAT
cosine_similarity	The cosine distance between images. This field is returned when the metric field is set to Cosine. A value of 1 indicates that the two images are the same. The smaller the value is, the more different the two images are.	[]	FLOAT

The following code provides an example of the output data:

```
{
    "request_id": "49f7da21-7e55-427d-a551-5952f104****",
    "success": true,
    "l2_distance": 0.4584488272666931
    #"cosine_similarity":0.8853877782821655
  }
```