ModelArts

Data Processing

Issue	01
Date	2024-06-12

HUAWEI TECHNOLOGIES CO., LTD.

Trademarks and Permissions


and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.
All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei Technologies Co., Ltd.

Address: Huawei Industrial Base
Bantian, Longgang
Shenzhen 518129
People's Republic of China

Website: https://www.huawei.com

Email: support@huawei.com

Security Declaration

Vulnerability

Huawei's regulations on product vulnerability management are subject to the *Vul. Response Process.* For details about this process, visit the following web page:
https://www.huawei.com/en/psirt/vul-response-process
For vulnerability information, enterprise customers can visit the following web page:
https://securitybulletin.huawei.com/enterprise/en/security-advisory

Contents

1 Data Processing Overview....1
2 Description of Built-in Operators for Data Processing....3
2.1 Data Validation....3
2.2 Data Cleansing....7
2.3 Data Selection....9
2.3.1 Data Deduplication....9
2.3.2 Data Deredundancy....11
2.4 Data Augmentation....13
2.4.1 Data Augmentation....13
2.4.2 Data Generation....19
2.4.3 Data Transfer Between Domains....22

1 Data Processing Overview

ModelArts provides the data processing function to extract valuable and meaningful data from a large amount of disordered and difficult-to-understand data. After data is collected and accessed, the data cannot directly meet the training requirements. Process data during R&D to ensure data quality and prevent negative impact on subsequent operations (such as data labeling and model training).

Common data processing types are as follows:

- **Data validation**: Generally, data needs to be validated after being collected to ensure data validity.

  Data validation is a process of determining and verifying data availability. Generally, the collected data cannot be further processed due to some format problems. Take image recognition as an example. Users often find some images from the Internet for training, and the image quality cannot be ensured. The name, path, and extension of the images may not meet the requirements of the training algorithm. Images may also be partially damaged. As a result, the images cannot be decoded or processed by the algorithm. Therefore, data validation is very important. It helps AI developers detect data problems and effectively prevent algorithm precision deterioration or training failures caused by noisy data.

- **Data cleansing**: refers to the process of removing, correcting, or supplementing data.

  Data cleansing is to check data consistency based on data validation and correct some invalid values. For example, in the deep learning field, data may be cleansed based on a positive sample and a negative sample that are input by a user, to retain a category that the user wants and remove a category that the user does not want.

- **Data selection**: refers to the process of selecting data subsets from full data.

  Data can be selected based on the similarity or deep learning algorithm. Data selection can avoid problems such as duplicate and similar images introduced during manual image collection. Among a batch of inference data input to an old model, data selection using built-in rules can further improve the precision of the old model.

- Data augmentation:

Data augmentation: increases data volumes directly or indirectly through simple data augmentation operations such as scaling, cropping, transformation, and composition.

Data transfer between domains: generates data transferred from the original domain to the target domain by applying deep learning models and learning the datasets of the original and target domains.

2 Description of Built-in Operators for Data Processing

2.1 Data Validation

MetaValidation Operator Overview

ModelArts data validation uses the MetaValidation operator and supports the following image formats: JPG, JPEG, BMP, and PNG. The object detection scenario supports the XML labeling format but does not support the non-rectangular box labeling format. The MetaValidation operator supports data validation for images and XML files.

Table 2-1 Image data validation

Exception	Solution
The images are damaged and cannot be decoded.	Filters out images that cannot be decoded.
The image channel can be channel 1 or channel 2. Channel 3 is not commonly used.	Converts images into RGB three-channel images.
The image format is not supported by ModelArts.	Converts the image format to JPG.
The image suffix is inconsistent with the actual format, but the format is supported by ModelArts.	Coverts the suffix to the actual format.
The image suffix is inconsistent with the actual format and the format is not supported by ModelArts.	Converts the image format to JPG.

Exception	Solution
The image resolution is too high.	The image width and height are cropped based on the specified size and ratio.

Table 2-2 Labeling file data validation

Exception	Solution
The XML structure is incomplete and cannot be parsed.	Filters XML files.
No labeled object is in the XML file.	Filters XML files.
The XML file does not contain rectangle **bndbox**.	Filters XML files.
Some labeled objects do not have rectangle **bndbox**.	Filters labeled objects.
After an image is cropped, the width and height of the image are inconsistent with those in the XML file.	Changes the values of the width and height parameters to the actual width and height of the image.
No **width** and **height** fields exist in XML files.	Supplements the **width** and **height** fields and values in the XML file based on the actual width and height of the image.
After an image is cropped, its size is inconsistent with the size of rectangle **bndbox** in the XML file.	Changes the value of **bndbox** in the XML file based on the image cropping ratio.
The width or height of rectangle **bndbox** in the XML file is too small and is displayed as a line.	If the difference between the width and height of the rectangle is less than 2, removes the current object.
In the XML file, the minimum value of rectangle **bndbox** is greater than the maximum value.	Removes the current object.
Rectangle **bndbox** exceeds the image boundaries, and the excess part occupies more than 50% of the frame area.	Removes the current object.
Rectangle **bndbox** exceeds the image boundaries, and the excess part occupies less than 50% of the frame area.	Rectangle **bndbox** is pulled back to the image boundaries.

NOTE

Original data is not changed during data validation. The newly validated image or XML file is saved in the specified output path.

Parameters

Table 2-3 Parameters of the MetaValidation operator for data validation

Name	Mandatory	Default	Description
image_max_width	No	-1	Maximum width of an input image. If the width of an input image exceeds the configured value, the image is cropped based on the ratio. The unit is pixel. The default value **-1** indicates that the image is not cropped.
image_max_height	No	-1	Maximum length of an input image. If the length of an input image exceeds the configured value, the image is cropped based on the ratio. The unit is pixel. The default value **-1** indicates that the image is not cropped.

Operator Input Requirements

The following two types of operator input are available:

- **Datasets**: Select a dataset and its version created on the ModelArts console from the drop-down list. Ensure that the dataset type be the same as the scenario type selected in this task.
- **OBSCatalog**: Select either of the following storage structures:
  - **Only images**: If the directory contains only images, the JPG, JPEG, PNG, and BMP formats are supported, and all images in the nested subdirectories are read.
  - **Images and labels**: The structure varies depending on the scenario type.

    The following shows the directory structure in the image classification scenario. The following directory structure supports only single-label scenarios.

    ```
    input_path/
      --label1/
          ----1.jpg
      --label2/
          ----2.jpg
      --../
    ```

    The following shows the directory structure in the object detection scenario. Images in JPG, JPEG, PNG, and BMP formats are supported. XML files are standard PACAL VOC files.

    ```
    input_path/
      --1.jpg
    ```



```
  --1.xml
  --2.jpg
  --2.xml
  ...
```

Output Description

- **Image classification**

  The output directory structure is as follows:

```
output_path/
   --Data/
       ----class1/  # If the input data has labeling information, the information is also output. class1 indicates the labeling class.
           ------1.jpg
           ------2_checked.jpg
       ----class2/
           ------3.jpg
           ------4_checked.jpg
       ----5_checked.jpg
   --output.manifest
```

  A manifest file example is as follows: The validation attribute **"property": {"@modelarts:data_checked":true}** is added for each data record.

```
{
  "id": "xss",
  "source": "obs://hard_example_path/Data/fc8e2688015d4a1784dcbda44d840307_14_checked.jpg",
  "property": {
    "@modelarts:data_checked": true
  },
  "usage": "train",
  "annotation": [
    {
      "name": "Cat",
      "type": "modelarts/image_classification"
    }
  ]
}
```

- **Object detection**

  The output directory structure is as follows:

```
output_path/
   --Data/
       ----1_checked.jpg
        ----1_checked.xml # If the input data is converted during validation, '_checked' is added to the file name.
       ----2.jpg          # If the input data is not converted, the file is saved with the original name.
       ----2.xml
   --output.manifest
```

  A manifest file example is as follows: The validation attribute **"property": {"@modelarts:data_checked":true}** is added for each data record.

```
{
  "source": "obs://hard_example_path/Data/be462ea9c5abc09f_checked.jpg",
  "property": {
    "@modelarts:data_checked": true
  },
  "annotation": [
    {
      "annotation-loc": "obs://hard_example_path/Data/be462ea9c5abc09f_checked.xml",
      "type": "modelarts/object_detection",
      "annotation-format": "PASCAL VOC",
      "annotated-by": "modelarts/hard_example_algo"
    }
  ]
}
```


2.2 Data Cleansing

PCC Operator Overview

ModelArts data cleansing is implemented by the PCC operator. The dataset used for image classification or object detection may contain images that do not belong to the required categories. These images need to be removed to avoid interference to labeling and model training.

Description

Table 2-4 Parameters of the PCC operator for data cleansing

Name	Mandatory	Default	Description
prototype_sample_path	Yes	None	Directory for storing positive data cleansing samples. The directory stores positive sample image files. The algorithm filters input data based on the positive sample images. That is, the data that is highly similar to the images in the **prototype_sample_path** directory is retained. Enter an existing OBS directory. The directory contains the provided positive sample images and starts with **obs://**, for example, *obs://obs_bucket_name/folder_name*.
criticism_sample_path	No	None	Directory for storing negative data cleansing samples. The directory stores negative sample image files. The algorithm filters input data based on the negative sample images. That is, the data that is less similar to the images in the **criticism_sample_path** directory is retained. It is recommended that this parameter be used together with **prototype_sample_path** to improve the accuracy of data cleansing. Enter an existing OBS directory that starts with **obs://**, for example, *obs://obs_bucket_name/folder_name*.
n_clusters	No	auto	Number of data sample types. The default value is **auto**. You can enter an integer less than the total number of samples or **auto**. **auto** indicates that the number of images in the positive sample directory is used as the number of data sample types.

Name	Mandatory	Default	Description
simlarity_threshold	No	0.9	Similarity threshold. If the similarity between two images exceeds the threshold, the two images are regarded as similar. Otherwise, they are regarded as dissimilar. The value ranges from 0 to 1.
embedding_distance	No	0.2	Distance between sample features. If the feature distance between two images is smaller than the specified value, the two images are regarded as similar. Otherwise, the two images are regarded as dissimilar. The value ranges from 0 to 1.
do_validation	No	True	Indicates whether to validate data. The value can be **True** or **False**. **True** indicates that data is validated before cleansing. **False** indicates that data is cleansed only.

Operator Input Requirements

The following two types of operator input are available:

- **Datasets**: Select a dataset and its version created on the ModelArts console from the drop-down list. Ensure that the dataset type be the same as the scenario type selected in this task.
- **OBSCatalog**: Select either of the following storage structures:
  - **Only images**: If the directory contains only images, the JPG, JPEG, PNG, and BMP formats are supported, and all images in the nested subdirectories are read.
  - **Images and labels**: The structure varies depending on the scenario type.

    The following shows the directory structure in the image classification scenario. The following directory structure supports only single-label scenarios.

    ```
    input_path/
      --label1/
          ----1.jpg
      --label2/
          ----2.jpg
      --../
    ```

    The following shows the directory structure in the object detection scenario. Images in JPG, JPEG, PNG, and BMP formats are supported. XML files are standard PACAL VOC files.

    ```
    input_path/
      --1.jpg
      --1.xml
      --2.jpg
      --2.xml
      ...
    ```


Output Description

- **Image classification**

  The output directory structure is as follows:

```
output_path/
    --Data/
        ----class1/  # If the input data has labeling information, the information is also output. class1 indicates the labeling class.
            ------1.jpg
        ----class2/
            ------2.jpg
        ----3.jpg
    --output.manifest
```

  A manifest file example is as follows:

```
{
    "id": "xss",
    "source": "obs://home/fc8e2688015d4a1784dcbda44d840307_14.jpg",
    "usage": "train",
    "annotation": [
        {
            "name": "Cat",
            "type": "modelarts/image_classification"
        }
    ]
}
```

- **Object detection**

  The output directory structure is as follows:

```
output_path/
    --Data/
        ----1.jpg
        ----1.xml  # If the input data has labeling information, the information is also output. xml indicates the label file.
        ----2.jpg
        ----3.jpg
    --output.manifest
```

  A manifest file example is as follows:

```
{
    "source":"obs://fake/be462ea9c5abc09f.jpg",
    "annotation":[
        {
            "annotation-loc":"obs://fake/be462ea9c5abc09f.xml",
            "type":"modelarts/object_detection",
            "annotation-format":"PASCAL VOC",
            "annotated-by":"modelarts/hard_example_algo"
        }
    ]
}
```

2.3 Data Selection

2.3.1 Data Deduplication

SimDeduplication Operator Overview

- The SimDeduplication operator can implement image deduplication based on the similarity threshold you set. Image deduplication is a common method for image data processing. Image duplication means that the image content is completely the same, or the scale, displacement, color, or brightness changes slightly, or a small amount of other content is added.

Table 2-5 Advanced parameters

Name	Mandatory	Default	Description
simlarity_threshold	No	0.9	Similarity threshold. When the similarity between two images is greater than the threshold, one of the images is filtered out as a duplicate image. The value ranges from 0 to 1.
do_validation	No	True	Indicates whether to validate data. The value can be **True** or **False**. **True** indicates that data is validated before deduplication. **False** indicates that data is deduplicated only.

Operator Input Requirements

The following two types of operator input are available:

- **Datasets**: Select a dataset and its version created on the ModelArts console from the drop-down list. Ensure that the dataset type be the same as the scenario type selected in this task.
- **OBSCatalog**: Select either of the following storage structures:
  - **Only images**: If the directory contains only images, the JPG, JPEG, PNG, and BMP formats are supported, and all images in the nested subdirectories are read.
  - **Images and labels**: The structure varies depending on the scenario type.

    The following shows the directory structure in the image classification scenario. The following directory structure supports only single-label scenarios.

    ```
    input_path/
      --label1/
          ----1.jpg
      --label2/
          ----2.jpg
      --../
    ```

    The following shows the directory structure in the object detection scenario. Images in JPG, JPEG, PNG, and BMP formats are supported. XML files are standard PACAL VOC files.

    ```
    input_path/
      --1.jpg
      --1.xml
      --2.jpg
      --2.xml
      ...
    ```

Output Description

- **Image classification**

  The output directory structure is as follows:


```
output_path/
    --Data/
        ----class1/  # If the input data has labeling information, the information is also output. class1 indicates the labeling class.
            ------1.jpg
        ----class2/
            ------2.jpg
            ------3.jpg
    --output.manifest
```

A manifest file example is as follows:

```
{
    "id": "xss",
    "source": "obs://home/fc8e2688015d4a1784dcbda44d840307_14.jpg",
    "usage": "train",
    "annotation": [
        {
            "name": "Cat",
            "type": "modelarts/image_classification"
        }
    ]
}
```

- **Object detection**

  The output directory structure is as follows:

```
output_path/
    --Data/
        ----1.jpg
        ----1.xml  # If the input data has labeling information, the information is also output. xml indicates the label file.
        ----2.jpg
        ----3.jpg
    --output.manifest
```

  A manifest file example is as follows:

```
{
    "source":"obs://fake/be462ea9c5abc09f.jpg",
    "annotation":[
        {
            "annotation-loc":"obs://fake/be462ea9c5abc09f.xml",
            "type":"modelarts/object_detection",
            "annotation-format":"PASCAL VOC",
            "annotated-by":"modelarts/hard_example_algo"
            }
    ]
}
```

2.3.2 Data Deredundancy

RRD Operator Overview

The data with the largest difference can be removed based on the preset proportion.

Table 2-6 Advanced parameters

Name	Mandatory	Default	Description
sample_ratio	No	0.9	Percentage of reserved data. The value ranges from 0 to 1. For example, **0.9** indicates that 90% of the original data is reserved.

Name	Mandatory	Default	Description
n_clusters	auto	auto	Number of data sample types. The default value is **auto**, indicating that the total number of types is obtained based on the number of images in the directory. For example, you can specify the number of types to **4**.
do_validation	No	True	Indicates whether to validate data. The value can be **True** or **False**. **True** indicates that data is validated before deredundancy. **False** indicates that data is deduplicated only.

Operator Input Requirements

The following two types of operator input are available:

- **Datasets**: Select a dataset and its version created on the ModelArts console from the drop-down list. Ensure that the dataset type be the same as the scenario type selected in this task.
- **OBSCatalog**: Select either of the following storage structures:
  - **Only images**: If the directory contains only images, the JPG, JPEG, PNG, and BMP formats are supported, and all images in the nested subdirectories are read.
  - **Images and labels**: The structure varies depending on the scenario type.

    The following shows the directory structure in the image classification scenario. The following directory structure supports only single-label scenarios.

    ```
    input_path/
      --label1/
          ----1.jpg
      --label2/
          ----2.jpg
      --../
    ```

    The following shows the directory structure in the object detection scenario. Images in JPG, JPEG, PNG, and BMP formats are supported. XML files are standard PACAL VOC files.

    ```
    input_path/
      --1.jpg
      --1.xml
      --2.jpg
      --2.xml
      ...
    ```

Output Description

- **Image classification**

  The output directory structure is as follows:

  ```
  output_path/
    --Data/
        ----class1/  # If the input data has labeling information, the information is also output. class1 indicates the labeling class.
  ```



```
        ------1.jpg
    ----class2/
        ------2.jpg
        ------3.jpg
--output.manifest
```

A manifest file example is as follows:

```
{
    "id": "xss",
    "source": "obs://home/fc8e2688015d4a1784dcbda44d840307_14.jpg",
    "usage": "train",
    "annotation": [
        {
            "name": "Cat",
            "type": "modelarts/image_classification"
        }
    ]
}
```

- **Object detection**

  The output directory structure is as follows:

```
output_path/
    --Data/
        ----1.jpg
        ----1.xml  # If the input data has labeling information, the information is also output. xml indicates the label file.
        ----2.jpg
        ----3.jpg
    --output.manifest
```

  A manifest file example is as follows:

```
{
    "source":"obs://fake/be462ea9c5abc09f.jpg",
    "annotation":[
        {
            "annotation-loc":"obs://fake/be462ea9c5abc09f.xml",
            "type":"modelarts/object_detection",
            "annotation-format":"PASCAL VOC",
            "annotated-by":"modelarts/hard_example_algo"
            }
    ]
}
```

2.4 Data Augmentation

2.4.1 Data Augmentation

Overview of Data Augmentation Operators

Data augmentation is mainly used in scenarios where training data is insufficient or simulation is required. You can transform a labeled dataset to increase the number of images for training and generate corresponding labels. In the deep learning field, augmentation is of great significance. It can improve model generalization and enhance anti-disturbance. Original data is not changed during data augmentation. A newly augmented image or XML file is saved in the specified output path.

ModelArts provides the following data augmentation operators:

Table 2-7 Description of data augmentation operators

Operator	Description	Advanced
AddNoise	Adds noises to simulate the noises that may be generated when common capture devices capture images.	• **noise_type**: type of noise added. **Gauss** indicates Gaussian noise. **Laplace** indicates Laplace noise. **Poisson** indicates Poisson noise. **Impulse** indicates impulse noise. **SaltAndPepper** indicates salt and pepper noise. The default value is **Gauss**. • **loc**: average noise distribution. This parameter is valid only in **Gauss** and **Laplace**. The default value is **0**. • **scale**: standard deviation of noise distribution. This parameter is valid only in **Gauss** and **Laplace**. The default value is **1**. • **lam**: lambda coefficient of Poisson distribution. This parameter is valid only in Poisson. The default value is **2**. • **p**: probability of pulse noise or salt-and-pepper noise for each pixel. This parameter is valid only for **Impulse** and **SaltAndPepper**. The default value is **0.01**. • **do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.
Blur	Uses filters to filter images and sometimes to simulate imaging of imaging devices.	• **blur_type**: The value can be **Gauss** or **Average**, which indicates Gaussian filtering and average filtering, respectively. The default value is **Gauss**. • **do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.

Operator	Description	Advanced
Crop	Randomly crops a part of an image to generate a new image.	• **crop_percent_min**: minimum value in the value range of the cropping ratio of each edge. The default value is **0.0**. • **crop_percent_max**: maximum value in the value range of the cropping ratio of each edge. The default value is **0.2**. • **do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.
CutOut	Random erase, which is a common method used in deep learning to simulate an object that is blocked by an obstacle.	**do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.
Flip	Flips along the horizontal or vertical axis of an image, which is a very common augmentation method.	• **lr_ud**: flipping direction. **lr** indicates horizontal flipping, and **ud** indicates vertical flipping. The default value is **lr**. • **flip_p**: flipping probability. The default value is **1**. • **do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.
Grayscale	Changes a three-channel color image to a three-channel grayscale image.	**do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.
HistogramEqual	Indicates the histogram equalization, which is mainly used to improve the visual effect of images. It is used in some scenarios.	**do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.
LightArithmetic	Implements linear enhancement on luminance space.	**do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.

Operator	Description	Advanced
LightContrast	Enhances luminance contrast. A certain non-linear function is used to change the luminance value of the luminance space.	**func**: The default value is **gamma**. • **gamma**: Gamma correction. Its formula is 255*((v/255)**gamma)'). • **sigmoid**: S-shaped curve function. Its formula is 255*1/(1+exp(gain*(cutoff-I_ij/255)))'). • **log**: logarithmic function. Its formula is 255*gain*log_2(1+v/255). • **linear**: linear function. Its formula is 127 + alpha*(v-127)'). **do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.
MotionBlur	Indicates the motion blur generated when an object moves.	**do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.
Padding	Pads an image with black edges.	• **px_top**: number of pixel lines added at the top of the image. The default value is **1**. • **px_right**: number of pixel lines added at the right of the image. The default value is **1**. • **px_left**: number of pixel lines added at the left of the image. The default value is **1**. • **px_bottom**: number of pixel lines added at the bottom of the image. The default value is **1**. • **do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.
Resize	Resizes an image.	• **height**: height of the image after conversion. The default value is **224**. • **width**: width of the image after conversion. The default value is **224**. • **do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.

Operator	Description	Advanced
Rotate	Rotates an image around the center point. After the operation is complete, the original shape of the image remains unchanged, and the blank part is filled with black.	• **angle_min**: minimum value in the range of rotation angles. Each image randomly obtains a value from the range. The default value is **90°**. • **angle_max**: maximum value in the range of rotation angles. Each image randomly obtains a value from the range. The default value is **-90°**. • **do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.
Saturation	Enhances chrominance and saturation. The H and S spaces in the HSV of an image are changed linearly to change the chrominance and saturation of the image.	**do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.
Scale	Zooms in or out an image. The length or width of an image is randomly zoomed in or out.	• **scaleXY**: scaling direction. **X** indicates horizontal, and **Y** indicates vertical. The default value is **X**. • **scale_min**: lower limit of the random scaling ratio range. The default value is **0.5**. • **scale_max**: upper limit of the random scaling ratio range. The default value is **1.5**. • **do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.
Sharpen	Indicates image sharpening, which is used to sharpen the edges of objects.	**do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.

Operator	Description	Advanced
Shear	Indicates image shearing, which is used for geometric transformation of images. Pixels are mapped using linear functions.	• **shearXY**: shearing direction. **X** indicates horizontal, and **Y** indicates vertical. The default value is **X**. • **shear_min**: lower limit of the random shearing angle range. The default value is **-30**. • **shear_max**: upper limit of the random shearing angle range. The default value is **30**. • **do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.
Translate	Moves an image along the x-axis or y-axis, discards the part that exceeds the original image, and fills the blank part with black.	• **translateXY**: translation direction. **X** indicates horizontal, and **Y** indicates vertical. The default value is **X**. • **do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.
Weather	Adds weather information to simulate the weather effect.	**weather_mode**: weather mode. The default value is **Rain**. • **Rain**: rain • **Fog**: fog • **Snow**: snow • **Clouds**: cloud **do_validation**: indicates whether to validate data before data augmentation. The default value is **True**.

Operator Input Requirements

The following two types of operator input are available:

- **Datasets**: Select a dataset and its version created on the ModelArts console from the drop-down list. Ensure that the dataset type be the same as the scenario type selected in this task.
- **OBS Catalog**: The storage structure supports **Images and labels**.

  Images and labels: The structure varies depending on the scenario type.

  The following shows the directory structure in the image classification scenario. The following directory structure supports only single-label scenarios.

  ```
  input_path/
    --label1/
  ```



```
    ----1.jpg
  --label2/
    ----2.jpg
  --../
```

The following shows the directory structure in the object detection scenario. Images in JPG, JPEG, PNG, and BMP formats are supported. XML files are standard PACAL VOC files.

```
input_path/
  --1.jpg
  --1.xml
  --2.jpg
  --2.xml
  ...
```

Output Description

Some data will be discarded due to some algorithm operations. Therefore, the output folder may not contain the full dataset. For example, **Rotate** will discard the images whose bounding boxes exceed the image boundaries.

The following shows the output directory structure. In this structure, the **Data** folder stores newly generated images and labeling information. The **manifest** file stores the structure of images in the folder and can be directly imported to the dataset in Data Management.

```
|----data_url
  |----Data
    |----xxx.jpg
    |----xxx.xml(xxx.txt)
  |----output.manifest
```

A manifest file example is as follows:

```
{
  "id": "xss",
  "source": "obs://home/fc8e2688015d4a1784dcbda44d840307_14.jpg",
  "usage": "train",
  "annotation": [
    {
      "name": "Cat",
      "type": "modelarts/image_classification"
    }
  ]
}
```

2.4.2 Data Generation

Introduction to Data Generation

The image generation uses a generative adversarial network (GAN) to generate a new dataset with the existing dataset. A GAN is a network that contains a generator and discriminator. The generator randomly selects samples from a latent space as the input, and outputs results similar to the real samples in the training set. The discriminator distinguishes the outputs of the generative network from the real samples by inputting real samples or outputs of the generative network. The generative network's training objective is to increase the error rate of the discriminative network (that is, "fool" the discriminative network). The two networks contest with each other and continuously adjust parameters to achieve the final purpose, that is, make the discriminative network unable to distinguish

whether the output of the generative network is true. The generative network obtained during training can be used to generate images similar to the input images, which can be used as new datasets for training. New datasets generated based on GANs have no labels. Original data is not changed during image generation. The newly generated image or XML file is saved in the specified output path.

StyleGAN Operator Overview

StyleGAN operator randomly generates similar images based on StyleGAN2 when a dataset is small. StyleGAN has a new generator architecture that can control the high-level attributes such as hairstyle and freckles of the generated images. These generated images perform even better in certain aspects. In addition, StyleGAN is equipped with the data augmentation algorithm, which can generate new satisfactory samples even with a small number of samples. However, there must be at least 70 samples to have rich image styles.

Table 2-8 Advanced parameters of the StyleGAN operator

Name	Default	Description
resolution	256	Height and width of the generated square image. The value must be 2 to the power of *n*.
batch-size	8	Number of samples for batch training
total-kimg	300	The total number of trained images is obtained by multiplying this parameter value by 1,000.
generate_num	300	Number of generated images. If the generated images have multiple classes, the value is the number of generated images of each class.
predict	False	Whether to perform inference and prediction. The default value is **False**. If this parameter is set to **True**, you need to set **resume** to the OBS path where the trained model is stored.
resume	empty	If **predict** is set to **True**, enter the OBS path where the TensorFlow model file is stored for inference and prediction. Currently, only models in **.pb** format are supported. Example: **obs://xxx/xxxx.pb**. The default value is **empty**.
do_validation	True	Whether to validate data. The default value is **True**, which indicates that data is validated before being generated. **False** indicates data is generated directly.

Data Input

The following two types of operator input are available:

- **Datasets**: Select a dataset and its version created on the ModelArts console from the drop-down list. Ensure that the dataset type be the same as the scenario type selected in this task.
- **OBSCatalog**: Operator image_generation does not require labeling information and its input supports single-level or dual-level directories, and the storage structure supports single-level or dual-level mode.

The single-level directory structure is as follows:

```
image_folder----0001.jpg
          ----0002.jpg
          ----0003.jpg
          ...
          ----1000.jpg
```

The dual-level directory structure is as follows:

```
image_folder----sub_folder_1----0001.jpg
                           ----0002.jpg
                           ----0003.jpg
                           ...
                           ----0500.jpg
          ----sub_folder_2----0001.jpg
                           ----0002.jpg
                           ----0003.jpg
                           ...
                           ----0500.jpg
                           ...
          ----sub_folder_100----0001.jpg
                            ----0002.jpg
                            ----0003.jpg
                            ...
                            ----0500.jpg
```

Output Description

The output directory structure is as follows: The folder **model** stores model **frozen PB** for inference, the folder **samples** stores the output images during training, and the folder **Data** stores the images generated by the training model.

```
train_url----model----CYcleGan_epoch_10.pb
                ----CYcleGan_epoch_20.pb
                ...
                ----CYcleGan_epoch_1000.pb
        ----samples----0000_0.jpg
                 ----0000_1.jpg
                 ...
                 ----0100_15.jpg
        ----Data----CYcleGan_0_0.jpg
                ----CYcleGan_0_1.jpg
                ...
                ----CYcleGan_16_8.jpg
        ----output_0.manifest
```

A manifest file example is as follows:

```
{
    "id": "xss",
    "source": "obs://home/fc8e2688015d4a1784dcbda44d840307_14.jpg",
    "usage": "train",
    "annotation": [
        {
            "name": "Cat",
            "type": "modelarts/image_classification"
        }
```



```
  ]
}
```

2.4.3 Data Transfer Between Domains

CycleGAN Operator Overview

CycleGAN operator generates images for domain transfer based on CycleGAN, that is, converts one type of images into another, or converts samples in the *X* space into samples in the *Y* space. CycleGAN can use non-paired data for training. During model training, two inputs are supported, indicating the source domain and target domain of data, respectively. After the training is complete, all images transferred from the source domain to the target domain are generated.

Table 2-9 Advanced parameters of the CycleGAN operator

Name	Default	Description
do_validation	True	Whether to validate data. The default value is **True**, which indicates that data is validated before being generated. **False** indicates data is generated directly.
image_channel	3	Number of channels of the image
image_height	256	Image height. The value must be 2 to the power of *n*.
image_width	256	Image width. The value must be 2 to the power of *n*.
batch_size	1	Number of samples for batch training
max_epoch	100	Number of dataset epochs during training
g_learning_rate	0.0001	Learning rate for the generator training
d_learning_rate	0.0001	Learning rate for the discriminator training
log_frequency	5	Logging frequency (counted by step)
save_frequency	5	Model save frequency (counted by epoch)
predict	False	Whether to perform inference and prediction. The default value is **False**. If this parameter is set to **True**, you need to set **resume** to the OBS path where the trained model is stored.

Name	Default	Description
resume	empty	If **predict** is set to **True**, enter the OBS path where the TensorFlow model file is stored for inference and prediction. Currently, only models in **.pb** format are supported. Example: **obs://xxx/xxxx.pb**. The default value is **empty**.

Data Input

The following two types of operator input are available:

- **Datasets**: Select a dataset and its version created on the ModelArts console from the drop-down list. Ensure that the dataset type be the same as the scenario type selected in this task.
- **OBSCatalog**: Operator image_generation does not require labeling information and its input supports single-level or dual-level directories, and the storage structure supports single-level or dual-level mode.

The single-level directory structure is as follows:

```
image_folder----0001.jpg
          ----0002.jpg
          ----0003.jpg
          ...
          ----1000.jpg
```

The dual-level directory structure is as follows:

```
image_folder----sub_folder_1----0001.jpg
                          ----0002.jpg
                          ----0003.jpg
                          ...
                          ----0500.jpg
          ----sub_folder_2----0001.jpg
                          ----0002.jpg
                          ----0003.jpg
                          ...
                          ----0500.jpg
                          ...
          ----sub_folder_100----0001.jpg
                            ----0002.jpg
                            ----0003.jpg
                            ...
                            ----0500.jpg
```

Output Description

The output directory structure is as follows: The folder **model** stores model **frozen PB** for inference, the folder **samples** stores the output images during training, and the folder **Data** stores the images generated by the training model.

```
train_url----model----CYcleGan_epoch_10.pb
                ----CYcleGan_epoch_20.pb
                ...
                ----CYcleGan_epoch_1000.pb
        ----samples----0000_0.jpg
                 ----0000_1.jpg
                 ...
```



```
                ----0100_15.jpg
        ----Data----CYcleGan_0_0.jpg
                ----CYcleGan_0_1.jpg
                ...
                ----CYcleGan_16_8.jpg
        ----output_0.manifest
```

A manifest file example is as follows:

```
{
    "id": "xss",
    "source": "obs://home/fc8e2688015d4a1784dcbda44d840307_14.jpg",
    "usage": "train",
    "annotation": [
        {
            "name": "Cat",
            "type": "modelarts/image_classification"
        }
    ]
}
```

