

Incremental Training in Amazon SageMaker

[PDF \(sagemaker-dg.pdf#incremental-training\)](#)

[Kindle \(https://www.amazon.com/dp/B07JVSBS9J\)](https://www.amazon.com/dp/B07JVSBS9J)

[RSS \(amazon-sagemaker-release-notes.rss\)](#)

Over time, you might find that a model generates inference that are not as good as they were in the past. With incremental training, you can use the artifacts from an existing model and use an expanded dataset to train a new model. Incremental training saves both time and resources.

Use incremental training to:

- Train a new model using an expanded dataset that contains an underlying pattern that was not accounted for in the previous training and which resulted in poor model performance.
- Use the model artifacts or a portion of the model artifacts from a popular publicly available model in a training job. You don't need to train a new model from scratch.
- Resume a training job that was stopped.
- Train several variants of a model, either with different hyperparameter settings or using different datasets.

For more information about training jobs, see [Train a Model with Amazon SageMaker \(/how-it-works-training.html\)](#).

You can train incrementally using the SageMaker console or the [Amazon SageMaker Python SDK](#) [\(<https://sagemaker.readthedocs.io>\)](https://sagemaker.readthedocs.io).

Important

Only three built-in algorithms currently support incremental training: [Object Detection Algorithm \(/object-detection.html\)](#), [Image Classification Algorithm \(/image-classification.html\)](#), and [Semantic Segmentation Algorithm \(/semantic-segmentation.html\)](#).

Topics

- [Perform Incremental Training \(Console\) \(#incremental-training-console\)](#)
- [Perform Incremental Training \(API\) \(#incremental-training-api\)](#)


Perform Incremental Training (Console)

To complete this procedure, you need:

- The Amazon Simple Storage Service (Amazon S3) bucket URI where you've stored the training data.
- The S3 bucket URI where you want to store the output of the job.
- The Amazon Elastic Container Registry path where the training code is stored. For more information, see [Docker Registry Paths and Example Code \(/sagemaker-algo-docker-registry-paths.html\)](#).
- The URL of the S3 bucket where you've stored the model artifacts that you want to use in incremental training. To find the URL for the model artifacts, see the details page of the training job used to create the model. To find the details page, in the SageMaker console, choose **Inference**, choose **Models**, and then choose the model.

To restart a stopped training job, use the URL to the model artifacts that are stored in the details page as you would with a model or a completed training job.

To perform incremental training (console)

1. Open the Amazon SageMaker console at <https://console.aws.amazon.com/sagemaker/>  (<https://console.aws.amazon.com/sagemaker/>) .
2. In the navigation pane, choose **Training**, then choose **Training jobs**.
3. Choose **Create training job**.
4. Provide a name for the training job. The name must be unique within an AWS Region in an AWS account. The training job name must have 1 to 63 characters. Valid characters: a-z, A-Z, 0-9, and . : + = @ _ % - (hyphen).
5. Choose the algorithm that you want to use. For information about algorithms, see [Use Amazon SageMaker Built-in Algorithms \(./algos.html\)](#) .
6. (Optional) For **Resource configuration**, either leave the default values or increase the resource consumption to reduce computation time.
 1. (Optional) For **Instance type**, choose the ML compute instance type that you want to use. In most cases, **ml.m4.xlarge** is sufficient.
 2. For **Instance count**, use the default, 1.
 3. (Optional) For **Additional volume per instance (GB)**, choose the size of the ML storage volume that you want to provision. In most cases, you can use the default, 1. If you are using a large dataset, use a larger size.
7. Provide information about the input data for the training dataset.
 1. For **Channel name**, either leave the default (train) or enter a more meaningful name for the training dataset, such as expanded-training-dataset.
 2. For **InputMode**, choose **File**. For incremental training, you need to use file input mode.
 3. For **S3 data distribution type**, choose **FullyReplicated**. This causes each ML compute instance to use a full replicate of the expanded dataset when training incrementally.
 4. If the expanded dataset is uncompressed, set the **Compression type** to **None**. If the expanded dataset is compressed using Gzip, set it to **Gzip**.
 5. (Optional) If you are using File input mode, leave **Content type** empty. For Pipe input mode, specify the appropriate MIME type. *Content type* is the multipurpose internet mail extension (MIME) type of the data.
 6. For **Record wrapper**, if the dataset is saved in RecordIO format, choose **RecordIO**. If your dataset is not saved as a RecordIO formatted file, choose **None**.
 7. For **S3 data type**, if the dataset is stored as a single file, choose **S3Prefix**. If the dataset is stored as several files in a folder, choose **Manifest**.
 8. For **S3 location**, provide the URL to the path where you stored the expanded dataset.
 9. Choose **Done**.

8. To use model artifacts in a training job, you need to add a new channel and provide the needed information about the model artifacts.
 1. For **Input data configuration**, choose **Add channel**.
 2. For **Channel name**, enter `model` to identify this channel as the source of the model artifacts.
 3. For **InputMode**, choose **File**. Model artifacts are stored as files.
 4. For **S3 data distribution type**, choose **FullyReplicated**. This indicates that each ML compute instance should use all of the model artifacts for training.
 5. For **Compression type**, choose **None** because we are using a model for the channel.
 6. Leave **Content type** empty. Content type is the multipurpose internet mail extension (MIME) type of the data. For model artifacts, we leave it empty.
 7. Set **Record wrapper** to **None** because model artifacts are not stored in RecordIO format.
 8. For **S3 data type**, if you are using a built-in algorithm or an algorithm that stores the model as a single file, choose **S3Prefix**. If you are using an algorithm that stores the model as several files, choose **Manifest**.
 9. For **S3 location**, provide the URL to the path where you stored the model artifacts. Typically, the model is stored with the name `model.tar.gz`. To find the URL for the model artifacts, in the navigation pane, choose **Inference**, then choose **Models**. From the list of models, choose a model to display its details page. The URL for the model artifacts is listed under **Primary container**.
 10. Choose **Done**.
9. For **Output data configuration**, provide the following information:
 1. For **S3 location**, type the path to the S3 bucket where you want to store the output data.
 2. (Optional) For **Encryption key**, you can add your AWS Key Management Service (AWS KMS) encryption key to encrypt the output data at rest. Provide the key ID or its Amazon Resource Number (ARN). For more information, see [KMS-Managed Encryption Keys](https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingKMSEncryption.html) (<https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingKMSEncryption.html>).
10. (Optional) For **Tags**, add one or more tags to the training job. A *tag* is metadata that you can define and assign to AWS resources. In this case, you can use tags to help you manage your training jobs. A tag consists of a key and a value, which you define. For example, you might want to create a tag with `Project` as a key and a value referring to a project that is related to the training job, such as `Home value forecasts`.
11. Choose **Create training job**. SageMaker creates and runs training job.

After the training job has completed, the newly trained model artifacts are stored under the **S3 output path** that you provided in the **Output data configuration** field. To deploy the model to get predictions, see [Step 5: Deploy the Model to Amazon EC2 \(.ex1-model-deployment.html\)](#).

Perform Incremental Training (API)

This example shows how to use SageMaker APIs to train a model using the SageMaker image classification algorithm and the [Caltech 256 Image Dataset](http://www.vision.caltech.edu/Image_Datasets/Caltech256/) (http://www.vision.caltech.edu/Image_Datasets/Caltech256/), then train a new

model using the first one. It uses Amazon S3 for input and output sources. Please see the [incremental training sample notebook](https://sagemaker-examples.readthedocs.io/en/latest/introduction_to_amazon_algorithms/object_detection_pascalvoc_coco/object_detection_incremental_training.html) (https://sagemaker-examples.readthedocs.io/en/latest/introduction_to_amazon_algorithms/object_detection_pascalvoc_coco/object_detection_incremental_training.html) for more details on using incremental training.

Note

In this example we used the original datasets in the incremental training, however you can use different datasets, such as ones that contain newly added samples. Upload the new datasets to S3 and make adjustments to the `data_channels` variable used to train the new model.

Get an AWS Identity and Access Management (IAM) role that grants required permissions and initialize environment variables:

```
import sagemaker
from sagemaker import get_execution_role
```

```
role = get_execution_role()
print(role)
```

```
sess = sagemaker.Session()
```

```
bucket=sess.default_bucket()
print(bucket)
prefix = 'ic-incr-training'
```

Get the training image for the image classification algorithm:

```
from sagemaker.amazon.amazon_estimator import get_image_uri

training_image = get_image_uri(sess.boto_region_name, 'image-classification',
                               repo_version="latest")
#Display the training image
print (training_image)
```

Download the training and validation datasets, then upload them to Amazon Simple Storage Service (Amazon S3):

```
import os
import urllib.request
import boto3

# Define a download function
def download(url):
    filename = url.split("/")[-1]
    if not os.path.exists(filename):
        urllib.request.urlretrieve(url, filename)

# Download the caltech-256 training and validation datasets
download('http://data.mxnet.io/data/caltech-256/caltech-256-60-train.rec')
download('http://data.mxnet.io/data/caltech-256/caltech-256-60-val.rec')
```

```
# Create four channels: train, validation, train_lst, and validation_lst
s3train = 's3://{}/{}/train/'.format(bucket, prefix)
s3validation = 's3://{}/{}/validation/'.format(bucket, prefix)

# Upload the first files to the train and validation channels
!aws s3 cp caltech-256-60-train.rec $s3train --quiet
!aws s3 cp caltech-256-60-val.rec $s3validation --quiet
```

Define the training hyperparameters:

```
# Define hyperparameters for the estimator
hyperparams = { "num_layers": "18",
                "resize": "32",
                "num_training_samples": "50000",
                "num_classes": "10",
                "image_shape": "3,28,28",
                "mini_batch_size": "128",
                "epochs": "3",
                "learning_rate": "0.1",
                "lr_scheduler_step": "2,3",
                "lr_scheduler_factor": "0.1",
                "augmentation_type": "crop_color",
                "optimizer": "sgd",
                "momentum": "0.9",
                "weight_decay": "0.0001",
                "beta_1": "0.9",
                "beta_2": "0.999",
                "gamma": "0.9",
                "eps": "1e-8",
                "top_k": "5",
                "checkpoint_frequency": "1",
                "use_pretrained_model": "0",
                "model_prefix": "" }
```

Create an estimator object and train the first model using the training and validation datasets:

```
# Fit the base estimator
s3_output_location = 's3://{}/{}/output'.format(bucket, prefix)
ic = sagemaker.estimator.Estimator(training_image,
                                    role,
                                    instance_count=1,
                                    instance_type='ml.p2.xlarge',
                                    volume_size=50,
                                    max_run=360000,
                                    input_mode='File',
                                    output_path=s3_output_location,
                                    sagemaker_session=sess,
                                    hyperparameters=hyperparams)
```

```
train_data = sagemaker.inputs.TrainingInput(s3train, distribution='FullyReplicated',
                                             content_type='application/x-recordio',
                                             s3_data_type='S3Prefix')
validation_data = sagemaker.inputs.TrainingInput(s3validation, distribution='FullyReplicated',
                                                  content_type='application/x-recordio',
                                                  s3_data_type='S3Prefix')

data_channels = {'train': train_data, 'validation': validation_data}

ic.fit(inputs=data_channels, logs=True)
```

To use the model to incrementally train another model, create a new estimator object and use the model artifacts (`ic.model_data`, in this example) for the `model_uri` input argument:

```
# Given the base estimator, create a new one for incremental training
incr_ic = sagemaker.estimator.Estimator(training_image,
                                         role,
                                         instance_count=1,
                                         instance_type='ml.p2.xlarge',
                                         volume_size=50,
                                         max_run=360000,
                                         input_mode='File',
                                         output_path=s3_output_location,
                                         sagemaker_session=sess,
                                         hyperparameters=hyperparams,
                                         model_uri=ic.model_data) # This parameter will ingest
the previous job's model as a new channel
incr_ic.fit(inputs=data_channels, logs=True)
```

After the training job has completed, the newly trained model artifacts are stored under the S3 output path that you provided in `Output_path`. To deploy the model to get predictions, see [Step 5: Deploy the Model to Amazon EC2 \(/ex1-model-deployment.html\)](#).

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.