Building Machine Learning Models Using AWS SageMaker

Jorge Vásquez
SOFTWARE ENGINEER

@jorvasquez2301

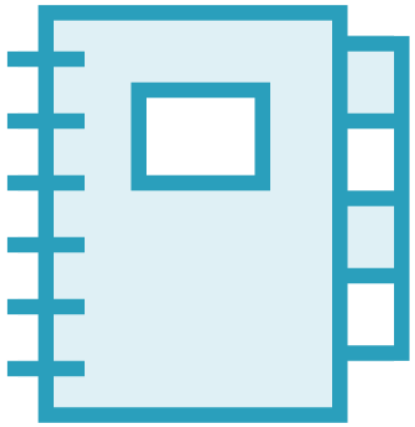
Overview

Building a model for breast cancer detection

- Creating a notebook instance
- Creating the model using:
  - A built-in algorithm
  - Tensorflow
  - Apache MXNet

Creating and Configuring a Notebook Instance for Creating Breast Cancer Detection Models

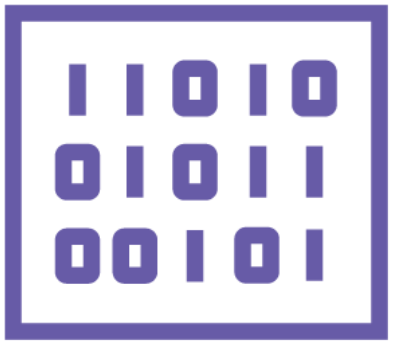
AWS SageMaker Notebook Instance



Fully managed ML compute instance running the Jupyter Notebook App, including related resources

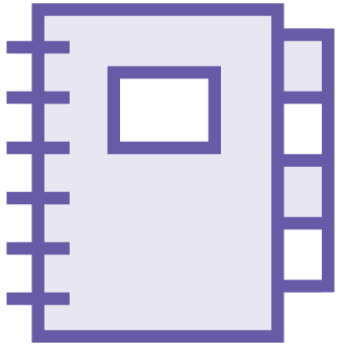
AWS SageMaker Notebook Instance

Anaconda packages



Tensorflow and Apache MXNet

Storage volume



Example Jupyter notebooks

AWS SageMaker Notebook Instance

Prepare and process data

Write code to train models

Deploy models to AWS SageMaker hosting

Test or validate your models

Demo

Creating an AWS SageMaker Notebook Instance:

- Configuring instance name and type
- Assigning an IAM role for allowing access to S3
- Configuring storage volume size

Building a Model in AWS SageMaker for Breast Cancer Detection Using a Built-in Algorithm

Image Classification Algorithm

Supervised learning algorithm

Takes an image as input and classifies it into one of multiple output categories

Uses a Convolutional Neural Network (ResNet)

Image Classification Algorithm

Full training mode

Network is initialized with random weights

Network is trained from scratch

Needs a lot of input images

Transfer learning mode

Network is initialized with pre-trained weights

Just the top fully connected layer is initialized with random weights

Needs a smaller number of input images

The whole network is fine tuned with user images

Image Classification Algorithm

Apache MXNet RecordIO (recommended)

Raw images (JPEG, PNG)

Image Classification Algorithm

Using RecordIO format for input

Pipe Mode

Your training job streams data directly from S3

Faster start times

Better throughput

Reduced storage volume usage

Using Raw Images for input

File Mode

Loads all your training data from S3 to the training instance volumes

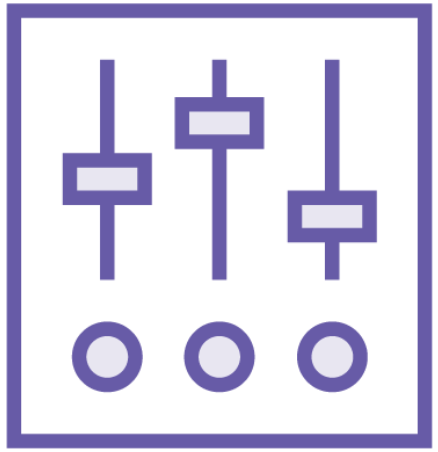
Slower start times

Lower throughput

Higher storage volume usage

Can work on Pipe Mode, if an augmented manifest file is provided

Image Classification Algorithm



The Image Classification Algorithm has several hyperparameters that can be adjusted for better performance

Hyperparameter

A hyperparameter is a parameter that is set before the learning process begins. These parameters are tunable and can directly affect how well a model trains (deepai.org)

Image Classification Required Hyperparameters

Number of classes

Number of training samples

Available APIs for Building Models Using Built-in Algorithms

Low-level AWS SDK for Python

High-level SageMaker Python library

Demo

Creating a Jupyter notebook

Obtaining the histopathology images

Exploring the images

Converting images to the RecordIO format and upload to S3

Demo

Configuring the Image Classification Algorithm using the low-level AWS SDK for Python

Demo

Configuring the Image Classification Algorithm using the high-level SageMaker Python library

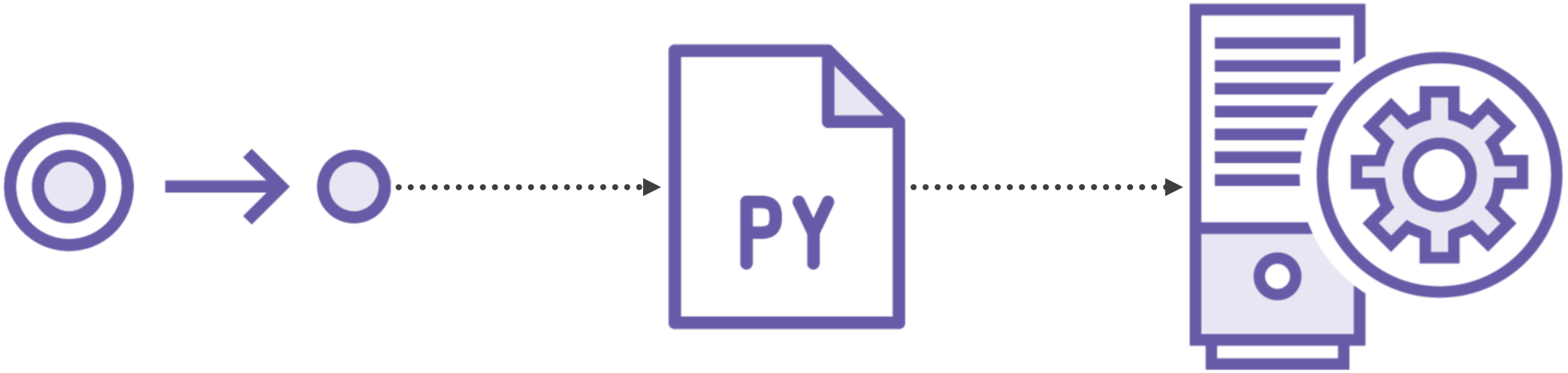
Building a Model in SageMaker for Breast Cancer Detection Using Tensorflow

Latest supported version of Tensorflow is 1.12.0

Using Tensorflow with the SageMaker Python SDK

It’s recommended to use the latest supported version of Tensorflow, because that's where AWS focuses most of its development efforts

Building Tensorflow Models in SageMaker



Transform input images to the TFRecord format

Prepare the training script

Build a model using the Tensorflow estimator

Training Tensorflow Models

Tensorflow Training Script

Since Tensorflow 1.11, Script mode is the training script format

Training Tensorflow Models

Tensorflow Training Script

Script mode supports training with a:

- Python script
- Python module
- Shell script

Training Tensorflow Models

Tensorflow Training Script

SageMaker gives your script access to useful environment variables

Available Environment Variables

SM_MODEL_DIR

SM_NUM_GPUS

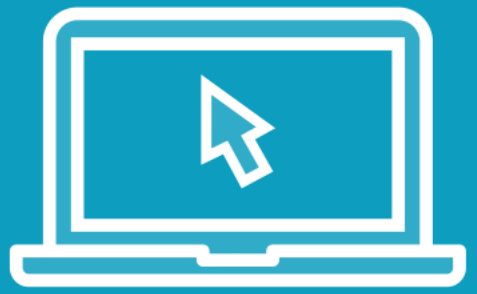
SM_OUTPUT_DATA_DIR

SM_CHANNEL_XXXX

Demo

Converting images to the TFRecord format and upload to S3

Demo



Preparing a training Python script

Configuring a Tensorflow Estimator using the high-level SageMaker Python library

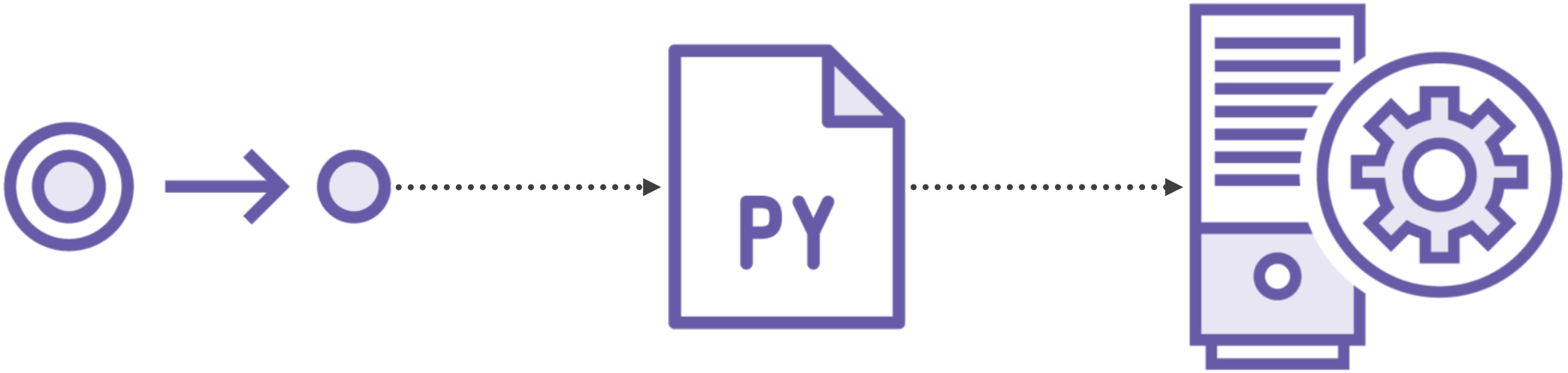
Building a Model in SageMaker for Breast Cancer Detection Using Apache MXNet

Latest supported version of MXNet is 1.3.0

Using Apache MXNet with the SageMaker Python SDK

It’s recommended to use the latest supported version of MXNet, because that’s where AWS focuses most of its development efforts

Building MXNet Models in SageMaker



Transform input images to the RecordIO format

Prepare the training script

Build a model using the MXNet estimator

Training MXNet Models

MXNet Training Script

The structure for training scripts changed with MXNet version 1.3, and that's the version we are going to use

Training MXNet Models

MXNet Training Script

SageMaker gives your script access to useful environment variables

Available Environment Variables

SM_MODEL_DIR

SM_NUM_GPUS

SM_OUTPUT_DATA_DIR

SM_CHANNEL_XXXX

Demo

Preparing the training script

Configuring a MXNet Estimator using the high-level SageMaker Python library

Summary

How to create a notebook instance

Exploring and preparing input data with Jupyter

Configuring the built-in Image Classification Algorithm

- Low-level AWS SDK for Python
- High-level SageMaker Python Library

Creating a custom script with Tensorflow and MXNet using the high-level SageMaker Python Library