Machine Learning For Medical Diagnosis

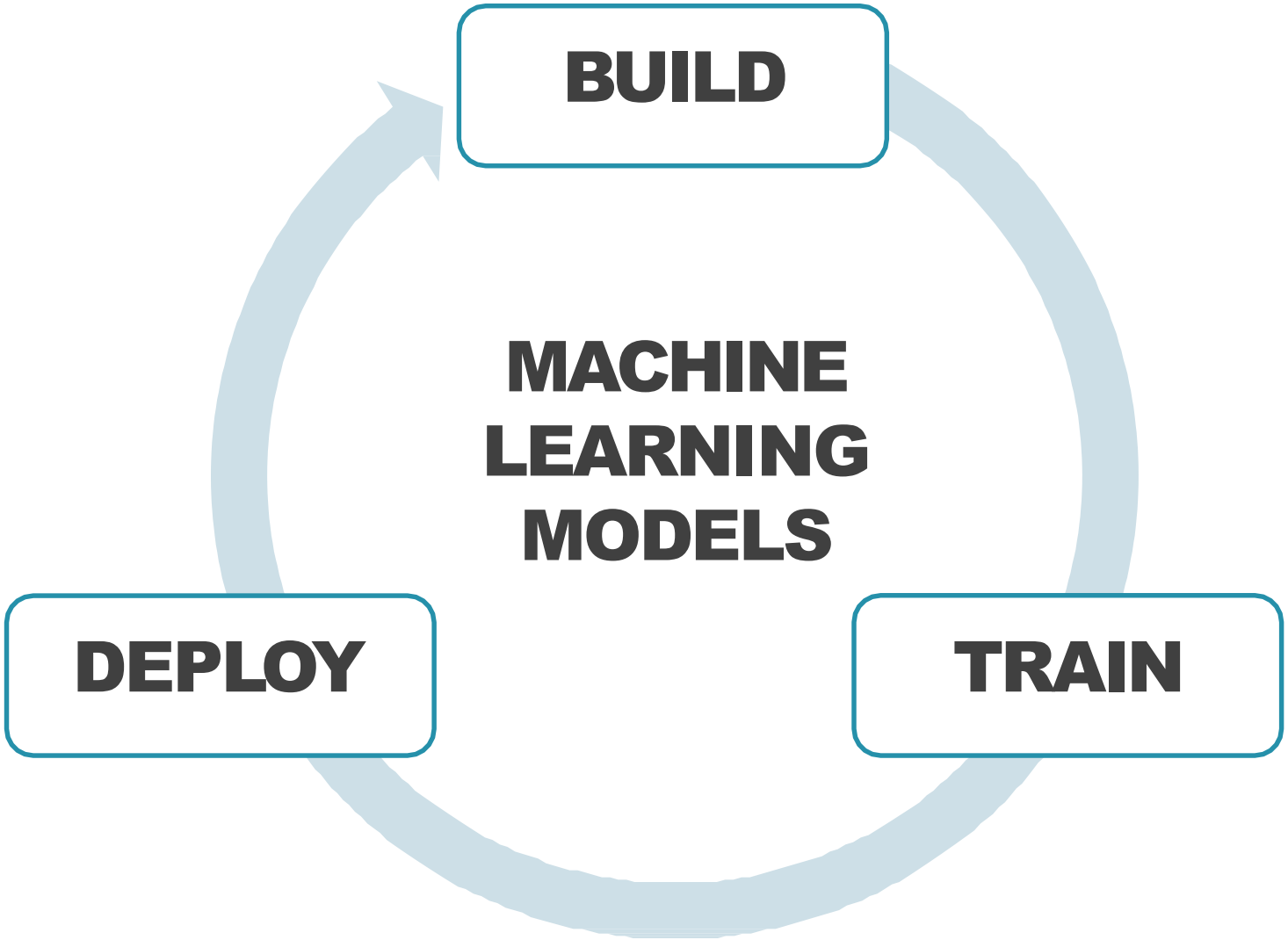
Nitin Pathania

Overview

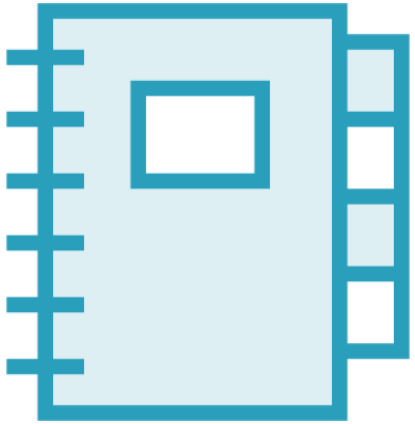
REST API for doing breast cancer detection from histopathology images.

REST API can be integrated later into a full web application that doctors could use for doing diagnostics

What Is AWS SageMaker?



AWS SageMaker Notebook Instance



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

AWS SageMaker Notebook Instance

Prepare and process data

Write code to train models

Deploy models to AWS SageMaker hosting

Test or validate your models

Building a Model in AWS SageMaker for Breast Cancer Detection Using a Built in Algorithm(Image Classification)

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

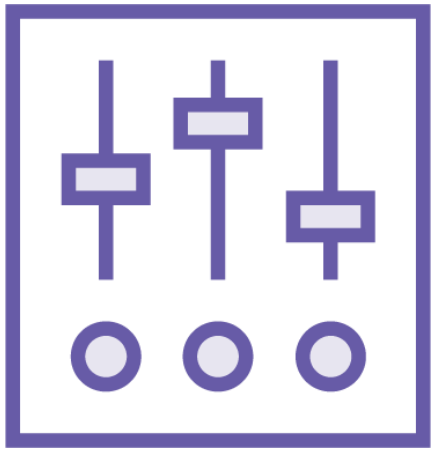
• Slower start times

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)

Number of classes

Number of training samples

Image Classification Required Hyperparameters

Low-level AWS SDK for Python

High-level SageMaker Python library

- Available APIs for Building Models Using Built-in Algorithms

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

- Configuring instance name and type

- Assigning an IAM role for allowing access to S3

- Configuring storage volume size

Preprocessing Notebook

- Creating a Jupyter notebook

- Obtaining the histopathology images

- Exploring the images

- Converting images to the RecordIO format and upload to S3

Image Classification Notebooks

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

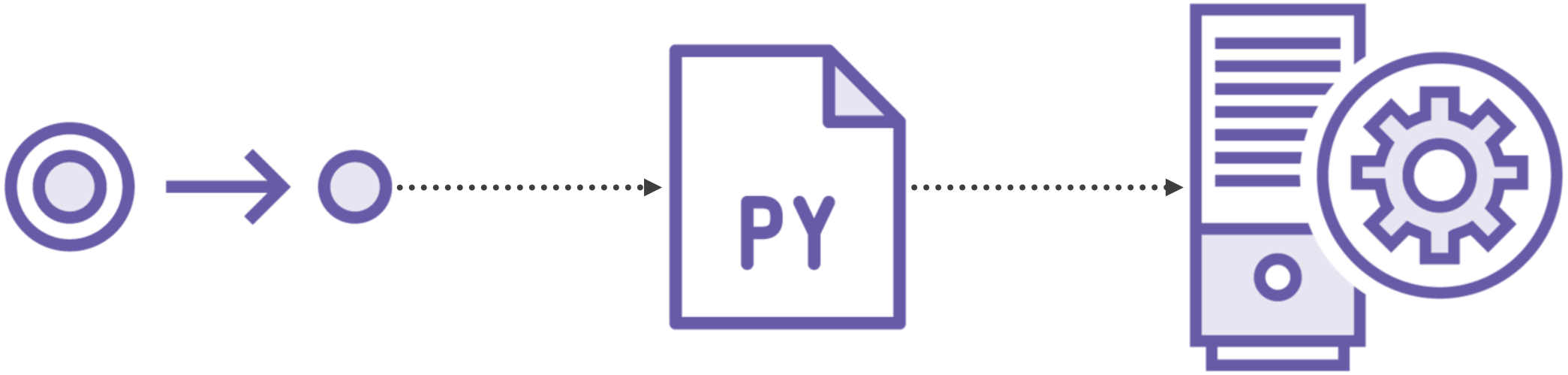
Configuring the Image Classification Algorithm using the high-level SageMaker Python library

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

Building Tensorflow Models in SageMaker



Transform input images to the TFRecord format

Prepare the training script

Build a model using the Tensorflow estimator

Available Environment Variables

SM_MODEL_DIR

SM_NUM_GPUS

SM_OUTPUT_DATA_DIR

SM_CHANNEL_XXXX

Tensorflow Notebook

Converting images to the TFRecord format and upload to S3

Preparing a training Python script

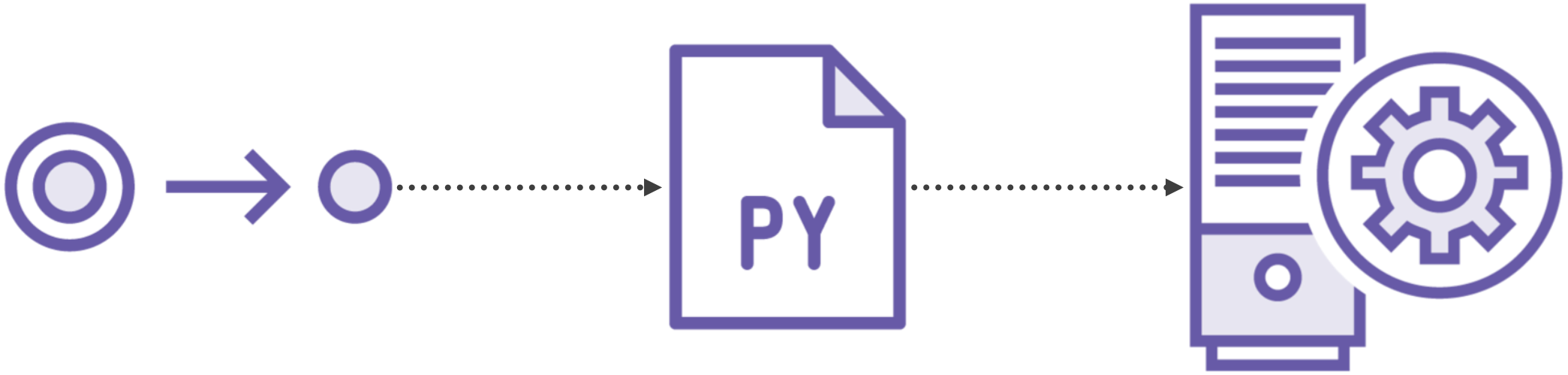
Configuring a Tensorflow Estimator using the high-level SageMaker Python library

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

Building MXNet Models in SageMaker



Transform input images to the RecordIO format

Prepare the training script

Build a model using the MXNet estimator

Available Environment Variables

SM_MODEL_DIR

SM_NUM_GPUS

SM_OUTPUT_DATA_DIR

SM_CHANNEL_XXXX

Apache MXNET Notebook

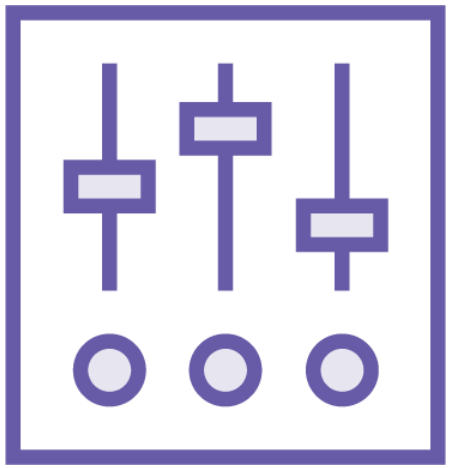
Preparing the training script

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

Training Model

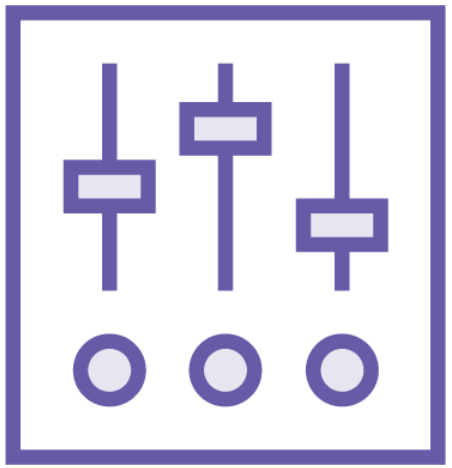
- Creating and monitoring a training job for the built-in Image Classification algorithm, using the low-level AWS SDK for Python.
- Creating and monitoring a training job for the built-in Image Classification algorithm, using the high-level SageMaker Python library.
- Creating and monitoring a training job for the custom Tensorflow algorithm, using the high-level SageMaker Python library.
- Creating and monitoring a training job for the custom MXNet algorithm, using the high-level SageMaker Python library

Automatic Hyperparameter Optimization



Automatic HPO Findsthe best version of a model by running many trainingjobs

Automatic Hyperparameter Optimization



It uses the algorithm and ranges of hyperparameters that you specify.

Chooses the hyperparameter values that result in a model that performs the best, as measured by a metric that you choose.

Tunable Image Classification Hyperparameters

mini_batch_size

learning_rate

optimizer

Tunable Image Classification Hyperparameters

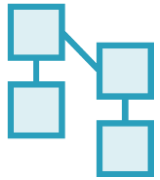
beta_1	beta_2	eps
gamma	momentum	weight_decay

Tuning Notebook

Creating and monitoring a tuning job for the built-in Image Classification algorithm, using the low-level AWS SDK for Python.

- Creating and monitoring a tuning job for the built-in Image Classification algorithm, using the high-level SageMaker Python library.
- Creating and monitoring a tuning job for the custom Tensorflow algorithm, using the high-level SageMaker Python library.
- Creating and monitoring a tuning job for the custom MXNet algorithm, using the high-level SageMaker Python library

Deploying to AWS SageMaker

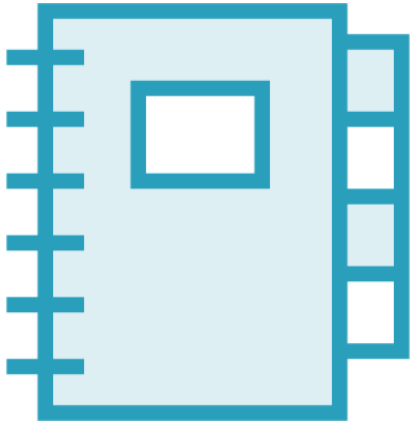


Create a Model

Create an Endpoint Configuration for an HTTPS Endpoint

Create an HTTPS Endpoint

Testing Deployed Models in AWS SageMaker



You can send sample requests and get inferences directly from Jupyter

Deployment Notebook

- Deploying and testing the trained model based on the built-in Image Classification algorithm, using the high-level SageMaker Python Library

- Deploying and testing the trained model based on the built-in Image Classification algorithm, using the low-level AWS SDK for Python

- Deploying and testing the trained model based on a custom MXNet algorithm, using the high-level SageMaker Python Library

- Deploying and testing the trained model based on a custom Tensorflow algorithm, using the high-level SageMaker Python Library

AWS API Gateway

With a few clicks, you can create REST APIs that act as a “front door” to external applications

AWS API Gateway

Create

Publish

Maintain

Monitor

Secure

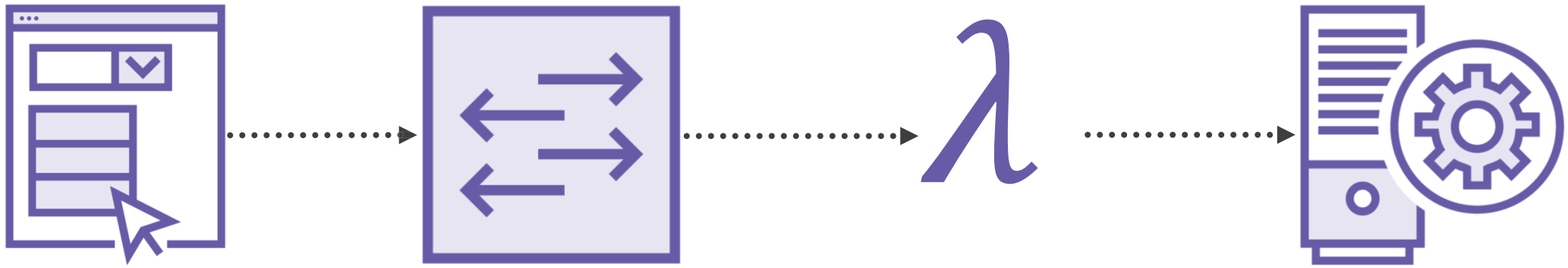
AWS Lambda

Run code without provisioning or managing servers

Pay only for the compute time you consume

Automatically triggered from other AWS services

Integrating AWS SageMaker with AWS API Gateway and AWS Lambda



Client application

AWS API Gateway

AWS Lambda Function

AWS SageMaker Endpoint

Integrating an AWS SageMaker Endpoint with AWS API Gateway and AWS Lambda:

- Creating aLambda function
- Creating an API Gateway
- Testing the API Gateway with Postman

Thank You