Apache MXNet on AWS

Developer Guide

Apache MXNet on AWS: Developer Guide

Table of Contents

What Is Apache MXNet?

Apache MXNet (MXNet) is an open source deep learning framework that allows you to define, train, and deploy deep neural networks on a wide array of platforms, from cloud infrastructure to mobile devices. It is highly scalable, which allows for fast model training, and it supports a flexible programming model and multiple languages.

The MXNet library is portable and lightweight. It scales seamlessly on multiple GPUs on multiple machines.

MXNet supports programming in various languages including Python, R, Scala, Julia, and Perl.

About This Guide

This guide focuses on setting up MXNet on AWS. It provides step-by-step instructions for launching an Amazon EC2 instance and setting up MXNet and its dependencies to configure a stable, secure, and high-performance execution environment for deep learning applications.

When working with large amounts of data, you might choose to run MXNet on a cluster of EC2 instances. This allows you to scale horizontally—to add as many EC2 instances as you need to scale the compute and memory resources.

AWS provides Deep Learning Amazon Machine Images (AMIs) and AWS CloudFormation templates optimized for both CPU and GPU EC2 instances. This guide explains how to set up MXNet on AWS using these AMIs and templates.

More Info

MXNet is an Apache open source project. For more information about MXNet, see the following open source documentation:

- Getting Started – Provides details for setting up MXNet on various platforms, such as macOS, Windows, and Linux. It also explains how to set up MXNet for use with different front-end languages, such as Python, Scala, R, Julia, and Perl.
- Tutorials – Provides step-by-step procedures for various deep learning tasks using MXNet.
- API Reference – For more information, see MXNet. At the top of this page, choose a language from the API menu.

For all other information, see MXNet.

Getting Started

Depending on your application needs, you can set up MXNet on a single Amazon EC2 instance or set up a cluster of EC2 instances for distributed deep learning.

In both cases, you launch EC2 instances using the Deep Learning Amazon Machine Image (AMI), provided by AWS Marketplace. The AMI comes preinstalled with MXNet, CUDA, cuDNN, Python 2, and Python 3 so that you can begin writing code using MXNet soon after launching the EC2 instance. The resulting EC2 instance provides a stable, secure, and high-performance execution environment for deep learning applications.

> **Note**
> The AMI also include other deep learning frameworks, such as TensorFlow, Caffe, and Keras. The AMI is available for Ubuntu, Amazon Linux and Windows. For more information, see Deep Learning AMI Developer Guide for AMI options.

When working with large amounts of data, you might choose to run MXNet on a cluster of EC2 instances to scale horizontally. This adds as many EC2 instances as required to scale up to the available compute and memory resources. To quickly set up a cluster, you use the predefined AWS CloudFormation template .

This guide provides the following step-by-step setup instructions.

Set Up Apache MXNet on an EC2 Instance (p. 3)

Set Up Apache MXNet on a Cluster of EC2 Instances (p. 10)

These exercises provide a complete end-to-end experience using MXNet. To set up MXNet, you need an AWS account. If you don't have one, use the following steps to sign up.

Sign up for an AWS account

1. Open https://aws.amazon.com/, and then choose **Create an AWS Account**.

   > **Note**
   > This might be unavailable in your browser if you previously signed into the AWS Management Console. In that case, choose **Sign in to a different account**, and then choose **Create a new AWS account**.

2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a PIN using the phone keypad.

Set Up an EC2 Instance for Deep Learning Using Apache MXNet

If you need to run MXNet on only one EC2 instance, use this procedure to set it up. In this procedure, you launch an EC2 instance using the Deep Learning Amazon Machine Image (AMI) for Ubuntu. The AMI installs MXNet, the deep learning framework of choice at AWS, and other components, such as Python 2 and Python 3, launch configuration tools, and many popular AWS libraries and tools. The Deep Learning AMI is provided on AWS Marketplace. After installing MXNet, you use it with the MNIST dataset—a collection of 60,000 images of handwritten digits—to train a model to recognize handwritten digits.

When working with large training data, you might choose to set up a cluster of EC2 instances to provide the required compute power and memory. To set up a cluster and train a model on multiple EC2 instances, see Set Up a Stack for Distributed Deep Learning Using Apache MXNet (p. 10).

> **Note**
> The AMI also includes other deep learning frameworks, such as TensorFlow, Caffe, and Keras. The AMI is available for Ubuntu, Amazon Linux and Windows. For more information, see Deep Learning AMI Developer Guide for AMI options.

Topics

Step 1: Launch an EC2 Instance

Launch an EC2 instance using the Ubuntu version of the Deep Learning AMI. The AMI includes everything that you need to set up an instance and test sample MXNet code using Python.

> **Note**
> Although the Deep Learning AMI is available for both Ubuntu and Amazon Linux, in this exercise, we use the Ubuntu version.

Launch the instance

1. In the Amazon EC2 console, launch an instance. For step-by-step instructions, see Launching an AWS Marketplace Instance in the *Amazon EC2 User Guide for Linux Instances*. As you follow the steps, use the following values:

   - On the **Choose an Amazon Machine Image (AMI)** page, choose the **AWS Marketplace** tab, and search for the **Deep Learning AMI Ubuntu Version** AMI.
   - On the **Choose an Instance Type** page, choose the **`c4.4xlarge`** instance type.
   - On the **Configure Instance Details** page, do the following:
     - From the **Network** drop-down, choose your VPC.
     - From the **Auto-assign Public IP** list, choose **Enable**
   - On the **Add Storage** page, choose the default storage size. The AMI uses about 38 GB of disk space.
   - On the **Add Tags** page, add one tag with the key **`Name`** and any value.

- On the **Configure Security Group** page, choose **Add Rule**, and then add the following custom Transmission Control Protocol (TCP) rule.

  Type : Custom **`TCP Rule`**

  Protocol: **`TCP`**

  Port Range: **`8888`**

  Source: **`Anywhere`** (**`0.0.0.0/0,::/0`**)

  Note
  In this exercise, you also set up a Jupyter notebook server on the EC2 instance. After you set up the environment, the server for the Jupyter notebook starts on port 8888.

2. Wait for the instance to be ready. You can verify the status of the instance in the Amazon EC2 console.

Next Step

Step 2: Connect to the EC2 Instance (p. 4)

Step 2: Connect to the EC2 Instance

Connect to the EC2 instance that you launched from a client (Windows, Mac, or Linux). For more information, see Connect to Your Linux Instance in the *Amazon EC2 User Guide for Linux Instances*.

Next Step

Step 3: Run an Apache MXNet Code Example (p. 4)

Step 3: Run an Apache MXNet Code Example

The Deep Learning AMI that you used to launch your EC2 instance also installs Python 2 and Python 3. To test your installation of MXNet, use Python 2 to write some simple MXNet code and run it on the EC2 instance. The code creates a two-dimensional array using the `NDArray` API, and prints it. For more information about the API, see NDArray API.

You can also create a Jupyter notebook, and write and run the MXNet code in Python. For more information, see Set up a Jupyter Notebook (p. 18).

Write code to test your installation

1. Start the Python terminal.

```
$ python
```

   In the following steps, >>> represents the Python prompt.

2. Import MXNet.

```
>>> import mxnet as mx;
```

3. Create a 5X5 matrix, an instance of the `NDArray`, with elements initialized to 0, and print the array.

```
>>> mx.ndarray.zeros((5,5)).asnumpy();
```

Verify that the result looks as follows:

```
array([[ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.]], dtype=float32)
```

4. Exit the Python terminal.

```
>>> exit();
```

Next Step

Step 4: Train a Model (p. 5)

Step 4: Train a Model

Now, train a simple model that recognizes handwritten digits using the MNIST dataset. First, create a Jupyter notebook, and then write Python code to train the model.

For more information about MNIST, see the MNIST documentation.

Topics

Step 4.1: Create a Jupyter Notebook

To write MXNet code for training a model, you use a Jupyter notebook.

Create the notebook

1. Set up the Jupyter notebook. For instructions, see Set up a Jupyter Notebook (p. 18). Follow the steps to configure both the server (the EC2 instance) and your client.
2. Connect your client to the Jupyter server. For more information, see Step 4: Test by Logging in to the Jupyter Server (p. 25).

   a. In a browser window, type the URL in the address bar.

   - For Windows clients, use the public DNS name of the EC2 instance followed by the port number, which is typically 8888.

     ```
     https://EC2-instance-publicDNS:port/
     ```

     For example,

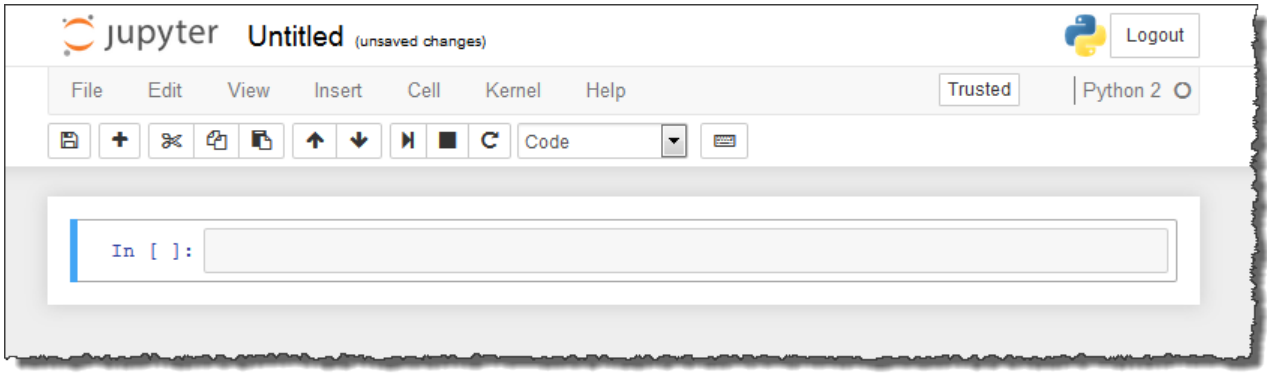
     ```
     https://ec2-111-22-33-44.compute-1.amazonaws.com:8888/
     ```

   - For macOS and Linux clients, use the localhost (127.0.0.1) followed by the port number.

     ```
     https://127.0.0.1:8157
     ```

b. If the connection is successful, the home page of the Jupyter notebook server appears. Type the password that you created when you configured the Jupyter server.

3. Create a Jupyter notebook, choosing the Python 2 option.



Now you are ready to write code.

Step 4.2: Train the Model

Use Python to incrementally add Apache MXNet code to the Jupyter notebook. In each step, run the code before going to the next step.

The code uses the MNIST dataset to train the model to recognize handwritten digits.

Train the model

1. Download the MNIST dataset and prepare training and validation data by copying and pasting the following code into the Jupyter notebook and running it.

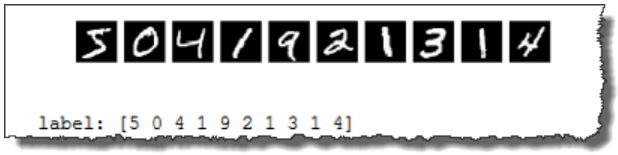
```
import numpy as np
import os
import urllib
import gzip
import struct

def download_data(url, force_download=True):
    '''
        Download the file from the given url and return the file name.
    '''
    fname = url.split("/")[-1]
    if force_download or not os.path.exists(fname):
        urllib.urlretrieve(url, fname)
    return fname

def read_data(label_url, image_url):
    '''
        Download labels (from the label_url) and images
        (from the image_url). Load them into Numpy arrays (label and images).
        Example: label[0] corresponds to label of the image at image[0].
    '''
    with gzip.open(download_data(label_url)) as flbl:
        magic, num = struct.unpack(">II", flbl.read(8))
        label = np.fromstring(flbl.read(), dtype=np.int8)
    with gzip.open(download_data(image_url), 'rb') as fimg:
        magic, num, rows, cols = struct.unpack(">IIII", fimg.read(16))
        image = np.fromstring(fimg.read(), dtype=np.uint8).reshape(len(label), rows,
 cols)
```

```
    return (label, image)

'''
    Download MNIST 10k dataset. This is a 28x28 hand-written digit
    images dataset with 60,000 samples.
    Download training data to train the model.
    Download validation data to test the model.
'''

path='http://yann.lecun.com/exdb/mnist/'
(train_lbl, train_img) = read_data(
    path+'train-labels-idx1-ubyte.gz', path+'train-images-idx3-ubyte.gz')
(val_lbl, val_img) = read_data(
    path+'t10k-labels-idx1-ubyte.gz', path+'t10k-images-idx3-ubyte.gz')
```

2. Copy and paste the following code into the notebook and run it. The code plots the first 10 images in the training dataset.

```
'''
    Plot 10 images in the training dataset.
'''
%matplotlib inline
import matplotlib.pyplot as plt
for i in range(10):
    plt.subplot(1,10,i+1)
    plt.imshow(train_img[i], cmap='Greys_r')
    plt.axis('off')
plt.show()
print('label: %s' % (train_lbl[0:10],))
```

You should see the following output:



3. Prepare the MXNet data iterators for training and validating data by copying and pasting the following code into the notebook and running it. Subsequent code uses these data iterators when training the model on the dataset.

```
'''
    Create MXNet data iterators to iterate through the training and
    validation dataset. These iterators are used during both the training and testing.
'''
import mxnet as mx

def to4d(img):
    '''
        Convert a batch of images into 4D matrix (batch_size, num_channels, width,
 height).
        In this example, MNIST dataset is a collection of 28X28 grey scale images.
        Grey scale images will have only one color channel.
    '''
    return img.reshape(img.shape[0], 1, 28, 28).astype(np.float32)/255

'''
    Prepare data iterators for training and validation.

    We use batches of 100 images for training and validation.
    We enable shuffling of data. This helps model to train faster.
'''
```

```
batch_size = 100
train_iter = mx.io.NDArrayIter(to4d(train_img), train_lbl, batch_size, shuffle=True)
val_iter = mx.io.NDArrayIter(to4d(val_img), val_lbl, batch_size)
```

4. Define the structure of the neural network for model training by copying and pasting the following code into the notebook and running it. The code uses a fully connected multilayer perceptron network (MLP).

```
'''
    Define the neural network structure for training the model.
    We use a fully connected multi layer perceptron (MLP) network.
'''

# Create a placeholder variable for the input data.
data = mx.sym.Variable('data')
# Flatten the data from 4-D shape (batch_size, num_channel, width, height)
# into 2-D (batch_size, num_channel*width*height).
data = mx.sym.Flatten(data=data)

# The first fully-connected layer with relu activation function.
fc1  = mx.sym.FullyConnected(data=data, name='fc1', num_hidden=128)
act1 = mx.sym.Activation(data=fc1, name='relu1', act_type="relu")

# The second fully-connected layer with relu activation function.
fc2  = mx.sym.FullyConnected(data=act1, name='fc2', num_hidden = 64)
act2 = mx.sym.Activation(data=fc2, name='relu2', act_type="relu")

# The third fully-connected layer, note that the hidden size should be 10,
# which is the number of unique digits (0-9).
fc3  = mx.sym.FullyConnected(data=act2, name='fc3', num_hidden=10)
# The softmax loss layer.
mlp  = mx.sym.SoftmaxOutput(data=fc3, name='softmax')

# We visualize the network structure with output size (the batch_size is ignored).
shape = {"data" : (batch_size, 1, 28, 28)}
mx.viz.plot_network(symbol=mlp, shape=shape)
```

5. Train the model by copying and pasting the following code into the notebook and then running it.

```
import logging
logging.getLogger().setLevel(logging.DEBUG)

model = mx.mod.Module(
    symbol = mlp        # network structure
)
model.fit(
    train_iter,         # training data
    eval_data=val_iter, # validation data
    batch_end_callback = mx.callback.Speedometer(batch_size, 200),    # output progress
 for each 200 data batches
    num_epoch = 10,     # number of data passes for training
    optimizer = 'sgd',
    optimizer_params=(('learning_rate', 0.1),)
)
```

Next Step

Step 5: Test the Model (p. 9)

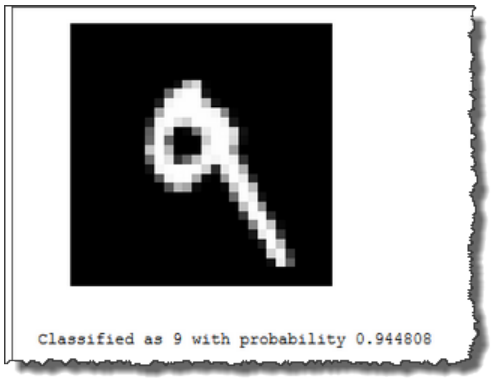
Step 5: Test the Model

The testing code uses the validation data from the model that you trained to perform predictions.

Copy and paste the following code into the notebook and run it.

```
'''
    Perform predictions on validation data with 10000 images.
'''

predictions = model.predict(eval_data=val_iter)

# Let us print an example prediction.

# Let us see what is the 8th image in our validation data is.
# And check if our model predicts it correctly.
plt.imshow(val_img[7], cmap='Greys_r')
plt.axis('off')
plt.show()
prob = predictions[7].asnumpy()
print("Classified as %d with probability %f" %(prob.argmax(), max(prob)))
```

The code prints the input image and the digit that the model predicts.



Next Step

Step 6: Clean Up (p. 9)

Step 6: Clean Up

When you no longer need the EC2 instance, terminate it to avoid incurring continuing charges. For instructions, see Terminate Your Instance in the *Amazon EC2 User Guide for Linux Instances*.

Set Up a Stack for Distributed Deep Learning Using Apache MXNet

When working with large amounts of data, running MXNet on a cluster of EC2 instances allows you to scale horizontally—to add as many EC2 instances as you need to provide the required compute power and memory. This topic explains how to set up a cluster of EC2 instances for distributed deep learning with MXNet on AWS. In addition to setting up the cluster using an AWS CloudFormation template and installing MXNet and its dependencies, you learn how to run distributed training on an image classification model using the CIFAR-10 dataset. For more information, see the CIFAR-10 dataset.

Topics

Step 1: Prepare

Make sure you have the following information:

- A range of IP addresses from which you want to access the cluster. You can provide an IP address (for example, 192.168.1.1) or a CIDR range (for example, 192.168.1.1/24). You can find the IP address of your client at the IP check URL .
- The key pair for the private key file. For more information about accessing your key pair, see Amazon EC2 Key Pairs in the *Amazon EC2 User Guide for Linux Instances*.

Next Step

Step 2: Create an AWS CloudFormation Stack (p. 10)

Step 2: Create an AWS CloudFormation Stack

To quickly create and configure the resources that you need for a cluster of EC2 instances, use the AWS CloudFormation Deep Learning template. The template creates an AWS CloudFormation *stack*, which is a group of related resources that you manage as a single unit. The stack includes a cluster of EC2 instances and an Amazon Elastic File System (Amazon EFS) file system that the instances can share, among other resources. For a complete list of the resources that the template creates, see Resources Created by the Deep Learning Template.

The Deep Learning template can use either the Ubuntu version or the Amazon Linux version of the AWS Deep Learning Amazon Machine Image (AMI). For this exercise, we use the Ubuntu version.

The AMI installs Apache MXNet and other components, such as Python 2, Python 3, launch configuration tools, and many popular AWS libraries and tools. For more information about these AMIs, see Deep Learning AMI Ubuntu Version and Deep Learning AMI Amazon Linux Version.

Create the stack

1. Download the template from the awslabs/deeplearning-cfn GitHub repository, and save it.
2. Sign in to the AWS Management Console and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
3. Create a stack as follows:

   - On the **Select Template** page, choose **Upload a template to Amazon S3**. Navigate to the folder where you saved the AWS CloudFormation template and choose the template file.
   - On the **Specify Details** page, provide the following parameter values:
     - For **ImageType**, choose **`Ubuntu`**.
     - For **InstanceType**, choose **`c4.4xlarge`**. This instance type provides the compute power and memory required for this exercise. In a production environment, you might choose a different instance type based on the requirements of your use case.
     - For **KeyName**, specify the EC2 key pair.
     - Choose an **SSHLocation**. This is the range of IP addresses from which you want to access the cluster. You can provide a CIDR range (for example, 192.168.1.1/24). You can find the IP address at http://checkip.amazonaws.com/.
     - For the **WorkerCount**, leave the default value of 1. This gives you two EC2 instances (one master and one worker). In your application scenario, you might choose a different worker count.
   - On the **Options** page, leave the defaults, and choose **Next**. Then on the **Review** page, choose **Create stack**.

     Note
     The stack is ready to use when the creation status changes to `CREATE_COMPLETE`.
4. After AWS CloudFormation creates the stack, find the public DNS name of the master node. You use the public DNS name to connect to the master node in Step 3.

   a. Navigate to the AWS CloudFormation console, and choose the stack.
   b. Expand the **Resources** section. In the **Logical ID** list, choose the **Physical ID** associated with the **Master Auto Scaling Group**. This opens the **Auto Scaling Groups** page in the Amazon Elastic Compute Cloud(Amazon EC2) console.
   c. Choose the **Instances** tab (in the console UI, you choose the **Instances** tab, not the instance link on the left pane). The tab shows the master instance.
   d. Choose the **Instance ID** to open the **Instances** page.
   e. Find the **Public DNS** name of the master node and record it.

Next Step

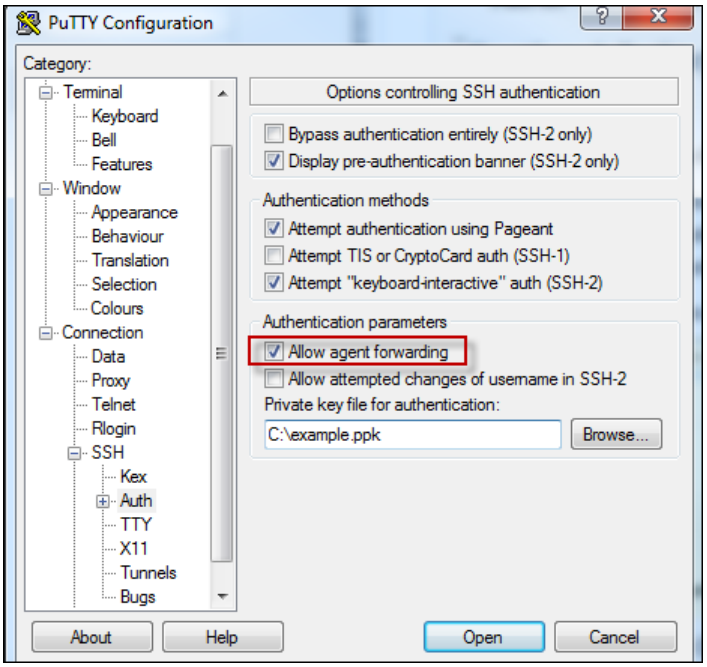
Step 3: Connect to the Master Node in the Stack (p. 11)

Step 3: Connect to the Master Node in the Stack

Connect to the master node in the cluster of EC2 instances launched by the AWS CloudFormation template.

For step-by-step instructions, see Connect to Your Linux Instance in the *Amazon EC2 User Guide for Linux Instances.* As you follow the steps, enable SSH agent forwarding when you connect to the master node. The master node uses SSH agent forwarding to securely connect with worker nodes to run distributed deep learning applications.

- On Windows client, in PuTTY choose the **Allow agent forwarding** authentication option.



- On Linux and macOS clients, use the `ssh` command to connect to the instance. Add the `-A` parameter when using this command. For example:

```
$ ssh-add -K /path/my-key-pair.pem
$ ssh -A -i /path/my-key-pair.pem ubuntu@public-DNS
```

Next Step

Step 4: Run Sample Apache MXNet Code (p. 12)

Step 4: Run Sample Apache MXNet Code

Test your setup by writing simple MXNet code in Python and running it on the master node in the cluster. This code sample creates an array using the `NDArray` API and prints the array. For more information about this API, see NDArray API.

Run test code

1. Start Python.

```
$ python
```

   In the following steps, >>> is the prompt for the Python terminal.

2. Import MXNet.

```
>>> import mxnet as mx;
```

3. Create a 5x5 matrix, an instance of the `NDArray`, with elements initialized to 0. Print the array.

```
>>> mx.ndarray.zeros((5,5)).asnumpy();
```

4. Verify the result:

```
array([[ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  0.]], dtype=float32)
```

5. Exit the Python terminal.

```
>>> exit()
```

Next Step

Step 5: Train a Sample Image Classification Model Using the Cluster (p. 13)

Step 5: Train a Sample Image Classification Model Using the Cluster

In this step, you train and build an image classification model to classify input images (for example, airplane, automobile, cat, dog, bird, etc.) using the CIFAR-10 dataset. This shows how to run Apache MXNet for distributed training. For more information about the dataset, see The CIFAR-10 dataset.

Download the code and dependencies from the awslabs/deeplearning-cfn GitHub repository.

Train the model

1. Clone the required GitHub repositories and update the submodules.

   Clone the `deeplearning-cfn` repository from AWS `Labs/deeplearning-cfn`. This repository includes MXNet, along with other dependencies, as a submodule. The MXNet submodule contains the `dmlc-core` submodule. This steps updates both of these submodules.

   ```
   $ git clone --recursive https://github.com/awslabs/deeplearning-cfn $EFS_MOUNT/
   deeplearning-cfn
   ```

   The MXNet submodule contains the Python code file, `train_cifar10.py`, which you will run to train an image classification model.

   > **Note**
   > The AWS CloudFormation template sets the `$EFS_MOUNT` environment variable when it creates the stack. The variable points to the directory where the Amazon EFS file system is mounted. The file system is shared across EC2 instances in the cluster.

2. Train the image classification model using the CIFAR-10 training dataset on the cluster of EC2 instances. First create a directory where the resulting model is saved.

   ```
   $ mkdir $EFS_MOUNT/cifar_model/

   $ cd $EFS_MOUNT/deeplearning-cfn/examples/mxnet/example/image-classification/

   $ ../../tools/launch.py -n $DEEPLEARNING_WORKERS_COUNT \
   -H $DEEPLEARNING_WORKERS_PATH python train_cifar10.py \
   --network resnet --num-layers 50 --kv-store dist_device_sync \
   ```

```
--model-prefix $EFS_MOUNT/cifar_model/cifar --num-epochs 10
```

This training on two c4.4xlarge CPU instances (that you specified when creating the cluster) runs for about 15 minutes for 10 epochs you specified. It achieves a training accuracy of 78%. You can increase the number of epochs for higher training accuracy, however it increases the amount of time it takes to train the model.

In the command:

- `train_cifar10.py` contains the MXNet Python code to train a model on the CIFAR-10 dataset. The dataset is a collection of 60,000 images grouped into 10 classes. After training, the model can infer the class of an input image.
- `launch.py` is a utility script that launches distributed training of the model. This utility uses the following environment variables to provide MXNet with information about the AWS CloudFormation stack. The AWS CloudFormation template sets these variables when you created the MXNet stack. MXNet uses this information to run distributed training.
  - `$DEEPLEARNING_WORKERS_COUNT` provides the count of the workers in this stack.
  - `$DEEPLEARNING_WORKERS_PATH` provides the path to a file containing a list of worker instances.
  - `$DEEPLEARNING_WORKER_GPU_COUNT` provides the count of GPUs on an instance.

  Note
  For this exercise, you launched a cluster of CPU machines. If you launch a cluster of GPU machines (for example p2.xlarge, g2.xlarge instance types), you use the following command to train a model:

  ```
  $ ../../tools/launch.py -n $DEEPLEARNING_WORKERS_COUNT -H
   $DEEPLEARNING_WORKERS_PATH \
  python train_cifar10.py --gpus $(seq -s , 0 1 $(($DEEPLEARNING_WORKER_GPU_COUNT
   - 1))) \
  --network resnet --num-layers 50 --kv-store dist_device_sync \
  --model-prefix $EFS_MOUNT/cifar_model/cifar --num-epochs 10
  ```

  This training takes only about 2 minutes on the two EC2 p2.8xlarge (GPU) instances. The model provides 79% accuracy. If you increase the number of epochs to 100, the training completes in about 25 minutes, and provides an accuracy of 92%.

3. Get the path where the model is saved. You need to test the model later.

```
$ echo $EFS_MOUNT/cifar_model/
```

If you find that you need to terminate the MXNet processes across the EC2 worker instances in the cluster, use the following command. Beware that it terminates all Python procesess, not just the MXNet processes.

```
$ while read -u 10 host; do ssh -o "StrictHostKeyChecking no" $host "pkill -f python" ; \
done 10<$DEEPLEARNING_WORKERS_PATH
```

The `$DEEPLEARNING_WORKERS_PATH` environment variable (which was set by the AWS CloudFormation template when you created the stack), provides a path to a file that contains a list of all worker instances in the stack.

Next Step

Step 6: Test the Image Classification Model (p. 15)

Step 6: Test the Image Classification Model

Test the model by first predicting image classes using the model for images in the CIFAR-10 dataset. Then verify the predictions with actual image classes. You use Jupyter notebook to write and run your code.

Topics

Step 6.1: Create a Jupyter Notebook

In the step, you use a Jupyter notebook to write Apache MXNet code for training a model in Python.

Create the notebook

1. Set up the Jupyter notebook. For instructions, see Set up a Jupyter Notebook. Follow the steps to configure both the server (the EC2 instance) and your client.
2. Connect your client to the Jupyter notebook server. For more information, see Step 4: Test by Logging in to the Jupyter Server (p. 25).

   a. In a browser window, type the URL in the address bar.

   - On Windows client, use the public DNS name of the EC2 instance followed by the port number, which is typically 8888.

     ```
     https://EC2-instance-publicDNS:port/
     ```

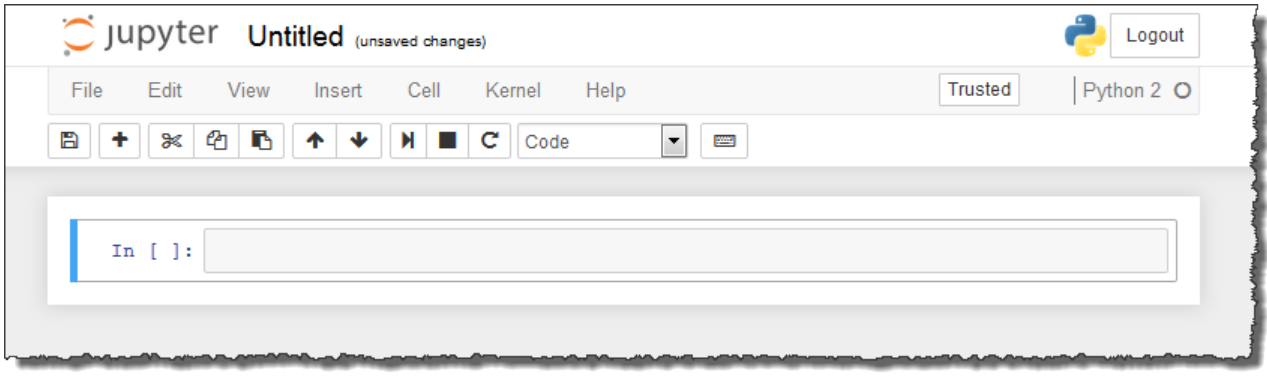
     For example:

     ```
     https://ec2-111-22-33-44.compute-1.amazonaws.com:8888/
     ```

   - On macOS and Linux clients, use the following URL:

     ```
     https://127.0.0.1:8157
     ```

   b. If the connection is successful, the homepage of the Jupyter notebook server appears. Type the password that you created when you configured the Jupyter server.
3. Create a Jupyter notebook, choosing the Python 2 option.



Now you are ready to write code.

Step 6.2: Test the Image Classification Model

Now test the image classification model that you trained in the preceding section, as follows:

- Using the model, perform image class predictions on all of the images provided in the CIFAR-10 validation dataset.
- Validate the image classification predictions. Compare the image class predicted by the model for one of the images with the image's actual image class.

In each step in the following procedure, copy and paste the code into your Jupyter notebook, and then run the code.

Test the model

1. Load the model we trained in the preceding section.

```
import mxnet as mx

symbol, arg_params, aux_params = mx.model.load_checkpoint('model-path/cifar', 300)
cifar_model = mx.mod.Module(symbol=symbol)
cifar_model.bind(for_training=False, data_shapes=[('data', (128,3,28,28))])
cifar_model.set_params(arg_params, aux_params)
```

   In `load_checkpoint()`, the parameter `300` is the number of epochs we used to train the model in the preceding section. This loads the model checkpoint at the end of the 300th epoch.

   Note
   If you trained your model using a different number of epochs, make sure to use that number.

2. To download the CIFAR validation dataset, run the following code.

```
import urllib
urllib.urlretrieve('http://data.mxnet.io/data/cifar10/cifar10_val.rec', '/tmp/
cifar10_val.rec')
```

3. Prepare a data iterator for the CIFAR-10 validation dataset (which has 10,000 images). You use this iterator in the next step to perform image classification predictions on all of these images.

   You also get the true label of the fourth image from the validation dataset. In the next step, you perform a prediction and then compare the predicted label against this true label.

```
rgb_mean = [123.68,116.779,103.939]
validation_data_iter = mx.io.ImageRecordIter(
        path_imgrec         = "/tmp/cifar10_val.rec",
        label_width         = 1,
        mean_r              = rgb_mean[0],
        mean_g              = rgb_mean[1],
        mean_b              = rgb_mean[2],
        data_name           = 'data',
        label_name          = 'softmax_label',
        batch_size          = 128,
        data_shape          = (3,28,28),
        rand_crop           = False,
        rand_mirror         = False)

# Get the actual label for the 4th image.
first_batch = validation_data_iter.first_batch
true_label = first_batch.label[0][4].asnumpy()[0]
```

4. Perform image classification prediction on all of the images in the validation dataset.

```
predictions = cifar_model.predict(validation_data_iter)
```

   In this step, you use the model to identify the image class of each of the 10,000 images in the dataset. The result is saved in `predictions`.
5. Now validate the image class predictions by comparing the actual image class with the image class predicted by the model.

```
# Print the actual label for the 4th image.
print("Actual label for the image - ", true_label)
# Print the label predicted by the model for the same image.
predicted_label = predictions[4].asnumpy().argmax()
print("Predicted label for the image - ", predicted_label)
```

Next Step

Step 7: Clean Up (p. 17)

Step 7: Clean Up

To avoid incurring unnecessary charges, when you are done with the stack, delete it and the resources that it created.

Delete the stack

1. Sign in to the AWS Management Console, and open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation/.
2. Choose the stack that you want to delete.
3. Choose **Actions**, and then choose **Delete Stack**.

Set up a Jupyter Notebook

Jupyter Notebook is a web application that allows you to manage notebook documents using a web browser.

To set up a Jupyter notebook, you:

- Configure the Jupyter notebook server on your EC2 instance.
- Configure your client so that you can connect to the Jupyter notebook server. We provide configuration instructions for Windows, macOS, and Linux clients.
- Test the setup by logging in to the Jupyter notebook server.

For more information about Jupyter notebooks, see Jupyter.

Topics

Step 1: Set Up Apache MXNet on AWS

Before you begin, set up MXNet on AWS. For instructions, see the following topics:

- Set Up an EC2 Instance for Deep Learning Using Apache MXNet (p. 3)
- Set Up a Stack for Distributed Deep Learning Using Apache MXNet (p. 10)

Both procedures use the AWS Deep Leaning AMI to set up MXNet. This AMI also installs Jupyter, so you only need to configure your Jupyter notebook.

Next Step

Step 2: Configure Jupyter Notebook on Your EC2 Instance (p. 18)

Step 2: Configure Jupyter Notebook on Your EC2 Instance

To configure the Jupyter notebook server on your EC2 instance, you create a configuration file. In the configuration file, you set some of the values to use for web authentication, including the SSL certificate file path, and a password.

Connect to the EC2 instance and then complete the following procedure. If you have set up a cluster of EC2 instances, connect to the master node.

Configure the Jupyter server

1. Create an SSL certificate.

```
$ cd
$ mkdir ssl
$ cd ssl
$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout "cert.key" -out
 "cert.pem" -batch
```

2. Create a password. You use this password to log in to the Jupyter notebook server from your client so you can access notebooks on the server.

   a. Open the iPython terminal.

   ```
   $ ipython
   ```

   At the iPython prompt, run the `passwd()`command to set the password.

   ```
   iPythonPrompt> from IPython.lib import passwd
   iPythonPrompt> passwd()
   ```

   You get the password hash (For example, `sha1:examplefc216:3a35a98ed...`).

   b. Record the password hash.

   c. Exit the iPython terminal.

   ```
   $ exit
   ```

3. Create a Jupyter configuration file.

```
$ jupyter notebook --generate-config
```

The command creates a configuration file (`jupyter_notebook_config.py`) in the `~/.jupyter` directory.

4. Update the configuration file to store your password and SSL certificate information.

   a. Open the .config file.

   ```
   $ vi ~/.jupyter/jupyter_notebook_config.py
   ```

   b. Paste the following text at the end of the file. You will need to provide your password hash.

   ```
   c = get_config()  # Get the config object.
   c.NotebookApp.certfile = u'/home/ubuntu/ssl/cert.pem' # path to the certificate we
    generated
   c.NotebookApp.keyfile = u'/home/ubuntu/ssl/cert.key' # path to the certificate key
    we generated
   c.IPKernelApp.pylab = 'inline'  # in-line figure when using Matplotlib
   c.NotebookApp.ip = '*'  # Serve notebooks locally.
   c.NotebookApp.open_browser = False  # Do not open a browser window by default when
    using notebooks.
   c.NotebookApp.password = 'sha1:fc216:3a35a98ed980b9...'
   ```

   This completes Jupyter server configuration.

Next Step

Step 3: Configure the Client to Connect to the Jupyter Server (p. 20)

Step 3: Configure the Client to Connect to the Jupyter Server

After configuring your client to connect to the Jupyter server, you can create and access notebooks on the server in your workspace and run your MXNet code on the server.

For configuration information, choose one of the following links.

Topics

Configure a Windows Client

To connect your Windows client to the Jupyter server, do the following:

- Configure proxy settings

  Configure your Internet browser to use an add-on, such as FoxyProxy, to manage your Socket Secure (SOCKS) proxy settings. The proxy management add-on enables you to limit the proxy settings to domains that match the form of the public DNS name of your EC2 instance. The add-on automatically handles turning the proxy on and off when you switch between viewing websites hosted on the EC2 instance and those on the Internet.
- Set up an SSH tunnel to your EC2 instance

  This is also known as port forwarding. If you create the tunnel using dynamic port forwarding, all traffic that is routed to a specified unused local port is forwarded to the Jupyter server on the EC2 instance. This creates a SOCKS proxy.

Topics

Step 1: Prepare

Make sure you have the following information, which you need to set up the SSH tunnel:

- The public DNS name of your EC2 instance. You can find the public DNS name in the EC2 console. If you set up a cluster of EC2 instances, see Set Up a Stack for Distributed Deep Learning Using Apache MXNet (p. 10)) for information about finding the public DNS name of the master node.
- The key pair for the private key file. For more information about accessing your key pair, see Amazon EC2 Key Pairs in the *Amazon EC2 User Guide for Linux Instances*.

Step 2: Configure Browser Proxy Settings

Configure the client browser to use FoxyProxy to manage SOCKS proxy settings. We provide procedures for both Firefox and Chrome.

Option 1: Configure proxy settings for Firefox

1. Download and install the Standard version of FoxyProxy from http://getfoxyproxy.org/downloads.html.
2. Using a text editor, create a file named `foxyproxy-settings.xml`. Save the following to the file:

```
<?xml version="1.0" encoding="UTF-8"?>
<foxyproxy>
   <proxies>
      <proxy name="ec2-socks-proxy" id="2322596116" notes="" fromSubscription="false"
 enabled="true" mode="manual" selectedTabIndex="2" lastresort="false"
 animatedIcons="true" includeInCycle="true" color="#0055E5" proxyDNS="true"
 noInternalIPs="false" autoconfMode="pac" clearCacheBeforeUse="false"
 disableCache="false" clearCookiesBeforeUse="false" rejectCookies="false">
         <matches>
            <match enabled="true" name="*ec2*.amazonaws.com*"
 pattern="*ec2*.amazonaws.com*" isRegEx="false" isBlackList="false" isMultiLine="false"
 caseSensitive="false" fromSubscription="false" />
            <match enabled="true" name="*ec2*.compute*" pattern="*ec2*.compute*"
 isRegEx="false" isBlackList="false" isMultiLine="false" caseSensitive="false"
 fromSubscription="false" />
            <match enabled="true" name="*.compute.internal*"
 pattern="*.compute.internal*" isRegEx="false" isBlackList="false" isMultiLine="false"
 caseSensitive="false" fromSubscription="false"/>
            <match enabled="true" name="*.ec2.internal* " pattern="*.ec2.internal*"
 isRegEx="false" isBlackList="false" isMultiLine="false" caseSensitive="false"
 fromSubscription="false"/>
   </matches>
         <manualconf host="localhost" port="8157" socksversion="5" isSocks="true"
 username="" password="" domain="" />
      </proxy>
   </proxies>
</foxyproxy>
```

   You use this settings file in the next step when configuring FoxyProxy. The settings do the following:

   - Port 8157 is the local port number used to establish the SSH tunnel with the Jupyter server. This must match the port number you used in PuTTY or the terminal.
   - The *ec2*.amazonaws.com* pattern matches the public DNS name of clusters in AWS US Regions.
   - The *ec2*.compute* pattern matches the public DNS name of clusters in all other AWS Regions.
3. Configure FoxyProxy.
   a. In Firefox, choose **Add-ons**.
   b. On the **add-ons** page, choose **Extensions**.
   c. Next to **FoxyProxy Standard**, choose **Options**.
   d. In the **FoxyProxy Standard** dialog box, do the following.
      i. Import the settings file.

         Choose **File**, choose **Import Settings**, and then open the `foxyproxy-settings.xml` file that you saved in the preceding step. If you are prompted to overwrite the settings, do so, and then restart Firefox.
      ii. Choose **Select Mode**, and then choose **`Use proxies based on their pre-defined patterns and priorities`**.
      iii. After Firefox restarts, in the **FoxyProxy Standard** dialog box, verify that **Select Mode** is set to **`Use proxies based on their pre-defined patterns and priorities`**.
   e. If the installation and configuration is successful, you see the FoxyProxy icon next to the URL address bar in the browser.

Option 2: Configure proxy settings for Chrome

1. Download and install the Standard version of FoxyProxy from http://getfoxyproxy.org/downloads.html.
2. Using a text editor, create a file named `foxyproxy-settings.xml`. Save the following to the file:

```
<?xml version="1.0" encoding="UTF-8"?>
<foxyproxy>
   <proxies>
      <proxy name="ec2-socks-proxy" id="2322596116" notes="" fromSubscription="false"
 enabled="true" mode="manual" selectedTabIndex="2" lastresort="false"
 animatedIcons="true" includeInCycle="true" color="#0055E5" proxyDNS="true"
 noInternalIPs="false" autoconfMode="pac" clearCacheBeforeUse="false"
 disableCache="false" clearCookiesBeforeUse="false" rejectCookies="false">
         <matches>
            <match enabled="true" name="*ec2*.amazonaws.com*"
 pattern="*ec2*.amazonaws.com*" isRegEx="false" isBlackList="false" isMultiLine="false"
 caseSensitive="false" fromSubscription="false" />
            <match enabled="true" name="*ec2*.compute*" pattern="*ec2*.compute*"
 isRegEx="false" isBlackList="false" isMultiLine="false" caseSensitive="false"
 fromSubscription="false" />
            <match enabled="true" name="*.compute.internal*"
 pattern="*.compute.internal*" isRegEx="false" isBlackList="false" isMultiLine="false"
 caseSensitive="false" fromSubscription="false"/>
            <match enabled="true" name="*.ec2.internal* " pattern="*.ec2.internal*"
 isRegEx="false" isBlackList="false" isMultiLine="false" caseSensitive="false"
 fromSubscription="false"/>
   </matches>
         <manualconf host="localhost" port="8157" socksversion="5" isSocks="true"
 username="" password="" domain="" />
      </proxy>
   </proxies>
</foxyproxy>
```

   You use these settings file in the next step when configuring FoxyProxy. The settings do the following:

   - Port 8157 is the local port number used to establish the SSH tunnel with the Jupyter server. This must match the port number you used in PuTTY or in the terminal.
   - The *ec2*.amazonaws.com* pattern matches the public DNS name of clusters in AWS US Regions.
   - The *ec2*.compute* pattern matches the public DNS name of clusters in all other AWS Regions.
3. Configure FoxyProxy.
   a. Choose **Customize and Control Google Chrome**, choose **Tools** (or **More Tools**), and then choose **Extensions**.
   b. Next to **FoxyProxy Standard**, choose **Options**.
   c. Choose **Extensions**, choose **Options**, and then do the following.
      i. Import the settings file.

         On the **Import/Export** page, choose **Choose File**, and then open the `foxyproxy-settings.xml` file that you saved in the preceding step. If you are prompted to overwrite the settings, do so.

ii. Choose **Proxy mode**, and then choose **`Use proxies based on their predefined patterns and priorities`**.

d. If the installation and configuration is successful, you see the FoxyProxy icon next to the URL address bar in the browser.

Step 3: Set Up an SSH Tunnel to Your EC2 Instance on the Client

Your client uses an SSH tunnel to your EC2 instance for dynamic port forwarding. For Windows, we use a PuTTY SSH client. First download and install PuTTY and PuTTYgen tools. For more information, see PuTTY download page.

> **Note**
> You need both the PuTTY and PuTTYgen tools. PuTTY does not natively support the key-pair private key file format (.pem) generated by Amazon Elastic Compute Cloud (Amazon EC2). You use PuTTYgen to convert your key file to the required PuTTY format (.ppk). Convert your key into this format (.ppk) before attempting to connect to the master node using PuTTY. For more information about converting your key, see Connecting to Your Linux Instance from Windows Using PuTTY in the *Amazon EC2 User Guide for Linux Instances*.

Set up an SSH tunnel using dynamic port forwarding on Windows

1. Start PuTTY by double-clicking `putty.exe`.
2. In the **Categories** section, choose **Sessions** and for **Host Name**, type **`ubuntu@`*`MasterPublicDNS`***. For example:

```
ubuntu@ec2-##-###-###-###.compute-1.amazonaws.com
```

3. In the **Categories** section, choose **Connection**. Choose **SSH**, and then choose **Auth**.
4. For **Private key file for authentication**, choose **Browse**, and then choose the `.ppk` file.
5. In the **Categories** section, choose **Connection-**. Choose **SSH**, and then choose **Tunnels**. Configure the tunnel.

   a. For **Source port**, type **`8157`** (an unused local port).
   b. Choose **Dynamic** and **Auto**.
   c. Choose **Add**, and then choose **Open**.

   This opens the tunnel.
6. Create a directory on the EC2 instance for storing Jupyter notebooks. This is your Jupyter workspace.

```
$ mkdir ~/mynotebooks
$ cd ~/mynotebooks
```

7. Start the Jupyter notebook server.

```
$ jupyter notebook
```

   By default, the server runs on port 8888. If the port is not available, it uses the next available port. The Jupyter terminal shows the port on which the server is listening.

Next Step

Step 4: Test by Logging in to the Jupyter Server (p. 25)

Configure a Linux Client

Configure the Linux client

1. Open a terminal.
2. Run the following command to forward all requests on local port 8157 to port 8888 on your remote EC2 instance. Update the command by replacing *ec2-###-##-##-###.compute-1.amazonaws.com* with the public DNS name of your EC2 instance.

```
$ ssh -i ~/mykeypair.pem -L 8157:127.0.0.1:8888 ubuntu@ec2-###-##-##-
###.compute-1.amazonaws.com
```

   This command opens a tunnel between your client and the remote EC2 instance that is running Jupyter server. After running the command, you can access the Jupyter notebook server at **https://127.0.0.1:8157**.
3. Create a directory for storing Jupyter notebooks. This is your Jupyter workspace.

```
$ mkdir ~/mynotebooks
$ cd ~/mynotebooks
```

4. Start the Jupyter notebook server.

```
$ jupyter notebook
```

   By default, the server runs on port 8888. If the port is not available, the server uses the next available port. The Jupyter terminal shows the port on which the server is listening.

Next Step

Step 4: Test by Logging in to the Jupyter Server (p. 25)

Configure a macOS Client

Configure the macOS client

1. Open a terminal.
2. Run the following command to forward all requests on local port 8157 to port 8888 on your remote EC2 instance. Update the command by replacing *ec2-###-##-##-###.compute-1.amazonaws.com* with the public DNS name of your EC2 instance.

```
$ ssh -i ~/mykeypair.pem -L 8157:127.0.0.1:8888 ubuntu@ec2-###-##-##-
###.compute-1.amazonaws.com
```

   This command opens a tunnel between your client and the remote EC2 instance that is running Jupyter server. After running the command, you can access the Jupyter notebook server at **https://127.0.0.1:8157**.
3. Create a directory for storing Jupyter notebooks. This is your Jupyter workspace.

```
$ mkdir ~/mynotebooks
```

```
$ cd ~/mynotebooks
```

4. Start the Jupyter notebook server.

```
$ jupyter notebook
```

By default, the server runs on port 8888. If that port is not available, Jupyter uses the next available port. The Jupyter terminal shows the port on which the server is listening.

Next Step

Step 4: Test by Logging in to the Jupyter Server (p. 25)

Step 4: Test by Logging in to the Jupyter Server

Now you are ready to log in to the Jupyter notebook server.

Test the server connection

1. In the address bar of your browser, type the following URL.

   - For windows client, using the public DNS name of the EC2 instance and the Jupyter port number. The Jupyter port is typically 8888.

   ```
   https://EC2-instance-public-DNS:port/
   ```

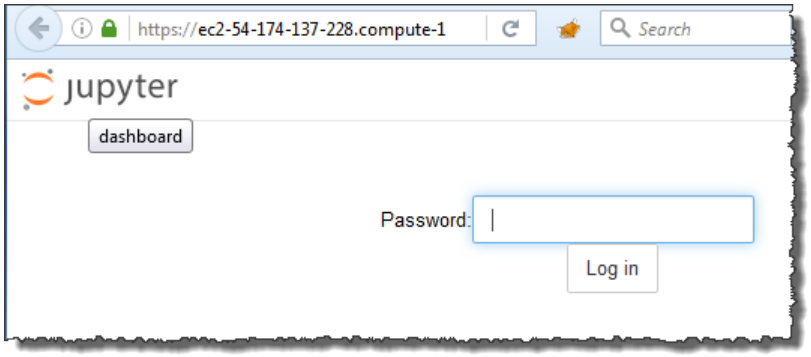
   For example:

   ```
   https://ec2-111-22-33-11.compute-1.amazonaws.com:8888/
   ```

   - For macOS and Linux clients, type the following URL:

   ```
   https://127.0.0.1:8157
   ```

2. If the connection is successful, you see the Jupyter notebook server home page. Type the password that you created when you configured the Jupyter server.



Now you have access to the Jupyter notebook server that is running on the remote EC2 instance. You can create new notebooks and write Apache MXNet code in Python.

Appendix

The appendix provides the following information:

Topics

Deep Learning AMI: Release Notes

Note: The release notes and other developer resources for the Deep Learning AMI have been moved to the new Developer Guide for Deep Learning AMI

The following sections provide release notes for both the CUDA 9 and the CUDA 8 versions of the Amazon Linux and Ubuntu Deep Learning AMI. The release notes list the contents of the AMI.

CUDA 9

CUDA 8

Deep Learning AMI: CUDA 9 (Amazon Linux Versions)

Deep Learning CUDA 9 AMI Amazon Linux Version: 1.0 (p. 35)

	Name	Version
Deep learning frameworks	Apache MXNet	0.12.0 Release Candidate
	Caffe2	0.8.1
	TensorFlow	master*
NVIDIA tools	CUDA Toolkit	9.0
	cuDNN	7.0
	NCCL	2.0.5
	NVIDIA Driver	384.81

*Please read the detailed release notes for Deep Learning CUDA 9 AMI Amazon Linux Version: 1.0 (p. 35) for more information on release tags/versions used for this release. Also refer to the AWS AI blog post to get started with the new CUDA 9 AMIs.

Deep Learning AMI: CUDA 9 (Ubuntu Versions)

Deep Learning CUDA 9 AMI Ubuntu Version: 1.0 (p. 37)

	Name	Version
Deep learning frameworks	Apache MXNet	0.12 Release Candidate
	Caffe2	0.8.1
	TensorFlow	master*
NVIDIA tools	CUDA Toolkit	9.0
	cuDNN	7.0
	NCCL	2.0.5
	NVIDIA Driver	384.81

*Please read the detailed release notes for Deep Learning CUDA 9 AMI Ubuntu Version: 1.0 (p. 37) for more information on release tags/versions used for this release. Also refer to the AWS AI blog post to get started with the new CUDA 9 AMIs.

Deep Learning AMI: CUDA 8 (Amazon Linux Versions)

Deep Learning AMI Amazon Linux Version: 3.3_Oct2017 (p. 40)

	Name	Version
Deep learning frameworks	Apache MXNet	0.11.0
	TensorFlow	1.3.0
	Caffe2	0.8.0*
	Caffe	1.0
	PyTorch	0.2.0
	Keras2	2.0.8**
	Keras	1.2.2 (DMLC fork) for Apache MXNet
	Theano	0.9.0
	CNTK	2.0
	Torch	master
NVIDIA tools***	CUDA Toolkit	8.0
	cuDNN	5.1
	NVIDIA Driver	375.66

*Caffe2 0.8.0 does not support g2.xxx instances. Please refer to release notes for Deep Learning AMI Amazon Linux Version: 3.1_Sep2017 (p. 44).

**You can swap between Keras 2.0.8 and Keras 1.2.2 using Conda virtual environments. See this blog post for detailed instructions.

***With NVidia CUDA 8 and NVidia Driver 375.66, this AMI is not compatible with latest EC2 P3 instance type. Use Deep Learning AMI CUDA 9 version for EC2 P3 instances.

Deep Learning AMI Amazon Linux Version: 3.1_Sep2017 (p. 44)

	Name	Version
Deep learning frameworks	Apache MXNet	0.11.0
	TensorFlow	1.3.0
	Theano	rel-0.9.0
	Keras	1.2.2 (with MXNet support)
	Caffe	1.0
	Caffe2	0.8.0*
	CNTK	2.0
	Torch	master
NVIDIA tools	CUDA Toolkit	8.0
	cuDNN	5.1
	NVIDIA Driver	375.66

*Caffe2 0.8.0 does not support g2.xxx instances. You can get Caffe2 0.7.0 on the following AMI ids using AMI version 2.3_Jun2017:

- us-east-1: ami-4b44745d
- us-east-2: ami-305d7c55
- us-west-2: ami-296e7850
- eu-west-1: ami-d36386aa
- ap-southeast-2: ami-52332031
- ap-northeast-1: ami-b44050d3
- ap-northeast-2: ami-1523fc7b

Deep Learning AMI Amazon Linux Version: 2.3_June2017 (p. 47)

	Name	Version
Deep learning frameworks	Apache MXNet	0.10.0
	TensorFlow	1.1.0
	Theano	rel-0.8.2
	Keras	1.2.2
	Caffe	rc5

	Name	Version
	Caffe2	0.7.0
	CNTK	2.0.rc1
	Torch	master
NVIDIA tools	CUDA Toolkit	7.5
	cuDNN	5.1
	NVIDIA Driver	367.57

Deep Learning AMI Amazon Linux Version: 2.2_June2017 (p. 51)

	Name	Version
Deep learning frameworks	Apache MXNet	0.10.0
	TensorFlow	1.1.0
	Theano	rel-0.8.2
	Keras	1.2.2
	Caffe	rc5
	Caffe2	0.7.0
	CNTK	2.0.rc1
	Torch	master
NVIDIA tools	CUDA Toolkit	7.5
	cuDNN	5.1
	NVIDIA Driver	367.57

Deep Learning AMI Amazon Linux Version: 2.1_Apr2017 (p. 55)

	Name	Version
Deep learning frameworks	Apache MXNet	0.9.3
	TensorFlow	1.0.1
	Theano	rel-0.8.2
	Keras	1.2.2
	Caffe	rc5
	Caffe2	0.7.0
	CNTK	2.0.rc1
	Torch	master

	Name	Version
NVIDIA tools	CUDA Toolkit	7.5
	cuDNN	5.1
	NVIDIA Driver	367.57

Deep Learning AMI Amazon Linux Version: 2.0 (p. 59)

	Name	Version
Deep learning frameworks	Apache MXNet	0.9.3
	TensorFlow	1.0.0
	Theano	rel-0.8.2
	Keras	1.2.2
	Caffe	rc5
	CNTK	2.0beta12.0
	Torch	master
NVIDIA tools	CUDA Toolkit	7.5
	cuDNN	5.1
	NVIDIA Driver	367.57

Deep Learning AMI Amazon Linux Version: 1.5 (p. 62)

	Name	Version
Deep learning frameworks	Apache MXNet	0.7.0
	TensorFlow	0.10.0
	Theano	rel-0.8.2
	Keras	1.0.8
	Caffe	rc5
	CNTK	2.0beta4.0
	Torch	master
NVIDIA tools	CUDA Toolkit	7.5
	cuDNN	5.1
	NVIDIA Driver	352.99

Deep Learning AMI: CUDA 8 (Ubuntu Versions)

Deep Learning AMI Ubuntu Version: 2.4_Oct2017 (p. 65)

	Name	Version
Deep learning frameworks	Apache MXNet	0.11.0
	TensorFlow	1.3.0
	Caffe2	0.8.0*
	Caffe	1.0
	PyTorch	0.2.0
	Keras2	2.0.8**
	Keras	1.2.2 (DMLC fork) for Apache MXNet
	Theano	0.9.0
	CNTK	2.0
	Torch	master
NVIDIA tools***	CUDA Toolkit	8.0
	cuDNN	5.1
	NVIDIA Driver	375.66

*Caffe2 0.8.0 does not support g2.xxx instances. Please refer to release notes for Deep Learning AMI Ubuntu Version: 2.3_Sep2017 (p. 69).

**You can swap between Keras 2.0.8 and Keras 1.2.2 using Conda virtual environments. Follow this blog post for detailed instructions.

With NVidia CUDA 8 and NVidia Driver 375.66, this AMI is not compatible with latest EC2 P3 instance type. Use Deep Learning AMI CUDA 9 version for EC2 P3 instances.

Deep Learning AMI Ubuntu Version: 2.3_Sep2017 (p. 69)

	Name	Version
Deep learning frameworks	Apache MXNet	0.11.0
	TensorFlow	1.3.0
	Theano	rel-0.9.0
	Keras	1.2.2 (with MXNet support)
	Caffe	1.0
	Caffe2	0.8.0*
	CNTK	2.0

	Name	Version
	Torch	master
NVIDIA tools	CUDA Toolkit	8.0
	cuDNN	5.1
	NVIDIA Driver	375.66

*Caffe2 0.8.0 does not support g2.xxx instances. You can get Caffe2 0.7.0 on the following AMI ids using AMI version 2.2_Aug2017:

- us-east-1: ami-2edccb38
- us-east-2: ami-2797b642
- us-west-2: ami-7fd7c906
- eu-west-1: ami-19896660
- ap-southeast-2: ami-b32b37d0
- ap-northeast-1: ami-55816633
- ap-northeast-2: ami-585a8436

Deep Learning AMI Ubuntu Version: 2.2_August2017 (p. 73)

	Name	Version
Deep learning frameworks	Apache MXNet	0.10.0
	TensorFlow	1.2.0
	Theano	rel-0.9.0
	Keras	1.2.2
	Caffe	1.0
	Caffe2	0.7.0
	CNTK	2.0
	Torch	master
NVIDIA tools	CUDA Toolkit	8.0
	cuDNN	5.1
	NVIDIA Driver	375.66

Deep Learning AMI Ubuntu Version: 1.5_June2017 (p. 76)

	Name	Version
Deep learning frameworks	Apache MXNet	0.10.0
	TensorFlow	1.1.0
	Theano	rel-0.8.2

	Name	Version
	Keras	1.2.2
	Caffe	rc5
	Caffe2	0.7.0
	CNTK	2.0.rc1
	Torch	master
NVIDIA tools	CUDA Toolkit	7.5
	cuDNN	5.1
	NVIDIA Driver	367.57

Deep Learning AMI Ubuntu Version: 1.4_June2017 (p. 80)

	Name	Version
Deep learning frameworks	Apache MXNet	0.10.0
	TensorFlow	1.1.0
	Theano	rel-0.8.2
	Keras	1.2.2
	Caffe	rc5
	Caffe2	0.7.0
	CNTK	2.0.rc1
	Torch	master
NVIDIA tools	CUDA Toolkit	7.5
	cuDNN	5.1
	NVIDIA Driver	367.57

Deep Learning AMI Ubuntu Version: 1.3_Apr2017 (p. 84)

	Name	Version
Deep learning frameworks	Apache MXNet	0.9.3
	TensorFlow	1.0.1
	Theano	rel-0.8.2
	Keras	1.2.2
	Caffe	rc5
	Caffe2	0.6.0

	Name	Version
	CNTK	2.0.rc1
	Torch	master
NVIDIA tools	CUDA Toolkit	7.5
	cuDNN	5.1
	NVIDIA Driver	367.57

Deep Learning AMI Ubuntu Version: 1.2 (p. 88)

	Name	Version
Deep learning frameworks	Apache MXNet	0.9.3
	TensorFlow	1.0.0
	Theano	rel-0.8.2
	Keras	1.2.0
	Caffe	rc5
	CNTK	2.0beta12.0
	Torch	master
NVIDIA tools	CUDA Toolkit	7.5
	cuDNN	5.1
	NVIDIA Driver	367.57

Deep Learning AMI Ubuntu Version: 1.1 (p. 91)

	Name	Version
Deep learning frameworks	Apache MXNet	0.9.3
	TensorFlow	0.12.1
	Theano	rel-0.8.2
	Keras	1.2.0
	Caffe	rc3
	CNTK	2.0beta7.0
	Torch	master
NVIDIA tools	CUDA Toolkit	7.5
	cuDNN	5.1
	NVIDIA Driver	367.57

Deep Learning AMI Ubuntu Version: 1.0 (p. 94)

	Name	Version
Deep learning frameworks	Apache MXNet	0.9.3
	TensorFlow	0.12.1
	Theano	rel-0.8.2
	Keras	1.2.0
	Caffe	rc3
	CNTK	2.0beta7.0
	Torch	master
NVIDIA tools	CUDA Toolkit	7.5
	cuDNN	5.1
	NVIDIA Driver	367.57

Deep Learning CUDA 9 AMI Amazon Linux Version: 1.0

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with CUDA9 and MXNet and also contain the Anaconda Platform(Python2 and Python3).

Highlights of the Release

1. Used Amazon Linux 2017.09 (ami-8c1be5f6) as the base AMI
2. CUDA 9
3. CuDNN 7
4. NCCL 2.0
5. CUDA 9 support
6. MXNet with CUDA9 Support

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.12.0 Release Candidate tag
  - Justification: Stable and well tested
  - Source_Directories:

    - /home/ec2-user/src/mxnet
- Caffe2: Caffe2 is a cross-platform framework made with expression, speed, and modularity in mind.
  - branch/tag used: v0.8.1 tag
  - Justification: Stable and well tested
  - Note: Available for Python2.7 only
  - Source_Directories:
    - For Python2.7+ - /home/ec2-user/src/caffe2
    - For Anaconda Python2.7+ - /home/ec2-user/src/caffe2_anaconda2
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
  - branch/tag used : Master tag
  - Justification : Stable and well tested
  - Source_Directories :
    - For Python2.7+ - /home/ec2-user/src/tensorflow
    - For Python3+ - /home/ec2-user/src/tensorflow3
    - For Anaconda Python2.7+ - /home/ec2-user/src/tensorflow_anaconda
    - For Anaconda Python3+ - /home/ec2-user/src/tensorflow_anaconda3

Python 2.7 and Python 3.5 Support

Python 2.7 and Python 3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. MXNet
2. Caffe2
3. Tensorflow

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support.

GPU Drivers Installed

- CuDNN 7
- Nvidia 384.81
- CUDA 9.0

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch P2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks.

The scripts are available in the /home/ec2-user/src/bin directory.

The following scripts test the various frameworks:

/home/ec2-user/src/bin/testMXNet : tests MXNet

/home/ec2-user/src/bin/testTensorFlow : tests TensorFlow

/home/ec2-user/src/bin/testCaffe2 : tests Caffe2

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Caffe2: Based on this example inside the Caffe2 repository. Validation accuracy threshold is 90%.

Amazon Linux AMI

Amazon Linux based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

References

MXNet

Test Environments

- Built on p2.16xlarge.
- Also tested on p2.xlarge, c4.4xlarge.

Deep Learning CUDA 9 AMI Ubuntu Version: 1.0

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with CUDA9 and MXNet and also contain the Anaconda Platform (Python2 and Python3).

Highlights of the Release

1. Used Ubuntu 16.04 (ami-d15a75c7) as the base AMI
2. CUDA 9
3. CuDNN 7
4. NCCL 2.0

5. MXNet with CUDA9 Support

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.12.0 Release Candidate tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe2: Caffe2 is a cross-platform framework made with expression, speed, and modularity in mind.
  - branch/tag used: v0.8.1 tag
  - Justification: Stable and well tested
  - Note: Available for Python2.7 only
  - Source_Directories:
    - For Python2.7+ - /home/ec2-user/src/caffe2
    - For Anaconda Python2.7+ - /home/ec2-user/src/caffe2_anaconda2
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
  - branch/tag used : Master tag
  - Justification : Stable and well tested
  - Source_Directories :
    - For Python2.7+ - /home/ubuntu/src/caffe
    - For Python3+ - /home/ubuntu/src/caffe3
    - For Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - For Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - For CPU_ONLY : /home/ubuntu/src/caffe_cpu

Python 2.7 and Python 3.5 Support

Python 2.7 and Python 3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. MXNet
2. Caffe2
3. Tensorflow

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support.

GPU Drivers Installed

- CuDNN 7

- Nvidia 384.81
- CUDA 9.0

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch P2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks.

The scripts are available in the /home/ubuntu/src/bin directory.

The following scripts test the various frameworks:

/home/ubuntu/src/bin/testMXNet : tests MXNet

/home/ubuntu/src/bin/testTensorFlow : tests TensorFlow

/home/ubuntu/src/bin/testCaffe2 : tests Caffe2

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Caffe2: Based on this example inside the Caffe2 repository. Validation accuracy threshold is 90%.

Ubuntu AMI

Ubuntu based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

References

MXNet

Test Environments

- Built on p2.16xlarge.
- Also tested on p2.xlarge, c4.4xlarge.

Deep Learning AMI Amazon Linux Version: 3.3_Oct2017

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with popular Deep Learning frameworks and also contain the Anaconda Platform (Python2 and Python3).

Highlights of the Release

1. The Linux AMI is now build with 2017.09 base Amazon Linux AMI
2. CUDA 8 support
3. Framework upgrades for Tensorflow(v1.3.0), Caffe2(v0.8.0), Caffe(1.0), CNTK(v2.0), Theano(rel-0.9.0)

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.11.0 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ec2-user/src/mxnet
- Caffe: Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
  - branch/tag used: 1.0 tag
  - Justification: Supports cuda8.0 and cudnn 5.1
  - Source_Directories:
    - For Python2.7+ - /home/ec2-user/src/caffe
    - For Python3+ - /home/ec2-user/src/caffe3
    - For Anaconda Python2.7+ - /home/ec2-user/src/caffe_anaconda2
    - For Anaconda3 Python3+ - /home/ec2-user/src/caffe_anaconda3
    - For CPU_ONLY : /home/ec2-user/src/caffe_cpu
- Caffe2: Caffe2 is a cross-platform framework made with expression, speed, and modularity in mind.
  - branch/tag used: v0.8.0 tag
  - Justification: Stable and well tested
  - Note: Available for Python2.7 only
  - Source_Directories:
    - For Python2.7+ - /home/ec2-user/src/caffe2
    - For Anaconda Python2.7+ - /home/ec2-user/src/caffe2_anaconda2
- Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
  - branch/tag used: rel-0.9.0 tag
  - Justification: Stable and well tested

  - Source_Directories:
    - /home/ec2-user/src/Theano
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
  - branch/tag used : v1.3.0 tag
  - Justification : Stable and well tested
  - Source_Directories :
    - For Python2.7+ - /home/ec2-user/src/tensorflow
    - For Python3+ - /home/ec2-user/src/tensorflow3
    - For Anaconda Python2.7+ - /home/ec2-user/src/tensorflow_anaconda
    - For Anaconda Python3+ - /home/ec2-user/src/tensorflow_anaconda3
- Torch: Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
  - branch/tag used : master branch
  - Justification : No other stable branch or tag available
  - Source_Directories :
    - /home/ec2-user/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - is a unified deep-learning toolkit by Microsoft Research.
  - branch/tag used : v2.0 tag
  - Justification : Latest release
  - Source_Directories :
    - /home/ec2-user/src/cntk
- Keras: Keras - Deep Learning Library for Python)
  - branch/tag used : v2.0.8
  - Justification : Stable release
  - Source_Directories:
    - /home/ec2-user/src/keras

Python 2.7 and Python 3.5 Support

Python 2.7 and Python 3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support. If you want to use the caffe binary for the CPU instance, then you should use the binary inside /home/ec2-user/src/caffe_cpu/

CNTK Python Support

You can run CNTK for Python inside a conda environment. To do this:

```
cd /home/ec2-user/src/anaconda3/bin
source activate cntk-py34
```

GPU Drivers Installed

- CuDNN 5.1
- Nvidia 375.66
- CUDA 8.0

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch P2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks. The test checks if the validation accuracy is above a specific threshold. The threshold is different for each of the frameworks.

The scripts are available in the /home/ec2-user/src/bin directory.

The following scripts test the various frameworks:

/home/ec2-user/src/bin/testAll : tests all frameworks

/home/ec2-user/src/bin/testMXNet : tests MXNet

/home/ec2-user/src/bin/testTheano : tests Theano

/home/ec2-user/src/bin/testTensorFlow : tests TensorFlow

/home/ec2-user/src/bin/testTorch : tests Torch

/home/ec2-user/src/bin/testCNTK : tests CNTK

/home/ec2-user/src/bin/testCaffe2 : tests Caffe2

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Theano: The same example above. Validation accuracy threshold is 95%.
- Torch: This example inside the Torch tree. Validation accuracy threshold is 93%.
- Caffe: This example inside the Caffe repository. Validation accuracy threshold is 98%.
- CNTK: This example inside the CNTK repository. Validation accuracy threshold is 97%.
- Caffe2: Based on this example inside the Caffe2 repository. Validation accuracy threshold is 90%.

Amazon Linux AMI

Amazon Linux based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

References

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Test Environments

- Built on p2.16xlarge.
- Also tested on p2.xlarge, p2.8xlarge, p2.16xlarge, c4.4xlarge.

Known Issues

- Need to use sudo to run the testCNTK script.

  eg. sudo ./testCNTK
- The conda environments keras1.2_p3 and keras1.2_p2 come with CPU only version of MXnet.

  To use Keras with an MXNet backend to train on GPUs, you can workaround this issue by running the following:

```
pip install mxnet-cu80
```

  from inside the conda environment.

Not Supported

- Functioning of multiple frameworks together in the same Python process has not been tested.

  For example, a code snippet like the following:

```
import mxnet as mx
import tensorflow as tf
```

  in the same Python process may cause an issue.

Deep Learning AMI Amazon Linux Version: 3.1_Sep2017

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with popular Deep Learning frameworks and also contain the Anaconda Platform (Python2 and Python3).

Highlights of the Release

1. Used Ubuntu 16.04 (ami-d15a75c7) as the base AMI
2. CUDA 8 support
3. Framework upgrades for Tensorflow(v1.3.0), Caffe2(v0.8.0), Caffe(1.0), CNTK(v2.0), Theano(rel-0.9.0)

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.11.0 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ec2-user/src/mxnet
- Caffe: Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
  - branch/tag used: 1.0 tag
  - Justification: Supports cuda8.0 and cudnn 5.1
  - Source_Directories:
    - For Python2.7+ - /home/ec2-user/src/caffe
    - For Python3+ - /home/ec2-user/src/caffe3
    - For Anaconda Python2.7+ - /home/ec2-user/src/caffe_anaconda2
    - For Anaconda3 Python3+ - /home/ec2-user/src/caffe_anaconda3
    - For CPU_ONLY : /home/ec2-user/src/caffe_cpu
- Caffe2: Caffe2 is a cross-platform framework made with expression, speed, and modularity in mind.
  - branch/tag used: v0.8.0 tag
  - Justification: Stable and well tested
  - Note: Available for Python2.7 only
  - Source_Directories:
    - For Python2.7+ - /home/ec2-user/src/caffe2
    - For Anaconda Python2.7+ - /home/ec2-user/src/caffe2_anaconda2
- Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
  - branch/tag used: rel-0.9.0 tag
  - Justification: Stable and well tested

  - Source_Directories:
    - /home/ec2-user/src/Theano
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
  - branch/tag used : v1.3.0 tag
  - Justification : Stable and well tested
  - Source_Directories :
    - For Python2.7+ - /home/ec2-user/src/tensorflow
    - For Python3+ - /home/ec2-user/src/tensorflow3
    - For Anaconda Python2.7+ - /home/ec2-user/src/tensorflow_anaconda
    - For Anaconda Python3+ - /home/ec2-user/src/tensorflow_anaconda3
- Torch: Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
  - branch/tag used : master branch
  - Justification : No other stable branch or tag available
  - Source_Directories :
    - /home/ec2-user/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - is a unified deep-learning toolkit by Microsoft Research.
  - branch/tag used : v2.0 tag
  - Justification : Latest release
  - Source_Directories :
    - /home/ec2-user/src/cntk
- Keras: Keras - Deep Learning Library for Python)
  - branch/tag used : master
  - Justification : Stable release (1.2.2 with MXNet support)
  - Source_Directories:
    - /home/ec2-user/src/keras

Python 2.7 and Python 3.5 Support

Python 2.7 and Python 3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support. If you want to use the caffe binary for the CPU instance, then you should use the binary inside /home/ec2-user/src/caffe_cpu/

CNTK Python Support

You can run CNTK for Python inside a conda environment. To do this:

```
cd /home/ec2-user/src/anaconda3/bin
source activate cntk-py34
```

GPU Drivers Installed

- CuDNN 5.1
- Nvidia 375.66
- CUDA 8.0

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch P2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks. The test checks if the validation accuracy is above a specific threshold. The threshold is different for each of the frameworks.

The scripts are available in the /home/ec2-user/src/bin directory.

The following scripts test the various frameworks:

/home/ec2-user/src/bin/testAll : tests all frameworks

/home/ec2-user/src/bin/testMXNet : tests MXNet

/home/ec2-user/src/bin/testTheano : tests Theano

/home/ec2-user/src/bin/testTensorFlow : tests TensorFlow

/home/ec2-user/src/bin/testTorch : tests Torch

/home/ec2-user/src/bin/testCNTK : tests CNTK

/home/ec2-user/src/bin/testCaffe2 : tests Caffe2

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Theano: The same example above. Validation accuracy threshold is 95%.
- Torch: This example inside the Torch tree. Validation accuracy threshold is 93%.
- Caffe: This example inside the Caffe repository. Validation accuracy threshold is 98%.
- CNTK: This example inside the CNTK repository. Validation accuracy threshold is 97%.
- Caffe2: Based on this example inside the Caffe2 repository. Validation accuracy threshold is 90%.

Amazon Linux AMI

Amazon Linux based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)

- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

References

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Test Environments

- Built on p2.16xlarge.
- Also tested on p2.xlarge, p2.8xlarge, p2.16xlarge, c4.4xlarge.

Known Issues

- Need to use sudo to run the testCNTK script.

  eg. sudo ./testCNTK

Not Supported

- Functioning of multiple frameworks together in the same Python process has not been tested.

  For example, a code snippet like the following:

```
import mxnet as mx
import tensorflow as tf
```

  in the same Python process may cause an issue.

Deep Learning AMI Amazon Linux Version: 2.3_June2017

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with popular Deep Learning frameworks and also contain the Anaconda Platform (Python2 and Python3).

Highlights of the Release

1. MXNet compiled with S3 Support(USE_S3=1).
2. Security fixes applied for Stackclash security issue. (https://alas.aws.amazon.com/ALAS-2017-845.html)

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.10.0 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ec2-user/src/mxnet
- Caffe: Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
  - branch/tag used: rc5 tag
  - Justification: Supports cuda7.5 and cudnn 5.1
  - Source_Directories:
    - For Python2.7+ - /home/ec2-user/src/caffe
    - For Python3+ - /home/ec2-user/src/caffe3
    - For Anaconda Python2.7+ - /home/ec2-user/src/caffe_anaconda2
    - For Anaconda3 Python3+ - /home/ec2-user/src/caffe_anaconda3
    - For CPU_ONLY : /home/ec2-user/src/caffe_cpu
- Caffe2: Caffe2 is a cross-platform framework made with expression, speed, and modularity in mind.
  - branch/tag used: v0.7.0 tag
  - Justification: Stable and well tested
  - Note: Available for Python2.7 only
  - Source_Directories:
    - For Python2.7+ - /home/ec2-user/src/caffe2
    - For Anaconda Python2.7+ - /home/ec2-user/src/caffe2_anaconda2
- Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
  - branch/tag used: rel-0.9.0 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ec2-user/src/Theano
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
  - branch/tag used : v1.1.0 tag
  - Justification : Stable and well tested
  - Source_Directories :
    - For Python2.7+ - /home/ec2-user/src/tensorflow

  - For Python3+ - /home/ec2-user/src/tensorflow3
  - For Anaconda Python2.7+ - /home/ec2-user/src/tensorflow_anaconda
  - For Anaconda Python3+ - /home/ec2-user/src/tensorflow_anaconda3
- Torch: Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
  - branch/tag used : master branch
  - Justification : No other stable branch or tag available
  - Source_Directories :
    - /home/ec2-user/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - is a unified deep-learning toolkit by Microsoft Research.
  - branch/tag used : v2.0.rc1 tag
  - Justification : Latest release
  - Source_Directories :
    - /home/ec2-user/src/cntk
- Keras: Keras - Deep Learning Library for Python)
  - branch/tag used : 1.2.2 tag
  - Justification : Stable release
  - Source_Directories:
    - /home/ec2-user/src/keras

Python 2.7 and Python 3.5 Support

Python 2.7 and Python 3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support. If you want to use the caffe binary for the CPU instance, then you should use the binary inside /home/ec2-user/src/caffe_cpu/

CNTK Python Support

You can run CNTK for Python inside a conda environment. To do this:

```
cd /home/ec2-user/src/anaconda3/bin
source activate cntk-py34
```

GPU Drivers Installed

- CuDNN 5.1
- Nvidia 367.57

- CUDA 7.5

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch G2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks. The test checks if the validation accuracy is above a specific threshold. The threshold is different for each of the frameworks.

The scripts are available in the /home/ec2-user/src/bin directory.

The following scripts test the various frameworks:

/home/ec2-user/src/bin/testAll : tests all frameworks

/home/ec2-user/src/bin/testMXNet : tests MXNet

/home/ec2-user/src/bin/testTheano : tests Theano

/home/ec2-user/src/bin/testTensorFlow : tests TensorFlow

/home/ec2-user/src/bin/testTorch : tests Torch

/home/ec2-user/src/bin/testCNTK : tests CNTK

/home/ec2-user/src/bin/testCaffe2 : tests Caffe2

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Theano: The same example above. Validation accuracy threshold is 95%.
- Torch: This example inside the Torch tree. Validation accuracy threshold is 93%.
- Caffe: This example inside the Caffe repository. Validation accuracy threshold is 98%.
- CNTK: This example inside the CNTK repository. Validation accuracy threshold is 97%.
- Caffe2: Based on this example inside the Caffe2 repository. Validation accuracy threshold is 90%.

Amazon Linux AMI

Amazon Linux based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)

- ap-northeast-2(ICN)

References

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Test Environments

- Built on p2.16xlarge.
- Also tested on g2.2xlarge, g2.8xlarge, p2.xlarge, p2.8xlarge, p2.16xlarge, c4.4xlarge.

Known Issues

- Need to use sudo to run the testCNTK script.

  eg. sudo ./testCNTK

Not Supported

- Functioning of multiple frameworks together in the same Python process has not been tested.

  For example, a code snippet like the following:

```
import mxnet as mx
import tensorflow as tf
```

  in the same Python process may cause an issue.

Deep Learning AMI Amazon Linux Version: 2.2_June2017

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with popular Deep Learning frameworks and also contain the Anaconda Platform (Python2 and Python3).

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized

over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.10.0 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ec2-user/src/mxnet
- Caffe: Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
  - branch/tag used: rc5 tag
  - Justification: Supports cuda7.5 and cudnn 5.1
  - Source_Directories:
    - For Python2.7+ - /home/ec2-user/src/caffe
    - For Python3+ - /home/ec2-user/src/caffe3
    - For Anaconda Python2.7+ - /home/ec2-user/src/caffe_anaconda2
    - For Anaconda3 Python3+ - /home/ec2-user/src/caffe_anaconda3
    - For CPU_ONLY : /home/ec2-user/src/caffe_cpu
- Caffe2: Caffe2 is a cross-platform framework made with expression, speed, and modularity in mind.
  - branch/tag used: v0.7.0 tag
  - Justification: Stable and well tested
  - Note: Available for Python2.7 only
  - Source_Directories:
    - For Python2.7+ - /home/ec2-user/src/caffe2
    - For Anaconda Python2.7+ - /home/ec2-user/src/caffe2_anaconda2
- Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
  - branch/tag used: rel-0.9.0 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ec2-user/src/Theano
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
  - branch/tag used : v1.1.0 tag
  - Justification : Stable and well tested
  - Source_Directories :
    - For Python2.7+ - /home/ec2-user/src/tensorflow
    - For Python3+ - /home/ec2-user/src/tensorflow3
    - For Anaconda Python2.7+ - /home/ec2-user/src/tensorflow_anaconda
    - For Anaconda Python3+ - /home/ec2-user/src/tensorflow_anaconda3
- Torch: Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
  - branch/tag used : master branch
  - Justification : No other stable branch or tag available
  - Source_Directories :
    - /home/ec2-user/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - is a unified deep-learning toolkit by Microsoft Research.
  - branch/tag used : v2.0.rc1 tag

  - Justification : Latest release
  - Source_Directories :
    - /home/ec2-user/src/cntk
- Keras: Keras - Deep Learning Library for Python)
  - branch/tag used : 1.2.2 tag
  - Justification : Stable release
  - Source_Directories:
    - /home/ec2-user/src/keras

Python 2.7 and Python 3.5 Support

Python 2.7 and Python 3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support. If you want to use the caffe binary for the CPU instance, then you should use the binary inside /home/ec2-user/src/caffe_cpu/

CNTK Python Support

You can run CNTK for Python inside a conda environment. To do this:

```
cd /home/ec2-user/src/anaconda3/bin
source activate cntk-py34
```

GPU Drivers Installed

- CuDNN 5.1
- Nvidia 367.57
- CUDA 7.5

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch G2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks. The test checks if the validation accuracy is above a specific threshold. The threshold is different for each of the frameworks.

The scripts are available in the /home/ec2-user/src/bin directory.

The following scripts test the various frameworks:

/home/ec2-user/src/bin/testAll : tests all frameworks

/home/ec2-user/src/bin/testMXNet : tests MXNet

/home/ec2-user/src/bin/testTheano : tests Theano

/home/ec2-user/src/bin/testTensorFlow : tests TensorFlow

/home/ec2-user/src/bin/testTorch : tests Torch

/home/ec2-user/src/bin/testCNTK : tests CNTK

/home/ec2-user/src/bin/testCaffe2 : tests Caffe2

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Theano: The same example above. Validation accuracy threshold is 95%.
- Torch: This example inside the Torch tree. Validation accuracy threshold is 93%.
- Caffe: This example inside the Caffe repository. Validation accuracy threshold is 98%.
- CNTK: This example inside the CNTK repository. Validation accuracy threshold is 97%.
- Caffe2: Based on this example inside the Caffe2 repository. Validation accuracy threshold is 90%.

Amazon Linux AMI

Amazon Linux based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

References

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Test Environments

- Built and Tested on g2.2xlarge.
- Also tested on g2.8xlarge, p2.16xlarge, c4.4xlarge.

Known Issues

- Need to use sudo to run the testCNTK script.

  eg. sudo ./testCNTK

Not Supported

- Functioning of multiple frameworks together in the same Python process has not been tested.

  For example, a code snippet like the following:

```
import mxnet as mx
import tensorflow as tf
```

  in the same Python process may cause an issue.

Deep Learning AMI Amazon Linux Version: 2.1_Apr2017

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with popular Deep Learning frameworks and also contain the Anaconda Platform(Python2 and Python3).

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.9.3 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ec2-user/src/mxnet
- Caffe: Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
  - branch/tag used: rc5 tag
  - Justification: Supports cuda7.5 and cudnn 5.1
  - Source_Directories:
    - For Python2.7+ - /home/ec2-user/src/caffe
    - For Python3+ - /home/ec2-user/src/caffe3

    - For Anaconda Python2.7+ - /home/ec2-user/src/caffe_anaconda2
    - For Anaconda3 Python3+ - /home/ec2-user/src/caffe_anaconda3
    - For CPU_ONLY : /home/ec2-user/src/caffe_cpu
- Caffe2: Caffe2 is a cross-platform framework made with expression, speed, and modularity in mind.
    - branch/tag used: v0.7.0 tag
    - Justification: Stable and well tested
    - Note: Available for Python2.7 only
    - Source_Directories:
        - For Python2.7+ - /home/ec2-user/src/caffe2
        - For Anaconda Python2.7+ - /home/ec2-user/src/caffe2_anaconda2
- Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
    - branch/tag used: rel-0.8.2 tag
    - Justification: Stable and well tested
    - Source_Directories:
        - /home/ec2-user/src/Theano
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
    - branch/tag used : v1.0.1 tag
    - Justification : Stable and well tested
    - Source_Directories :
        - For Python2.7+ - /home/ec2-user/src/tensorflow
        - For Python3+ - /home/ec2-user/src/tensorflow3
        - For Anaconda Python2.7+ - /home/ec2-user/src/tensorflow_anaconda
        - For Anaconda Python3+ - /home/ec2-user/src/tensorflow_anaconda3
- Torch: Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
    - branch/tag used : master branch
    - Justification : No other stable branch or tag available
    - Source_Directories :
        - /home/ec2-user/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - is a unified deep-learning toolkit by Microsoft Research.
    - branch/tag used : v2.0.rc1 tag
    - Justification : Latest release
    - Source_Directories :
        - /home/ec2-user/src/cntk
- Keras: Keras - Deep Learning Library for Python)
    - branch/tag used : 1.2.2 tag
    - Justification : Stable release
    - Source_Directories:
        - /home/ec2-user/src/keras

Python 2.7 and Python3.5 Support

Python2.7 and Python3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support. If you want to use the caffe binary for the CPU instance, then you should use the binary inside /home/ec2-user/src/caffe_cpu/

CNTK Python Support

You can run CNTK for Python inside a conda environment. To do this:

```
cd /home/ec2-user/src/anaconda3/bin
source activate cntk-py34
```

GPU Drivers Installed

- CuDNN 5.1
- Nvidia 367.57
- CUDA 7.5

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch G2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks. The test checks if the validation accuracy is above a specific threshold. The threshold is different for each of the frameworks.

The scripts are available in the /home/ec2-user/src/bin directory.

The following scripts test the various frameworks:

/home/ec2-user/src/bin/testAll : tests all frameworks

/home/ec2-user/src/bin/testMXNet : tests MXNet

/home/ec2-user/src/bin/testTheano : tests Theano

/home/ec2-user/src/bin/testTensorFlow : tests TensorFlow

/home/ec2-user/src/bin/testTorch : tests Torch

/home/ec2-user/src/bin/testCNTK : tests CNTK

/home/ec2-user/src/bin/testCaffe2 : tests Caffe2

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Theano: The same example above. Validation accuracy threshold is 95%.
- Torch: This example inside the Torch tree. Validation accuracy threshold is 93%.
- Caffe: This example inside the Caffe repository. Validation accuracy threshold is 98%.
- CNTK: This example inside the CNTK repository. Validation accuracy threshold is 97%.
- Caffe2: Based on this example inside the Caffe2 repository. Validation accuracy threshold is 90%.

Amazon Linux AMI

Amazon Linux based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)

References

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Test Environments

- Built and Tested on g2.2xlarge.
- Also tested on g2.8xlarge, p2.16xlarge, c4.4xlarge.

Known Issues

- Need to use sudo to run the testCNTK script.

  eg. sudo ./testCNTK

Not Supported

- Functioning of multiple frameworks together in the same Python process has not been tested.

  For example, a code snippet like the following:

```
import mxnet as mx
import tensorflow as tf
```

in the same Python process may cause an issue.

Deep Learning AMI Amazon Linux Version: 2.0

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with popular Deep Learning frameworks and also contain the Anaconda Platform (Python2 and Python3).

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.9.3 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ec2-user/src/mxnet
- Caffe: Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
  - branch/tag used: rc5 tag
  - Justification: Supports cuda7.5 and cudnn 5.1
  - Source_Directories:
    - For Python2.7+ - /home/ec2-user/src/caffe
    - For Python3+ - /home/ec2-user/src/caffe3
    - For Anaconda Python2.7+ - /home/ec2-user/src/caffe_anaconda2
    - For Anaconda3 Python3+ - /home/ec2-user/src/caffe_anaconda3
    - For CPU_ONLY : /home/ec2-user/src/caffe_cpu
- Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
  - branch/tag used: rel-0.8.2 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ec2-user/src/Theano
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
  - branch/tag used : v1.0.0 tag
  - Justification : Stable and well tested
  - Source_Directories :
    - For Python2.7+ - /home/ec2-user/src/tensorflow
    - For Python3+ - /home/ec2-user/src/tensorflow3

    - For Anaconda Python2.7+ - /home/ec2-user/src/tensorflow_anaconda
    - For Anaconda Python3+ - /home/ec2-user/src/tensorflow_anaconda3
- Torch: Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
    - branch/tag used : master branch
    - Justification : No other stable branch or tag available
    - Source_Directories :
        - /home/ec2-user/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - is a unified deep-learning toolkit by Microsoft Research.
    - branch/tag used : v2.0beta12.0 tag
    - Justification : Latest release
    - Source_Directories :
        - /home/ec2-user/src/cntk
- Keras: Keras - Deep Learning Library for Python)
    - branch/tag used : 1.2.2 tag
    - Justification : Stable release
    - Source_Directories:
        - /home/ec2-user/src/keras

Python 2.7 and Python 3.5 Support

Python 2.7 and Python 3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support. If you want to use the caffe binary for the CPU instance, then you should use the binary inside /home/ec2-user/src/caffe_cpu/

CNTK Python Support

You can run CNTK for Python inside a conda environment. To do this:

```
cd /home/ec2-user/src/anaconda3/bin
source activate cntk-py34
```

GPU Drivers Installed

- CuDNN 5.1
- Nvidia 367.57
- CUDA 7.5

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch G2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks. The test checks if the validation accuracy is above a specific threshold. The threshold is different for each of the frameworks.

The scripts are available in the /home/ec2-user/src/bin directory.

The following scripts test the various frameworks:

/home/ec2-user/src/bin/testAll : tests all frameworks

/home/ec2-user/src/bin/testMXNet : tests MXNet

/home/ec2-user/src/bin/testTheano : tests Theano

/home/ec2-user/src/bin/testTensorFlow : tests TensorFlow

/home/ec2-user/src/bin/testTorch : tests Torch

/home/ec2-user/src/bin/testCNTK : tests CNTK

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Theano: The same example above. Validation accuracy threshold is 95%.
- Torch: This example inside the Torch tree. Validation accuracy threshold is 93%.
- Caffe: This example inside the Caffe repository. Validation accuracy threshold is 98%.
- CNTK: This example inside the CNTK repository. Validation accuracy threshold is 97%.

Amazon Linux AMI

Amazon Linux based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)

References

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Test Environments

- Built and Tested on g2.2xlarge.
- Also tested on g2.8xlarge, p2.16xlarge, c4.4xlarge.

Known Issues

- Need to use sudo to run the testCNTK script.

  eg. sudo ./testCNTK

Not Supported

- Functioning of multiple frameworks together in the same Python process has not been tested.

  For example, a code snippet like the following:

```
import mxnet as mx
import tensorflow as tf
```

Deep Learning AMI Amazon Linux Version: 1.5

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with popular Deep Learning frameworks and also contain the Anaconda Platform(Python2 and Python3).

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: master branch
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ec2-user/src/mxnet
- Caffe: Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
  - branch/tag used: master branch
  - Justification: Supports cuda7.5 and cudnn 5.1
  - Source_Directories:
    - For Python2.7+ - /home/ec2-user/src/caffe
    - For Python3+ - /home/ec2-user/src/caffe3

    - For Python3+ - /home/ec2-user/src/caffe3
- Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
    - branch/tag used: rel-0.8.2 tag
    - Justification: Stable and well tested
    - Source_Directories:
        - /home/ec2-user/src/Theano
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
    - branch/tag used : r0.10 branch
    - Justification : Stable and well tested
    - Source_Directories :
        - /home/ec2-user/src/tensorflow
- Torch: Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
    - branch/tag used : rel-0.8.2 branch
    - Justification : No other stable branch or tag available
    - Source_Directories :
        - /home/ec2-user/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - is a unified deep-learning toolkit by Microsoft Research.
    - branch/tag used : v2.0beta2.0
    - Justification : Stable release
    - Source_Directories :
        - /home/ec2-user/src/cntk

Python 2.7 and Python 3.5 Support

Python2.7 and Python3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support. If you want to use the caffe binary for the CPU instance, then you should use the binary inside /home/ec2-user/src/caffe_cpu/

CNTK Python Support

You can run CNTK for Python inside a conda environment. To do this:

```
cd /home/ec2-user/src/anaconda3/bin
source activate cntk-py34
```

GPU Drivers Installed

- CuDNN 5.1
- Nvidia 352.99
- CUDA 7.5

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch G2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks. The test checks if the validation accuracy is above a specific threshold. The threshold is different for each of the frameworks.

The scripts are available in the /home/ec2-user/src/bin directory.

The following scripts test the various frameworks:

/home/ec2-user/src/bin/testAll : tests all frameworks

/home/ec2-user/src/bin/testMXNet : tests MXNet

/home/ec2-user/src/bin/testTheano : tests Theano

/home/ec2-user/src/bin/testTensorFlow : tests TensorFlow

/home/ec2-user/src/bin/testTorch : tests Torch

/home/ec2-user/src/bin/testCNTK : tests CNTK

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Theano: The same example above. Validation accuracy threshold is 95%.
- Torch: This example inside the Torch tree. Validation accuracy threshold is 93%.
- Caffe: This example inside the Caffe repository. Validation accuracy threshold is 98%.
- CNTK: This example inside the CNTK repository. Validation accuracy threshold is 97%.

Amazon Linux AMI

Amazon Linux based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)

References

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Test Environments

- Built and Tested on g2.2xlarge.
- Also tested on g2.8xlarge, p2.16xlarge, c4.4xlarge.

Deep Learning AMI Ubuntu Version: 2.4_Oct2017

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with popular Deep Learning frameworks and also contain the Anaconda Platform (Python2 and Python3).

Highlights of the Release

1. Used Ubuntu 16.04 (ami-d15a75c7) as the base AMI
2. CUDA 8 support
3. Framework upgrades for Tensorflow(v1.3.0), Caffe2(v0.8.0), Caffe(1.0), CNTK(v2.0), Theano(rel-0.9.0)

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.11.0 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
  - branch/tag used: v1.0 tag
  - Justification: Supports cuda8.0 and cudnn 5.1
  - Source_Directories:
    - For Python2.7+ - /home/ubuntu/src/caffe
    - For Python3+ - /home/ubuntu/src/caffe3
    - For Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - For Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - For CPU_ONLY : /home/ubuntu/src/caffe_cpu

- Caffe2: Caffe2 is a cross-platform framework made with expression, speed, and modularity in mind.
  - branch/tag used: v0.8.0 tag
  - Justification: Stable and well tested
  - Note: This is an experimental release and there may be some issues. Available for Python2.7 only
  - Source_Directories:
    - For Python2.7+ - /home/ubuntu/src/caffe2
    - For Anaconda Python2.7+ - /home/ubuntu/src/caffe2_anaconda2
- Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
  - branch/tag used: rel-0.9.0 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/Theano
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
  - branch/tag used : v1.3.0 tag
  - Justification : Stable and well tested
  - Source_Directories :
    - For Python2.7+ - /home/ubuntu/src/tensorflow
    - For Python3+ - /home/ubuntu/src/tensorflow3
    - For Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - For Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
  - branch/tag used : master branch
  - Justification : No other stable branch or tag available
  - Source_Directories :
    - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - is a unified deep-learning toolkit by Microsoft Research.
  - branch/tag used : v2.0 tag
  - Justification : Stable release
  - Source_Directories :
    - /home/ubuntu/src/cntk
- Keras: Keras - Deep Learning Library for Python)
  - branch/tag used : v2.0.8
  - Justification : Stable release
  - Source_Directories:
    - /home/ubuntu/src/keras

Python 2.7 and Python 3.5 Support

Python 2.7 and Python 3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. Caffe
2. Tensorflow
3. Theano

4. MXNet
5. CNTK

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support. If you want to use the caffe binary for the CPU instance, then you should use the binary inside /home/ubuntu/src/caffe_cpu/

CNTK Python Support

You can run CNTK for Python inside a conda environment. To do this:

```
cd /home/ubuntu/src/anaconda3/bin
source activate cntk-py34
```

GPU Drivers Installed

- CuDNN 5.1
- Nvidia 375.66
- CUDA 8.0

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch P2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks. The test checks if the validation accuracy is above a specific threshold. The threshold is different for each of the frameworks.

The scripts are available in the /home/ubuntu/src/bin directory.

The following scripts test the various frameworks:

/home/ubuntu/src/bin/testAll : tests all frameworks

/home/ubuntu/src/bin/testMXNet : tests MXNet

/home/ubuntu/src/bin/testTheano : tests Theano

/home/ubuntu/src/bin/testTensorFlow : tests TensorFlow

/home/ubuntu/src/bin/testTorch : tests Torch

/home/ubuntu/src/bin/testCNTK : tests CNTK

/home/ubuntu/src/bin/testCaffe2 : tests Caffe2

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.

- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Theano: The same example above. Validation accuracy threshold is 95%.
- Torch: This example inside the Torch tree. Validation accuracy threshold is 93%.
- Caffe: This example inside the Caffe repository. Validation accuracy threshold is 98%.
- CNTK: This example inside the CNTK repository. Validation accuracy threshold is 97%.
- Caffe2: Based on this example inside the Caffe2 repository. Validation accuracy threshold is 90%.

Ubuntu AMI

Ubuntu based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

References

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Test Environments

- Built on p2.16xlarge.
- Also tested on p2.xlarge, p2.8xlarge, p2.16xlarge, c4.4xlarge.

Known Issues

- Need to use sudo to run the testCNTK script.

  eg. sudo ./testCNTK
- The conda environments keras1.2_p3 and keras1.2_p2 come with CPU only version of MXnet.

  To use Keras with an MXNet backend to train on GPUs, you can workaround this issue by running the following:

```
pip install mxnet-cu80
```

  from inside the conda environment.

Not Supported

- Functioning of multiple frameworks together in the same Python process has not been tested.

  For example, a code snippet like the following:

```
import mxnet as mx
import tensorflow as tf
```

  in the same Python process may cause an issue.

Deep Learning AMI Ubuntu Version: 2.3_Sep2017

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with popular Deep Learning frameworks and also contain the Anaconda Platform (Python2 and Python3).

Highlights of the Release

1. Used Ubuntu 16.04 (ami-d15a75c7) as the base AMI
2. CUDA 8 support
3. Framework upgrades for Tensorflow(v1.3.0), Caffe2(v0.8.0), Caffe(1.0), CNTK(v2.0), Theano(rel-0.9.0)

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.11.0 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
  - branch/tag used: v1.0 tag
  - Justification: Supports cuda8.0 and cudnn 5.1
  - Source_Directories:
    - For Python2.7+ - /home/ubuntu/src/caffe
    - For Python3+ - /home/ubuntu/src/caffe3
    - For Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - For Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - For CPU_ONLY : /home/ubuntu/src/caffe_cpu
- Caffe2: Caffe2 is a cross-platform framework made with expression, speed, and modularity in mind.
  - branch/tag used: v0.8.0 tag
  - Justification: Stable and well tested

  - Note: This is an experimental release and there may be some issues. Available for Python2.7 only
  - Source_Directories:
    - For Python2.7+ - /home/ubuntu/src/caffe2
    - For Anaconda Python2.7+ - /home/ubuntu/src/caffe2_anaconda2
- Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
  - branch/tag used: rel-0.9.0 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/Theano
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
  - branch/tag used : v1.3.0 tag
  - Justification : Stable and well tested
  - Source_Directories :
    - For Python2.7+ - /home/ubuntu/src/tensorflow
    - For Python3+ - /home/ubuntu/src/tensorflow3
    - For Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - For Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
  - branch/tag used : master branch
  - Justification : No other stable branch or tag available
  - Source_Directories :
    - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - is a unified deep-learning toolkit by Microsoft Research.
  - branch/tag used : v2.0 tag
  - Justification : Stable release
  - Source_Directories :
    - /home/ubuntu/src/cntk
- Keras: Keras - Deep Learning Library for Python)
  - branch/tag used : master (with MXNet support)
  - Justification : Stable release (1.2.2)
  - Source_Directories:
    - /home/ubuntu/src/keras

Python 2.7 and Python 3.5 Support

Python 2.7 and Python 3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support. If you want to use the caffe binary for the CPU instance, then you should use the binary inside /home/ubuntu/src/caffe_cpu/

CNTK Python Support

You can run CNTK for Python inside a conda environment. To do this:

```
cd /home/ubuntu/src/anaconda3/bin
source activate cntk-py34
```

GPU Drivers Installed

- CuDNN 5.1
- Nvidia 375.66
- CUDA 8.0

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch P2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks. The test checks if the validation accuracy is above a specific threshold. The threshold is different for each of the frameworks.

The scripts are available in the /home/ubuntu/src/bin directory.

The following scripts test the various frameworks:

/home/ubuntu/src/bin/testAll : tests all frameworks

/home/ubuntu/src/bin/testMXNet : tests MXNet

/home/ubuntu/src/bin/testTheano : tests Theano

/home/ubuntu/src/bin/testTensorFlow : tests TensorFlow

/home/ubuntu/src/bin/testTorch : tests Torch

/home/ubuntu/src/bin/testCNTK : tests CNTK

/home/ubuntu/src/bin/testCaffe2 : tests Caffe2

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Theano: The same example above. Validation accuracy threshold is 95%.
- Torch: This example inside the Torch tree. Validation accuracy threshold is 93%.

- Caffe: This example inside the Caffe repository. Validation accuracy threshold is 98%.
- CNTK: This example inside the CNTK repository. Validation accuracy threshold is 97%.
- Caffe2: Based on this example inside the Caffe2 repository. Validation accuracy threshold is 90%.

Ubuntu AMI

Ubuntu based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

References

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Test Environments

- Built on p2.16xlarge.
- Also tested on p2.xlarge, p2.8xlarge, p2.16xlarge, c4.4xlarge.

Known Issues

- Need to use sudo to run the testCNTK script.

  eg. sudo ./testCNTK

Not Supported

- Functioning of multiple frameworks together in the same Python process has not been tested.

  For example, a code snippet like the following:

```
import mxnet as mx
import tensorflow as tf
```

  in the same Python process may cause an issue.

Deep Learning AMI Ubuntu Version: 2.2_August2017

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with popular Deep Learning frameworks and also contain the Anaconda Platform (Python2 and Python3).

Highlights of the Release

1. Used Ubuntu 16.04 (ami-d15a75c7) as the base AMI
2. CUDA 8 support
3. Framework upgrades for Tensorflow(v1.2.0), Caffe2(v0.7.0), Caffe(1.0), CNTK(v2.0), Theano(rel-0.9.0)

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.10.0 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
  - branch/tag used: v1.0 tag
  - Justification: Supports cuda8.0 and cudnn 5.1
  - Source_Directories:
    - For Python2.7+ - /home/ubuntu/src/caffe
    - For Python3+ - /home/ubuntu/src/caffe3
    - For Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - For Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - For CPU_ONLY : /home/ubuntu/src/caffe_cpu
- Caffe2: Caffe2 is a cross-platform framework made with expression, speed, and modularity in mind.
  - branch/tag used: v0.7.0 tag
  - Justification: Pre-release
  - Note: This is an experimental release and there may be some issues. Available for Python2.7 only
  - Source_Directories:
    - For Python2.7+ - /home/ubuntu/src/caffe2
    - For Anaconda Python2.7+ - /home/ubuntu/src/caffe2_anaconda2
- Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
  - branch/tag used: rel-0.9.0 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/Theano

- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
  - branch/tag used : v1.2.0 tag
  - Justification : Stable and well tested
  - Source_Directories :
    - For Python2.7+ - /home/ubuntu/src/tensorflow
    - For Python3+ - /home/ubuntu/src/tensorflow3
    - For Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - For Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
  - branch/tag used : master branch
  - Justification : No other stable branch or tag available
  - Source_Directories :
    - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - is a unified deep-learning toolkit by Microsoft Research.
  - branch/tag used : v2.0 tag
  - Justification : Latest release
  - Source_Directories :
    - /home/ubuntu/src/cntk
- Keras: Keras - Deep Learning Library for Python)
  - branch/tag used : 1.2.2 tag
  - Justification : Stable release
  - Source_Directories:
    - /home/ubuntu/src/keras

Python 2.7 and Python 3.5 Support

Python 2.7 and Python 3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support. If you want to use the caffe binary for the CPU instance, then you should use the binary inside /home/ubuntu/src/caffe_cpu/

CNTK Python Support

You can run CNTK for Python inside a conda environment. To do this:

```
cd /home/ubuntu/src/anaconda3/bin
```

```
source activate cntk-py34
```

GPU Drivers Installed

- CuDNN 5.1
- Nvidia 375.66
- CUDA 8.0

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch G2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks. The test checks if the validation accuracy is above a specific threshold. The threshold is different for each of the frameworks.

The scripts are available in the /home/ec2-user/src/bin directory.

The following scripts test the various frameworks:

/home/ubuntu/src/bin/testAll : tests all frameworks

/home/ubuntu/src/bin/testMXNet : tests MXNet

/home/ubuntu/src/bin/testTheano : tests Theano

/home/ubuntu/src/bin/testTensorFlow : tests TensorFlow

/home/ubuntu/src/bin/testTorch : tests Torch

/home/ubuntu/src/bin/testCNTK : tests CNTK

/home/ubuntu/src/bin/testCaffe2 : tests Caffe2

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Theano: The same example above. Validation accuracy threshold is 95%.
- Torch: This example inside the Torch tree. Validation accuracy threshold is 93%.
- Caffe: This example inside the Caffe repository. Validation accuracy threshold is 98%.
- CNTK: This example inside the CNTK repository. Validation accuracy threshold is 97%.
- Caffe2: Based on this example inside the Caffe2 repository. Validation accuracy threshold is 90%.

Ubuntu AMI

Ubuntu based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)

- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

References

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Test Environments

- Built on p2.16xlarge.
- Also tested on g2.2xlarge, g2.8xlarge, p2.xlarge, p2.8xlarge, p2.16xlarge, c4.4xlarge.

Known Issues

- Need to use sudo to run the testCNTK script.

  eg. sudo ./testCNTK

Not Supported

- Functioning of multiple frameworks together in the same Python process has not been tested.

  For example, a code snippet like the following:

```
import mxnet as mx
import tensorflow as tf
```

  in the same Python process may cause an issue.

Deep Learning AMI Ubuntu Version: 1.5_June2017

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with popular Deep Learning frameworks and also contain the Anaconda Platform (Python2 and Python3).

Highlights of the Release

1. MXNet compiled with S3 Support(USE_S3=1).
2. Used the latest base AMI for Ubuntu 14.04 (ami-b1143ba7).

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.10.0 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
  - branch/tag used: rc5 tag
  - Justification: Supports cuda7.5 and cudnn 5.1
  - Source_Directories:
    - For Python2.7+ - /home/ubuntu/src/caffe
    - For Python3+ - /home/ubuntu/src/caffe3
    - For Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - For Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - For CPU_ONLY : /home/ubuntu/src/caffe_cpu
- Caffe2: Caffe2 is a cross-platform framework made with expression, speed, and modularity in mind.
  - branch/tag used: v0.7.0 tag
  - Justification: Pre-release
  - Note: This is an experimental release and there may be some issues. Available for Python2.7 only
  - Source_Directories:
    - For Python2.7+ - /home/ubuntu/src/caffe2
    - For Anaconda Python2.7+ - /home/ubuntu/src/caffe2_anaconda2
- Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
  - branch/tag used: rel-0.9.0 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/Theano
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
  - branch/tag used : v1.1.0 tag
  - Justification : Stable and well tested
  - Source_Directories :
    - For Python2.7+ - /home/ubuntu/src/tensorflow
    - For Python3+ - /home/ubuntu/src/tensorflow3
    - For Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - For Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
  - branch/tag used : master branch

  - Justification : No other stable branch or tag available
  - Source_Directories :
    - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - is a unified deep-learning toolkit by Microsoft Research.
  - branch/tag used : v2.0.rc1 tag
  - Justification : Latest release
  - Source_Directories :
    - /home/ubuntu/src/cntk
- Keras: Keras - Deep Learning Library for Python)
  - branch/tag used : 1.2.2 tag
  - Justification : Stable release
  - Source_Directories:
    - /home/ubuntu/src/keras

Python 2.7 and Python 3.5 Support

Python 2.7 and Python 3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support. If you want to use the caffe binary for the CPU instance, then you should use the binary inside /home/ubuntu/src/caffe_cpu/

CNTK Python Support

You can run CNTK for Python inside a conda environment. To do this:

```
cd /home/ubuntu/src/anaconda3/bin
source activate cntk-py34
```

GPU Drivers Installed

- CuDNN 5.1
- Nvidia 367.57
- CUDA 7.5

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch G2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks. The test checks if the validation accuracy is above a specific threshold. The threshold is different for each of the frameworks.

The scripts are available in the /home/ec2-user/src/bin directory.

The following scripts test the various frameworks:

/home/ubuntu/src/bin/testAll : tests all frameworks

/home/ubuntu/src/bin/testMXNet : tests MXNet

/home/ubuntu/src/bin/testTheano : tests Theano

/home/ubuntu/src/bin/testTensorFlow : tests TensorFlow

/home/ubuntu/src/bin/testTorch : tests Torch

/home/ubuntu/src/bin/testCNTK : tests CNTK

/home/ubuntu/src/bin/testCaffe2 : tests Caffe2

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Theano: The same example above. Validation accuracy threshold is 95%.
- Torch: This example inside the Torch tree. Validation accuracy threshold is 93%.
- Caffe: This example inside the Caffe repository. Validation accuracy threshold is 98%.
- CNTK: This example inside the CNTK repository. Validation accuracy threshold is 97%.
- Caffe2: Based on this example inside the Caffe2 repository. Validation accuracy threshold is 90%.

Ubuntu AMI

Ubuntu based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

References

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Test Environments

- Built on p2.16xlarge.
- Also tested on g2.2xlarge, g2.8xlarge, p2.xlarge, p2.8xlarge, p2.16xlarge, c4.4xlarge.

Known Issues

- Need to use sudo to run the testCNTK script.

  eg. sudo ./testCNTK
- There is a known issue with the Dropout operator when running on GPU instance types. Please see the Github Issue for details and workaround.

Not Supported

- Functioning of multiple frameworks together in the same Python process has not been tested.

  For example, a code snippet like the following:

```
import mxnet as mx
import tensorflow as tf
```

  in the same Python process may cause an issue.

Deep Learning AMI Ubuntu Version: 1.4_June2017

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with popular Deep Learning frameworks and also contain the Anaconda Platform (Python2 and Python3).

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.10.0 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/mxnet

- Caffe: Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
  - branch/tag used: rc5 tag
  - Justification: Supports cuda7.5 and cudnn 5.1
  - Source_Directories:
    - For Python2.7+ - /home/ubuntu/src/caffe
    - For Python3+ - /home/ubuntu/src/caffe3
    - For Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - For Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - For CPU_ONLY : /home/ubuntu/src/caffe_cpu
- Caffe2: Caffe2 is a cross-platform framework made with expression, speed, and modularity in mind.
  - branch/tag used: v0.7.0 tag
  - Justification: Pre-release
  - Note: This is an experimental release and there may be some issues. Available for Python2.7 only
  - Source_Directories:
    - For Python2.7+ - /home/ubuntu/src/caffe2
    - For Anaconda Python2.7+ - /home/ubuntu/src/caffe2_anaconda2
- Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
  - branch/tag used: rel-0.9.0 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/Theano
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
  - branch/tag used : v1.1.0 tag
  - Justification : Stable and well tested
  - Source_Directories :
    - For Python2.7+ - /home/ubuntu/src/tensorflow
    - For Python3+ - /home/ubuntu/src/tensorflow3
    - For Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - For Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
  - branch/tag used : master branch
  - Justification : No other stable branch or tag available
  - Source_Directories :
    - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - is a unified deep-learning toolkit by Microsoft Research.
  - branch/tag used : v2.0.rc1 tag
  - Justification : Latest release
  - Source_Directories :
    - /home/ubuntu/src/cntk
- Keras: Keras - Deep Learning Library for Python)
  - branch/tag used : 1.2.2 tag
  - Justification : Stable release

- Source_Directories:
  - /home/ubuntu/src/keras

Python 2.7 and Python 3.5 Support

Python 2.7 and Python 3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support. If you want to use the caffe binary for the CPU instance, then you should use the binary inside /home/ubuntu/src/caffe_cpu/

CNTK Python Support

You can run CNTK for Python inside a conda environment. To do this:

```
cd /home/ubuntu/src/anaconda3/bin
source activate cntk-py34
```

GPU Drivers Installed

- CuDNN 5.1
- Nvidia 367.57
- CUDA 7.5

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch G2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks. The test checks if the validation accuracy is above a specific threshold. The threshold is different for each of the frameworks.

The scripts are available in the /home/ec2-user/src/bin directory.

The following scripts test the various frameworks:

/home/ubuntu/src/bin/testAll : tests all frameworks

/home/ubuntu/src/bin/testMXNet : tests MXNet

/home/ubuntu/src/bin/testTheano : tests Theano

/home/ubuntu/src/bin/testTensorFlow : tests TensorFlow

/home/ubuntu/src/bin/testTorch : tests Torch

/home/ubuntu/src/bin/testCNTK : tests CNTK

/home/ubuntu/src/bin/testCaffe2 : tests Caffe2

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Theano: The same example above. Validation accuracy threshold is 95%.
- Torch: This example inside the Torch tree. Validation accuracy threshold is 93%.
- Caffe: This example inside the Caffe repository. Validation accuracy threshold is 98%.
- CNTK: This example inside the CNTK repository. Validation accuracy threshold is 97%.
- Caffe2: Based on this example inside the Caffe2 repository. Validation accuracy threshold is 90%.

Ubuntu AMI

Ubuntu based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)
- us-east-2(CHM)
- ap-southeast-2(SYD)
- ap-northeast-1(NRT)
- ap-northeast-2(ICN)

References

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Test Environments

- Built on p2.16xlarge.

- Also tested on g2.2xlarge, g2.8xlarge, p2.xlarge, p2.8xlarge, p2.16xlarge, c4.4xlarge.

Known Issues

- Need to use sudo to run the testCNTK script.

  eg. sudo ./testCNTK
- There is a known issue with the Dropout operator when running on GPU instance types. Please see the Github Issue for details and workaround.

Not Supported

- Functioning of multiple frameworks together in the same Python process has not been tested.

  For example, a code snippet like the following:

```
import mxnet as mx
import tensorflow as tf
```

  in the same Python process may cause an issue.

Deep Learning AMI Ubuntu Version: 1.3_Apr2017

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with popular Deep Learning frameworks and also contain the Anaconda Platform (Python2 and Python3).

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.9.3 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
  - branch/tag used: rc5 tag
  - Justification: Supports cuda7.5 and cudnn 5.1
  - Source_Directories:
    - For Python2.7+ - /home/ubuntu/src/caffe
    - For Python3+ - /home/ubuntu/src/caffe3
    - For Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - For Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3

    - For CPU_ONLY : /home/ubuntu/src/caffe_cpu
- Caffe2: Caffe2 is a cross-platform framework made with expression, speed, and modularity in mind.
  - branch/tag used: v0.6.0 tag
  - Justification: Pre-release
  - Note: This is an experimental release and there may be some issues. Available for Python2.7 only
  - Source_Directories:
    - For Python2.7+ - /home/ubuntu/src/caffe2
    - For Anaconda Python2.7+ - /home/ubuntu/src/caffe2_anaconda2
- Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
  - branch/tag used: rel-0.8.2 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/Theano
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
  - branch/tag used : v1.0.1 tag
  - Justification : Stable and well tested
  - Source_Directories :
    - For Python2.7+ - /home/ubuntu/src/tensorflow
    - For Python3+ - /home/ubuntu/src/tensorflow3
    - For Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - For Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
  - branch/tag used : master branch
  - Justification : No other stable branch or tag available
  - Source_Directories :
    - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - is a unified deep-learning toolkit by Microsoft Research.
  - branch/tag used : v2.0.rc1 tag
  - Justification : Latest release
  - Source_Directories :
    - /home/ubuntu/src/cntk
- Keras: Keras - Deep Learning Library for Python)
  - branch/tag used : 1.2.2 tag
  - Justification : Stable release
  - Source_Directories:
    - /home/ubuntu/src/keras

Python 2.7 and Python 3.5 Support

Python 2.7 and Python 3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. Caffe
2. Tensorflow

3. Theano
4. MXNet
5. CNTK

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support. If you want to use the caffe binary for the CPU instance, then you should use the binary inside /home/ubuntu/src/caffe_cpu/

CNTK Python Support

You can run CNTK for Python inside a conda environment. To do this:

```
cd /home/ubuntu/src/anaconda3/bin
source activate cntk-py34
```

GPU Drivers Installed

- CuDNN 5.1
- Nvidia 367.57
- CUDA 7.5

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch G2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks. The test checks if the validation accuracy is above a specific threshold. The threshold is different for each of the frameworks.

The scripts are available in the /home/ec2-user/src/bin directory.

The following scripts test the various frameworks:

/home/ubuntu/src/bin/testAll : tests all frameworks

/home/ubuntu/src/bin/testMXNet : tests MXNet

/home/ubuntu/src/bin/testTheano : tests Theano

/home/ubuntu/src/bin/testTensorFlow : tests TensorFlow

/home/ubuntu/src/bin/testTorch : tests Torch

/home/ubuntu/src/bin/testCNTK : tests CNTK

/home/ubuntu/src/bin/testCaffe2 : tests Caffe2

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Theano: The same example above. Validation accuracy threshold is 95%.
- Torch: This example inside the Torch tree. Validation accuracy threshold is 93%.
- Caffe: This example inside the Caffe repository. Validation accuracy threshold is 98%.
- CNTK: This example inside the CNTK repository. Validation accuracy threshold is 97%.
- Caffe2: Based on this example inside the Caffe2 repository. Validation accuracy threshold is 90%.

Ubuntu AMI

Ubuntu based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)

References

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Test Environments

- Built and Tested on g2.2xlarge.
- Also tested on g2.8xlarge, p2.16xlarge, c4.4xlarge.

Known Issues

- Need to use sudo to run the testCNTK script.

  eg. sudo ./testCNTK
- There is a known issue with the Dropout operator when running on GPU instance types. Please see the Github Issue for details and workaround.

Not Supported

- Functioning of multiple frameworks together in the same Python process has not been tested.

  For example, a code snippet like the following:

```
import mxnet as mx
import tensorflow as tf
```

in the same Python process may cause an issue.

Deep Learning AMI Ubuntu Version: 1.2

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with popular Deep Learning frameworks and also contain the Anaconda Platform (Python2 and Python3).

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.9.3 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
  - branch/tag used: rc5 tag
  - Justification: Supports cuda7.5 and cudnn 5.1
  - Source_Directories:
    - For Python2.7+ - /home/ubuntu/src/caffe
    - For Python3+ - /home/ubuntu/src/caffe3
    - For Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
    - For Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
    - For CPU_ONLY : /home/ubuntu/src/caffe_cpu
- Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
  - branch/tag used: rel-0.8.2 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/Theano
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
  - branch/tag used : v1.0.0 tag
  - Justification : Stable and well tested
  - Source_Directories :
    - For Python2.7+ - /home/ubuntu/src/tensorflow
    - For Python3+ - /home/ubuntu/src/tensorflow3

    - For Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - For Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
    - branch/tag used : master branch
    - Justification : No other stable branch or tag available
    - Source_Directories :
        - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - is a unified deep-learning toolkit by Microsoft Research.
    - branch/tag used : v2.0beta12.0 tag
    - Justification : Latest release
    - Source_Directories :
        - /home/ubuntu/src/cntk
- Keras: Keras - Deep Learning Library for Python)
    - branch/tag used : 1.2.2 tag
    - Justification : Stable release
    - Source_Directories:
        - /home/ubuntu/src/keras

Python 2.7 and Python 3.5 Support

Python 2.7 and Python 3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support. If you want to use the caffe binary for the CPU instance, then you should use the binary inside /home/ubuntu/src/caffe_cpu/

CNTK Python Support

You can run CNTK for Python inside a conda environment. To do this:

```
cd /home/ubuntu/src/anaconda3/bin
source activate cntk-py34
```

GPU Drivers Installed

- CuDNN 5.1
- Nvidia 367.57
- CUDA 7.5

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch G2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks. The test checks if the validation accuracy is above a specific threshold. The threshold is different for each of the frameworks.

The scripts are available in the /home/ec2-user/src/bin directory.

The following scripts test the various frameworks:

/home/ubuntu/src/bin/testAll : tests all frameworks

/home/ubuntu/src/bin/testMXNet : tests MXNet

/home/ubuntu/src/bin/testTheano : tests Theano

/home/ubuntu/src/bin/testTensorFlow : tests TensorFlow

/home/ubuntu/src/bin/testTorch : tests Torch

/home/ubuntu/src/bin/testCNTK : tests CNTK

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Theano: The same example above. Validation accuracy threshold is 95%.
- Torch: This example inside the Torch tree. Validation accuracy threshold is 93%.
- Caffe: This example inside the Caffe repository. Validation accuracy threshold is 98%.
- CNTK: This example inside the CNTK repository. Validation accuracy threshold is 97%.

Ubuntu AMI

Ubuntu based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)

References

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Test Environments

- Built and Tested on g2.2xlarge.
- Also tested on g2.8xlarge, p2.16xlarge, c4.4xlarge.

Known Issues

- Need to use sudo to run the testCNTK script.

  eg. sudo ./testCNTK
- There is a known issue with the Dropout operator when running on GPU instance types. Please see the Github Issue for details and workaround.

Not Supported

- Functioning of multiple frameworks together in the same Python process has not been tested.

  For example, a code snippet like the following:

```
import mxnet as mx
import tensorflow as tf
```

  in the same Python process may cause an issue.

Deep Learning AMI Ubuntu Version: 1.1

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with popular Deep Learning frameworks and also contain the Anaconda Platform (Python2 and Python3).

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.9.3 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
  - branch/tag used: rc4 tag
  - Justification: Supports cuda7.5 and cudnn 5.1
  - Source_Directories:

  - For Python2.7+ - /home/ubuntu/src/caffe
  - For Python3+ - /home/ubuntu/src/caffe3
  - For Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
  - For Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
  - For CPU_ONLY : /home/ubuntu/src/caffe_cpu
- Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
  - branch/tag used: rel-0.8.2 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/theano
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
  - branch/tag used : v0.12.1 tag
  - Justification : Stable and well tested
  - Source_Directories :
    - For Python2.7+ - /home/ubuntu/src/tensorflow
    - For Python3+ - /home/ubuntu/src/tensorflow3
    - For Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - For Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
  - branch/tag used : master branch
  - Justification : No other stable branch or tag available
  - Source_Directories :
    - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - is a unified deep-learning toolkit by Microsoft Research.
  - branch/tag used : v2.0beta7.0 tag
  - Justification : Latest release
  - Source_Directories :
    - /home/ubuntu/src/cntk
- Keras: Keras - Deep Learning Library for Python)
  - branch/tag used : 1.2.1 tag
  - Justification : Stable release
  - Source_Directories:
    - /home/ubuntu/src/keras

Python 2.7 and Python 3.5 Support

Python 2.7 and Python 3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support. If you want to use the caffe binary for the CPU instance, then you should use the binary inside /home/ubuntu/src/caffe_cpu/

CNTK Python Support

You can run CNTK for Python inside a conda environment. To do this:

```
cd /home/ubuntu/src/anaconda3/bin
source activate cntk-py34
```

GPU Drivers Installed

- CuDNN 5.1
- Nvidia 352.99
- CUDA 7.5

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch G2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks. The test checks if the validation accuracy is above a specific threshold. The threshold is different for each of the frameworks.

The scripts are available in the /home/ec2-user/src/bin directory.

The following scripts test the various frameworks:

/home/ubuntu/src/bin/testAll : tests all frameworks

/home/ubuntu/src/bin/testMXNet : tests MXNet

/home/ubuntu/src/bin/testTheano : tests Theano

/home/ubuntu/src/bin/testTensorFlow : tests TensorFlow

/home/ubuntu/src/bin/testTorch : tests Torch

/home/ubuntu/src/bin/testCNTK : tests CNTK

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Theano: The same example above. Validation accuracy threshold is 95%.
- Torch: This example inside the Torch tree. Validation accuracy threshold is 93%.
- Caffe: This example inside the Caffe repository. Validation accuracy threshold is 98%.
- CNTK: This example inside the CNTK repository. Validation accuracy threshold is 97%.

Ubuntu AMI

Ubuntu based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)

References

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Test Environments

- Built and Tested on g2.2xlarge.
- Also tested on g2.8xlarge, p2.16xlarge, c4.4xlarge.

Deep Learning AMI Ubuntu Version: 1.0

Deep Learning Amazon Machine Image

The Deep Learning AMIs are prebuilt with popular Deep Learning frameworks and also contain the Anaconda Platform (Python2 and Python3).

Prebuilt Deep Learning Frameworks

- MXNet: MXNet is a flexible, efficient, portable and scalable open source library for deep learning. It supports declarative and imperative programming models, across a wide variety of programming languages, making it powerful yet simple to code deep learning applications. MXNet is efficient, inherently supporting automatic parallel scheduling of portions of source code that can be parallelized over a distributed environment. MXNet is also portable, using memory optimizations that allow it to run on mobile phones to full servers.
  - branch/tag used: v0.9.3 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/mxnet
- Caffe: Caffe is a deep learning framework made with expression, speed, and modularity in mind. It is developed by the Berkeley Vision and Learning Center (BVLC) and by community contributors.
  - branch/tag used: master branch
  - Justification: Supports cuda7.5 and cudnn 5.1
  - Source_Directories:
    - For Python2.7+ - /home/ubuntu/src/caffe

  - For Python3+ - /home/ubuntu/src/caffe3
  - For Anaconda Python2.7+ - /home/ubuntu/src/caffe_anaconda2
  - For Anaconda3 Python3+ - /home/ubuntu/src/caffe_anaconda3
  - For CPU_ONLY : /home/ubuntu/src/caffe_cpu
- Theano: Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.
  - branch/tag used: rel-0.8.2 tag
  - Justification: Stable and well tested
  - Source_Directories:
    - /home/ubuntu/src/theano
- TensorFlow: TensorFlow™ is an open source software library for numerical computation using data flow graphs.
  - branch/tag used : v0.12.1 tag
  - Justification : Stable and well tested
  - Source_Directories :
    - For Python2.7+ - /home/ubuntu/src/tensorflow
    - For Python3+ - /home/ubuntu/src/tensorflow3
    - For Anaconda Python2.7+ - /home/ubuntu/src/tensorflow_anaconda
    - For Anaconda Python3+ - /home/ubuntu/src/tensorflow_anaconda3
- Torch: Torch is a scientific computing framework with wide support for machine learning algorithms that puts GPUs first. It is easy to use and efficient, thanks to an easy and fast scripting language, LuaJIT, and an underlying C/CUDA implementation.
  - branch/tag used : master branch
  - Justification : No other stable branch or tag available
  - Source_Directories :
    - /home/ubuntu/src/torch
- CNTK: CNTK - Microsoft Cognitive Toolkit - is a unified deep-learning toolkit by Microsoft Research.
  - branch/tag used : v2.0beta7.0 tag
  - Justification : Latest release
  - Source_Directories :
    - /home/ubuntu/src/cntk

Python 2.7 and Python 3.5 Support

Python 2.7 and Python 3.5 are supported in the AMI for the following Deep Learning Frameworks:

1. Caffe
2. Tensorflow
3. Theano
4. MXNet
5. CNTK

CPU Instance Type Support

The AMI supports CPU Instance Types for all frameworks. MXNet is built with support for Intel MKL2017 DNN library support. If you want to use the caffe binary for the CPU instance, then you should use the binary inside /home/ubuntu/src/caffe_cpu/

CNTK Python Support

You can run CNTK for Python inside a conda environment. To do this:

```
cd /home/ubuntu/src/anaconda3/bin
source activate cntk-py34
```

GPU Drivers Installed

- CuDNN 5.1
- Nvidia 352.99
- CUDA 7.5

Launching Deep Learning Instance

Choose the flavor of the AMI from the list below in the region of your choice and follow the steps at:

EC2 Documentation to launch G2 Instance

Testing the FrameWorks

The Deep Learning frameworks have been tested with MNIST data. The AMI contains scripts to train and test with MNIST for each of the frameworks. The test checks if the validation accuracy is above a specific threshold. The threshold is different for each of the frameworks.

The scripts are available in the /home/ec2-user/src/bin directory.

The following scripts test the various frameworks:

/home/ubuntu/src/bin/testAll : tests all frameworks

/home/ubuntu/src/bin/testMXNet : tests MXNet

/home/ubuntu/src/bin/testTheano : tests Theano

/home/ubuntu/src/bin/testTensorFlow : tests TensorFlow

/home/ubuntu/src/bin/testTorch : tests Torch

/home/ubuntu/src/bin/testCNTK : tests CNTK

The following tests have been run against each of the frameworks:

- MXNet: This example inside the MXNet repository. Validation accuracy threshold tested for is 97%.
- Tensorflow: This example inside the keras repository. Validation accuracy threshold tested for is 95%.
- Theano: The same example above. Validation accuracy threshold is 95%.
- Torch: This example inside the Torch tree. Validation accuracy threshold is 93%.
- Caffe: This example inside the Caffe repository. Validation accuracy threshold is 98%.
- CNTK: This example inside the CNTK repository. Validation accuracy threshold is 97%.

Ubuntu AMI

Ubuntu based Deep Learning AMIs are available in the following regions:

- eu-west-1(DUB)
- us-east-1(IAD)
- us-west-1(PDX)

References

MXNet

Caffe

Theano

TensorFlow

Torch

CNTK

Test Environments

- Built and Tested on g2.2xlarge.
- Also tested on g2.8xlarge, p2.16xlarge, c4.4xlarge.

Document History for Apache MXNet on AWS

The following table describes the documentation for this release of MXNet on AWS.

- **Latest documentation update:** May 16, 2017

Change	Description	Date
Initial release	This is the initial public release of Apache MXNet on AWS.	May 16, 2017

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the *AWS General Reference*.