

University of the Immaculate C x Amazon SageMaker | us-east-1 x Home x CreateAnMXNetImageClassifier x +

notebook-akntfeb4ctfe.notebook.us-east-1.sagemaker.aws/notebooks/CreateAnMXNetImageClassifier.ipynb#

jupyter CreateAnMXNetImageClassifier Last Checkpoint: 11 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

code in each cell is trying to accomplish, and then take the time to experiment: make changes, break it, fix it, and learn! You can always pull the code down again to get a clean copy.

1) Preparing our Environment

First, we'll need to install the `mxnet` package to run everything. Since it's not included in the `conda_python3` image by default, we'll need to use `pip` to install it.

```
In [1]: %pip install mxnet
```

Requirement already satisfied: mxnet in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (1.9.1)
 Requirement already satisfied: numpy<2.0.0,>1.16.0 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from mxnet) (1.22.4)
 Requirement already satisfied: requests<3,>=2.20.0 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from mxnet) (2.31.0)
 Requirement already satisfied: graphviz<0.9.0,>=0.8.1 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from mxnet) (0.8.4)
 Requirement already satisfied: charset-normalizer<4,>=2 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from requests<3,>=2.20.0->mxnet) (3.3.2)
 Requirement already satisfied: idna<4,>=2.5 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from requests<3,>=2.20.0->mxnet) (3.6)
 Requirement already satisfied: urllib3<3,>=1.21.1 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from requests<3,>=2.20.0->mxnet) (2.2.1)
 Requirement already satisfied: certifi>=2017.4.17 in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from requests<3,>=2.20.0->mxnet) (2024.2.2)
 Note: you may need to restart the kernel to use updated packages.

Now that MXNet has been installed, we can import all of the modules we'll need to build and run our image classification model.

```
In [5]: import mxnet as mx
from mxnet.gluon.data.vision import transforms
```

University of the Immaculate C x Amazon SageMaker | us-east-1 x Home x CreateAnMXNetImageClassifier x +

notebook-akntfeb4ctfe.notebook.us-east-1.sagemaker.aws/notebooks/CreateAnMXNetImageClassifier.ipynb#

jupyter CreateAnMXNetImageClassifier Last Checkpoint: 11 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

```
In [5]: import mxnet as mx
from mxnet.gluon.data.vision import transforms

from mxnet import nd, gluon, autograd
from mxnet.gluon import nn

import pickle

import numpy as np
import matplotlib.pyplot as plt
```

Since this is part of a learning environment, we'll also include a fixed seed. This means your results should follow what you experience in the hands-on lab, although you can also change this, or comment it out entirely for different results.

And of course, there's only one seed that which would be appropriate to help us figure out life, the universe, and everything

```
In [10]: mx.random.seed(42)
```

Lastly, we need to set the processor type MXNet will use. This Jupyter Notebook server doesn't have access to GPU's and we let MXNet discover that here. That will be okay for what we are doing.

```
In [11]: ctx = mx.gpu(0) if mx.context.num_gpus() > 0 else mx.cpu(0)
```

2) Load the Data

We have a dataset created from a set of photos of LEGO bricks. In total we have 2 sets of data saved to files as NDArrays.

1. `lego-simple-mx-train` - Training images and labels combined, around 80% of the data collected.

University of the Immaculate C x Amazon SageMaker | us-east-1 x Home x CreateAnMXNetImageClassifier x +

notebook-akntfeb4ctfe.notebook.us-east-1.sagemaker.aws/notebooks/CreateAnMXNetImageClassifier.ipynb#

jupyter CreateAnMXNetImageClassifier Last Checkpoint: 11 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3 O

2) Load the Data

We have a dataset created from a set of photos of LEGO bricks. In total we have 2 sets of data saved to files as NDArrays.

1. **lego-simple-mx-train** - Training images and labels combined, around 80% of the data collected.
2. **lego-simple-mx-test** - Testing or validation images and labels combined, around 20% of the data collected.

First we load the data into runtime arrays. `Pickle` has been used to create these object files, so we use `pickle` to load them as well.

```
In [23]: train_fh = open('lego-simple-mx-train', 'rb')
test_fh = open('lego-simple-mx-test', 'rb')

train_data = pickle.load(train_fh)
test_data = pickle.load(test_fh)
```

The label data we loaded are integer values (1,2,3). We want human names for the data classes we're working with.

```
In [27]: # For humans:
class_names = ['2x3 Brick', '2x2 Brick', '1x3 Brick', '2x1 Brick', '1x1 Brick',
               '2x2 Macaroni', '2x2 Curved End', 'Cog 16 Tooth', '1x2 Handles', '1x2 Grill']

# Or the real LEGO codes:
class_names = ['3002', '3003', '3622', '3004', '3005', '3063', '47457', '94925', '3839a', '2412b']
```

Convert to MXNet Tensors

We have the data loaded into NDArrays. Now we transfer the data into MXNet tensors.

Tensors act like arrays but with extra capability.

University of the Immaculate C x Amazon SageMaker | us-east-1 x Home x CreateAnMXNetImageClassifier x +

notebook-akntfeb4ctfe.notebook.us-east-1.sagemaker.aws/notebooks/CreateAnMXNetImageClassifier.ipynb#

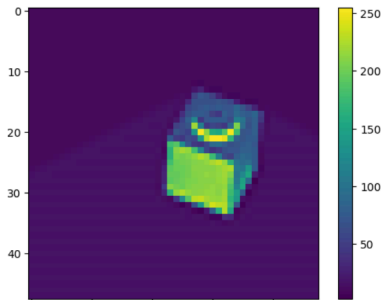
jupyter CreateAnMXNetImageClassifier Last Checkpoint: 18 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3 O

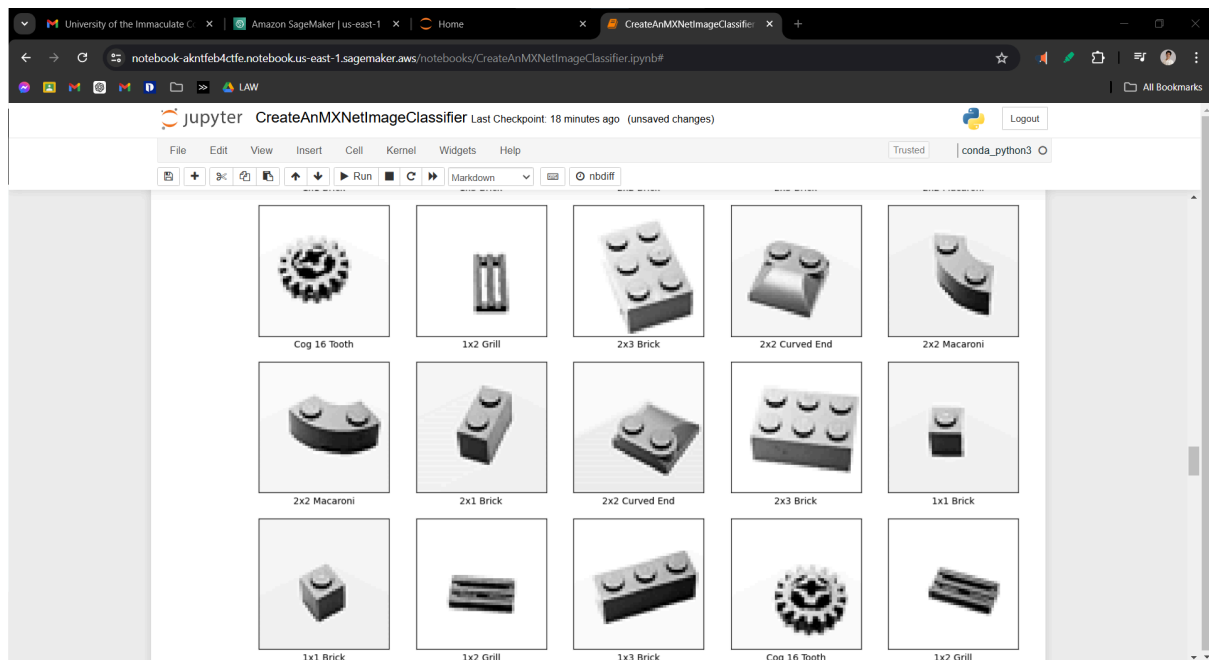
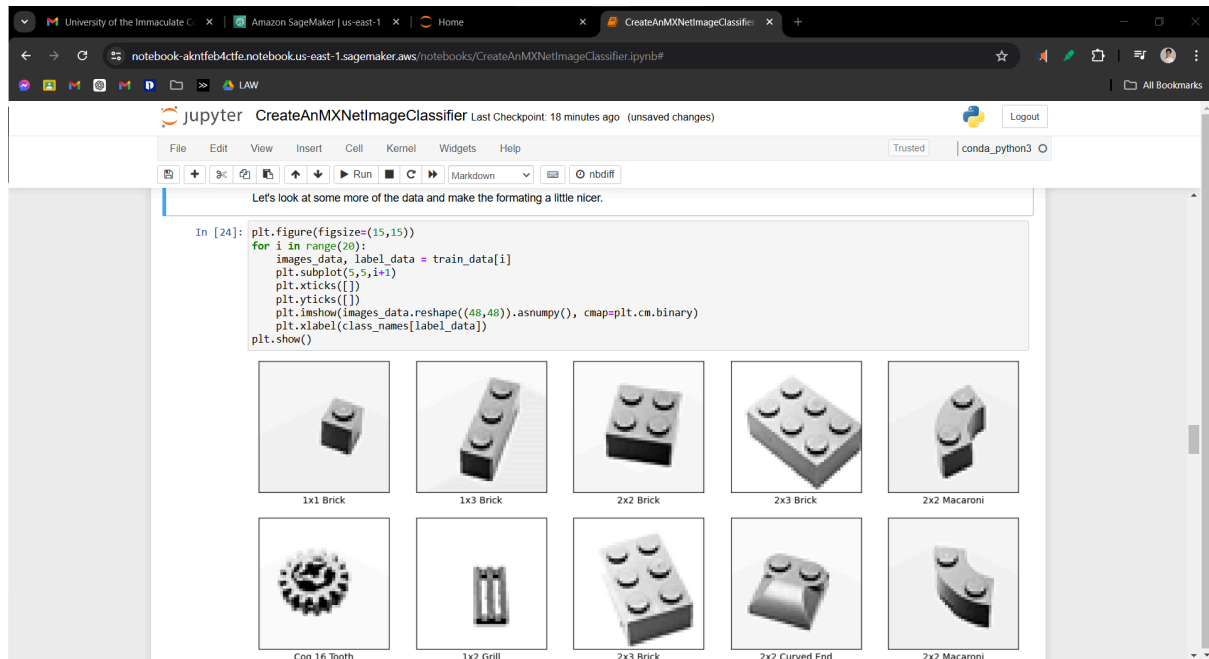
Let's take a look at one of the images loaded with the data.

```
In [25]: train_image_no = 0

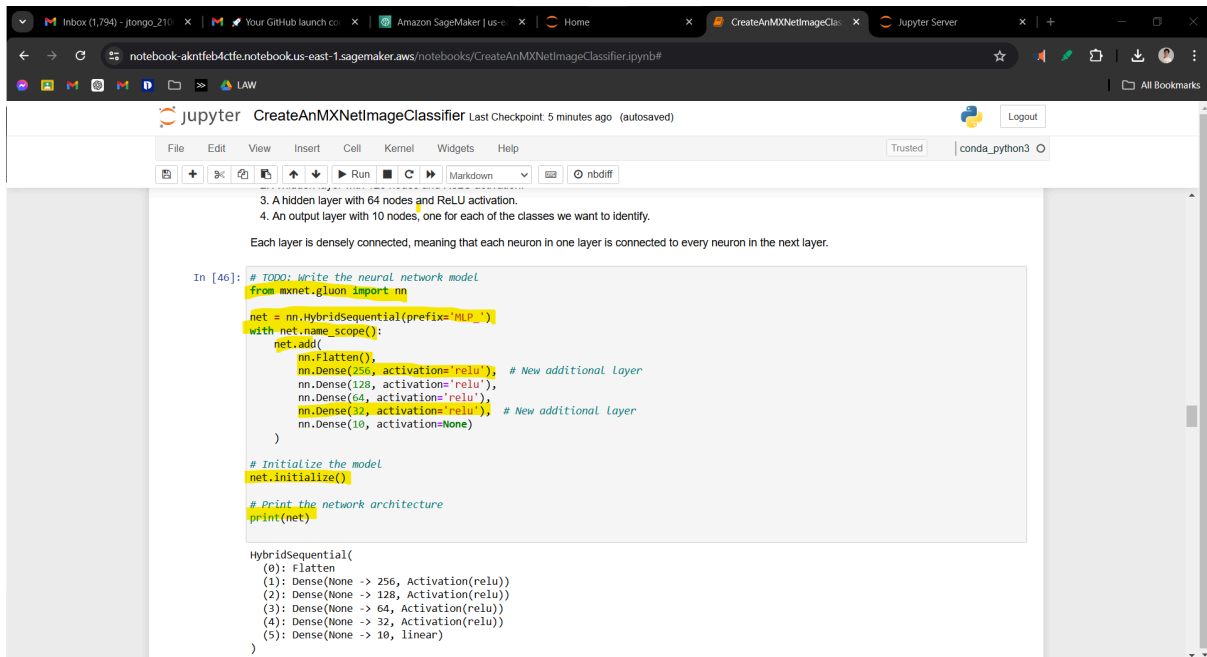
images_data, label_data = train_data[train_image_no]
plt.figure()
plt.imshow(images_data.reshape((48,48)).asnumpy())
plt.colorbar()
plt.xlabel(class_names[label_data])
plt.show()
```



TONGO, JOHN FRANKIE A.



TONGO, JOHN FRANKIE A.



The screenshot shows a Jupyter Notebook interface with the title 'CreateAnMXNetImageClassifier'. The notebook is running on a browser with the URL 'notebook-akntfeb4ctfe.notebook.us-east-1.sagemaker.aws/notebooks/CreateAnMXNetImageClassifier.ipynb#'. The notebook has a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. The 'Cell' menu is open, showing options like 'Run', 'Markdown', and 'nbdiff'. The notebook content includes a text block with instructions for creating a neural network model, followed by a code cell. The code cell contains the following Python code:

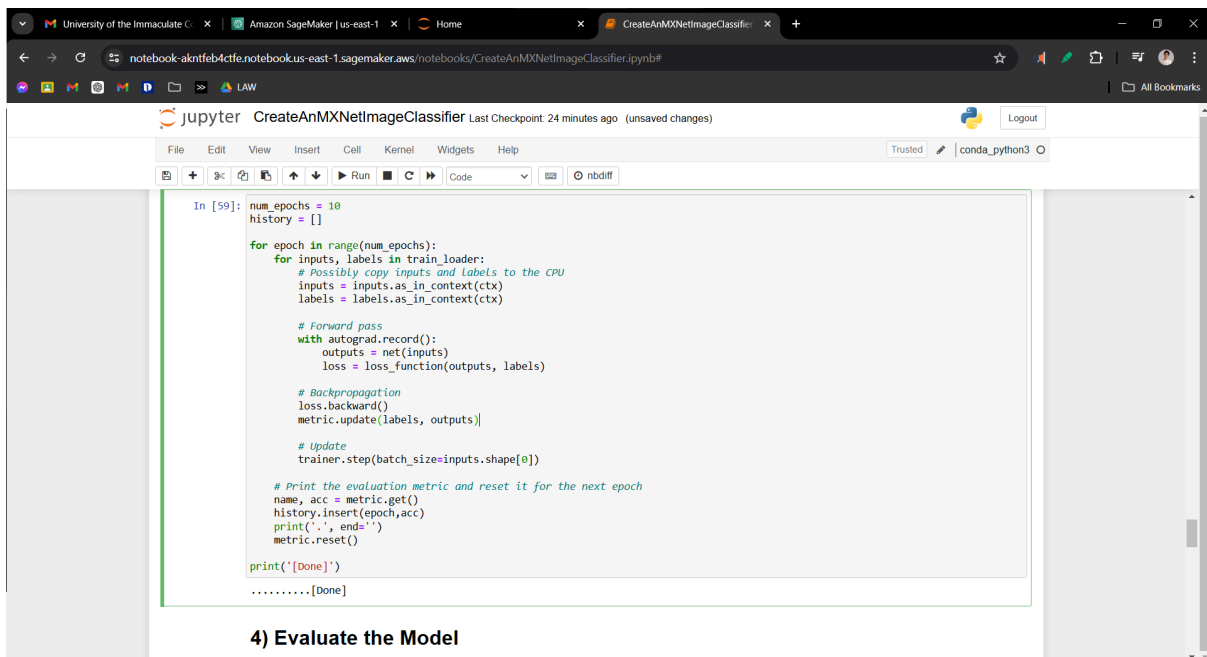
```
In [46]: # TODO: write the neural network model
from mxnet.gluon import nn

net = nn.HybridSequential(prefix='MLP_')
with net.name_scope():
    net.add(
        nn.Flatten(),
        nn.Dense(256, activation='relu'), # New additional layer
        nn.Dense(128, activation='relu'),
        nn.Dense(64, activation='relu'),
        nn.Dense(32, activation='relu'), # New additional layer
        nn.Dense(10, activation=None)
    )

# Initialize the model
net.initialize()

# Print the network architecture
print(net)

HybridSequential(
  (0): Flatten
  (1): Dense(None -> 256, Activation(relu))
  (2): Dense(None -> 128, Activation(relu))
  (3): Dense(None -> 64, Activation(relu))
  (4): Dense(None -> 32, Activation(relu))
  (5): Dense(None -> 10, linear)
)
```



The screenshot shows a Jupyter Notebook interface with the title 'CreateAnMXNetImageClassifier'. The notebook is running on a browser with the URL 'notebook-akntfeb4ctfe.notebook.us-east-1.sagemaker.aws/notebooks/CreateAnMXNetImageClassifier.ipynb#'. The notebook has a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. The 'Cell' menu is open, showing options like 'Run', 'Code', and 'nbdiff'. The notebook content includes a code cell. The code cell contains the following Python code:

```
In [59]: num_epochs = 10
history = []

for epoch in range(num_epochs):
    for inputs, labels in train_loader:
        # Possibly copy inputs and labels to the CPU
        inputs = inputs.as_in_context(ctx)
        labels = labels.as_in_context(ctx)

        # Forward pass
        with autograd.record():
            outputs = net(inputs)
            loss = loss_function(outputs, labels)

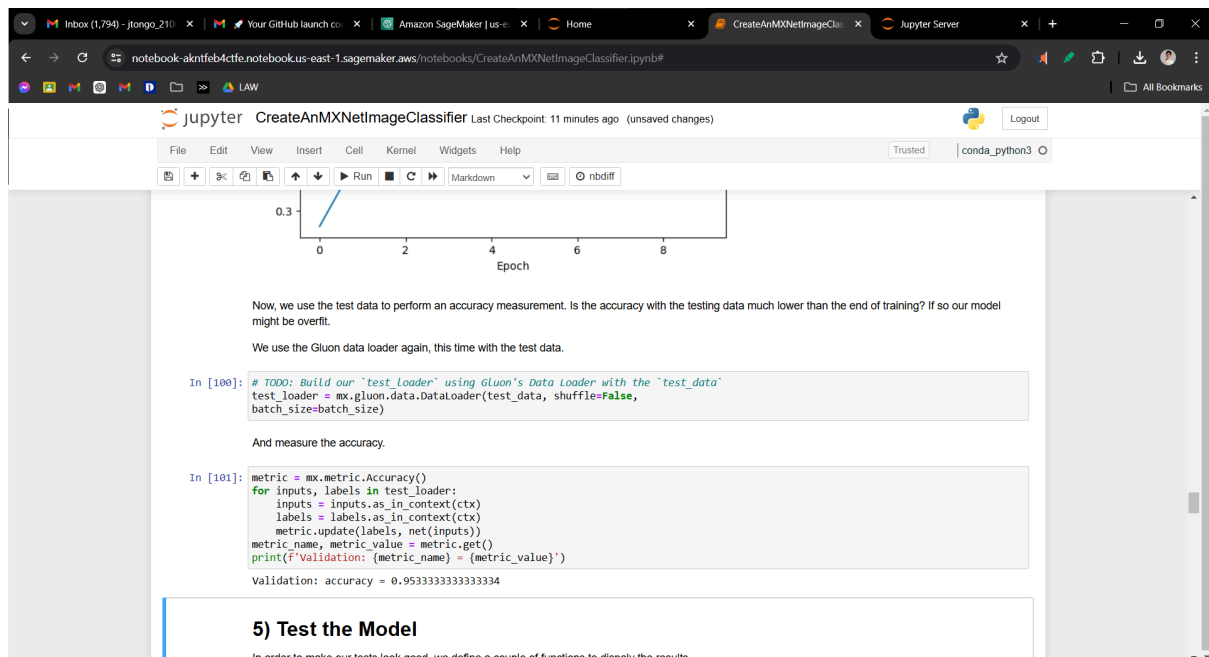
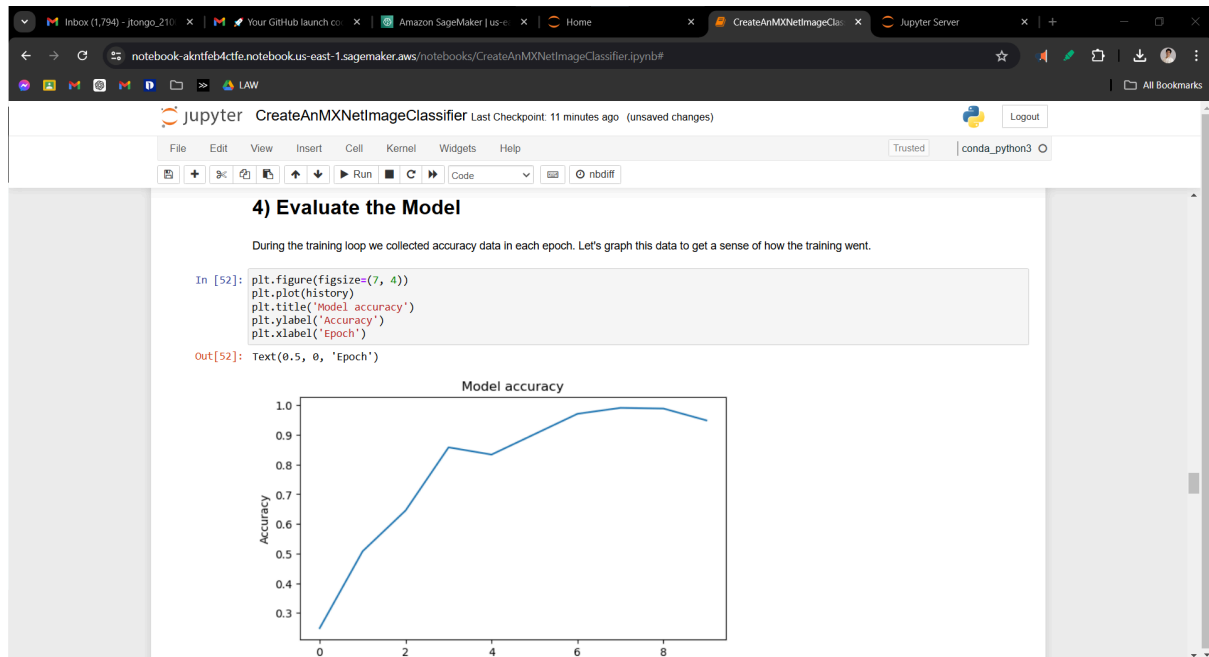
        # Backpropagation
        loss.backward()
        metric.update(labels, outputs)

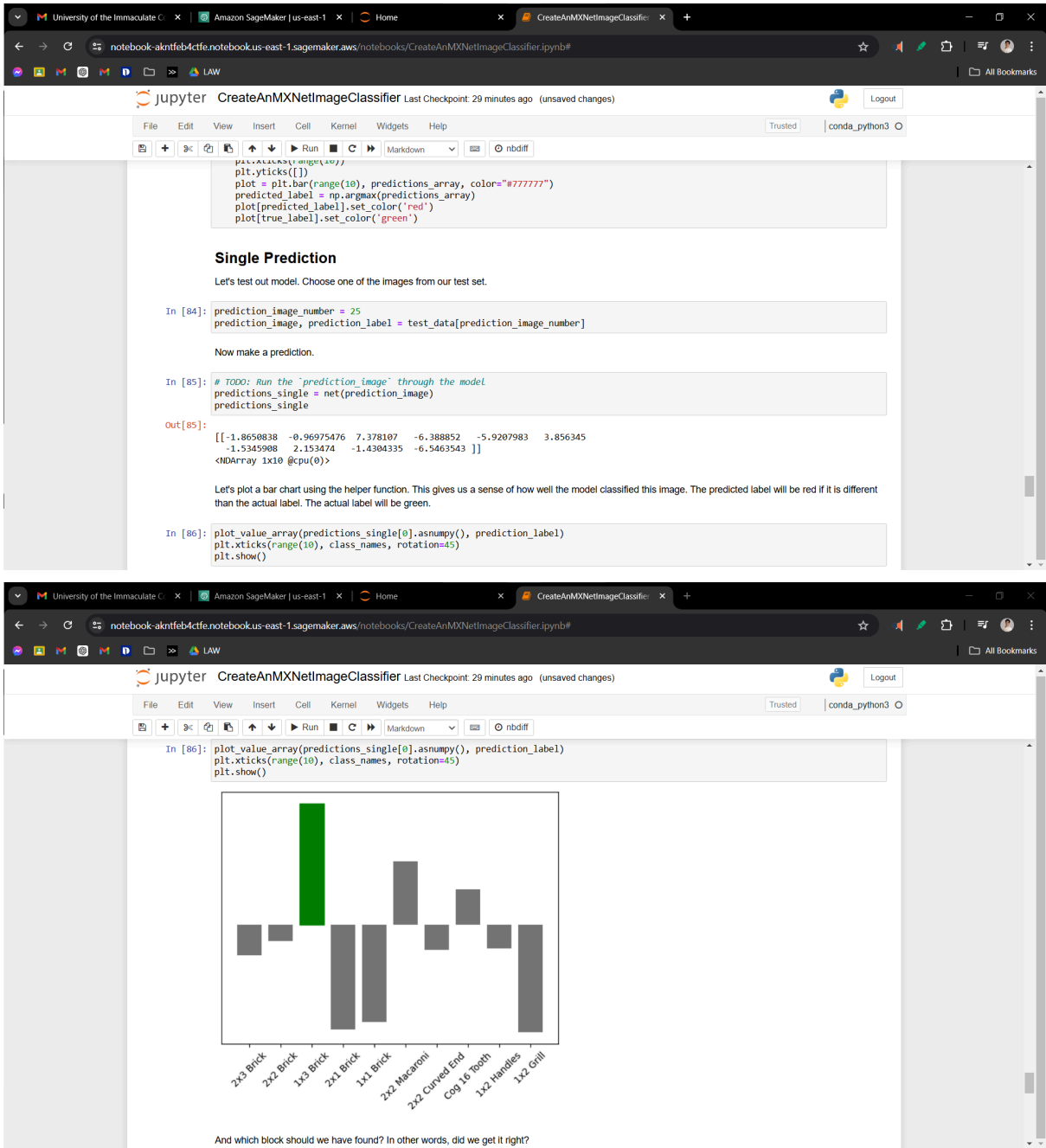
        # Update
        trainer.step(batch_size=inputs.shape[0])

    # Print the evaluation metric and reset it for the next epoch
    name, acc = metric.get()
    history.insert(epoch, acc)
    print('.', end='')
    metric.reset()

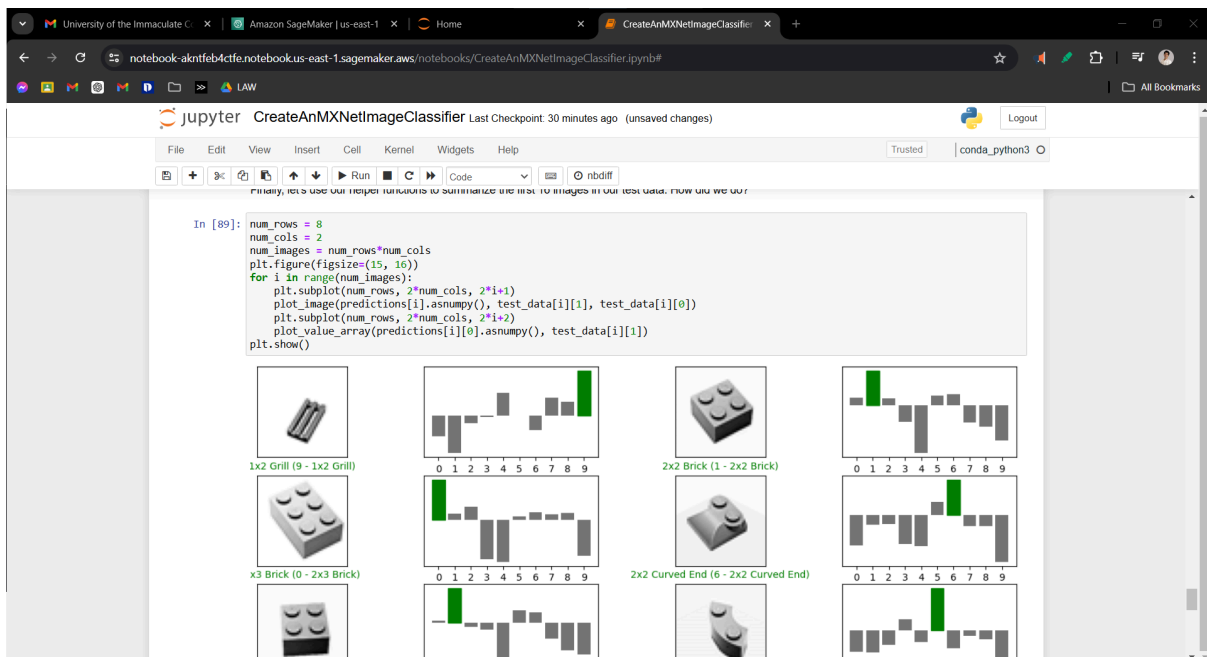
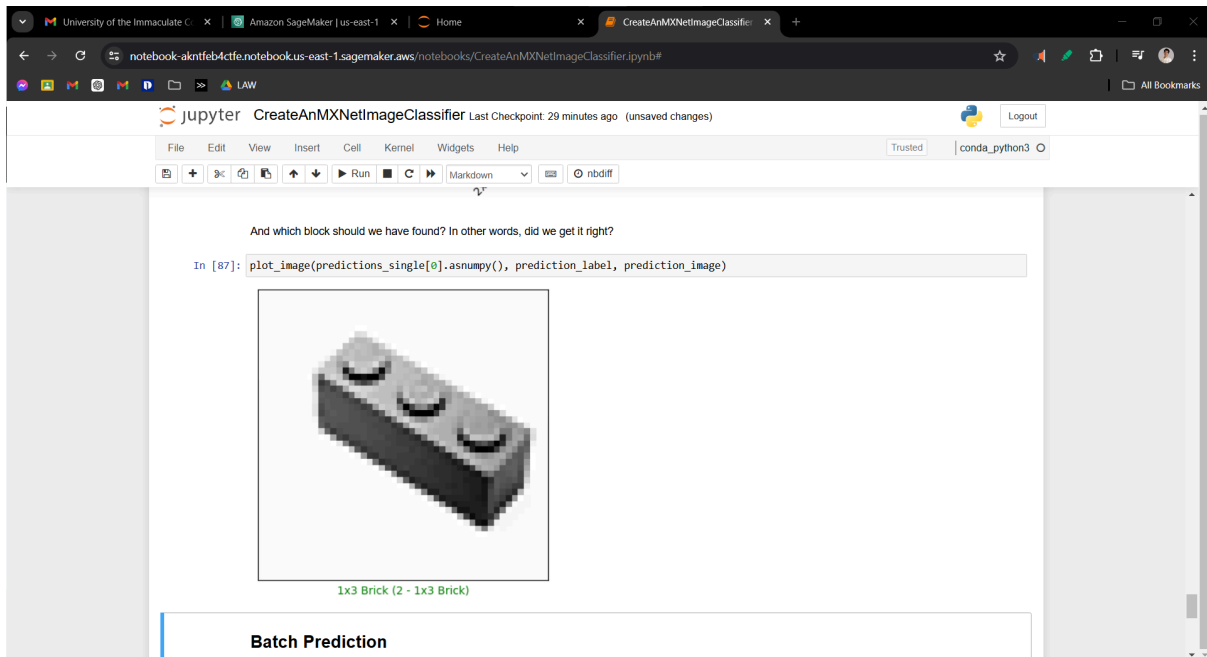
print("[Done]")
.....[Done]
```

4) Evaluate the Model





TONGO, JOHN FRANKIE A.



TONGO, JOHN FRANKIE A.

