

The image displays two sequential screenshots of a Jupyter Notebook interface titled "CreateAnMXNetImageClassifier". The notebook is running on Amazon SageMaker, as indicated by the browser tabs and the URL in the address bar.

First Screenshot:

- Section 1) Preparing our Environment:** The text explains that MXNet is not included in the default conda_python3 image, so it needs to be installed using pip.
- Code Cell [1]:** The command `%pip install mxnet` is executed. The output shows that various dependencies (numpy, requests, graphviz, charset-normalizer, idna, urllib3, certifi) are already satisfied or installed, and MXNet (1.9.1) is being installed.
- Text:** "Now that MXNet has been installed, we can import all of the modules we'll need to build and run our image classification model."
- Code Cell [5]:** The following modules are imported: `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`, and `import matplotlib.pyplot as plt`.

Second Screenshot:

- Text:** "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."
- Text:** "And of course, there's only one seed that which would be appropriate to help us figure out life, the universe, and everything"
- Code Cell [10]:** The command `mx.random.seed(42)` is executed.
- Text:** "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."
- Code Cell [11]:** The command `ctx = mx.gpu(0) if mx.context.num_gpus() > 0 else mx.cpu(0)` is executed.
- Section 2) Load the Data:** The text states: "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."
- List Item:** "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

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#

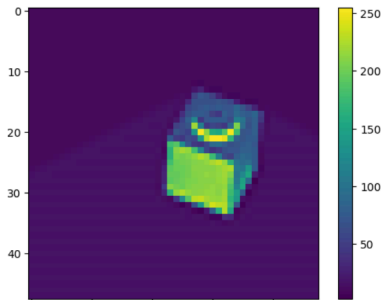
jupyter CreateAnMXNetImageClassifier Last Checkpoint: 18 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

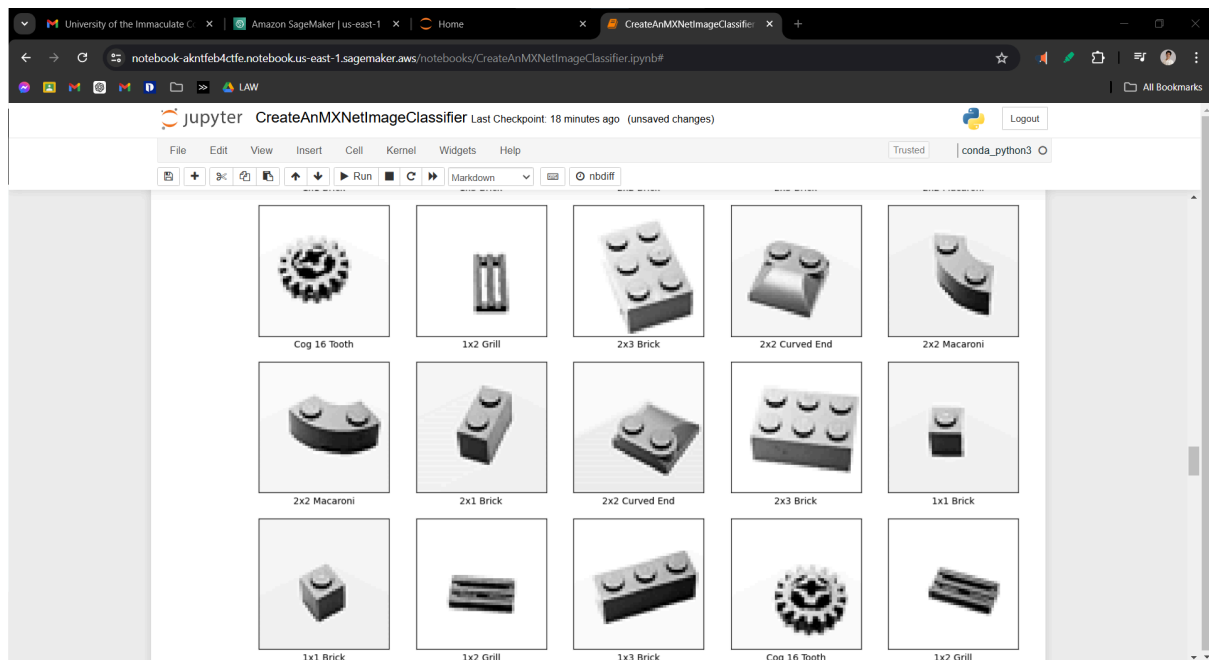
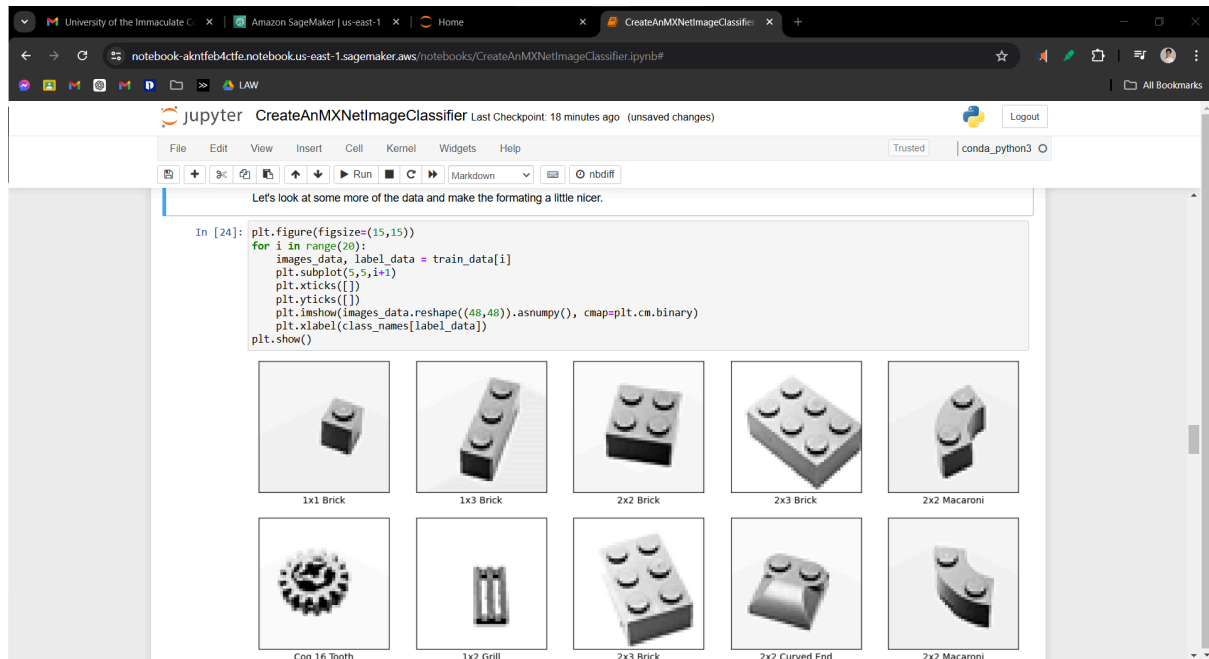
Let's take a look at one of the images loaded with the data.

```
In [25]: train_image_no = 0

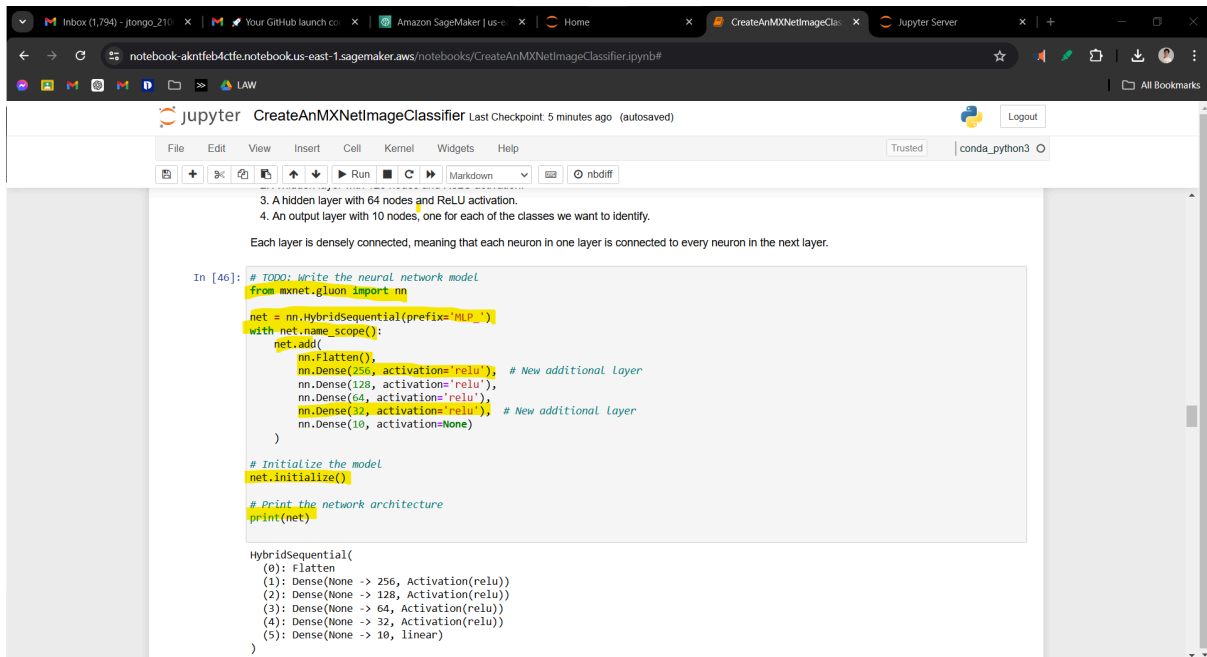
images_data, label_data = train_data[train_image_no]
plt.figure()
plt.imshow(images_data.reshape((48,48)).asnumpy())
plt.colorbar()
plt.xlabel(class_names[label_data])
plt.show()
```



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 'Run' button is highlighted. The code in the notebook is as follows:

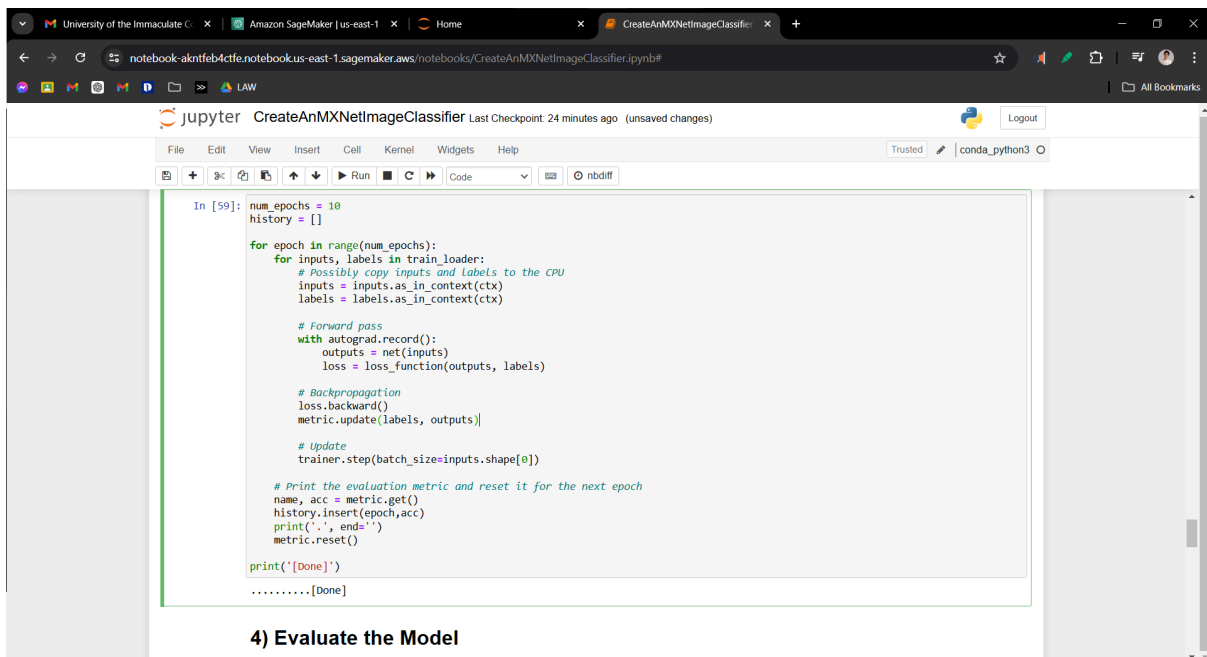
```
In [46]: # TODO: write the neural network model
from mxnet.gluon import nn

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 the same Jupyter Notebook interface, but now the code is for evaluating the model. 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 'Run' button is highlighted. The code in the notebook is as follows:

```
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)

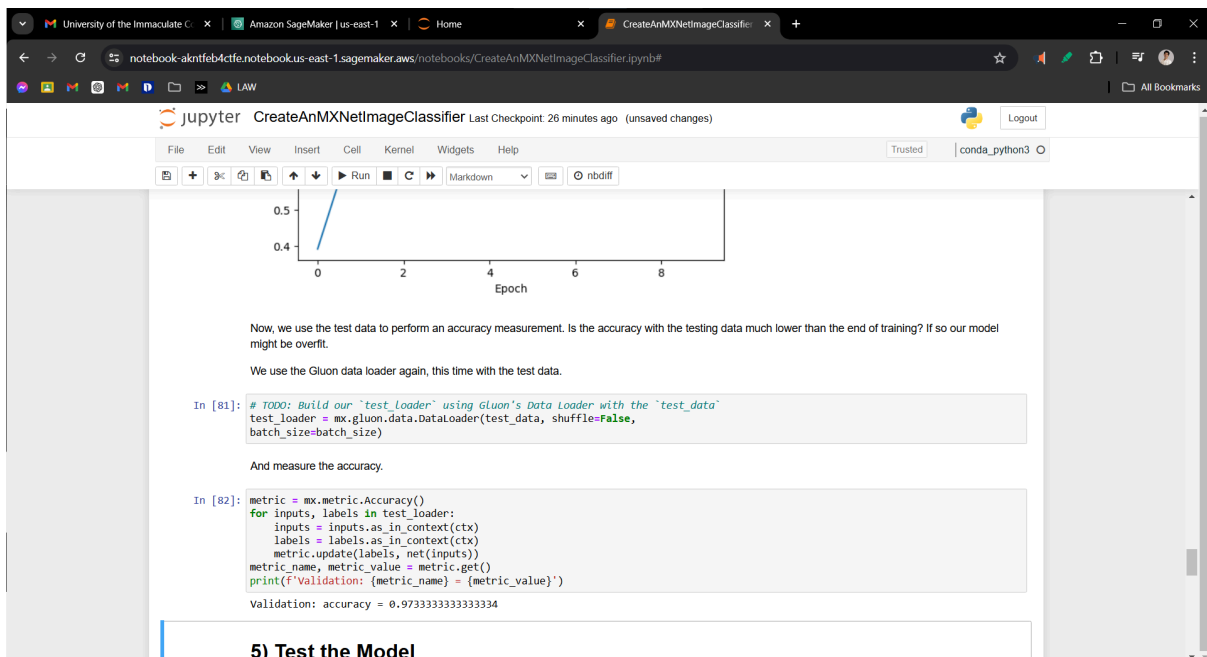
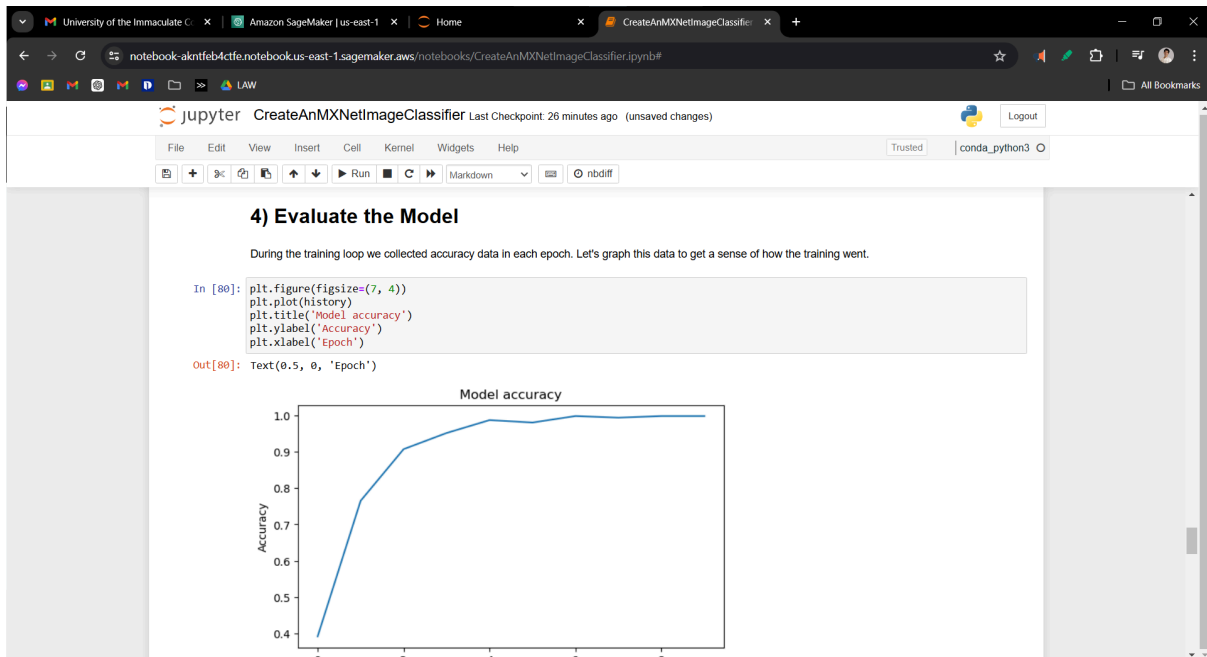
        # Backpropagation
        loss.backward()
        metric.update(labels, outputs)

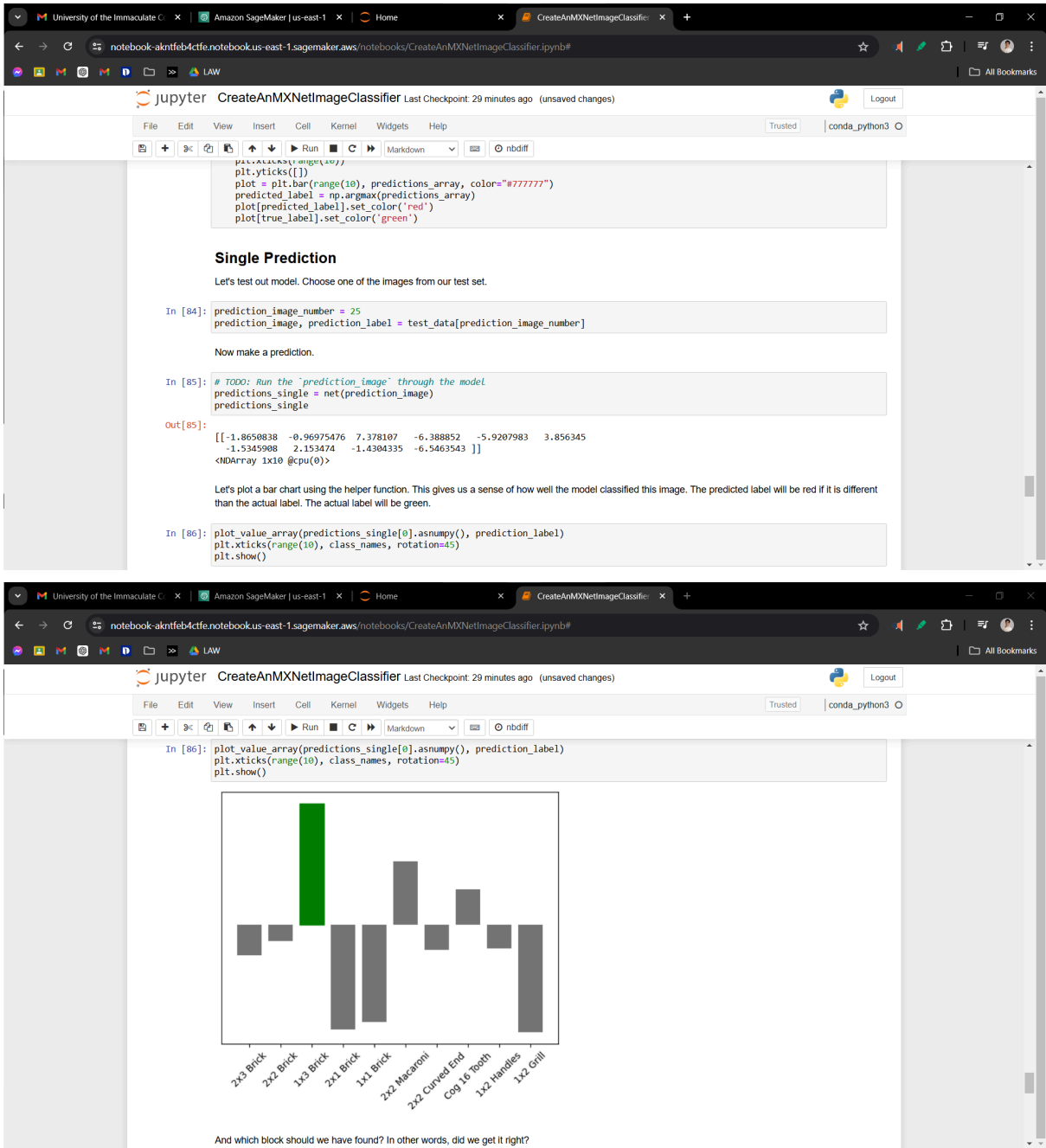
        # Update
        trainer.step(batch_size=inputs.shape[0])

    # 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.

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: 29 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

And which block should we have found? In other words, did we get it right?

```
In [87]: plot_image(predictions_single[0].asnumpy(), prediction_label, prediction_image)
```



1x3 Brick (2 - 1x3 Brick)

Batch Prediction

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






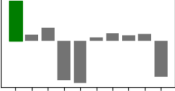

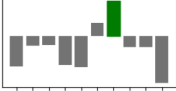


notebook-akntfeb4ctfe.notebook.us-east-1.sagemaker.aws/notebooks/CreateAnMXNetImageClassifier.ipynb#

jupyter CreateAnMXNetImageClassifier Last Checkpoint: 30 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Finally, let's use our helper functions to summarize the first 10 images in our test data. Now and we do:

```
In [89]: num_rows = 8
num_cols = 2
num_images = num_rows * num_cols
plt.figure(figsize=(15, 16))
for i in range(num_images):
    plt.subplot(num_rows, 2*num_cols, 2*i+1)
    plot_image(predictions[i].asnumpy(), test_data[i][1], test_data[i][0])
    plt.subplot(num_rows, 2*num_cols, 2*i+2)
    plot_value_array(predictions[i][0].asnumpy(), test_data[i][1])
plt.show()
```

 1x2 Grill (9 - 1x2 Grill)	 0 1 2 3 4 5 6 7 8 9	 2x2 Brick (1 - 2x2 Brick)	 0 1 2 3 4 5 6 7 8 9
 x3 Brick (0 - 2x3 Brick)	 0 1 2 3 4 5 6 7 8 9	 2x2 Curved End (6 - 2x2 Curved End)	 0 1 2 3 4 5 6 7 8 9
 3x3 Brick (0 - 2x3 Brick)	 0 1 2 3 4 5 6 7 8 9		

