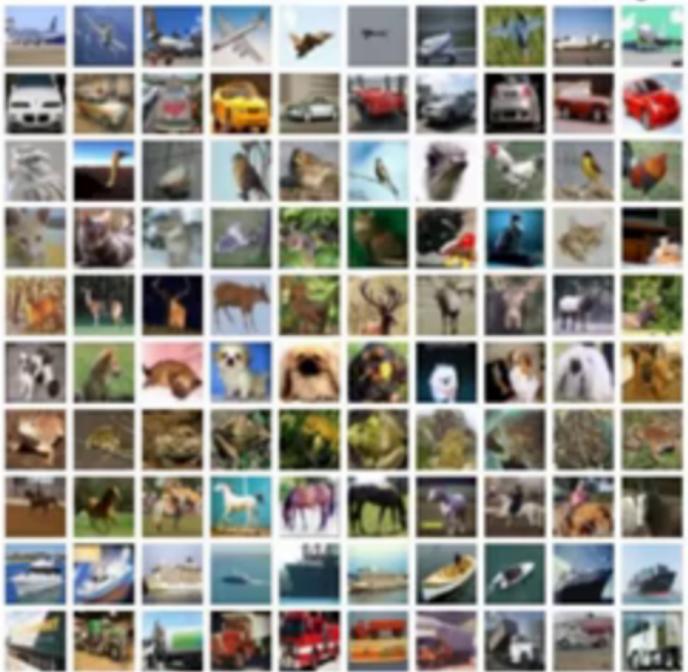






airplane



automobile

bird

cat

deer

dog

frog

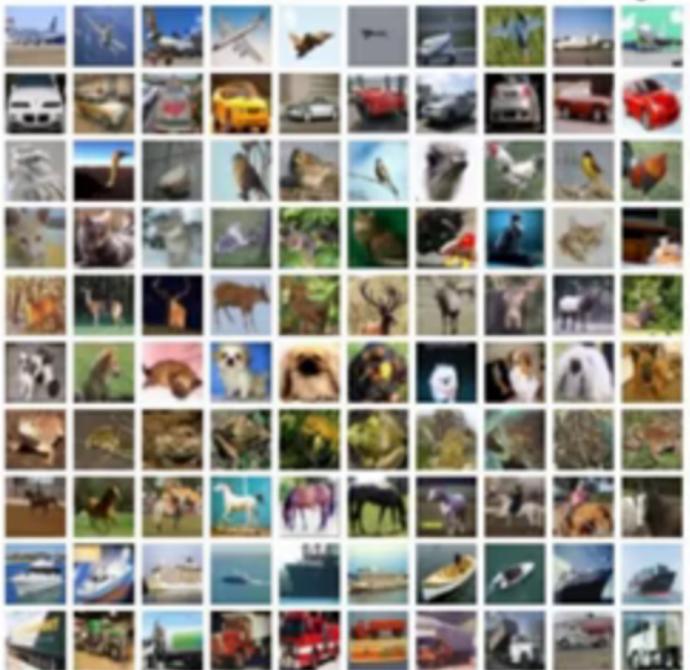
horse

ship

truck



airplane



automobile

bird

cat

deer

dog

frog

horse

ship

truck



[< Back to Alex Krizhevsky's home page](#)

The CIFAR-10 and CIFAR-100 are labeled subsets of the [80 million tiny images](#) dataset. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

The CIFAR-10 dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:



The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Airplane" includes seaplanes. So I've thrown in that one. "Truck" includes

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:



The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

Download

If you're going to use this dataset, please cite the tech report at the bottom of this page.

Version

Size

md5sum

Version	Size	md5sum
1.0	~17.5 GB	~e6433a3a3d0a2a3a3a3a3a3a3a3a3a3a

Type here to search



CRM-10 and CRM-100 dataset... Untitled - Jupyter Notebook 16_cifar10_small_image_classification/Untitled.ipynb Kernel name: python3

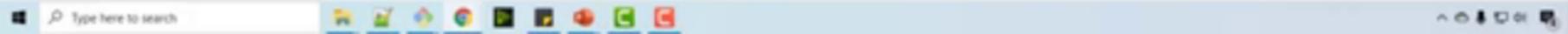
jupyter Untitled Last Checkpoint: 7 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [1]: `import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np`

In []: `datasets.cifar10.load_data()`

The screenshot shows a Jupyter Notebook window with two code cells. The first cell contains imports for TensorFlow, Keras, Matplotlib, and NumPy. The second cell contains the command to load the CIFAR-10 dataset. A toolbar at the top includes File, Edit, View, Insert, Cell, Kernel, Help, Trusted, and Python 3. Below the toolbar are standard notebook controls for cell selection, execution, and modification.



CRM-10 and CRM-100 dataset X Untitled - Jupyter Notebook X 18_cnn_cifar10_small_image_classification/Untitled.ipynb Kernel name: python3

localhost:8888/notebooks/18_cnn_cifar10_small_image_classification/Untitled.ipynb?kernel_name=python3

jupyter Untitled Last Checkpoint: 7 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help Trusted Python 3 Logout

In [1]:

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np
```

In []:

```
[datasets.cifar10.load_data()]
```



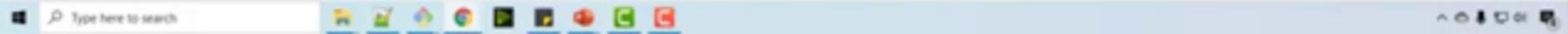
CRM-10 and CRM-100 dataset CRM-10 Untitled - Jupyter Notebook CRM-100 18_cnn_cifar10_small_image_classification/Untitled.ipynb Kernel name: python3

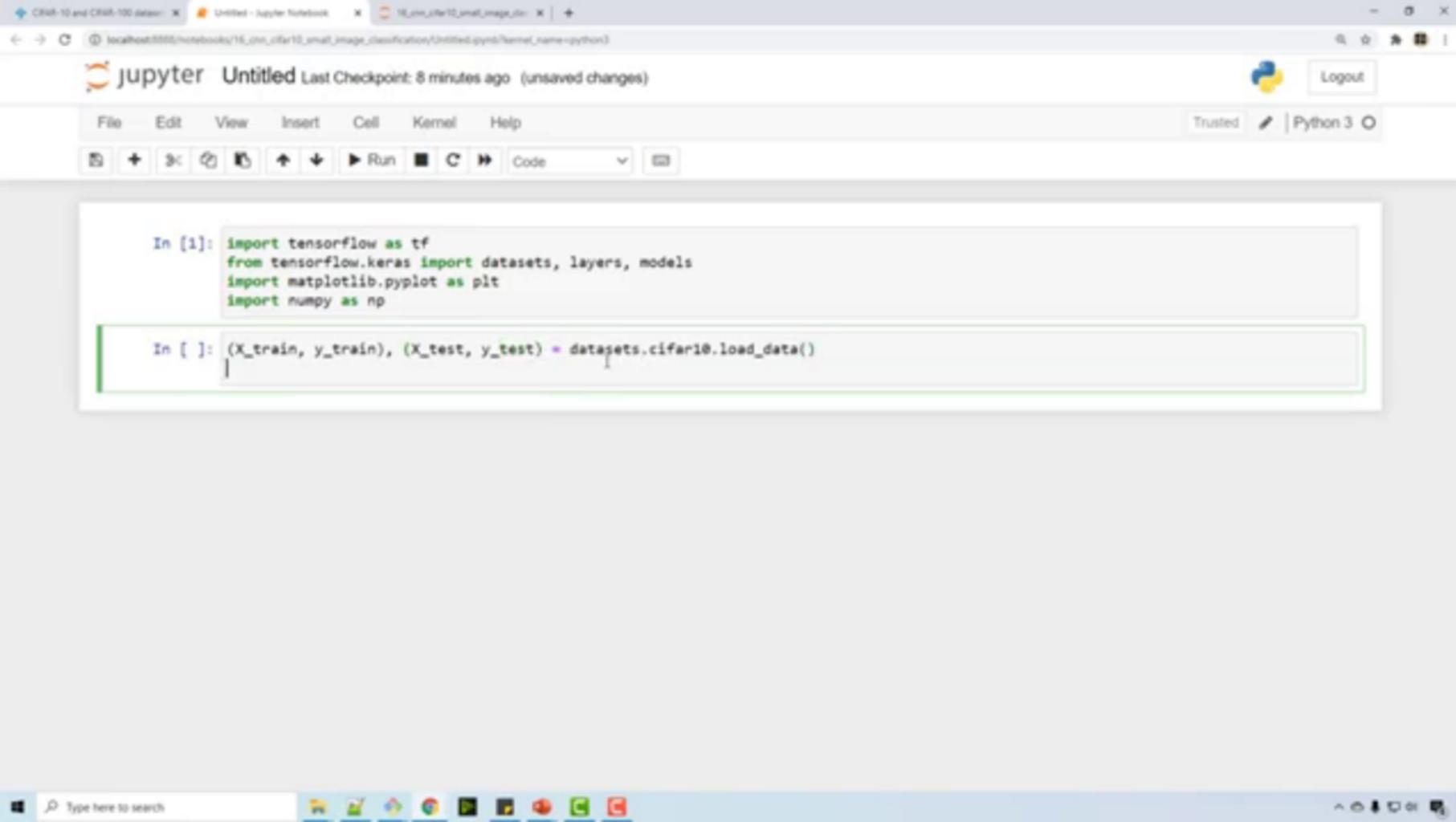
jupyter Untitled Last Checkpoint: 8 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [1]: `import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np`

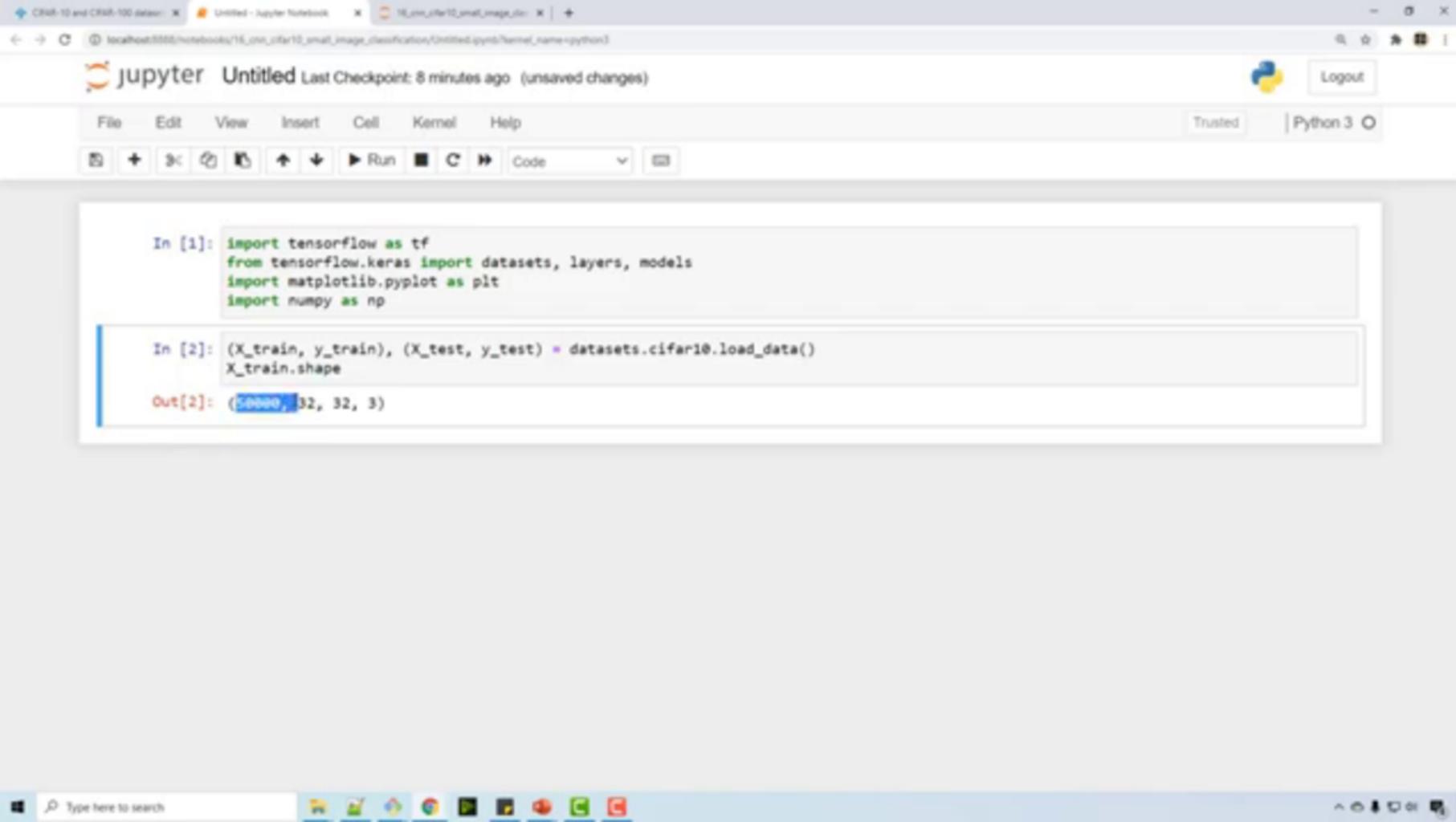
In []: `(X_train, y_train) = datasets.cifar10.load_data()`

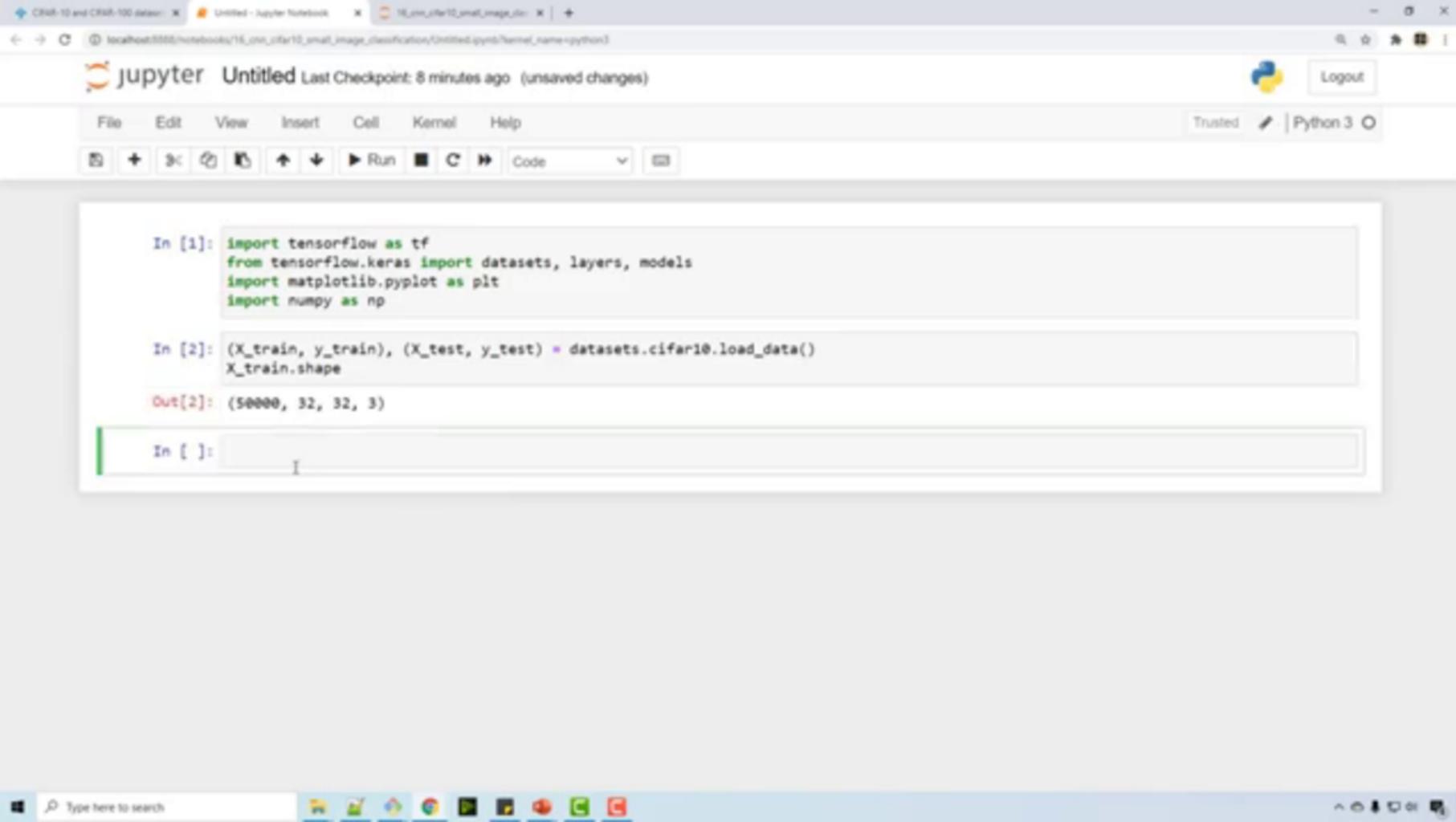


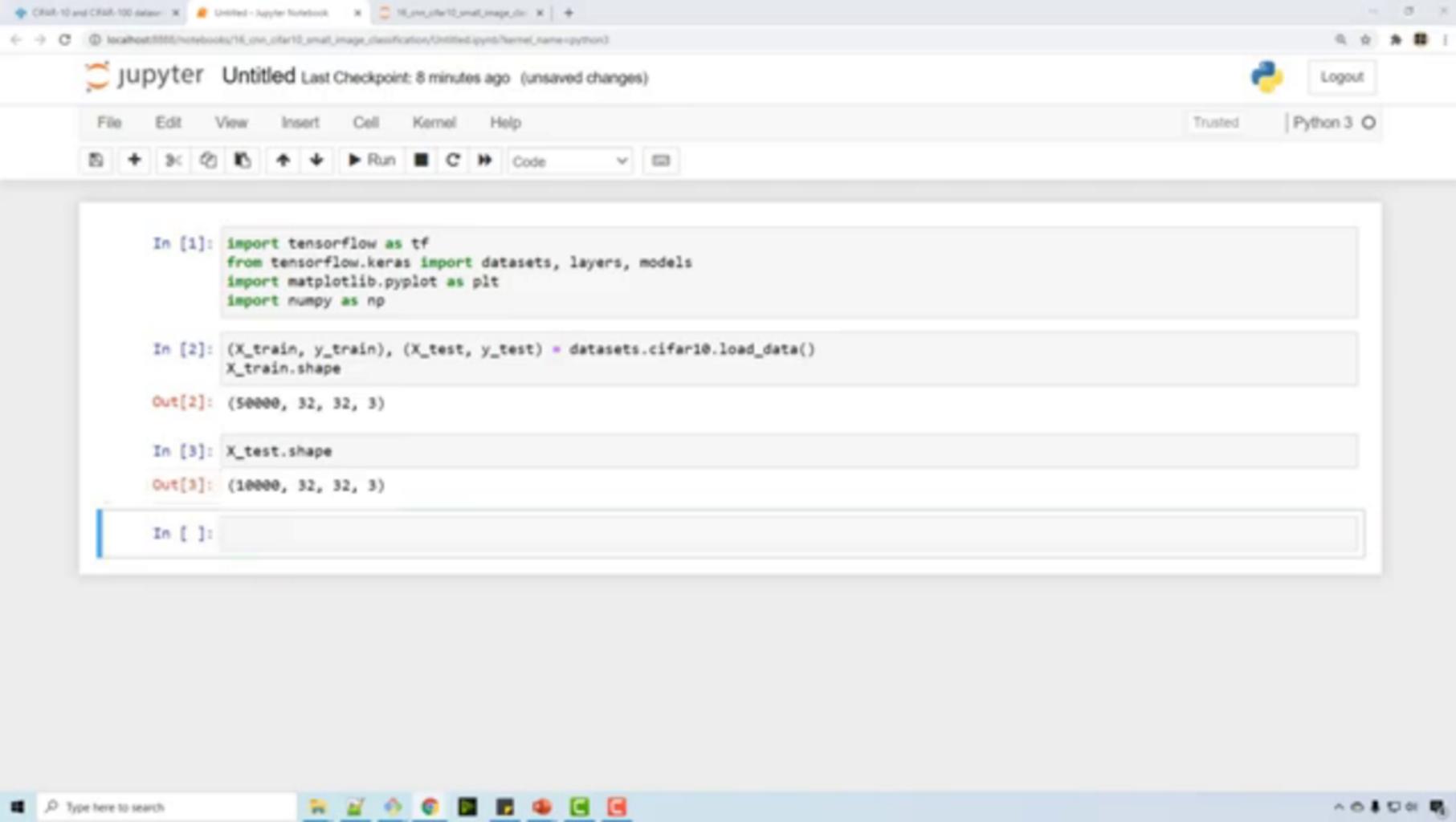


```
In [1]: import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
import numpy as np
```

```
In [ ]: (X_train, y_train), (X_test, y_test) = datasets.cifar10.load_data()
```







```
In [1]: import tensorflow as tf  
from tensorflow.keras import datasets, layers, models  
import matplotlib.pyplot as plt  
import numpy as np
```

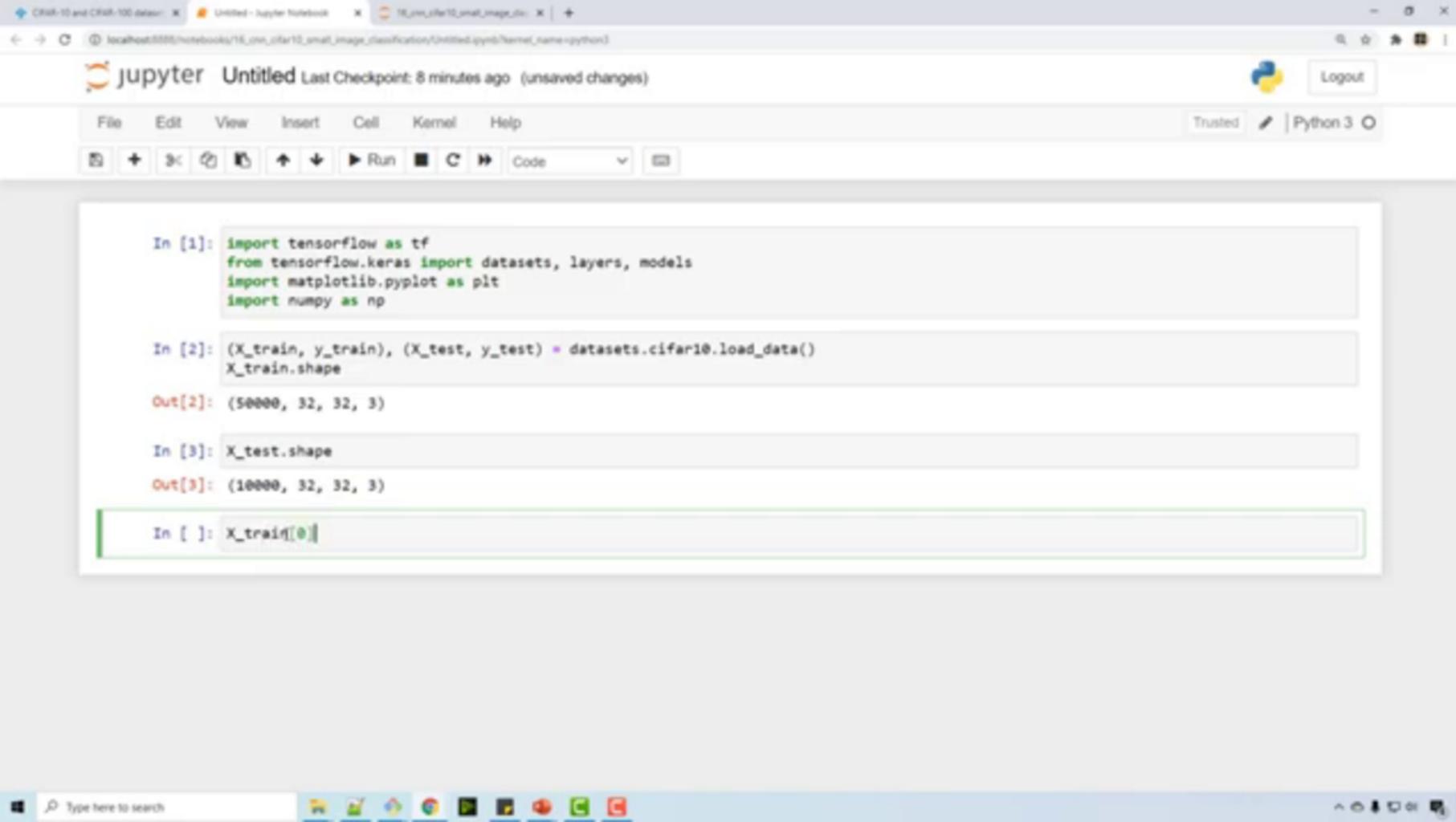
```
In [2]: (X_train, y_train), (X_test, y_test) = datasets.cifar10.load_data()  
X_train.shape
```

```
Out[2]: (50000, 32, 32, 3)
```

```
In [3]: X_test.shape
```

```
Out[3]: (10000, 32, 32, 3)
```

```
In [ ]:
```



CRM-10 and CRM-100 dataset CRM-10 Untitled - Jupyter Notebook CRM-100 18_cnn_cifar10_small_image_classification/Untitled.ipynb Kernel name: python3

jupyter Untitled Last Checkpoint: 9 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [2]: `(X_train, y_train), (X_test, y_test) = datasets.cifar10.load_data()
X_train.shape`

Out[2]: `(50000, 32, 32, 3)`

In [3]: `X_test.shape`

Out[3]: `(10000, 32, 32, 3)`

In [4]: `X_train[0]`

Out[4]: `array([[[59, 62, 63],
 [43, 46, 45],
 [50, 48, 43],
 ...,
 [158, 132, 108],
 [152, 125, 102],
 [148, 124, 103]],

 [[16, 20, 20],
 [0, 0, 0],
 [18, 8, 0],
 ...,
 [123, 88, 55],
 [119, 83, 50],
 [122, 87, 57]],

 [[25, 24, 21],
 [16, 7, 0],
 [49, 27, 8],
 ...])`

Type here to search

CRM-10 and CRM-100 dataset CRM-10 Untitled - Jupyter Notebook CRM-100 18_cnn_cifar10_small_image_classification/Untitled.ipynb Kernel name: python3

jupyter Untitled Last Checkpoint: 9 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [2]: `(X_train, y_train), (X_test, y_test) = datasets.cifar10.load_data()`
X_train.shape

Out[2]: `(50000, 32, 32, 3)`

In [3]: `X_test.shape`

Out[3]: `(10000, 32, 32, 3)`

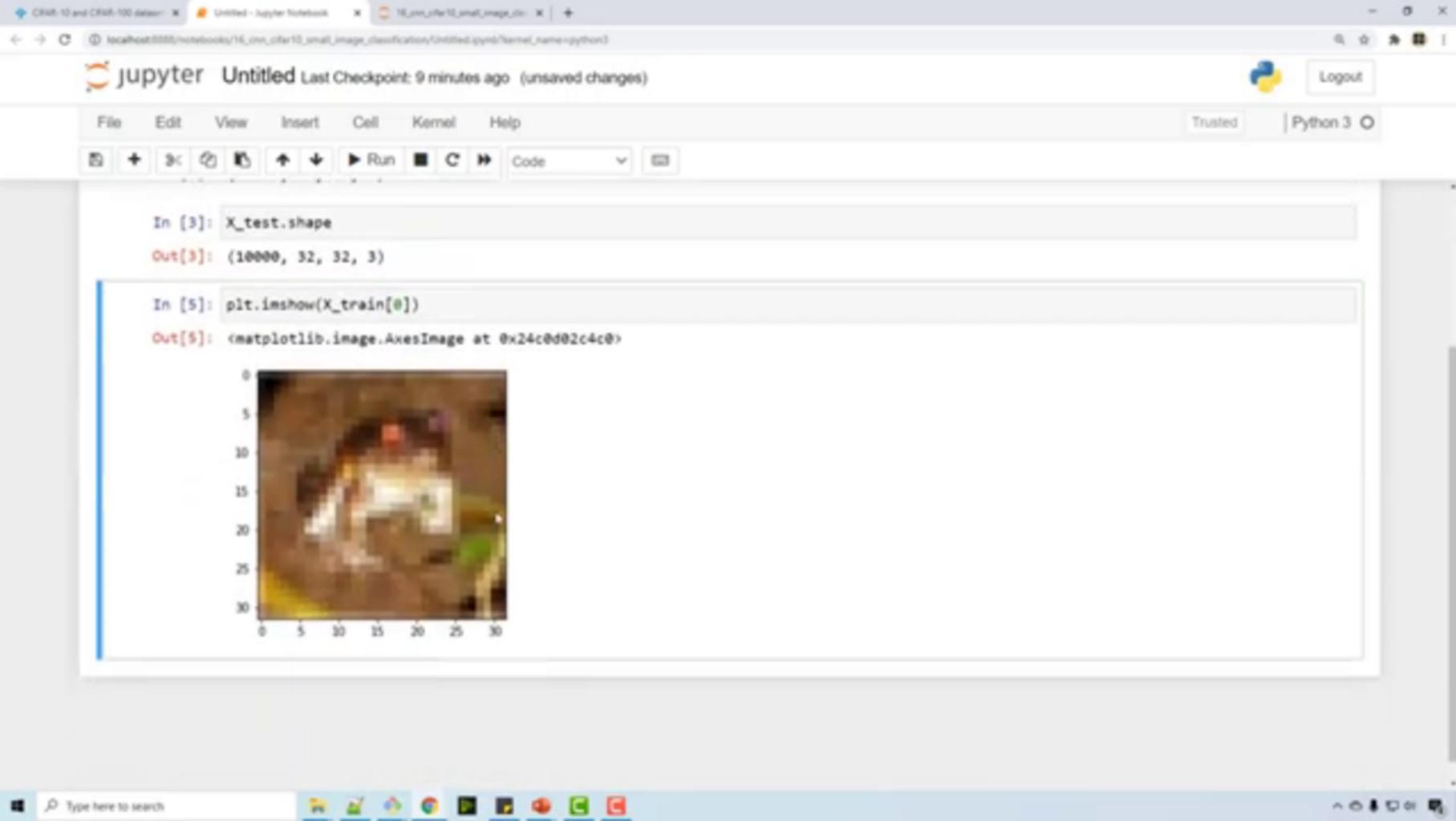
In [4]: `plt.X_train[0]`

Out[4]: `array([[[59, 62, 63],
 [43, 46, 45],
 [50, 48, 43],
 ...,
 [158, 132, 108],
 [152, 125, 102],
 [148, 124, 103]],

 [[16, 20, 20],
 [0, 0, 0],
 [18, 8, 0],
 ...,
 [123, 88, 55],
 [119, 83, 50],
 [122, 87, 57]],

 [[25, 24, 21],
 [16, 7, 0],
 [49, 27, 8],
 ...])`

Type here to search

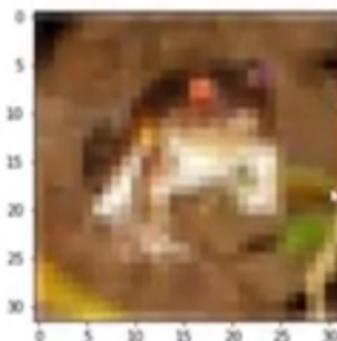


```
In [3]: X_test.shape
```

```
Out[3]: (10000, 32, 32, 3)
```

```
In [5]: plt.imshow(X_train[0])
```

```
Out[5]: <matplotlib.image.AxesImage at 0x24c0d02c4c0>
```



jupyter Untitled Last Checkpoint: 9 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3

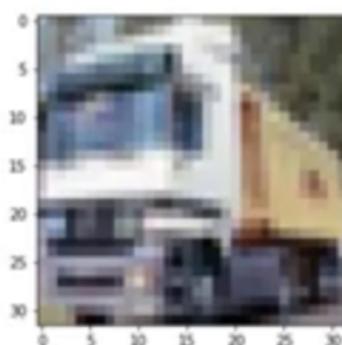


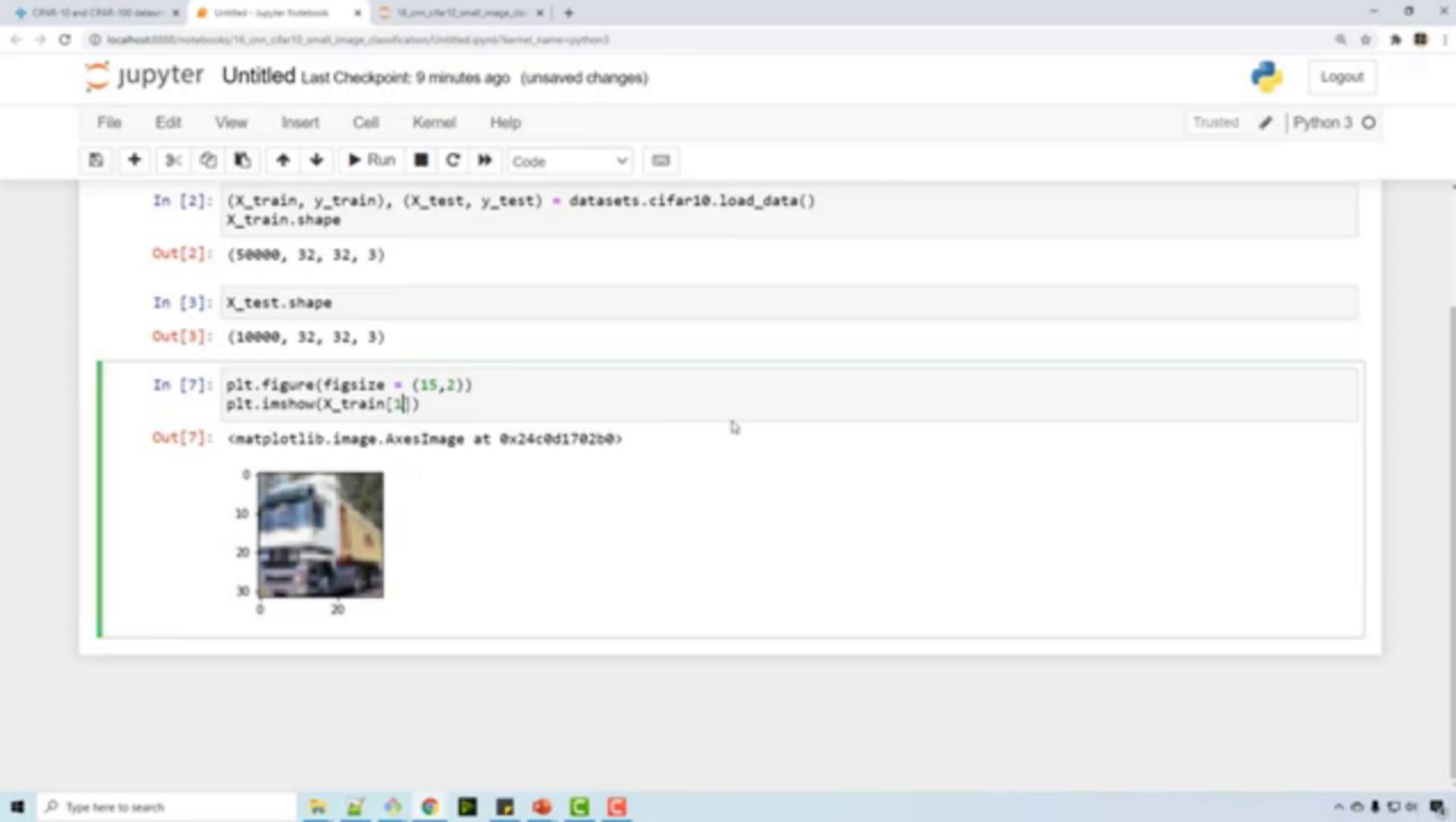
In [3]: X_test.shape

Out[3]: (10000, 32, 32, 3)

In [6]: plt.imshow(X_train[1])

Out[6]: <matplotlib.image.AxesImage at 0x24c0d109730>





jupyter Untitled Last Checkpoint: 9 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3



```
In [2]: (X_train, y_train), (X_test, y_test) = datasets.cifar10.load_data()
X_train.shape
```

```
Out[2]: (50000, 32, 32, 3)
```

```
In [3]: X_test.shape
```

```
Out[3]: (10000, 32, 32, 3)
```

```
In [7]: plt.figure(figsize = (15,2))
plt.imshow(X_train[1])
```

```
Out[7]: <matplotlib.image.AxesImage at 0x24c0d1702b0>
```



Type here to search





Logout

jupyter Untitled Last Checkpoint: 9 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help

Trusted Python 3



```
In [2]: (X_train, y_train), (X_test, y_test) = datasets.cifar10.load_data()
X_train.shape
```

```
Out[2]: (50000, 32, 32, 3)
```

```
In [3]: X_test.shape
```

```
Out[3]: (10000, 32, 32, 3)
```

```
In [8]: %def plot_sample(x, y, index):
    plt.figure(figsize=(15,2))
    plt.imshow(x[index])
```

```
Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>
```



jupyter Untitled



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3



```
In [2]: (X_train, y_train), (X_test, y_test) = datasets.cifar10.load_data()
X_train.shape
```

```
Out[2]: (50000, 32, 32, 3)
```

```
In [3]: X_test.shape
```

```
Out[3]: (10000, 32, 32, 3)
```

```
In [8]: def plot_sample(X, y, index):
    plt.figure(figsize=(15,2))
    plt.imshow(X[index])
```

```
Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>
```



[< Back to Alex Krizhevsky's home page](#)

The CIFAR-10 and CIFAR-100 are labeled subsets of the [80 million tiny images](#) dataset. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

The CIFAR-10 dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

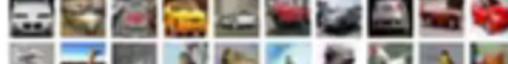
The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:

airplane



automobile



bird



cat



deer



dog



frog



horse



ship



truck



The classes are nonmutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes vans. So we think none of that count. "Truck" includes

Type here to search



Logout

jupyter Untitled Last Checkpoint: 10 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help

Trusted | Python 3



```
In [2]: (X_train, y_train), (X_test, y_test) = datasets.cifar10.load_data()
X_train.shape
```

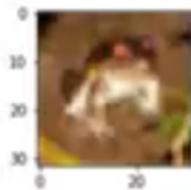
```
Out[2]: (50000, 32, 32, 3)
```

```
In [3]: X_test.shape
```

```
Out[3]: (10000, 32, 32, 3)
```

```
In [8]: def plot_sample(X, y, index):
    plt.figure(figsize = (15,2))
    plt.imshow(X[index])
    plt.xlabel('')
```

```
Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>
```



 jupyter Untitled Last Checkpoint: 10 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help

Trusted Python 3



Out[2]: (50000, 32, 32, 3)

In [3]: X_test.shape

Out[3]: (10000, 32, 32, 3)

In []: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

In [8]: def plot_sample(x, y, index):
 plt.figure(figsize=(15, 2))
 plt.imshow(x[index])
 plt.xlabel()

Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>



 jupyter Untitled Last Checkpoint: 11 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help

Trusted Python 3



Out[2]: (50000, 32, 32, 3)

In [3]: X_test.shape

Out[3]: (10000, 32, 32, 3)

In [9]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

In [8]: def plot_sample(x, y, index):
 plt.figure(figsize=(15, 2))
 plt.imshow(x[index])
 plt.xlabel(classes[y])

Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>



jupyter Untitled Last Checkpoint: 11 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3



In [3]: X_test.shape

Out[3]: (10000, 32, 32, 3)

In []:

In [9]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

In [8]: def plot_sample(x, y, index):
 plt.figure(figsize=(15, 2))
 plt.imshow(x[index])
 plt.xlabel(classes[y])

Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>



CRM-10 and CRM-100 dataset CRM-10 Untitled - Jupyter Notebook CRM-100 16_mnist cifar10_smali_image_classification/Untitled.ipynb Kernel Name: python3

jupyter Untitled Last Checkpoint: 11 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

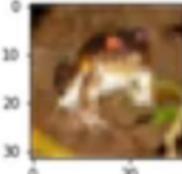
In [3]: X_test.shape
Out[3]: (10000, 32, 32, 3)

In [10]: y_train.shape
Out[10]: (50000, 1)

In [9]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

In [8]: def plot_sample(X, y, index):
 plt.figure(figsize=(15, 2))
 plt.imshow(X[index])
 plt.xlabel(classes[index])

Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>



< Back to Alex Krizhevsky's home page

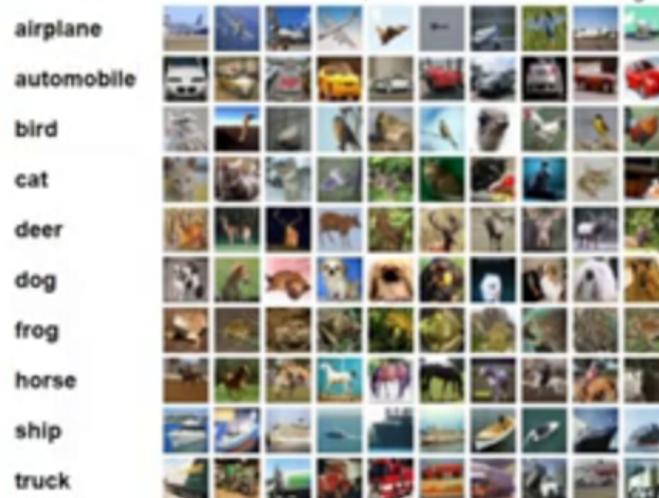
The CIFAR-10 and CIFAR-100 are labeled subsets of the [80 million tiny images](#) dataset. They were collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

The CIFAR-10 dataset

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:



The classes are completely mutually exclusive. There is no overlap between airplanes and trucks. "Antennahila" includes earlars. So I've thrown that out. "Truck" includes

Type here to search



CRM-10 and CRM-100 dataset Untitled - Jupyter Notebook 18_imagenet2_imat_image_classification/Untitled.ipynb (kernel_name:python3)

jupyter Untitled Last Checkpoint: 11 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [3]: X_test.shape

Out[3]: (10000, 32, 32, 3)

In [11]: y_train[:5]

Out[11]: array([[6],
[3],
[9],
[4],
[1]], dtype=uint8)

In [9]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

In [8]: def plot_sample(X, y, index):
 plt.figure(figsize = (15,2))
 plt.imshow(X[index])
 plt.xlabel(classes[y])

Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>



Type here to search Back Home Stop Refresh

CRM-10 and CRM-100 dataset CRM-10 Untitled - Jupyter Notebook CRM-100 18_cnn_cifar10_small_image_classification/Untitled.ipynb Kernel Name: python3

jupyter Untitled Last Checkpoint: 11 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [3]: X_test.shape

Out[3]: (10000, 32, 32, 3)

In [11]: y_train[:5]

Out[11]: array([1, 9, 9, 9, 4], dtype=uint8)

In [9]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

In [8]: def plot_sample(X, y, index):
 plt.figure(figsize=(15, 2))
 plt.imshow(X[index])
 plt.xlabel(classes[y])

Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>



Type here to search

CRM-10 and CRM-100 dataset x Untitled - Jupyter Notebook x 18_mnist_cifar10_smali_image_classification/Untitled.ipynb Kernel name: python3

jupyter Untitled Last Checkpoint: 12 minutes ago (autosaved) Logout

File Edit View Insert Cell Kernel Help

Notebook saved Trusted Python 3

In [3]: X_test.shape

Out[3]: (10000, 32, 32, 3)

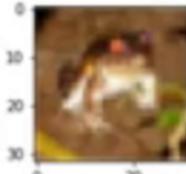
In [11]: y_train[:5]

Out[11]: array([6, 9, 9, 4, 1], dtype=uint8)

In [9]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

In [8]: def plot_sample(X, y, index):
 plt.figure(figsize=(15,2))
 plt.imshow(X[index])
 plt.xlabel(classes[y])

Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>



Type here to search

CRM-10 and CRM-100 dataset CRM-10 Untitled - Jupyter Notebook CRM-100 18_cnn_cifar10_small_image_classification/Untitled.ipynb Kernel Name: python3

jupyter Untitled Last Checkpoint: 12 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [3]: X_test.shape

Out[3]: (10000, 32, 32, 3)

In [11]: y_train[:5]

Out[11]: array([[6],
[9],
[9],
[4],
[1]], dtype=uint8)

In []: y_train = y_train.s

In [9]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

In [8]: def plot_sample(x, y, index):
 plt.figure(figsize=(15, 2))
 plt.imshow(x[index])
 plt.xlabel(classes[y])

Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>



Type here to search

CRM-10 and CRM-100 dataset * Untitled - Jupyter Notebook * 16_imagenet10_small_image_classification/Untitled.ipynb Kernel name: python3

jupyter Untitled Last Checkpoint: 12 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted | Python 3

In [3]: X_test.shape

Out[3]: (10000, 32, 32, 3)

In [11]: y_train[:5]

Out[11]: array([[6],
[9],
[9],
[4],
[1]], dtype=uint8)

In []: y_train = y_train.reshape()

In [9]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

In [8]: def plot_sample(x, y, index):
 plt.figure(figsize=(15, 2))
 plt.imshow(x[index])
 plt.xlabel(classes[y])

Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>

0
10
20



Type here to search

CRM-10 and CRM-100 dataset CRM-10 Untitled - Jupyter Notebook CRM-100 16_imagenet10_small_image_classification/Untitled.ipynb Kernel Name: python3

jupyter Untitled Last Checkpoint: 12 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [3]: X_test.shape

Out[3]: (10000, 32, 32, 3)

In [11]: y_train[:5]

Out[11]: array([0, 9, 9, 4, 1], dtype=uint8)

In []: y_train = y_train.reshape(-1,)

In [9]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

In [8]: def plot_sample(x, y, index):
 plt.figure(figsize=(15, 2))
 plt.imshow(x[index])
 plt.xlabel(classes[y])

Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>



Type here to search CRM-10 CRM-100

CRM-10 and CRM-100 dataset x Untitled - Jupyter Notebook x 16_mnist cifar10_smali_image_classification/Untitled.ipynb x kernel_name:python3

jupyter Untitled Last Checkpoint: 12 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [3]: X_test.shape

Out[3]: (10000, 32, 32, 3)

In [11]: y_train[:5]

Out[11]: array([6, 9, 9, 4, 1], dtype=uint8)

In []: y_train = y_train.reshape(-1,)

In [9]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

In [8]: def plot_sample(X, y, index):
 plt.figure(figsize = (15,2))
 plt.imshow(X[index])
 plt.xlabel(classes[y])

Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>



Type here to search

CRM-10 and CRM-100 dataset CRM-10 Untitled - Jupyter Notebook CRM-100 16_cnn_cifar10_small_image_classification/Untitled.ipynb Kernel Name: python3

jupyter Untitled Last Checkpoint: 12 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [3]: X_test.shape
Out[3]: (10000, 32, 32, 3)

In [11]: y_train[:5]
Out[11]: array([6, 9, 9, 4, 1], dtype=uint8)

In [13]: y_train = y_train.reshape(-1,)
y_train[:5]
Out[13]: array([6, 9, 9, 4, 1], dtype=uint8)

In [9]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

In [8]: def plot_sample(X, y, index):
 plt.figure(figsize=(15, 2))
 plt.imshow(X[index])
 plt.xlabel(classes[y])
Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>

0 
10

Type here to search

CRM-10 and CRM-100 dataset CRM-10 Untitled - Jupyter Notebook CRM-100 16_mnist_cifar10_small_image_classification/Untitled.ipynb Kernel Name: python3

jupyter Untitled Last Checkpoint: 13 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Out[11]: array([[6,
 [9],
 [9],
 [4],
 [1]], dtype=uint8)

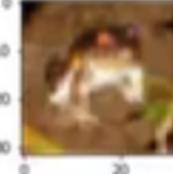
In [13]: y_train = y_train.reshape(-1,)
y_train[:5]

Out[13]: array([6, 9, 9, 4, 1], dtype=uint8)

In [9]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

In [8]: def plot_sample(X, y, index):
 plt.figure(figsize=(15,2))
 plt.imshow(X[index])
 plt.xlabel(classes[index])

Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>



Type here to search

CRM-10 and CRM-100 dataset CRM-10 Untitled - Jupyter Notebook CRM-100 16_mns_cifar10_small_image_classification/Untitled.ipynb Kernel name: python3

jupyter Untitled Last Checkpoint: 13 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Out[11]: array([[6,
 [9],
 [9],
 [4],
 [1]], dtype=uint8)

In [13]: y_train = y_train.reshape(-1,)
y_train[:5]

Out[13]: array([6, 9, 9, 4, 1], dtype=uint8)

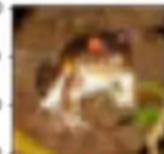
In [9]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

In [14]: classes[9] |
Out[14]: 'truck'

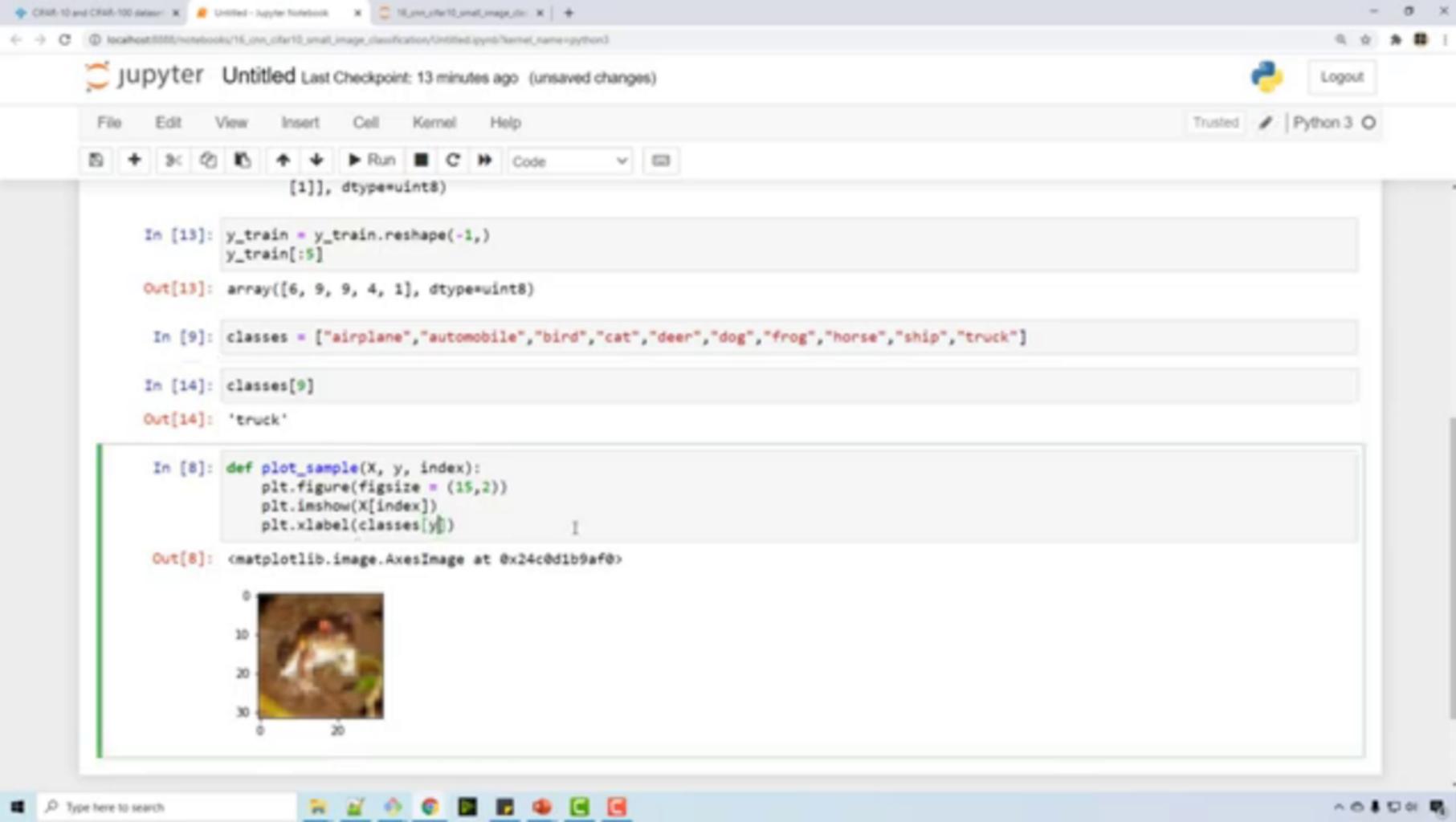
In [8]: def plot_sample(X, y, index):
 plt.figure(figsize=(15,2))
 plt.imshow(X[index])
 plt.xlabel(classes[0])

Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>

0
10
20
30



Type here to search



jupyter Untitled

Last Checkpoint: 13 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

[1]], dtype='uint8')

In [13]: y_train = y_train.reshape(-1,)
y_train[:5]

Out[13]: array([6, 9, 9, 4, 1], dtype='uint8')

In [9]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]

In [14]: classes[9]

Out[14]: 'truck'

In [8]: def plot_sample(x, y, index):
 plt.figure(figsize = (15,2))
 plt.imshow(x[index])
 plt.xlabel(classes[y])

Out[8]: <matplotlib.image.AxesImage at 0x24c0d1b9af0>



Type here to search



CRM-10 and CRM-100 dataset CRM-10 Untitled - Jupyter Notebook CRM-100 16_mnist cifar10_smali_image_classification/Untitled.ipynb Kernel Name: python3

jupyter Untitled Last Checkpoint: 13 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [11]: `y_train[:5]`

Out[11]: `array([[6],
[9],
[9],
[4],
[1]], dtype=uint8)`

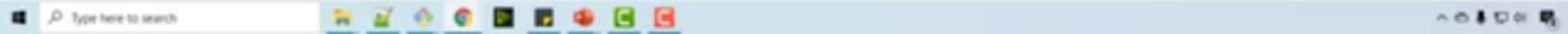
In [13]: `y_train = y_train.reshape(-1,)
y_train[:5]`

Out[13]: `array([6, 9, 9, 4, 1], dtype=uint8)`

In [9]: `classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]`

In [15]: `def plot_sample(X, y, index):
 plt.figure(figsize=(15,2))
 plt.imshow(X[index])
 plt.xlabel(classes[y[index]])`

In []: |



CRM-10 and CRM-100 dataset CRM-10 Untitled - Jupyter Notebook CRM-100 18_cnn_cifar10_small_image_classification/Untitled.ipynb Kernel Name: python3

jupyter Untitled Last Checkpoint: 13 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [11]: `y_train[:5]`

Out[11]: `array([[6],
[9],
[9],
[4],
[1]], dtype=uint8)`

In [13]: `y_train = y_train.reshape(-1,)
y_train[:5]`

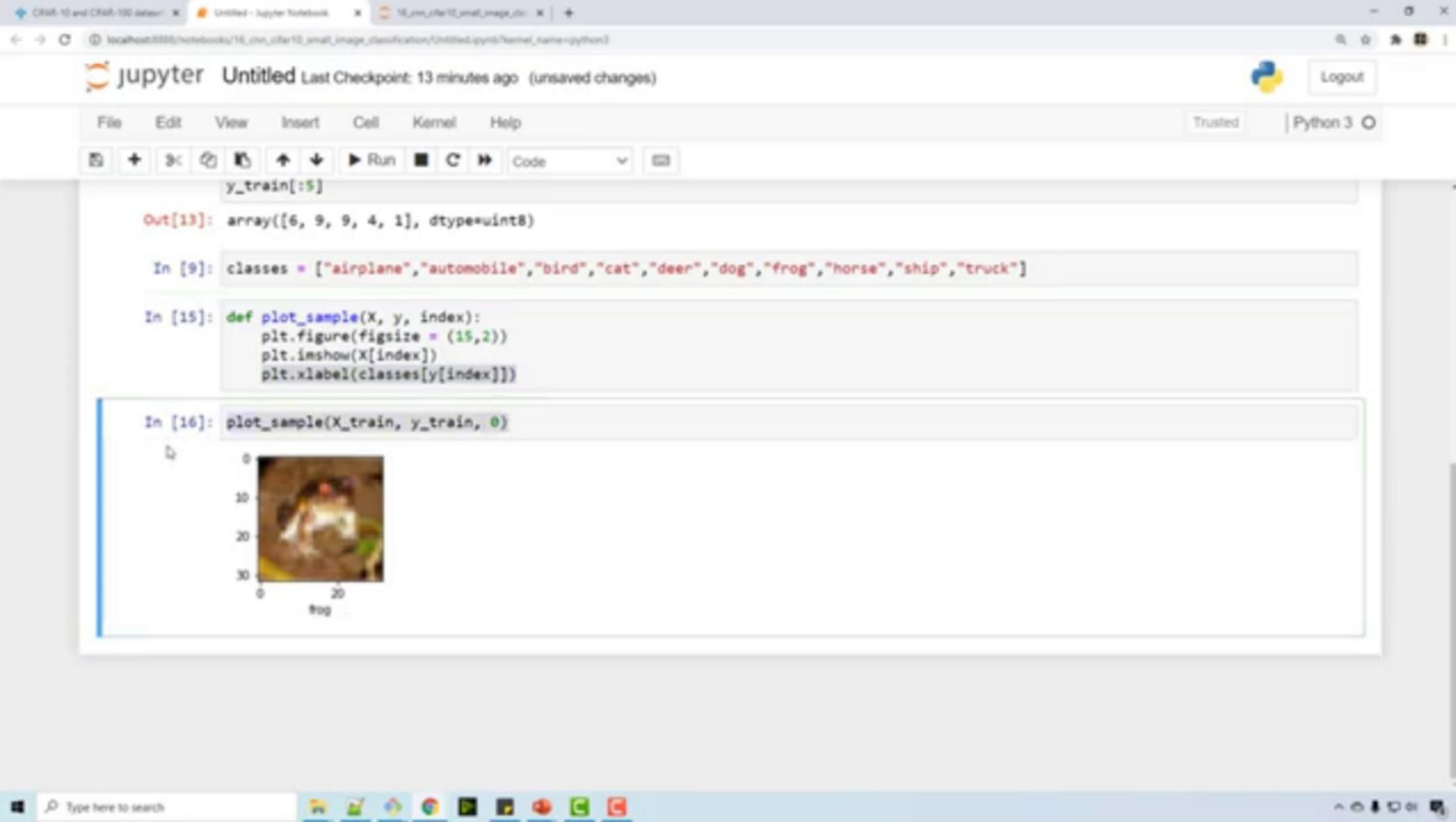
Out[13]: `array([6, 9, 9, 4, 1], dtype=uint8)`

In [9]: `classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]`

In [15]: `def plot_sample(X, y, index):
 plt.figure(figsize=(15,2))
 plt.imshow(X[index])
 plt.xlabel(classes[y[index]])`

In []: `plot_sample(X_train, y_train, 0)`





```
y_train[:5]
```

```
Out[13]: array([6, 9, 9, 4, 1], dtype=uint8)
```

```
In [9]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]
```

```
In [15]: def plot_sample(X, y, index):
    plt.figure(figsize = (15,2))
    plt.imshow(X[index])
    plt.xlabel(classes[y[index]])
```

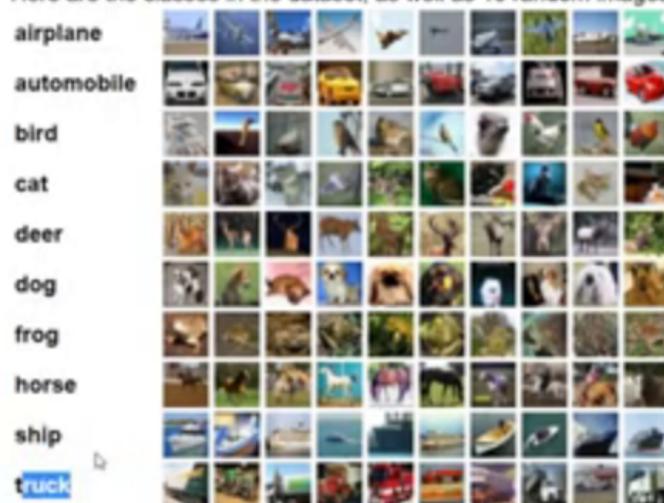
```
In [16]: plot_sample(X_train, y_train, 0)
```



The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:



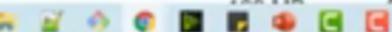
The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

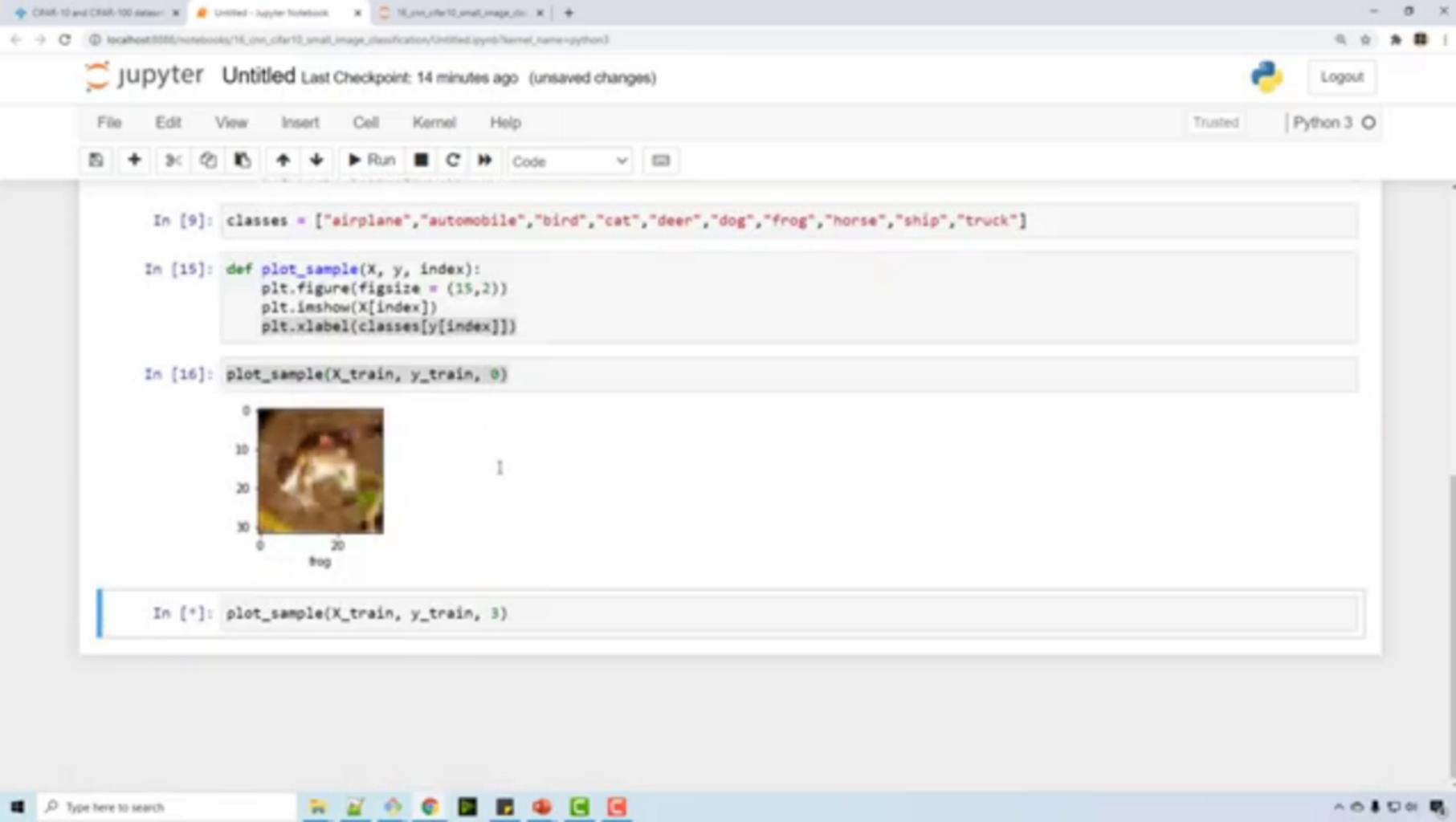
Download

If you're going to use this dataset, please cite the tech report at the bottom of this page.

Version	Size	md5sum
v1	~1.8GB	MD5: 238d94d849bc35f5bf2a3b73cde73ad0

Type here to search





```
In [9]: classes = ["airplane", "automobile", "bird", "cat", "deer", "dog", "frog", "horse", "ship", "truck"]
```

```
In [15]: def plot_sample(X, y, index):
    plt.figure(figsize = (15,2))
    plt.imshow(X[index])
    plt.xlabel(classes[y[index]])
```

```
In [16]: plot_sample(X_train, y_train, 0)
```



```
In [*]: plot_sample(X_train, y_train, 3)
```

 jupyter Untitled Last Checkpoint: 14 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3

In [16]: `plot_sample(X_train, y_train, 0)`In [19]: `plot_sample(X_train, y_train, 3)`

 jupyter Untitled Last Checkpoint: 14 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3 o

In [16]: plot_sample(X_train, y_train, 0)



In [19]: plot_sample(X_train, y_train, 3)



 jupyter Untitled Last Checkpoint: 14 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3 oIn [16]: `plot_sample(X_train, y_train, 0)`In [19]: `plot_sample(X_train, y_train, 3)`

jupyter Untitled Last Checkpoint: 14 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3

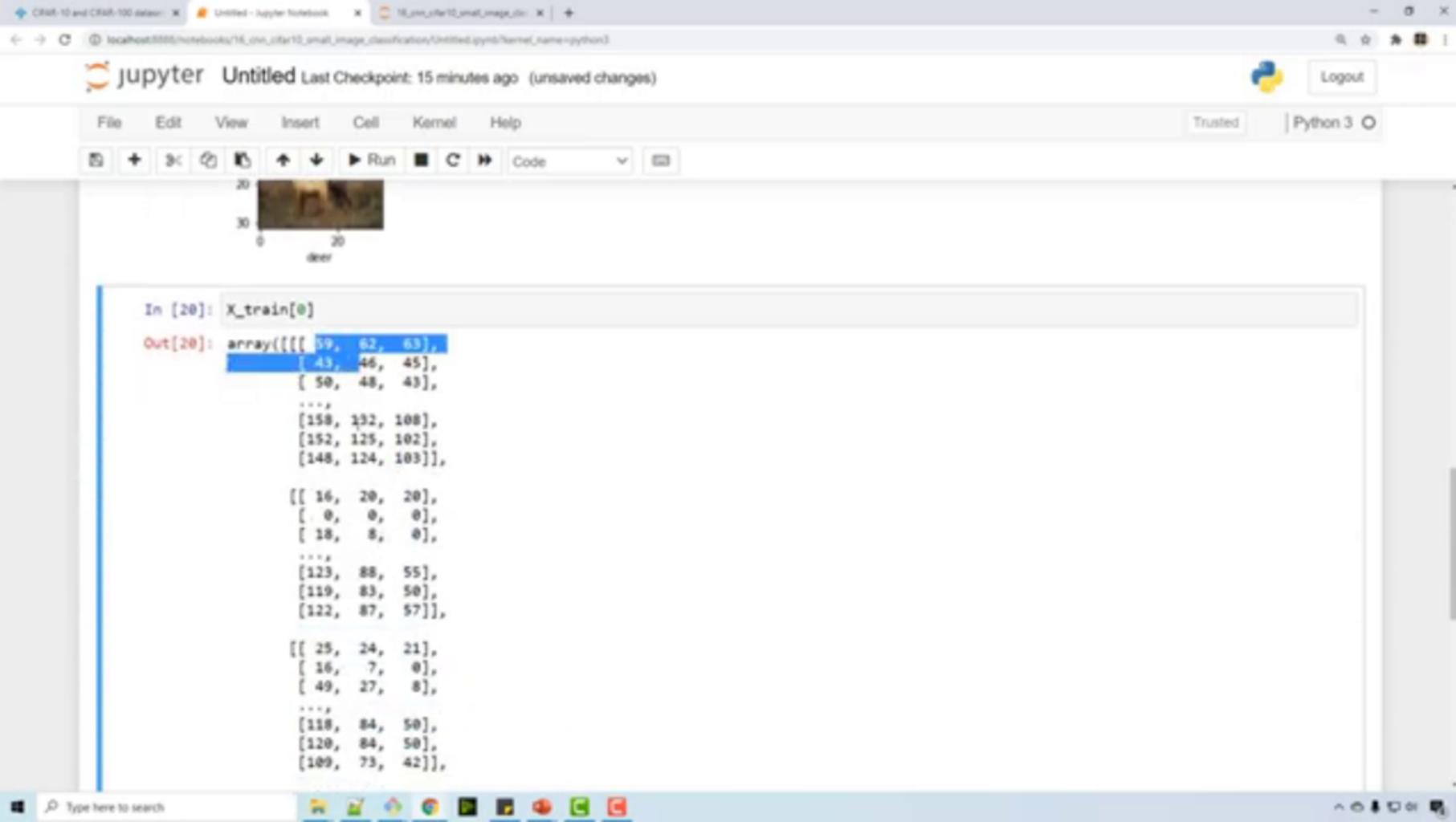


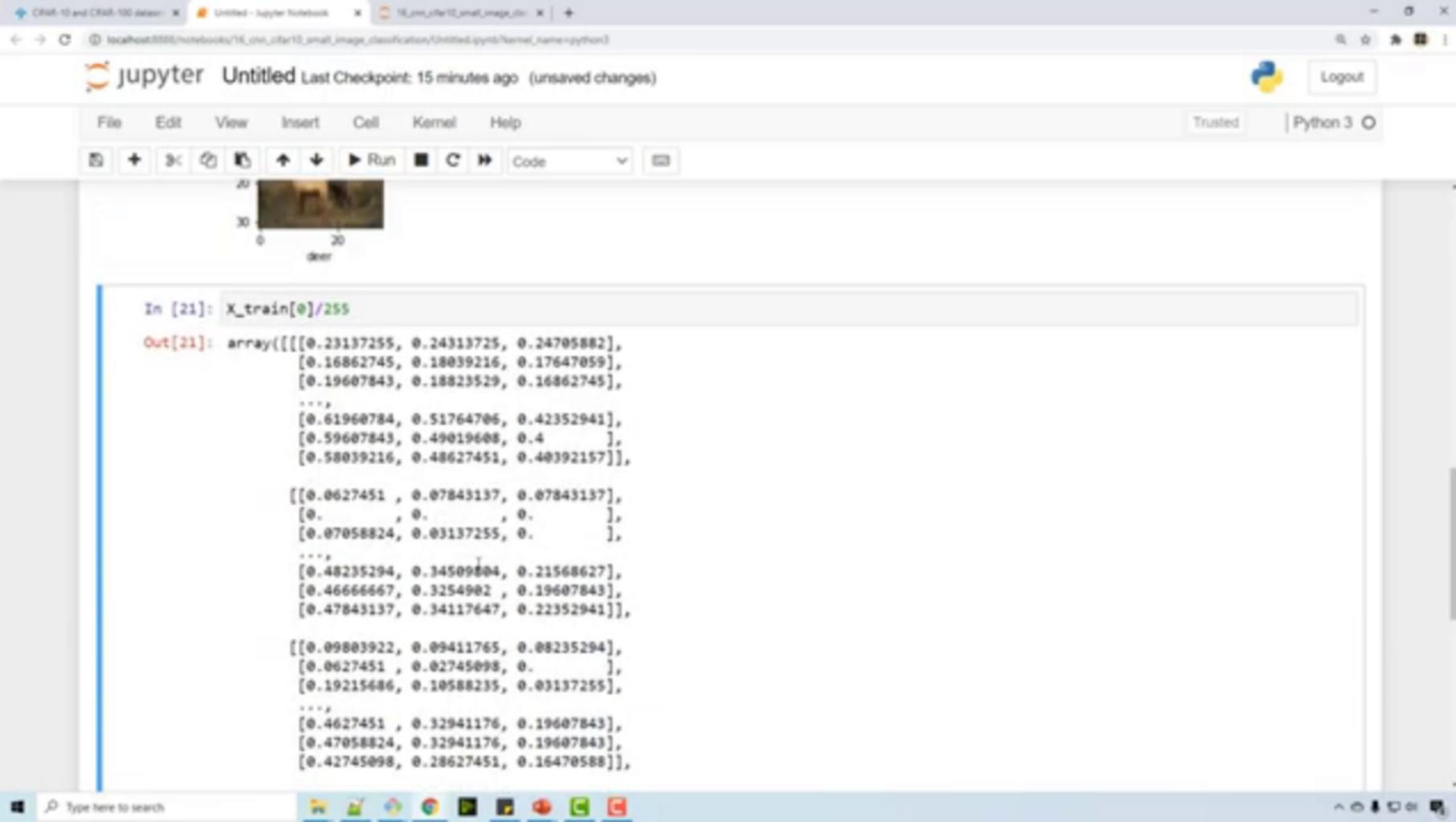
In [16]: plot_sample(X_train, y_train, 0)

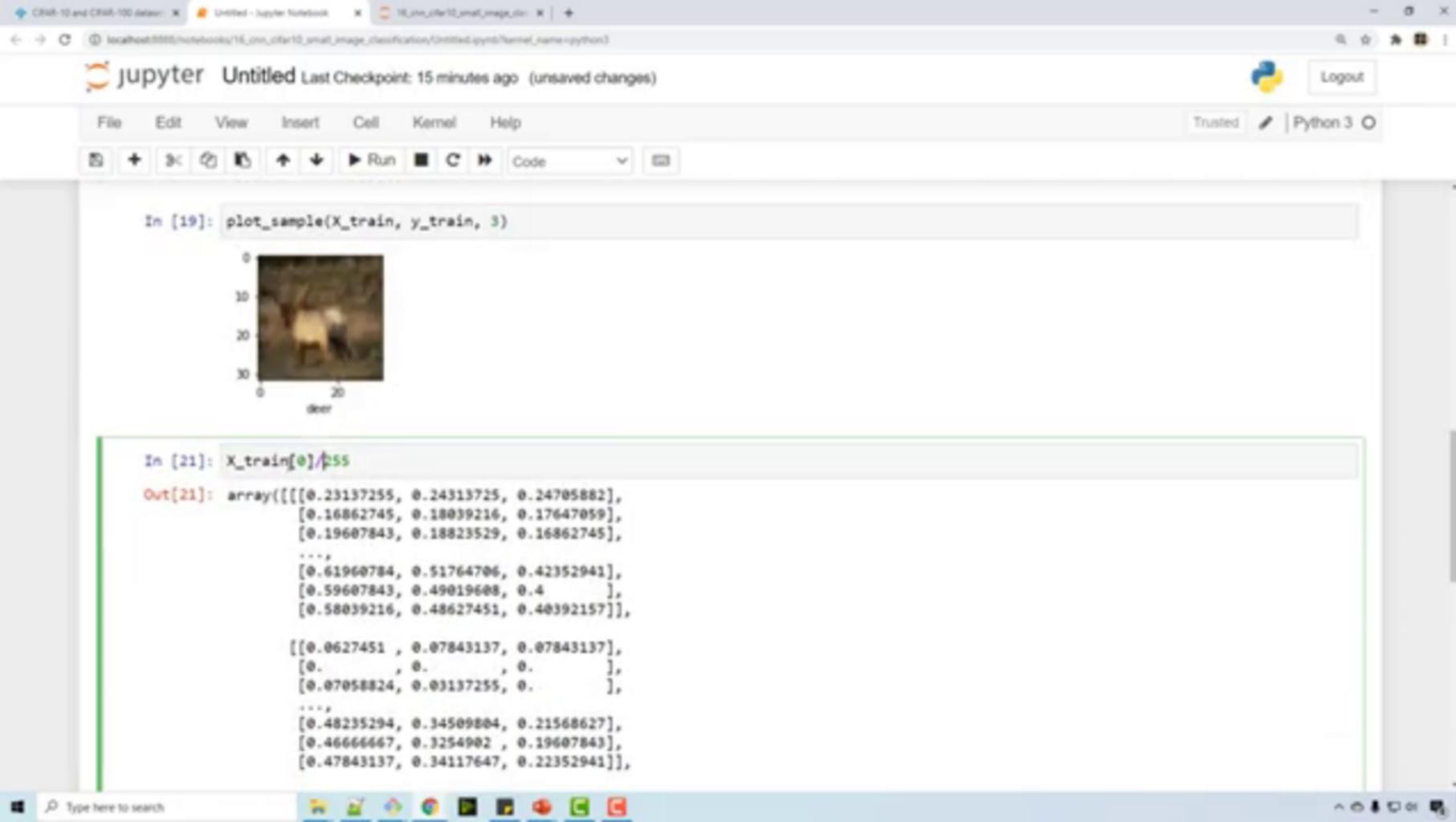


In [19]: plot_sample(X_train, y_train, 3)









```
In [19]: plot_sample(X_train, y_train, 3)
```



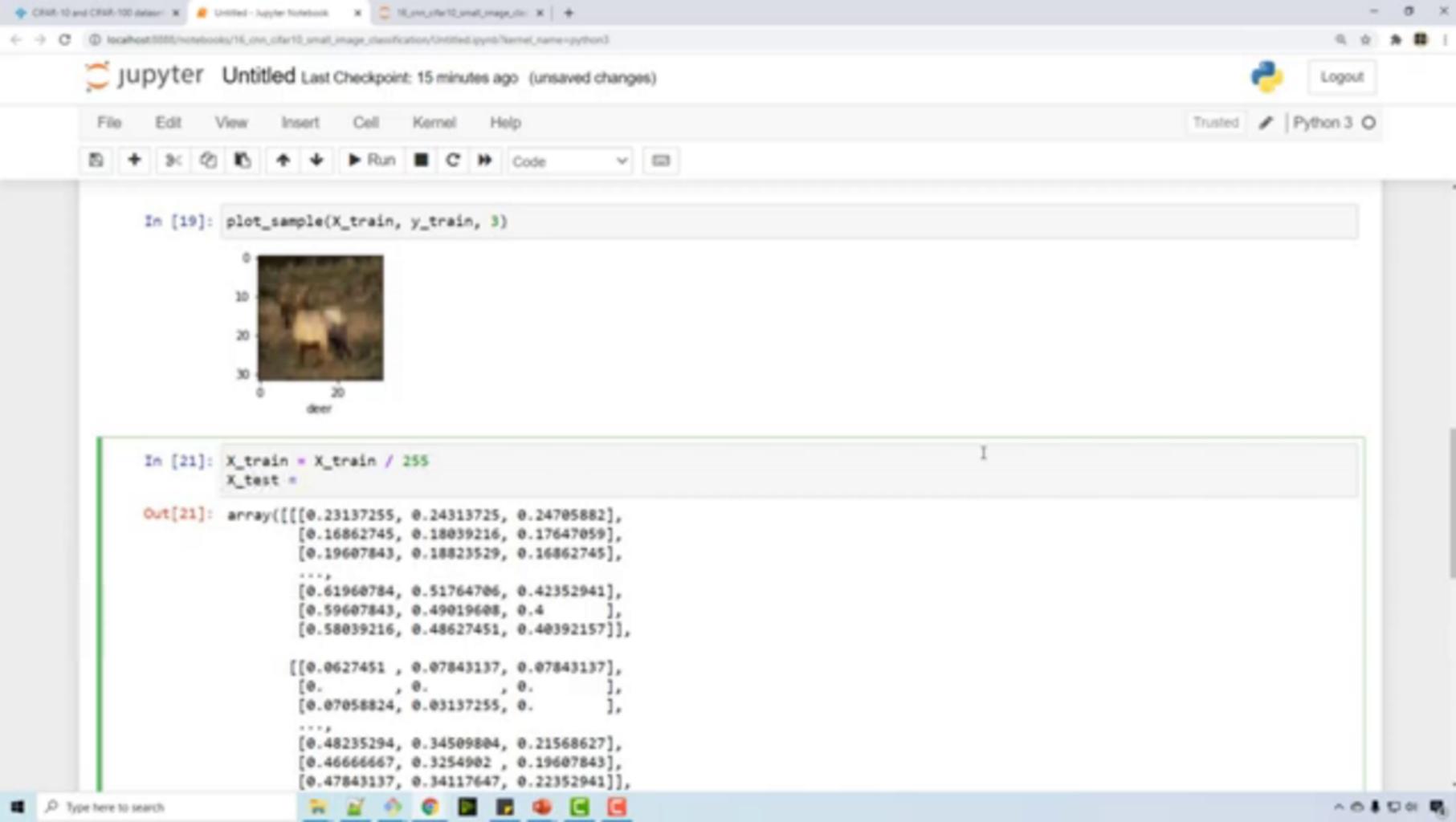
```
In [21]: X_train[0]/255
```

```
Out[21]: array([[[0.23137255, 0.24313725, 0.24705882],
   [0.16862745, 0.18039216, 0.17647059],
   [0.19607843, 0.18823529, 0.16862745],
   ...,
   [0.61960784, 0.51764706, 0.42352941],
   [0.59607843, 0.49019608, 0.4       ],
   [0.58039216, 0.48627451, 0.40392157]],

   [[0.0627451 , 0.07843137, 0.07843137],
   [0.          , 0.          , 0.          ],
   [0.07058824, 0.03137255, 0.          ],
   ...,
   [0.48235294, 0.34509804, 0.21568627],
   [0.46666667, 0.3254902 , 0.19607843],
   [0.47843137, 0.34117647, 0.22352941]],
```

Type here to search



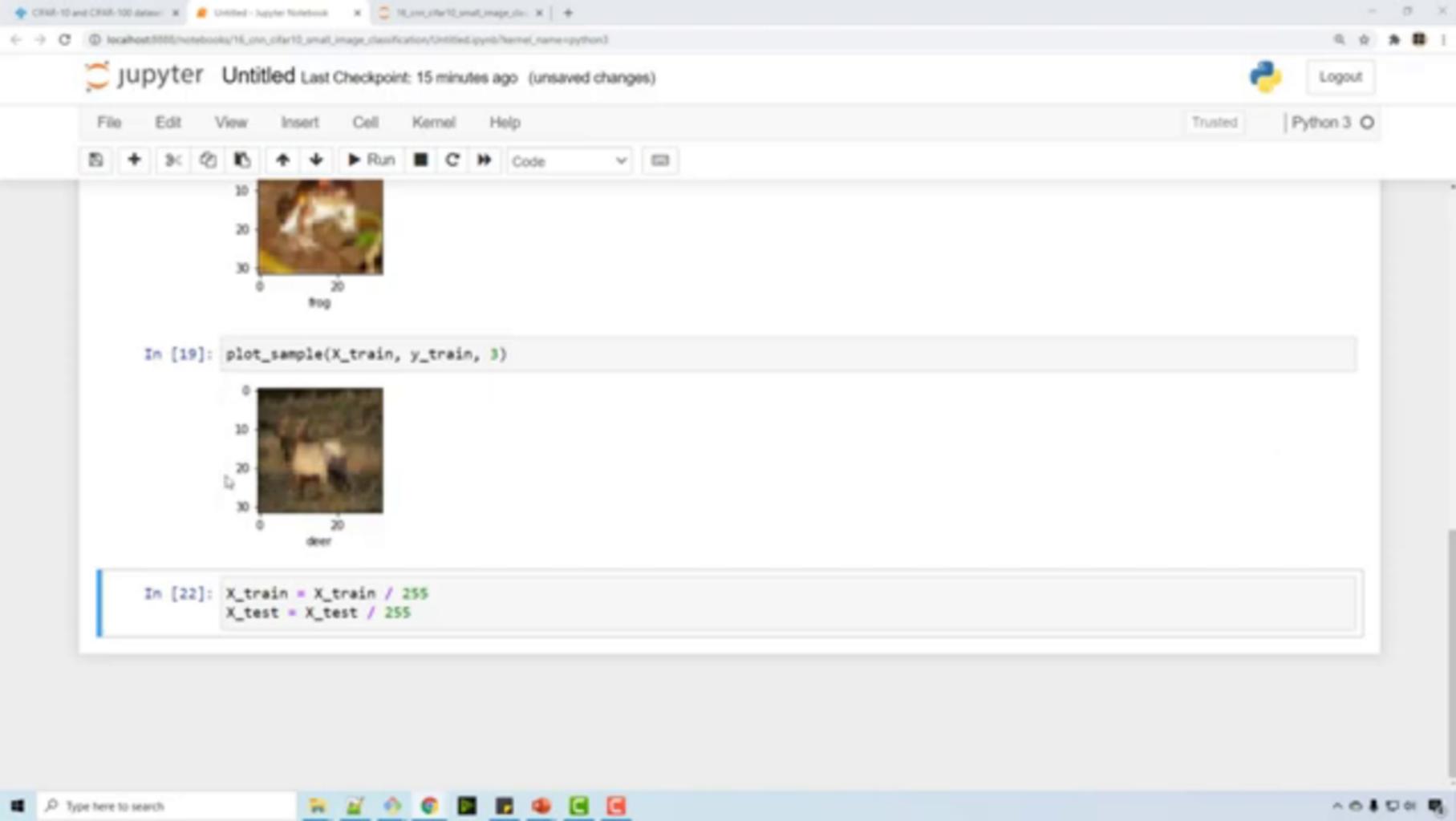


```
In [19]: plot_sample(X_train, y_train, 3)
```



```
In [21]: X_train = X_train / 255  
X_test =
```

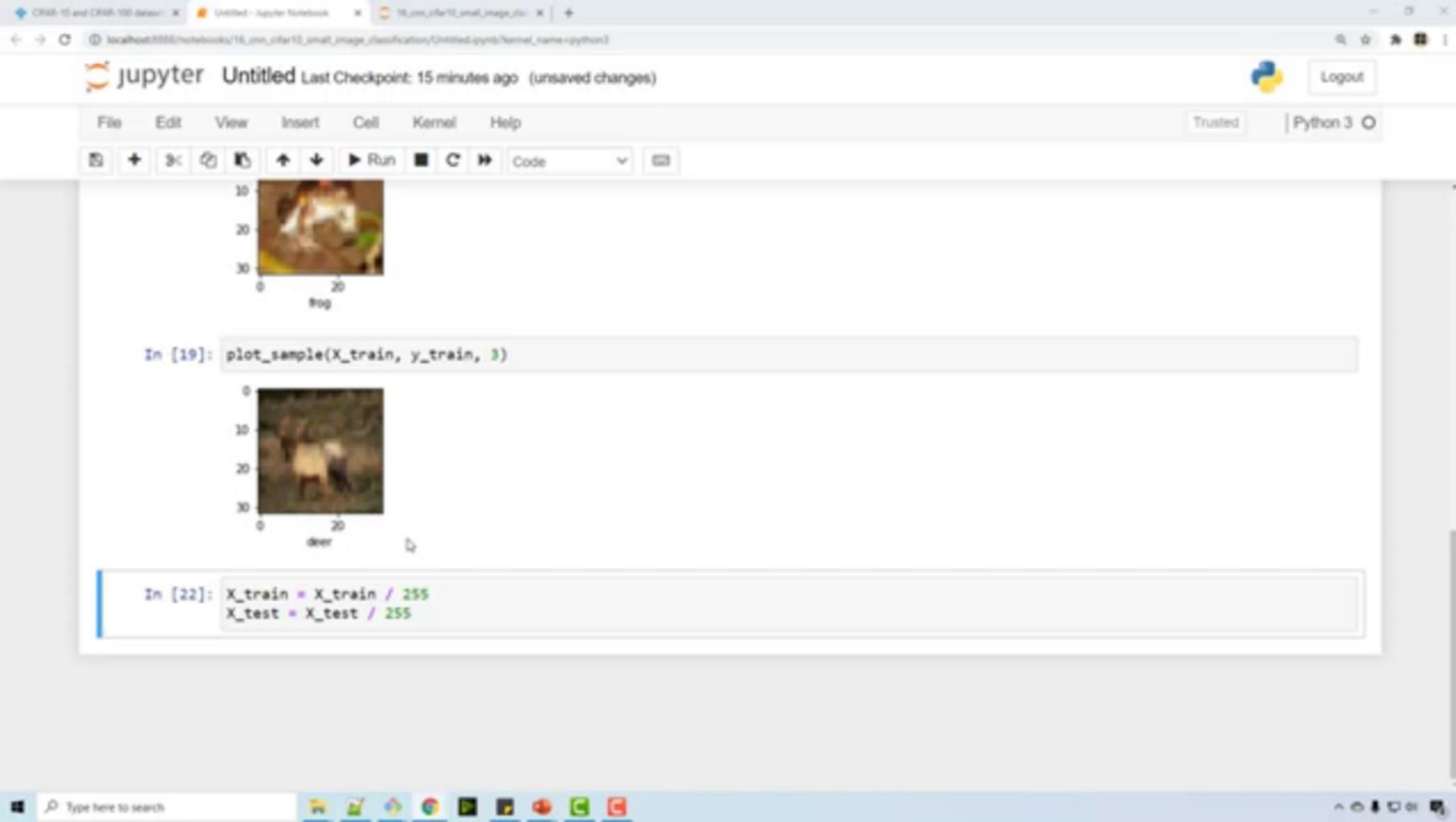
```
Out[21]: array([[0.23137255, 0.24313725, 0.24705882],  
[0.16862745, 0.18039216, 0.17647059],  
[0.19607843, 0.18823529, 0.16862745],  
...,  
[0.61960784, 0.51764706, 0.42352941],  
[0.59607843, 0.49019608, 0.4       ],  
[0.58039216, 0.48627451, 0.40392157]],  
  
[[0.0627451 , 0.07843137, 0.07843137],  
[0.        , 0.        , 0.        ],  
[0.07058824, 0.03137255, 0.        ],  
...,  
[0.48235294, 0.34509804, 0.21568627],  
[0.46666667, 0.3254982 , 0.19607843],  
[0.47843137, 0.34117647, 0.22352941]]]
```



```
In [19]: plot_sample(X_train, y_train, 3)
```



```
In [22]: X_train = X_train / 255  
X_test = X_test / 255
```



File Edit View Insert Cell Kernel Help

Trusted | Python 3

File Edit View Insert Cell Kernel Help

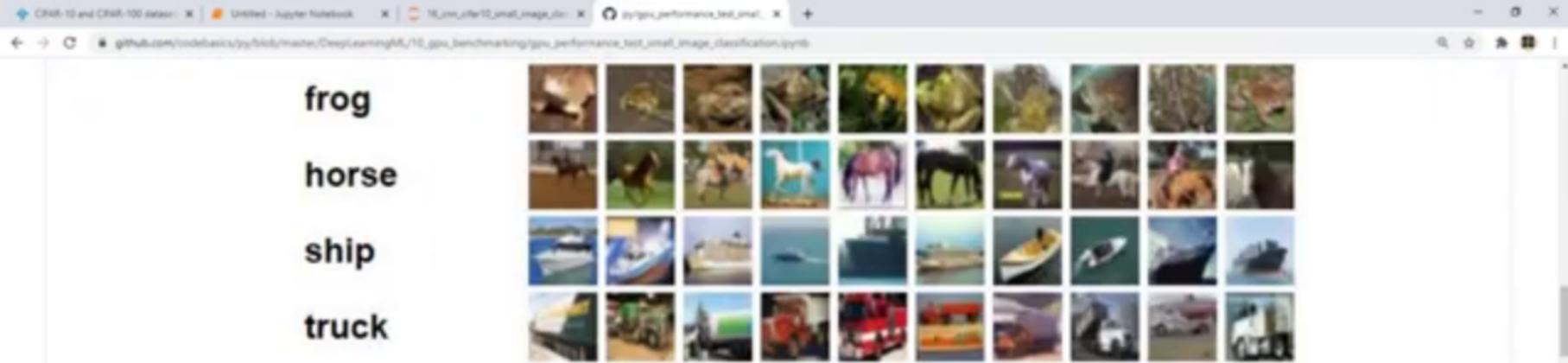
File Edit View Insert Cell Kernel Help



In [19]: `plot_sample(X_train, y_train, 3)`



In [22]: `X_train = X_train / 255
X_test = X_test / 255`



frog

horse

ship

truck

```
In [8]: (X_train, y_train), (X_test,y_test) = tf.keras.datasets.cifar10.load_data()
```

```
In [9]: X_train.shape
```

```
Out[9]: (50000, 32, 32, 3)
```

```
In [10]: y_train.shape
```

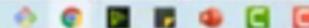
```
Out[10]: (50000, 1)
```

Data Visualization

```
In [11]: def plot_sample(index):
    plt.figure(figsize = (10,1))
    plt.imshow(X_train[index])
```

```
In [12]: plot_sample(0)
```

Type here to search



```
In [18]: y_train_categorical[0:5]
```

```
Out[18]: array([[0., 0., 0., 0., 0., 1., 0., 0., 0.],
   [0., 0., 0., 0., 0., 0., 0., 0., 1.],
   [0., 0., 0., 0., 0., 0., 0., 0., 1.],
   [0., 0., 0., 0., 1., 0., 0., 0., 0.],
   [0., 1., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```

Model building and training

```
In [19]: model = keras.Sequential([
    keras.layers.Flatten(input_shape=(32,32,3)),
    keras.layers.Dense(3000, activation='relu'),
    keras.layers.Dense(1000, activation='relu'),
    keras.layers.Dense(10, activation='sigmoid')
])

model.compile(optimizer='SGD',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(X_train_scaled, y_train_categorical, epochs=1)

1563/1563 [=====] - 3s 2ms/step - loss: 1.8642 - accuracy: 0.3328
```

```
Out[19]: <tensorflow.python.keras.callbacks.History at 0x24b348d8be0>
```

Let's make some predictions

```
In [ ]: np.argmax(model.predict(X_test_scaled)[0])
```

 jupyter Untitled Last Checkpoint: 17 minutes ago (unsaved changes)

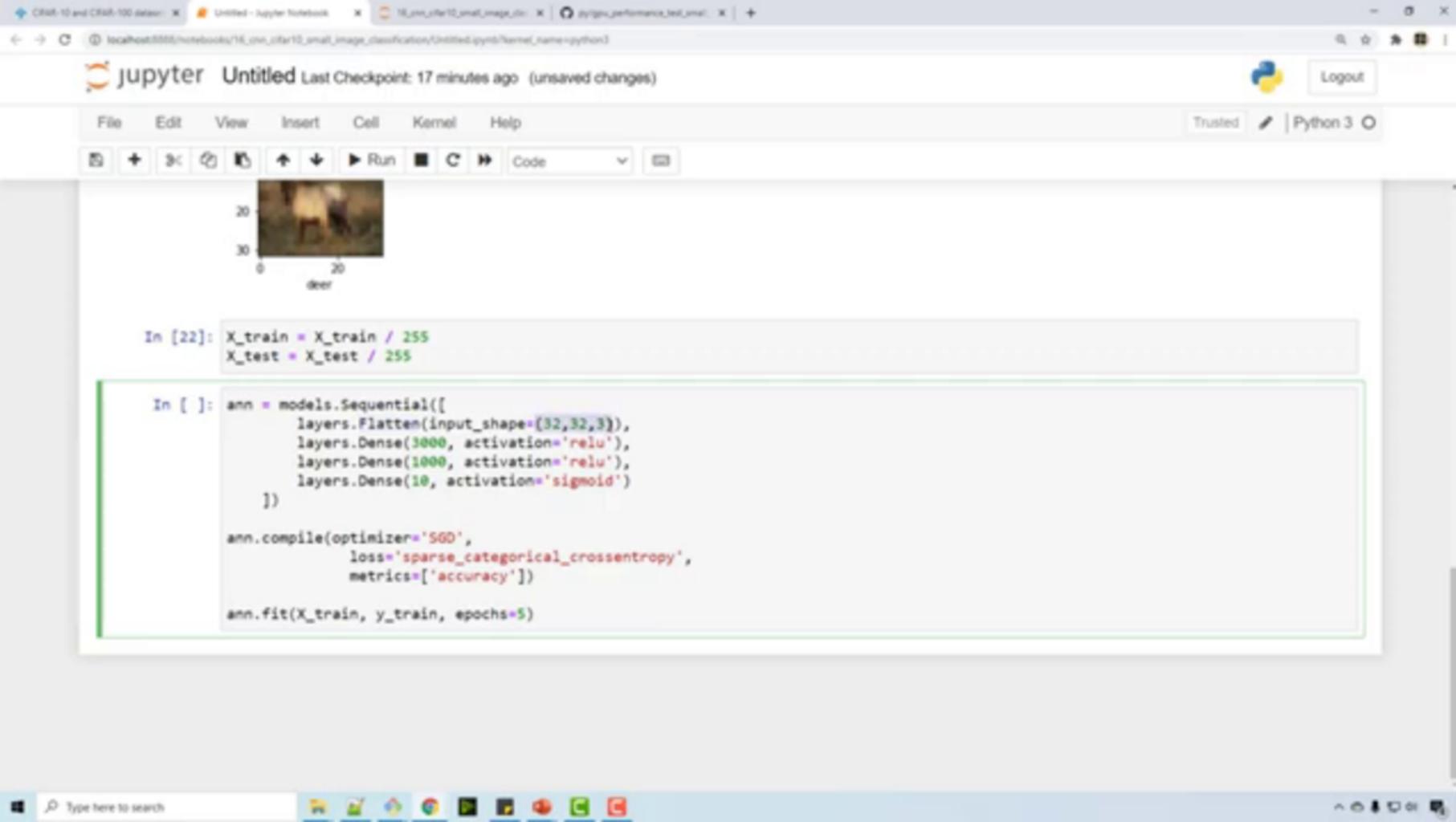
Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

In [19]: `plot_sample(X_train, y_train, 3)`In [22]: `X_train = X_train / 255
X_test = X_test / 255`

In []:



jupyter Untitled Last Checkpoint: 17 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

Cell Kernel Help



```
In [22]: X_train = X_train / 255
X_test = X_test / 255
```

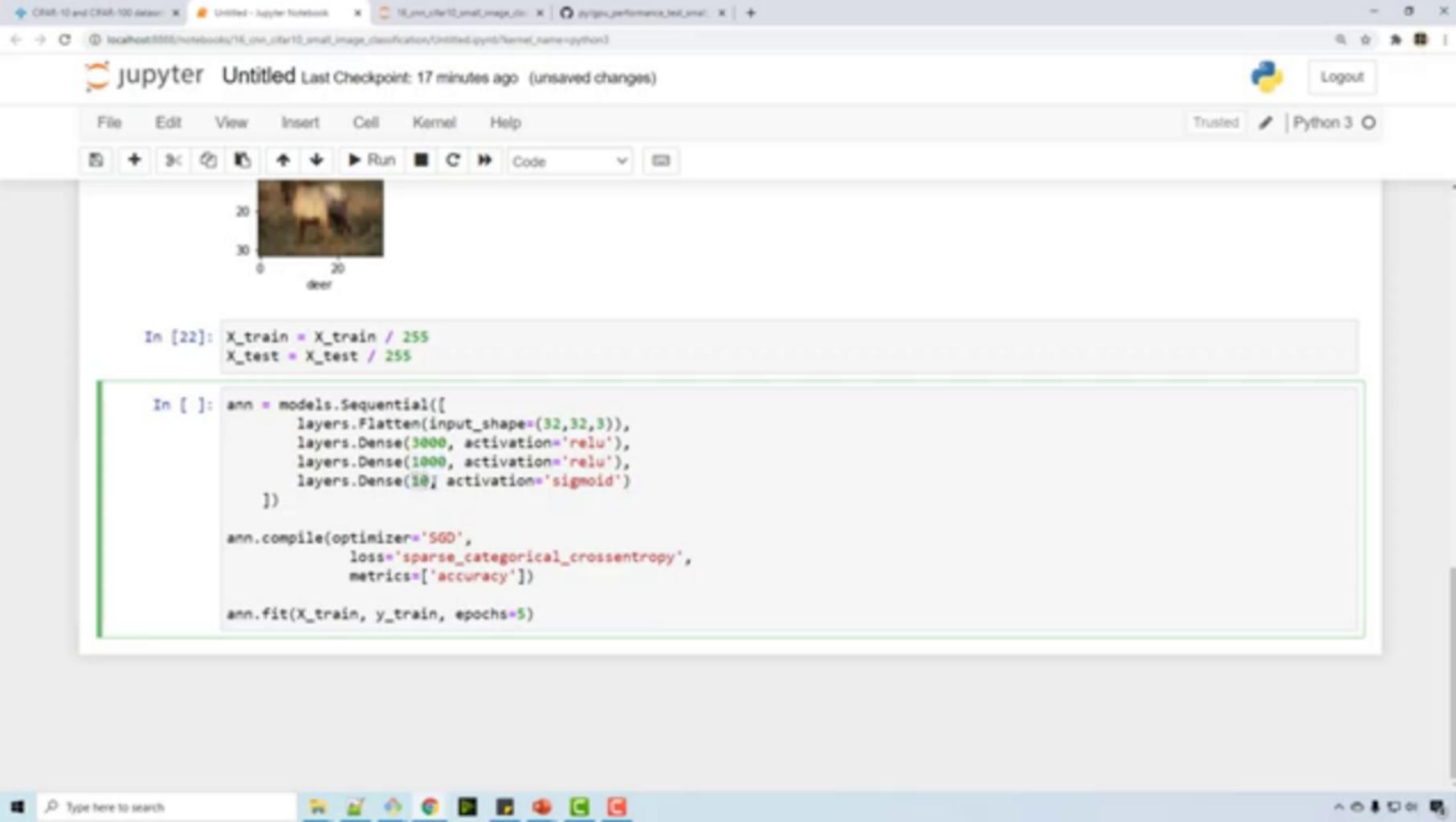
```
In [ ]: ann = models.Sequential([
    layers.Flatten(input_shape=(32,32,3)),
    layers.Dense(3000, activation='relu'),
    layers.Dense(1000, activation='relu'),
    layers.Dense(10, activation='sigmoid')
])

ann.compile(optimizer='SGD',
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])

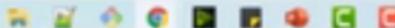
ann.fit(X_train, y_train, epochs=5)
```

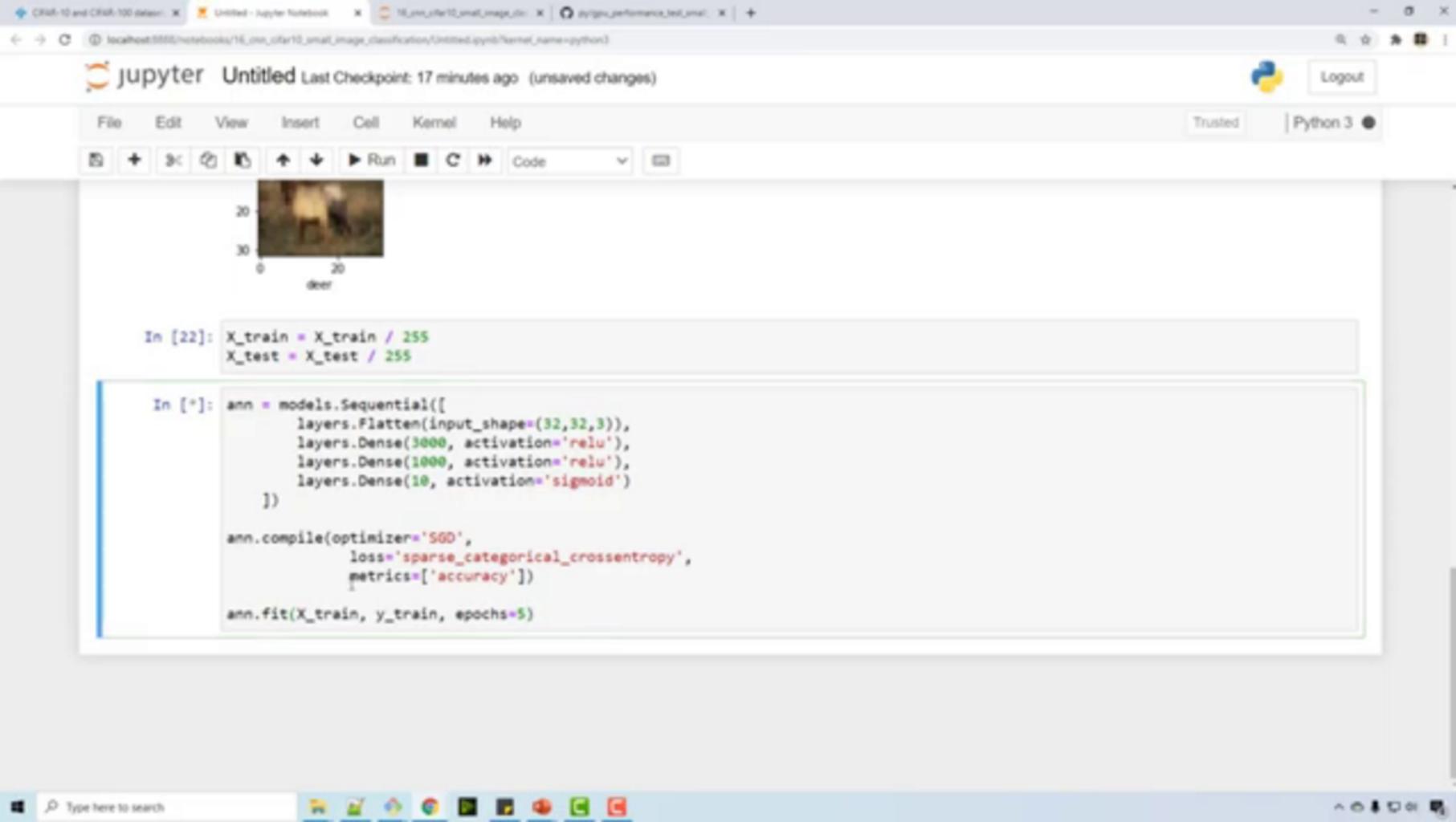
Type here to search





Type here to search





jupyter Untitled Last Checkpoint: 17 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3



```
In [22]: X_train = X_train / 255  
X_test = X_test / 255
```

```
In [*]: ann = models.Sequential([  
        layers.Flatten(input_shape=(32,32,3)),  
        layers.Dense(3000, activation='relu'),  
        layers.Dense(1000, activation='relu'),  
        layers.Dense(10, activation='sigmoid')  
    ])  
  
ann.compile(optimizer='SGD',  
            loss='sparse_categorical_crossentropy',  
            metrics=['accuracy'])  
  
ann.fit(X_train, y_train, epochs=5)  
  
Epoch 1/5  
1563/1563 [*****] - 3s 2ms/step - loss: 1.8591 - accuracy: 0.3366  
Epoch 2/5  
1563/1563 [*****] - 3s 2ms/step - loss: 1.6584 - accuracy: 0.4129  
Epoch 3/5  
1245/1563 [*****->.....] - ETA: 0s - loss: 1.5738 - accuracy: 0.4452
```

```
In [18]: y_train_categorical[0:5]
```

```
Out[18]: array([[0., 0., 0., 0., 0., 1., 0., 0., 0.],
   [0., 0., 0., 0., 0., 0., 0., 0., 1.],
   [0., 0., 0., 0., 0., 0., 0., 0., 1.],
   [0., 0., 0., 0., 1., 0., 0., 0., 0.],
   [0., 1., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```

Model building and training

```
In [19]: model = keras.Sequential([
    keras.layers.Flatten(input_shape=(32,32,3)),
    keras.layers.Dense(3000, activation='relu'),
    keras.layers.Dense(1000, activation='relu'),
    keras.layers.Dense(10, activation='sigmoid')
])

model.compile(optimizer='SGD',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

model.fit(X_train_scaled, y_train_categorical, epochs=1)

1563/1563 [=====] - 3s 2ms/step - loss: 1.8642 - accuracy: 0.3328
```

```
Out[19]: <tensorflow.python.keras.callbacks.History at 0x24b348d8be0>
```

Let's make some predictions

```
In [ ]: np.argmax(model.predict(X_test_scaled)[0])
```

Shared Panel

Search

Annotations Hide Show Review View Add My Help

Plane Automobile d : er g g rse ip ck

0 0 0 0 0 0 0 0 1 0

y = 8

loss='sparse_categorical_crossentropy'

91 - accuracy: 0.3366
34 - accuracy: 0.4129
34 - accuracy: 0.4474
33 - accuracy: 0.4677
76 - accuracy: 0.4857

Type here to search

Image

Drawing

Editing

Design Ideas

Visual

Share

Comments

File Replace Select Delete

Plane Automobile d : er g g rse ip ck

0 0 0 0 0 0 0 0 1 0

y = 8

loss='sparse_categorical_crossentropy'

91 - accuracy: 0.3366
34 - accuracy: 0.4129
34 - accuracy: 0.4474
33 - accuracy: 0.4677
76 - accuracy: 0.4857

airplane



automobile



bird



cat



deer



dog



frog



horse



ship



truck



y

0
0
0
0
0
0
0
0
1
0

y = 8

loss='sparse_categorical_crossentropy'

4

loss='categorical_crossentropy'

airplane



automobile



bird



cat



deer



dog



frog



horse



ship



b

truck



y

0
0
0
0
0
0
0
0
1
0

y = 8

`loss='sparse_categorical_crossentropy'`

`loss='categorical_crossentropy'`



airplane



automobile



bird



cat



deer



dog



frog



horse



ship



truck



y

0
0
0
0
0
0
0
0
1
0

y = 8

loss='sparse_categorical_crossentropy'

loss='categorical_crossentropy'

```
y_test, num_classes=10, dtype='float32'  
})
```

In [17]: y_train[0:5]

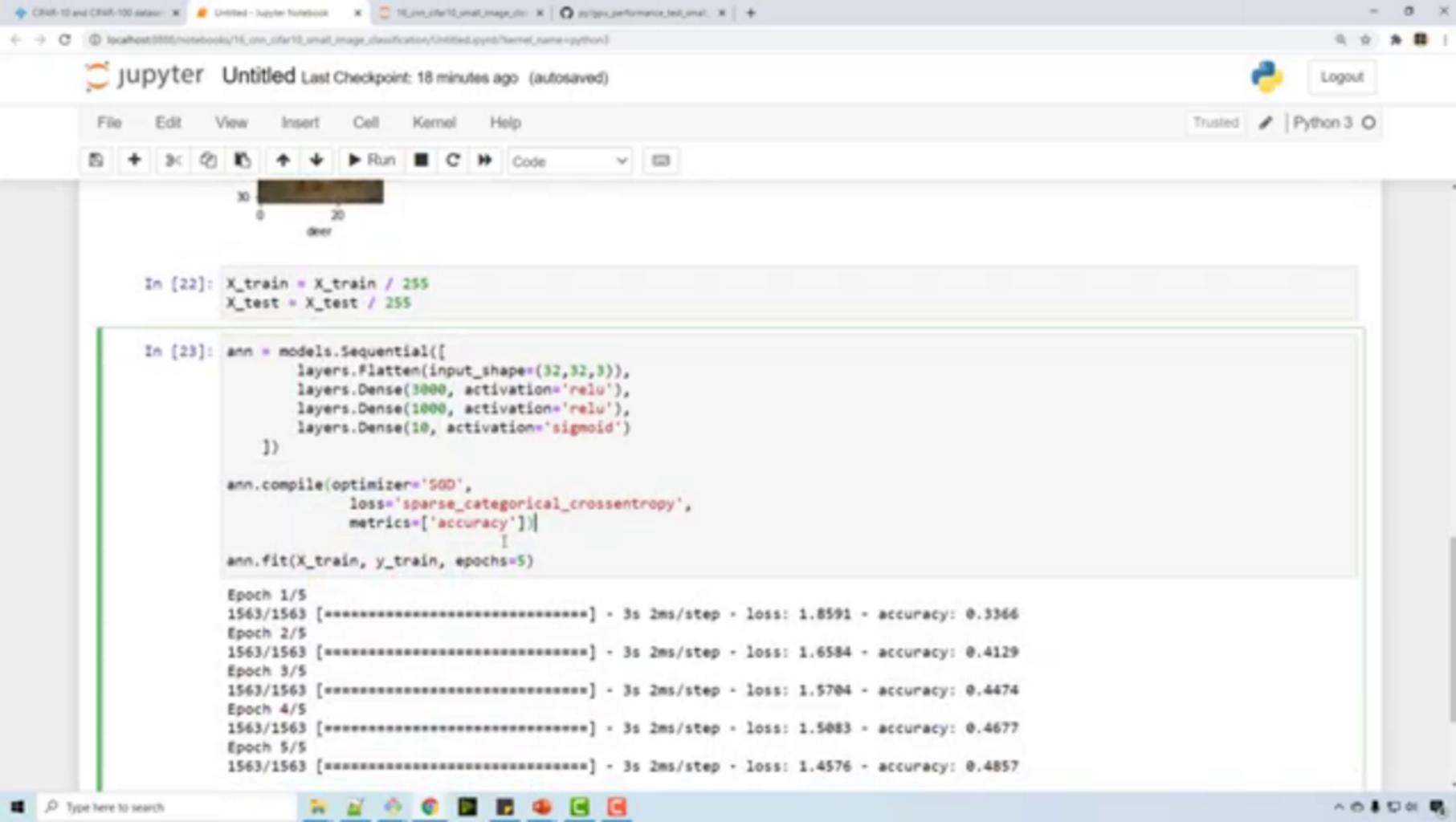
```
Out[17]: array([[6],  
                 [9],  
                 [9],  
                 [4],  
                 [1]], dtype=uint8)
```

In [18]: y_train_categorical[0:5]

```
Out[18]: array([[0., 0., 0., 0., 0., 0., 1., 0., 0., 0.],  
                 [0., 0., 0., 0., 0., 0., 0., 0., 0., 1.],  
                 [0., 0., 0., 0., 0., 0., 0., 0., 0., 1.],  
                 [0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],  
                 [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.]], dtype=float32)
```

Model building and training

```
In [19]: model = keras.Sequential([  
        keras.layers.Flatten(input_shape=(32,32,3)),  
        keras.layers.Dense(3000, activation='relu'),  
        keras.layers.Dense(1000, activation='relu'),  
        keras.layers.Dense(10, activation='sigmoid')  
])  
  
model.compile(optimizer='SGD',  
              loss='categorical_crossentropy',  
              metrics=['accuracy'])
```



CRM-10 and CRM-100 dataset... Untitled - Jupyter Notebook 18_cnn_cifar10_small_image_classification/Untitled.ipynb kernel_name:python3

jupyter Untitled Last Checkpoint: 22 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [22]: X_train = X_train / 255
X_test = X_test / 255

In [23]: ann = models.Sequential([
 layers.Flatten(input_shape=(32,32,3)),
 layers.Dense(3000, activation='relu'),
 layers.Dense(1000, activation='relu'),
 layers.Dense(10, activation='sigmoid')
])

ann.compile(optimizer='SGD',
 loss='sparse_categorical_crossentropy',
 metrics=['accuracy'])

ann.fit(X_train, y_train, epochs=5)

Epoch 1/5
1563/1563 [*****] - 3s 2ms/step - loss: 1.8591 - accuracy: 0.3366
Epoch 2/5
1563/1563 [*****] - 3s 2ms/step - loss: 1.6584 - accuracy: 0.4129
Epoch 3/5
1563/1563 [*****] - 3s 2ms/step - loss: 1.5704 - accuracy: 0.4474
Epoch 4/5
1563/1563 [*****] - 3s 2ms/step - loss: 1.5083 - accuracy: 0.4677
Epoch 5/5
1563/1563 [*****] - 3s 2ms/step - loss: 1.4576 - accuracy: 0.4857

Out[23]: <tensorflow.python.keras.callbacks.History at 0x24c0976edf0>

In [24]: ann.evaluate(X_test, y_test)

Type here to search

CRM 10 and CRM 100 dataset CRM 10 Untitled - Jupyter Notebook CRM 100 16_cnn_cifar10_small_image_classification.ipynb pygpu_performance_test_omni pyhandling_imbalanced_data.ipynb +

localhost:8888/notebooks/16_cnn_cifar10_small_image_classification.ipynb?kernel_name=python3

jupyter Untitled Last Checkpoint: 23 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [23]:

```
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])

ann.fit(X_train, y_train, epochs=5)

Epoch 1/5
1563/1563 [*****] - 3s 2ms/step - loss: 1.8591 - accuracy: 0.3366
Epoch 2/5
1563/1563 [*****] - 3s 2ms/step - loss: 1.6584 - accuracy: 0.4129
Epoch 3/5
1563/1563 [*****] - 3s 2ms/step - loss: 1.5704 - accuracy: 0.4474
Epoch 4/5
1563/1563 [*****] - 3s 2ms/step - loss: 1.5083 - accuracy: 0.4677
Epoch 5/5
1563/1563 [*****] - 3s 2ms/step - loss: 1.4576 - accuracy: 0.4857

Out[23]: <tensorflow.python.keras.callbacks.History at 0x24c0976edf0>
```

In [24]:

```
ann.evaluate(X_test, y_test)

313/313 [*****] - 0s 1ms/step - loss: 1.4823 - accuracy: 0.4759

Out[24]: [1.4822779893875122, 0.47589999437332153]
```

In [27]:

```
from sklearn.metrics import confusion_matrix , classification_report
import numpy as np
y_pred = ann.predict(X_test)
y_pred_classes = [np.argmax(element) for element in y_pred]

print("Classification Report: \n", classification_report(y_test, y_pred_classes))

Classification Report:
```

Type here to search CRM 10 CRM 100

jupyter Untitled Last Checkpoint: 23 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

```
In [22]: X_train = X_train / 255  
X_test = X_test / 255
```

```
In [23]: ann = models.Sequential([  
        layers.Flatten(input_shape=(32,32,3)),  
        layers.Dense(3000, activation='relu'),  
        layers.Dense(1000, activation='relu'),  
        layers.Dense(10, activation='sigmoid')  
    ])  
  
ann.compile(optimizer='SGD',  
            loss='sparse_categorical_crossentropy',  
            metrics=['accuracy'])  
  
ann.fit(X_train, y_train, epochs=5)
```

```
Epoch 1/5  
1563/1563 [*****] - 3s 2ms/step - loss: 1.8591 - accuracy: 0.3366  
Epoch 2/5  
1563/1563 [*****] - 3s 2ms/step - loss: 1.6584 - accuracy: 0.4129  
Epoch 3/5  
1563/1563 [*****] - 3s 2ms/step - loss: 1.5704 - accuracy: 0.4474  
Epoch 4/5  
1563/1563 [*****] - 3s 2ms/step - loss: 1.5083 - accuracy: 0.4677  
Epoch 5/5  
1563/1563 [*****] - 3s 2ms/step - loss: 1.4576 - accuracy: 0.4857
```

```
Out[23]: <tensorflow.python.keras.callbacks.History at 0x24c0976edf0>
```

```
In [24]: ann.evaluate(X_test, y_test)
```

Type here to search



jupyter Untitled Last Checkpoint: 23 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3



```
In [27]: from sklearn.metrics import confusion_matrix , classification_report
import numpy as np
y_pred = ann.predict(X_test)
y_pred_classes = [np.argmax(element) for element in y_pred]

print("Classification Report: \n", classification_report(y_test, y_pred_classes))
```

Classification Report:
precision recall f1-score support

	0	0.71	0.35	0.47	1000
1	0.55	0.71	0.62	1000	
2	0.33	0.41	0.36	1000	
3	0.44	0.15	0.23	1000	
4	0.34	0.51	0.40	1000	
5	0.43	0.33	0.37	1000	
6	0.43	0.68	0.52	1000	
7	0.63	0.45	0.53	1000	
8	0.57	0.69	0.62	1000	
9	0.59	0.48	0.53	1000	
accuracy			0.48	10000	
macro avg	0.50	0.48	0.47	10000	
weighted avg	0.50	0.48	0.47	10000	

jupyter Untitled Last Checkpoint: 23 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3



```
In [27]: from sklearn.metrics import confusion_matrix , classification_report
import numpy as np
y_pred = ann.predict(X_test)
y_pred_classes = [np.argmax(element) for element in y_pred]

print("Classification Report: \n", classification_report(y_test, y_pred_classes))
```

	precision	recall	f1-score	support
0	0.71	0.35	0.47	1000
1	0.55	0.71	0.62	1000
2	0.33	0.41	0.36	1000
3	0.44	0.15	0.23	1000
4	0.34	0.51	0.40	1000
5	0.43	0.33	0.37	1000
6	0.43	0.68	0.52	1000
7	0.63	0.45	0.53	1000
8	0.57	0.69	0.62	1000
9	0.59	0.48	0.53	1000
accuracy			0.48	10000
macro avg	0.50	0.48	0.47	10000
weighted avg	0.50	0.48	0.47	10000

jupyter Untitled Last Checkpoint: 23 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3



```
In [27]: from sklearn.metrics import confusion_matrix , classification_report
import numpy as np
y_pred = ann.predict(X_test)
y_pred_classes = [np.argmax(element) for element in y_pred]

print("Classification Report: \n", classification_report(y_test, y_pred_classes))
```

	precision	recall	f1-score	support
0	0.71	0.35	0.47	1000
1	0.55	0.71	0.62	1000
2	0.33	0.41	0.36	1000
3	0.44	0.15	0.23	1000
4	0.34	0.51	0.40	1000
5	0.43	0.33	0.37	1000
6	0.43	0.68]	0.52	1000
7	0.63	0.45	0.53	1000
8	0.57	0.69	0.62	1000
9	0.59	0.48	0.53	1000
accuracy			0.48	10000
macro avg	0.50	0.48	0.47	10000
weighted avg	0.50	0.48	0.47	10000

jupyter Untitled Last Checkpoint: 23 minutes ago (autosaved)



Logout

File Edit View Insert Cell Kernel Help

Notebook saved Trusted

Python 3



```
In [27]: from sklearn.metrics import confusion_matrix , classification_report
import numpy as np
y_pred = ann.predict(X_test)
y_pred_classes = [np.argmax(element) for element in y_pred]

print("Classification Report: \n", classification_report(y_test, y_pred_classes))
```

	precision	recall	f1-score	support
0	0.71	0.35	0.47	1000
1	0.55	0.71	0.62	1000
2	0.33	0.41	0.36	1000
3	0.44	0.15	0.23	1000
4	0.34	0.51	0.40	1000
5	0.43	0.33	0.37	1000
6	0.43	0.68	0.52	1000
7	0.63	0.45	0.53	1000
8	0.57	0.69	0.62	1000
9	0.59	0.48	0.53	1000
accuracy			0.48	10000
macro avg	0.50	0.48	0.47	10000
weighted avg	0.50	0.48	0.47	10000

jupyter Untitled Last Checkpoint: 24 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

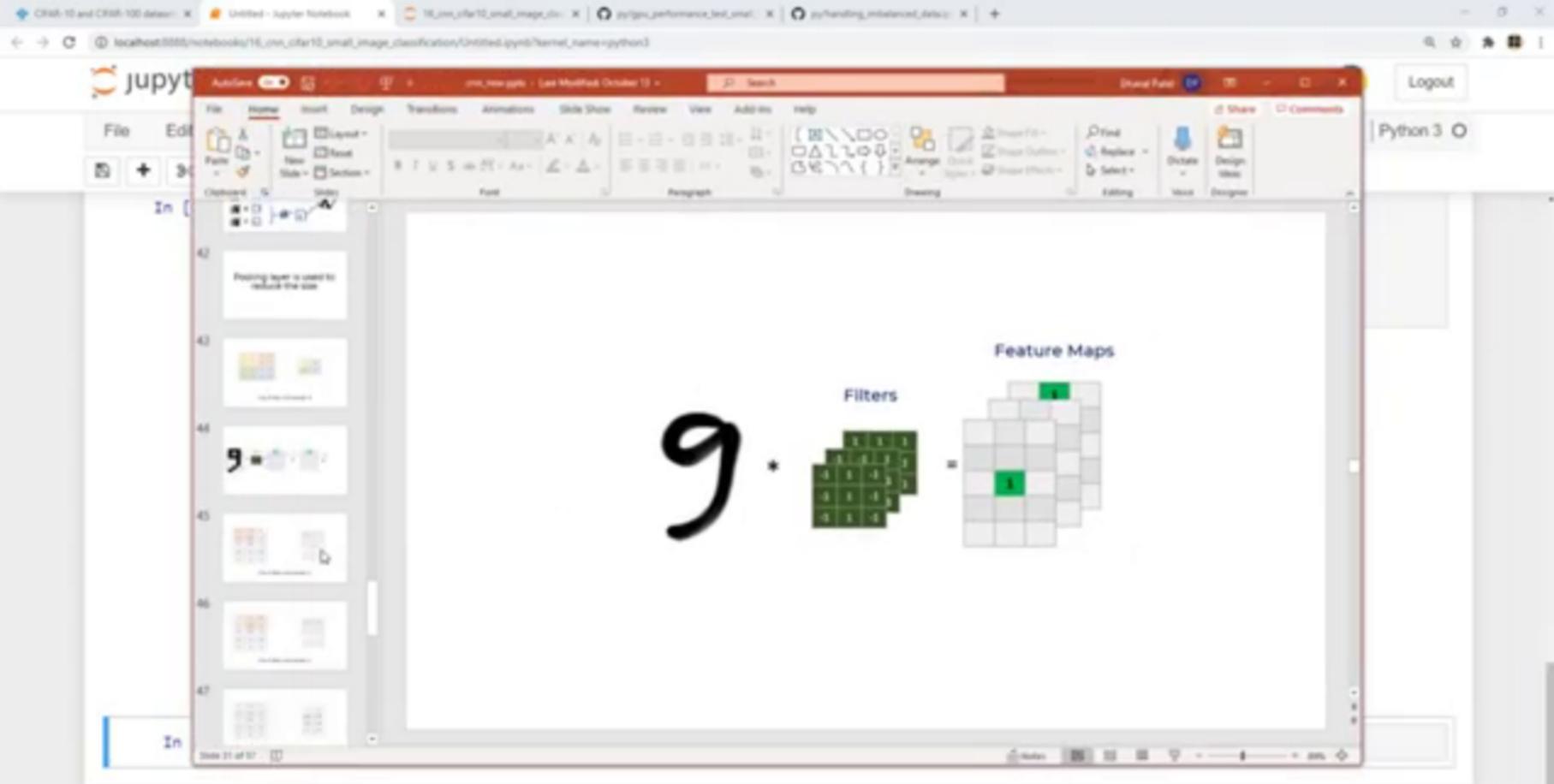
Python 3



```
In [27]: from sklearn.metrics import confusion_matrix , classification_report  
import numpy as np  
y_pred = ann.predict(X_test)  
y_pred_classes = [np.argmax(element) for element in y_pred]  
  
print("Classification Report: \n", classification_report(y_test, y_pred_classes))
```

	precision	recall	f1-score	support
0	0.71	0.35	0.47	1000
1	0.55	0.71	0.62	1000
2	0.33	0.41	0.36	1000
3	0.44	0.15	0.23	1000
4	0.34	0.51	0.40	1000
5	0.43	0.33	0.37	1000
6	0.43	0.68	0.52	1000
7	0.63	0.45	0.53	1000
8	0.57	0.69	0.62	1000
9	0.59	0.48	0.53	1000
accuracy			0.48	10000
macro avg	0.50	0.48	0.47	10000
weighted avg	0.50	0.48	0.47	10000

In []:



CRM-10 and CRM-100 dataset | Untitled - Jupyter Notebook | 16_mnist_cifar10_small_image_classification | pygpu_performance_test_email | pyhandling_imbalanced_data | +

localhost:8888/notebooks/16_mnist_cifar10_small_image_classification/Untitled.ipynb?kernel_name=python3

jupyter

Autosave **File** Home Insert Design Transitions Animations Slide Show Review View Add-ins Help

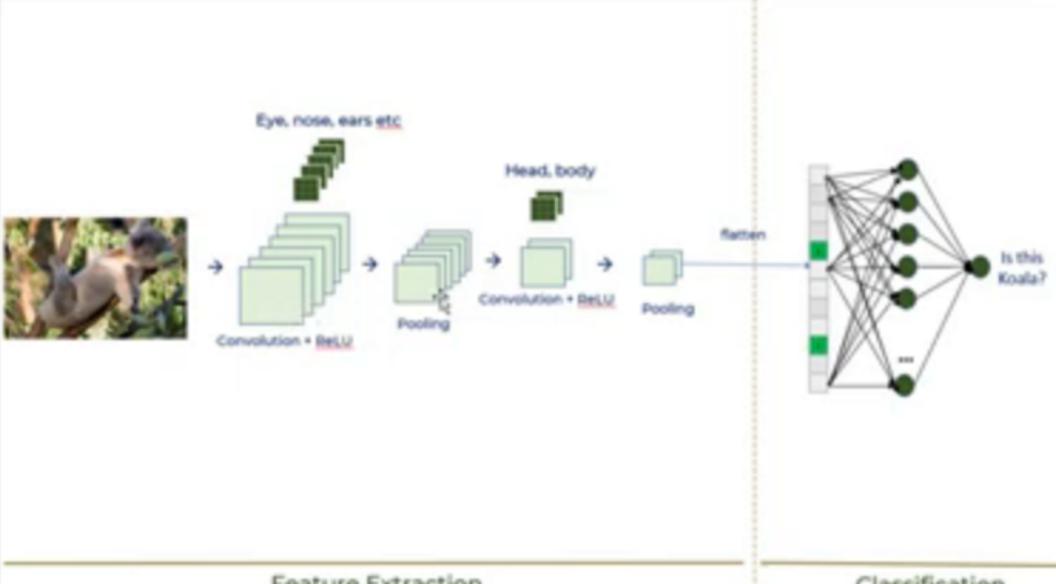
Search

Detailed Panel

Logout

In [50]:

```
50: Koala average bounding box.
51: Benefits of using
52: 
53: 
54: 
55: Check for small Koala's later part of
```

56: 

Find Replace Select Dictate Design Mode Designer

Python 3

In [56]: State 52 of 57

Type here to search

Windows Taskbar icons: File Explorer, Edge, Google Chrome, Microsoft Edge, File Manager, Task View, Taskbar Icons, Taskbar Icons (red), Taskbar Icons (green).

CRM-10 and CRM-100 dataset | Untitled - Jupyter Notebook | 16_imagenet10_small_image_classification | pygpu_performance_test_0ml | pyhandling_imbalanced_data | +

localhost:8888/notebooks/16_imagenet10_small_image_classification/Untitled.ipynb?kernel_name=python3

Logout

In [50]:

```
!pip install tensorflow==2.0.0
```

In [51]:

```
!git clone https://github.com/tensorflow/tensorflow.git
```

In [52]:

The diagram illustrates a neural network architecture for image classification. It starts with an input image of a koala. This image is processed through a series of layers: 'Convolution + ReLU' leads to a feature map; 'Pooling' reduces the spatial dimensions; another 'Convolution + ReLU' layer leads to another feature map; and another 'Pooling' layer further reduces the dimensions. The resulting feature vector is then flattened and passed into a classification layer consisting of multiple neurons. The output neuron is labeled 'Is this Koala?'.

Eye, nose, ears etc

Head, body

Convolution + ReLU

Pooling

Convolution + ReLU

Pooling

Is this Koala?

Feature Extraction

Classification

In [56]:

```
!git clone https://github.com/tensorflow/tensorflow.git
```

jupyter Untitled



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3



```
In [27]: from sklearn.metrics import confusion_matrix , classification_report
import numpy as np
y_pred = ann.predict(X_test)
y_pred_classes = [np.argmax(element) for element in y_pred]

print("Classification Report: \n", classification_report(y_test, y_pred_classes))
```

	precision	recall	f1-score	support
0	0.71	0.35	0.47	1000
1	0.55	0.71	0.62	1000
2	0.33	0.41	0.36	1000
3	0.44	0.15	0.23	1000
4	0.34	0.51	0.40	1000
5	0.43	0.33	0.37	1000
6	0.43	0.68	0.52	1000
7	0.63	0.45	0.53	1000
8	0.57	0.69	0.62	1000
9	0.59	0.48	0.53	1000
accuracy			0.48	10000
macro avg	0.50	0.48	0.47	10000
weighted avg	0.50	0.48	0.47	10000

In []:



Logout

jupyter Untitled Last Checkpoint: 24 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help

Trusted Python 3

	0	0.71	0.35	0.47	1000
1	0.55	0.71	0.62	0.62	1000
2	0.33	0.41	0.36	0.23	1000
3	0.44	0.15	0.23	0.40	1000
4	0.34	0.51	0.40	0.40	1000
5	0.43	0.33	0.37	0.37	1000
6	0.43	0.68	0.52	0.52	1000
7	0.63	0.45	0.53	0.62	1000
8	0.57	0.69	0.62	0.62	1000
9	0.59	0.48	0.53	0.53	1000
accuracy			0.48	0.48	10000
macro avg	0.50	0.48	0.47	0.47	10000
weighted avg	0.50	0.48	0.47	0.47	10000

```
In [ ]: ann = models.Sequential([
    layers.Flatten(input_shape=(32,32,3)),
    layers.Dense(3000, activation='relu'),
    layers.Dense(1000, activation='relu'),
    layers.Dense(10, activation='sigmoid')
])
```

jupyter Untitled Last Checkpoint: 25 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

	0	0.71	0.35	0.47	1000
1	0.55	0.71	0.62	1000	
2	0.33	0.41	0.36	1000	
3	0.44	0.15	0.23	1000	
4	0.34	0.51	0.40	1000	
5	0.43	0.33	0.37	1000	
6	0.43	0.68	0.52	1000	
7	0.63	0.45	0.53	1000	
8	0.57	0.69	0.62	1000	
9	0.59	0.48	0.53	1000	
accuracy			0.48	10000	
macro avg	0.50	0.48	0.47	10000	
weighted avg	0.50	0.48	0.47	10000	

```
In [ ]: cfn = models.Sequential([
    # cnn
    layers.Flatten(input_shape=(32,32,3)),
    layers.Dense(3000, activation='relu'),
    layers.Dense(1000, activation='relu'),
    layers.Dense(10, activation='sigmoid')
])
```

 jupyter Untitled Last Checkpoint: 25 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help

Trusted Python 3

	0.71	0.35	0.47	1000
0	0.55	0.71	0.62	1000
1	0.33	0.41	0.36	1000
2	0.44	0.15	0.23	1000
3	0.34	0.51	0.40	1000
4	0.43	0.33	0.37	1000
5	0.43	0.68	0.52	1000
6	0.63	0.45	0.53	1000
7	0.57	0.69	0.62	1000
8	0.59	0.48	0.53	1000
9				
accuracy			0.48	10000
macro avg	0.50	0.48	0.47	10000
weighted avg	0.50	0.48	0.47	10000

```
In [ ]: cfn = models.Sequential([
    # cnn
    # dense
    layers.Flatten(input_shape=(32,32,3)),
    layers.Dense(3000, activation='relu'),
    layers.Dense(1000, activation='relu'),
    layers.Dense(10, activation='sigmoid')
])
```

jupyter Untitled Last Checkpoint: 25 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

	0	0.71	0.35	0.47	1000
1	0.55	0.71	0.62	0.62	1000
2	0.33	0.41	0.36	0.23	1000
3	0.44	0.15	0.48	0.23	1000
4	0.34	0.51	0.48	0.48	1000
5	0.43	0.33	0.37	0.37	1000
6	0.43	0.68	0.52	0.52	1000
7	0.63	0.45	0.53	0.53	1000
8	0.57	0.69	0.62	0.62	1000
9	0.59	0.48	0.53	0.53	1000
accuracy			0.48	0.48	10000
macro avg	0.50	0.48	0.47	0.47	10000
weighted avg	0.50	0.48	0.47	0.47	10000

```
In [ ]: cfn = models.Sequential([
    # cnn
    # dense
    layers.Flatten(),
    layers.Dense(1000, activation='relu'),
    layers.Dense(10, activation='sigmoid')
])
```

jupyter Untitled Last Checkpoint: 25 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

	0.71	0.35	0.47	1000
0	0.55	0.71	0.62	1000
1	0.33	0.41	0.36	1000
2	0.44	0.15	0.23	1000
3	0.34	0.51	0.40	1000
4	0.43	0.33	0.37	1000
5	0.43	0.68	0.52	1000
6	0.63	0.45	0.53	1000
7	0.57	0.69	0.62	1000
8	0.59	0.48	0.53	1000
9				
accuracy			0.48	10000
macro avg	0.50	0.48	0.47	10000
weighted avg	0.50	0.48	0.47	10000

```
In [ ]: cfn = models.Sequential([
    # cnn
    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='sigmoid')
])
```



Logout

jupyter Untitled Last Checkpoint: 25 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help

Trusted | Python 3

	0.71	0.35	0.47	1000
0	0.55	0.71	0.62	1000
1	0.33	0.41	0.36	1000
2	0.44	0.15	0.23	1000
3	0.34	0.51	0.40	1000
4	0.43	0.33	0.37	1000
5	0.43	0.68	0.52	1000
6	0.63	0.45	0.53	1000
7	0.57	0.69	0.62	1000
8	0.59	0.48	0.53	1000
9				
accuracy			0.48	10000
macro avg	0.50	0.48	0.47	10000
weighted avg	0.50	0.48	0.47	10000

```
In [ ]: cnn = models.Sequential([
    # cnn
    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

jupyter Untitled Last Checkpoint: 25 minutes ago (autosaved)



Logout

File Edit View Insert Cell Kernel Help

Notebook saved Trusted | Python 3

	5	0.43	0.33	0.37	1000
	6	0.43	0.68	0.52	1000
	7	0.63	0.45	0.53	1000
	8	0.57	0.69	0.62	1000
	9	0.59	0.48	0.53	1000
accuracy				0.48	10000
macro avg		0.50	0.48	0.47	10000
weighted avg		0.50	0.48	0.47	10000

```
In [ ]: cnn = models.Sequential([
    # cnn
    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

```
In [ ]:
```

jupyter Untitled Last Checkpoint: 26 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

5	0.43	0.33	0.37	1000
6	0.43	0.68	0.52	1000
7	0.63	0.45	0.53	1000
8	0.57	0.69	0.62	1000
9	0.59	0.48	0.53	1000
accuracy			0.48	10000
macro avg	0.50	0.48	0.47	10000
weighted avg	0.50	0.48	0.47	10000

```
In [ ]: cnn = models.Sequential([
    # cnn
    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

I

```
In [ ]: 1: 0.45
2: 0.67
```

jupyter Untitled Last Checkpoint: 26 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

5	0.43	0.33	0.37	1000
6	0.43	0.68	0.52	1000
7	0.63	0.45	0.53	1000
8	0.57	0.69	0.62	1000
9	0.59	0.48	0.53	1000
accuracy			0.48	10000
macro avg	0.50	0.48	0.47	10000
weighted avg	0.50	0.48	0.47	10000

```
In [ ]: cnn = models.Sequential([
    # cnn
    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

I

```
In [ ]: 1: 0.45
2: 0.67
1: (0.45)
```

jupyter Untitled Last Checkpoint: 26 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

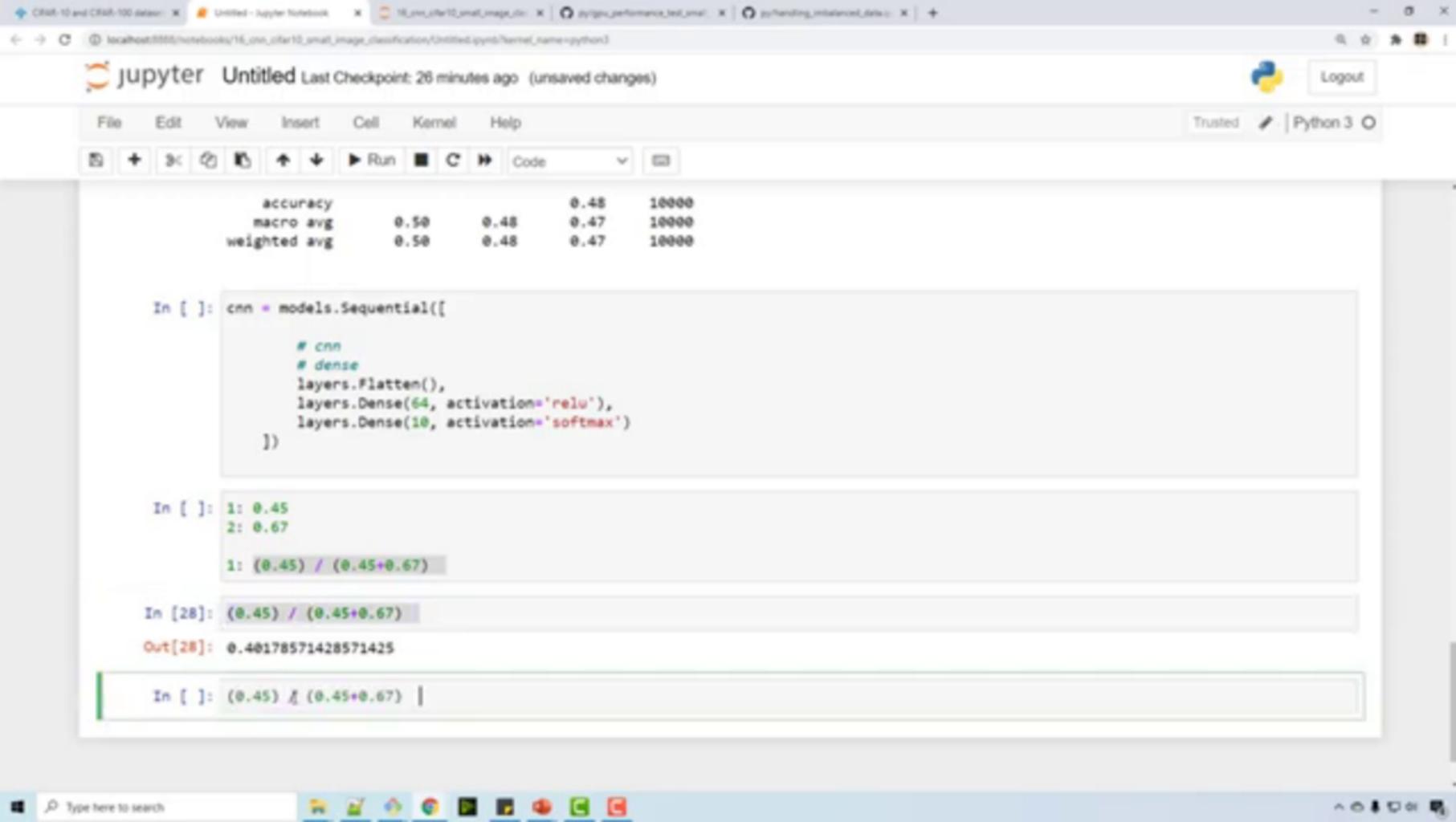
Trusted

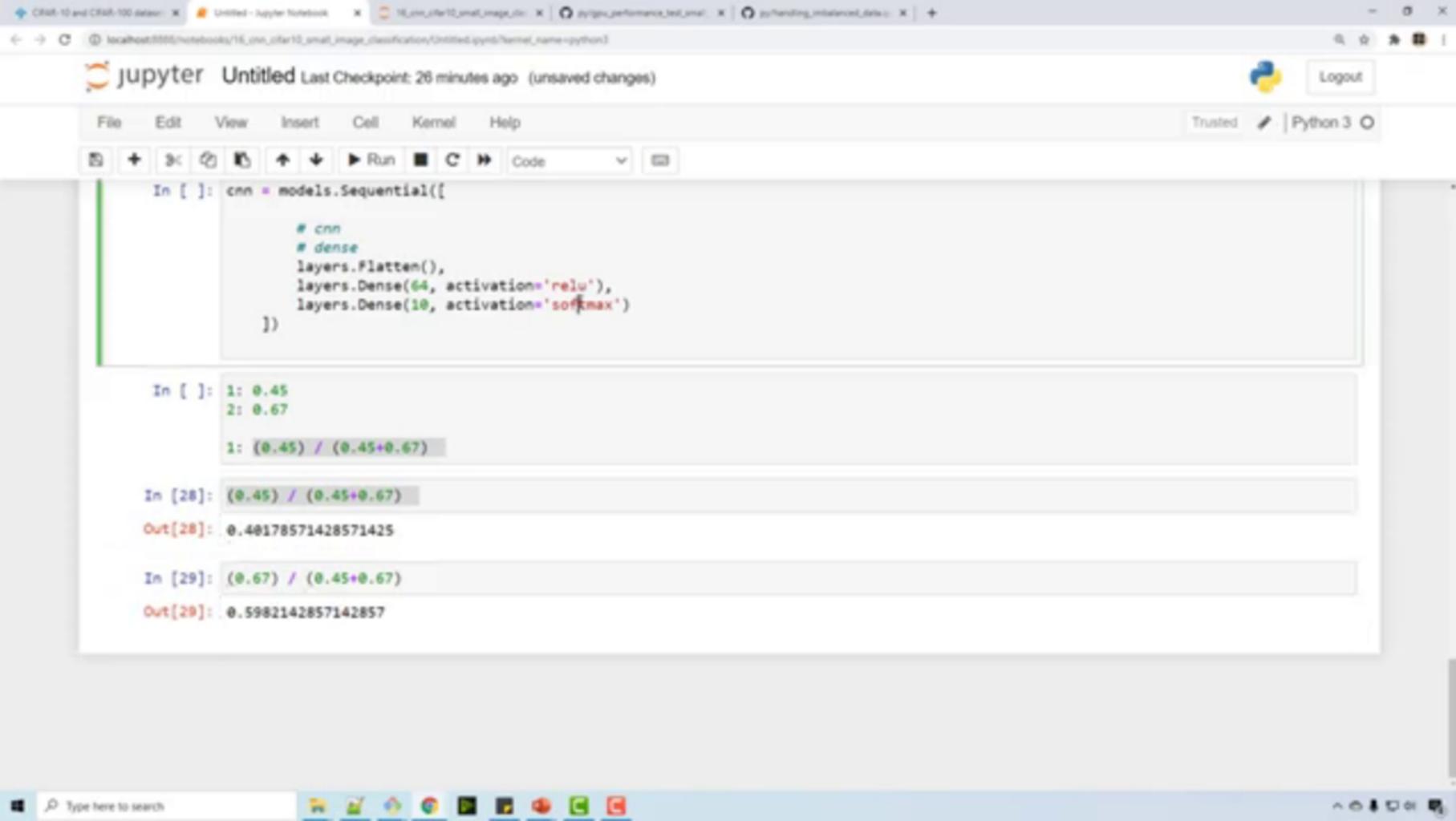
Python 3

5	0.43	0.33	0.37	1000
6	0.43	0.68	0.52	1000
7	0.63	0.45	0.53	1000
8	0.57	0.69	0.62	1000
9	0.59	0.48	0.53	1000
accuracy			0.48	10000
macro avg		0.50	0.48	0.47
weighted avg		0.50	0.48	0.47

```
In [ ]: cnn = models.Sequential([
    # cnn
    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

```
In [ ]: 1: 0.45
2: 0.67
In [ ]: 1: (0.45) / (0.45+0.67)
```





jupyter Untitled Last Checkpoint: 27 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

	2	0.33	0.41	0.36	1000
	3	0.44	0.15	0.23	1000
	4	0.34	0.51	0.48	1000
	5	0.43	0.33	0.37	1000
	6	0.43	0.68	0.52	1000
	7	0.63	0.45	0.53	1000
	8	0.57	0.69	0.62	1000
	9	0.59	0.48	0.53	1000
	accuracy			0.48	10000
	macro avg	0.50	0.48	0.47	10000
	weighted avg	0.50	0.48	0.47	10000

```
In [ ]: cnn = models.Sequential([
    # CNN
    layers.Conv2D(),
    layers.MaxPooling2D(),

    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

CRM-10 and CRM-100 dataset | Untitled - Jupyter Notebook | 16_cnn_cifar10_small_image_classification | 03gpu_performance_test_email | 04handling_imbalanced_data | +

localhost:8888/notebooks/16_cnn_cifar10_small_image_classification/Untitled.ipynb?kernel_name=python3

jupyter

File Home Insert Design Transitions Animations Slide Show Review View Add-ons Help

Search

Dashed Panel Logout

Python 3

In

50 

51 

52 

53 

54 

55 

56 

Eye, nose, ears etc.

Head, body

Convolution + ReLU

Pooling

Convolution + ReLU

Pooling

Is this Koala?

Feature Extraction Classification

Type here to search

File Insert Design Transitions Animations Slide Show Review View Add-ons Help

Search

Dashed Panel Logout

Python 3

CRM-10 and CRM-100 dataset | Untitled - Jupyter Notebook | 16_cnn_cifar10_small_image_classification | pytorch_performance_tutorial | pythontesting_imbalanced_data | +

localhost:8888/notebooks/16_cnn_cifar10_small_image_classification/Untitled.ipynb?kernel_name=python3

Logout

File Home Insert Design Transitions Animations Slide Show Review View Add-ins Help

Search Drawing Editing More Designer

Find Replace Select Dictate Design Notes

41

42 Resting layer is used to reduce the size

43

44

45

46

47

In

Feature Extraction Classification

Eye, nose, ears etc

Head, body

Convolution + ReLU

Pooling

Convolution + ReLU

Pooling

Is this Koala?

Detailed Panel

Type here to search

Python 3

```
graph LR; Input[Input Image] --> Conv1[Convolution + ReLU]; Conv1 --> Pool1[Pooling]; Pool1 --> Conv2[Convolution + ReLU]; Conv2 --> Pool2[Pooling]; Pool2 --> FeatureExtraction[Feature Extraction]; FeatureExtraction --> HeadBody[Head, body]; HeadBody --> FeatureExtraction; FeatureExtraction --> Flatten[Flatten]; Flatten --> FC[Classification Layer]; FC --> Output[Is this Koala?]
```

CRM-10 and CRM-100 dataset | Untitled - Jupyter Notebook | 16_mnist_cifar10_small_image_classification | Untitled.ipynb | kernel_name=python3

localhost:8888/notebooks/16_mnist_cifar10_small_image_classification/Untitled.ipynb?kernel_name=python3

jupyter

Autosave Save

File Home Insert Design Transitions Animations Slide Show Review View Add-ins Help

Search

Detailed Panel Logout

Python 3

In

28

29

30

31

32

33

Slide 30 of 37

Loopy pattern detector

Vertical line detector

Diagonal line detector

Diagram illustrating the detection of specific patterns in a handwritten digit '9' using convolutional filters. The filters are represented as 3x3 grids of weights.

- Loopy pattern detector:** A filter with weights:

1	-1	1
-1	1	-1
1	-1	1

The result shows a green '1' at the center of the digit's loop.
- Vertical line detector:** A filter with weights:

-1	-1	-1
-1	1	-1
-1	-1	-1

The result shows a green '1' at the vertical stroke of the digit.
- Diagonal line detector:** A filter with weights:

1	-1	-1
-1	1	-1
-1	-1	1

The result shows a green '1' at the diagonal stroke of the digit.

Type here to search

Navigation icons: back, forward, search, etc.

Handwritten digit recognition

Loopy pattern detector

$g \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}$

Vertical line detector

$g \cdot \begin{bmatrix} 1 & 1 & -1 \\ 1 & 1 & -1 \\ -1 & -1 & -1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}$

Diagonal line detector

$g \cdot \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & & \\ & 1 & \\ & & 1 \end{bmatrix}$



Logout

jupyter Untitled Last Checkpoint: 28 minutes ago (autosaved)

File Edit View Insert Cell Kernel Help

Trusted

Python 3

	2	0.33	0.41	0.36	1000
	3	0.44	0.15	0.23	1000
	4	0.34	0.51	0.48	1000
	5	0.43	0.33	0.37	1000
	6	0.43	0.68	0.52	1000
	7	0.63	0.45	0.53	1000
	8	0.57	0.69	0.62	1000
	9	0.59	0.48	0.53	1000
	accuracy			0.48	10000
	macro avg	0.50	0.48	0.47	10000
	weighted avg	0.50	0.48	0.47	10000

```
In [ ]: cnn = models.Sequential([
    # CNN
    layers.Conv2D(),
    layers.MaxPooling2D(),
    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```



Type here to search



CRM-10 and CRM-100 datasets | Untitled - Jupyter Notebook | 16_cifar10_small_image_classification | pygpu_performance_test_omni | pyhandling_imbalanced_data | +

localhost:8888/notebooks/16_cifar10_small_image_classification/Untitled.ipynb?kernel_name=python3

jupyter

Autosave **File** Home Insert Design Transitions Animations Slide Show Review View Add-ins Help

Search

Dashed Panel

Logout

Python 3

In

28

29

30

31

32

33

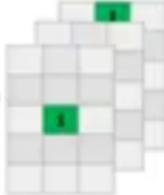
Slide 31 of 37

Feature Maps

Filters







Type here to search

jupyter Untitled

Last Checkpoint: 28 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

	2	0.33	0.41	0.36	1000
	3	0.44	0.15	0.23	1000
	4	0.34	0.51	0.48	1000
	5	0.43	0.33	0.37	1000
	6	0.43	0.68	0.52	1000
	7	0.63	0.45	0.53	1000
	8	0.57	0.69	0.62	1000
	9	0.59	0.48	0.53	1000
accuracy				0.48	10000
macro avg		0.50	0.48	0.47	10000
weighted avg		0.50	0.48	0.47	10000

↳

```
In [ ]: cnn = models.Sequential([
    # CNN
    layers.Conv2D(filters=32,
    layers.MaxPooling2D(),

    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```



Logout

jupyter Untitled Last Checkpoint: 29 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help

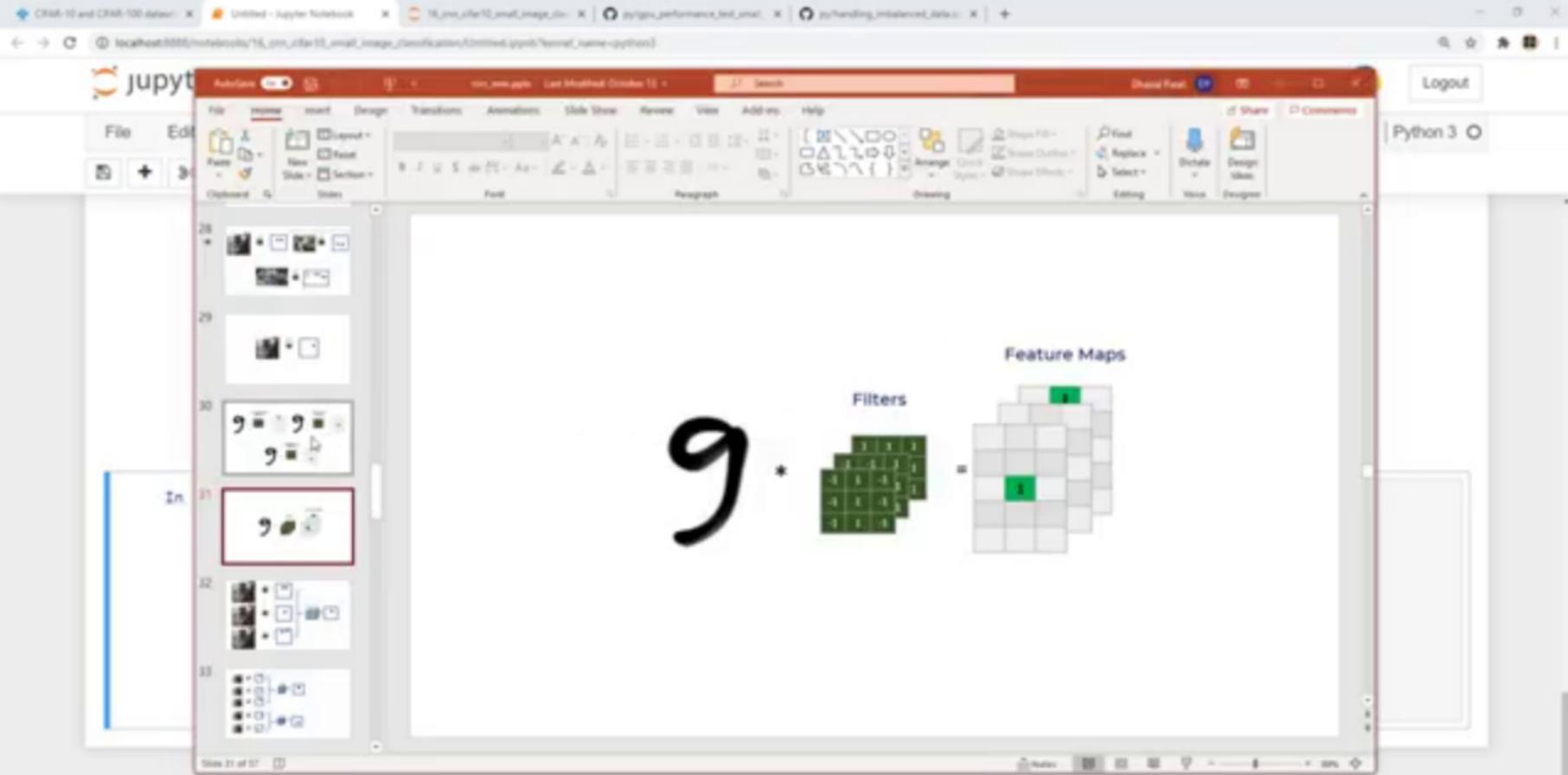
Trusted | Python 3

	2	0.33	0.41	0.36	1000
	3	0.44	0.15	0.23	1000
	4	0.34	0.51	0.48	1000
	5	0.43	0.33	0.37	1000
	6	0.43	0.68	0.52	1000
	7	0.63	0.45	0.53	1000
	8	0.57	0.69	0.62	1000
	9	0.59	0.48	0.53	1000
	accuracy			0.48	10000
	macro avg	0.50	0.48	0.47	10000
	weighted avg	0.50	0.48	0.47	10000

↳

```
In [ ]: cnn = models.Sequential([
    # CNN
    layers.Conv2D(filters=32, []),
    layers.MaxPooling2D(),

    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```



jupyter Untitled Last Checkpoint: 29 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

	2	0.33	0.41	0.36	1000
	3	0.44	0.15	0.23	1000
	4	0.34	0.51	0.48	1000
	5	0.43	0.33	0.37	1000
	6	0.43	0.68	0.52	1000
	7	0.63	0.45	0.53	1000
	8	0.57	0.69	0.62	1000
	9	0.59	0.48	0.53	1000
	accuracy			0.48	10000
	macro avg	0.50	0.48	0.47	10000
	weighted avg	0.50	0.48	0.47	10000

```
In [ ]: cnn = models.Sequential([
    # CNN
    layers.Conv2D(filters=32, activation='relu', kernel_size=(3,3)),
    layers.MaxPooling2D(),

    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

jupyter Untitled Last Checkpoint: 29 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

	2	0.33	0.41	0.36	1000
	3	0.44	0.15	0.23	1000
	4	0.34	0.51	0.48	1000
	5	0.43	0.33	0.37	1000
	6	0.43	0.68	0.52	1000
	7	0.63	0.45	0.53	1000
	8	0.57	0.69	0.62	1000
	9	0.59	0.48	0.53	1000
	accuracy			0.48	10000
	macro avg	0.50	0.48	0.47	10000
	weighted avg	0.50	0.48	0.47	10000

```
In [ ]: cnn = models.Sequential([
    # CNN
    layers.Conv2D(filters=32, activation='relu', kernel_size=(3,3)),
    layers.MaxPooling2D(),

    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

jupyter Untitled Last Checkpoint: 29 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

	2	0.33	0.41	0.36	1000
	3	0.44	0.15	0.23	1000
	4	0.34	0.51	0.48	1000
	5	0.43	0.33	0.37	1000
	6	0.43	0.68	0.52	1000
	7	0.63	0.45	0.53	1000
	8	0.57	0.69	0.62	1000
	9	0.59	0.48	0.53	1000
	accuracy			0.48	10000
	macro avg	0.50	0.48	0.47	10000
	weighted avg	0.50	0.48	0.47	10000

```
In [ ]: cnn = models.Sequential([
    # CNN
    layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu', input_shape=[28, 28, 3]),
    layers.MaxPooling2D(),

    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

Jupyter Notebook interface showing a presentation slide titled "Car Modified Oracle 10".

The slide content illustrates three types of line detectors applied to a handwritten digit '9':

- Loopy pattern detector:** Shows a 3x3 kernel with values [1, 1, 1; 1, -1, 1; 1, 1, 1]. The result shows a green '1' at the center of the digit's loop.
- Vertical line detector:** Shows a 3x3 kernel with values [1, 1, 1; 1, 1, 1; -1, -1, -1]. The result shows a green '1' at the vertical stroke of the digit.
- Diagonal line detector:** Shows a 3x3 kernel with values [1, 1, 1; 1, -1, 1; 1, 1, 1]. The result shows a green '1' at the diagonal stroke of the digit.

The Jupyter interface includes a toolbar, a sidebar with file operations, and a search bar at the top. The bottom navigation bar shows the current slide number (Slide 30 of 37) and other slide controls.

jupyter Untitled Last Checkpoint: 30 minutes ago (autosaved)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3

	2	0.33	0.41	0.36	1000
	3	0.44	0.15	0.23	1000
	4	0.34	0.51	0.48	1000
	5	0.43	0.33	0.37	1000
	6	0.43	0.68	0.52	1000
	7	0.63	0.45	0.53	1000
	8	0.57	0.69	0.62	1000
	9	0.59	0.48	0.53	1000
	accuracy			0.48	10000
	macro avg	0.50	0.48	0.47	10000
	weighted avg	0.50	0.48	0.47	10000

```
In [ ]: cnn = models.Sequential([
    # CNN
    layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu', input_shape=(32,32,3)),
    layers.MaxPooling2D(),

    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

jupyter Untitled Last Checkpoint: 30 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

2	0.33	0.41	0.36	1000
3	0.44	0.15	0.23	1000
4	0.34	0.51	0.48	1000
5	0.43	0.33	0.37	1000
6	0.43	0.68	0.52	1000
7	0.63	0.45	0.53	1000
8	0.57	0.69	0.62	1000
9	0.59	0.48	0.53	1000
accuracy			0.48	10000
macro avg		0.50	0.48	0.47
weighted avg		0.50	0.48	0.47

```
In [ ]: cnn = models.Sequential([
    # CNN
    layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu', input_shape=(32,32,3)),
    layers.MaxPooling2D((2,2)),

    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

jupyter Untitled Last Checkpoint: 30 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

2	0.33	0.41	0.36	1000
3	0.44	0.15	0.23	1000
4	0.34	0.51	0.48	1000
5	0.43	0.33	0.37	1000
6	0.43	0.68	0.52	1000
7	0.63	0.45	0.53	1000
8	0.57	0.69	0.62	1000
9	0.59	0.48	0.53	1000
accuracy			0.48	10000
macro avg		0.50	0.48	0.47
weighted avg		0.50	0.48	0.47

```
In [ ]: cnn = models.Sequential([
    # CNN
    layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu', input_shape=(32,32,3)),
    layers.MaxPooling2D((2,2)),
    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

jupyter Untitled Last Checkpoint: 30 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

2	0.33	0.41	0.36	1000
3	0.44	0.15	0.23	1000
4	0.34	0.51	0.48	1000
5	0.43	0.33	0.37	1000
6	0.43	0.68	0.52	1000
7	0.63	0.45	0.53	1000
8	0.57	0.69	0.62	1000
9	0.59	0.48	0.53	1000
accuracy			0.48	10000
macro avg		0.50	0.48	10000
weighted avg		0.50	0.48	10000

```
In [ ]: cnn = models.Sequential([
    # CNN
    layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu', input_shape=(32,32,3)),
    layers.MaxPooling2D((2,2)),
    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

jupyter Untitled Last Checkpoint: 30 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

	2	0.33	0.41	0.36	1000
	3	0.44	0.15	0.23	1000
	4	0.34	0.51	0.48	1000
	5	0.43	0.33	0.37	1000
	6	0.43	0.68	0.52	1000
	7	0.63	0.45	0.53	1000
	8	0.57	0.69	0.62	1000
	9	0.59	0.48	0.53	1000
accuracy				0.48	10000
macro avg		0.50	0.48	0.47	10000
weighted avg		0.50	0.48	0.47	10000

```
In [ ]: cnn = models.Sequential([
    # CNN
    layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu', input_shape=(32,32,3)),
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```



Logout

jupyter Untitled Last Checkpoint: 30 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help

Trusted | Python 3

	7	0.63	0.45	0.53	1000
	8	0.57	0.69	0.62	1000
	9	0.59	0.48	0.53	1000
	accuracy			0.48	10000
	macro avg	0.50	0.48	0.47	10000
	weighted avg	0.50	0.48	0.47	10000

```
In [ ]: cnn = models.Sequential([
    # CNN
    layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu', input_shape=(32,32,3)),
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

jupyter Untitled Last Checkpoint: 31 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3



accuracy			0.48	10000
macro avg	0.50	0.48	0.47	10000
weighted avg	0.50	0.48	0.47	10000

```
In [30]: cnn = models.Sequential([
    # CNN
    layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu', input_shape=(32,32,3)),
    layers.MaxPooling2D((2,2)),

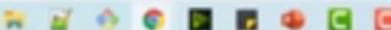
    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    # Dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

```
In [ ]: model.compile(optimizer='adam',
                      loss='sparse_categorical_crossentropy',
                      metrics=['accuracy'])
```



Type here to search



jupyter Untitled Last Checkpoint: 31 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3



accuracy			0.48	10000
macro avg	0.50	0.48	0.47	10000
weighted avg	0.50	0.48	0.47	10000

```
In [30]: cnn = models.Sequential([
    # cnn
    layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu', input_shape=(32,32,3)),
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

```
In [ ]: model.compile(optimizer='adam',
                      loss='sparse_categorical_crossentropy',
                      metrics=['accuracy'])
```

jupyter Untitled Last Checkpoint: 31 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3



accuracy	0.48	10000
macro avg	0.50	0.48
weighted avg	0.50	0.48

```
In [30]: cnn = models.Sequential([
    # cnn
    layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu', input_shape=(32,32,3)),
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

```
In [31]: cnn.compile(optimizer='adam',
                    loss='sparse_categorical_crossentropy',
                    metrics=['accuracy'])
```

```
-----
NameError                                 Traceback (most recent call last)
<ipython-input-31-beaf0aeade08> in <module>
----> 1 model.compile(optimizer='adam',
      2         loss='sparse_categorical_crossentropy',
      3         metrics=['accuracy'])

NameError: name 'model' is not defined
```



Type here to search

jupyter Untitled Last Checkpoint: 31 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3

accuracy			0.48	10000
macro avg	0.50	0.48	0.47	10000
weighted avg	0.50	0.48	0.47	10000

```
In [30]: cnn = models.Sequential([
    # cnn
    layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu', input_shape=(32,32,3)),
    layers.MaxPooling2D((2,2)),

    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),

    # dense
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])
```

```
In [32]: cnn.compile(optimizer='adam',
                    loss='sparse_categorical_crossentropy',
                    metrics=['accuracy'])
```

In []:

[]



Logout

jupyter Untitled Last Checkpoint: 32 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help

Trusted

Python 3



```
# dense
layers.Flatten(),
layers.Dense(64, activation='relu'),
layers.Dense(10, activation='softmax')
])
```

```
In [32]: cnn.compile(optimizer='adam',
                    loss='sparse_categorical_crossentropy',
                    metrics=['accuracy'])
```

```
In [33]: cnn.fit(X_train, y_train, epochs=10)
```

```
Epoch 1/10
1563/1563 [*****] - 3s 2ms/step - loss: 1.4027 - accuracy: 0.4997
Epoch 2/10
1563/1563 [*****] - 3s 2ms/step - loss: 1.0595 - accuracy: 0.6280
Epoch 3/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.9334 - accuracy: 0.6757
Epoch 4/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.8446 - accuracy: 0.7062
Epoch 5/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.7665 - accuracy: 0.7355
Epoch 6/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.7023 - accuracy: 0.7551
Epoch 7/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.6441 - accuracy: 0.7759
Epoch 8/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.5879 - accuracy: 0.7938
Epoch 9/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.5357 - accuracy: 0.8124
Epoch 10/10
```

Type here to search



jupyter Untitled Last Checkpoint: 32 minutes ago (unsaved changes)



1000

File Edit View Insert Cell Kernel Help

Trusted

```
loss='sparse_categorical_crossentropy',
metrics=['accuracy'])
```

```
In [33]: cnn.fit(X_train, y_train, epochs=10)
```

```
Epoch 1/10  
1563/1563 [=====] - 3s 2ms/step - loss: 1.4027 - accuracy: 0.499  
Epoch 2/10  
1563/1563 [=====] - 3s 2ms/step - loss: 1.0595 - accuracy: 0.628  
Epoch 3/10  
1563/1563 [=====] - 3s 2ms/step - loss: 0.9334 - accuracy: 0.675  
Epoch 4/10  
1563/1563 [=====] - 3s 2ms/step - loss: 0.8446 - accuracy: 0.706  
Epoch 5/10  
1563/1563 [=====] - 3s 2ms/step - loss: 0.7665 - accuracy: 0.735  
Epoch 6/10  
1563/1563 [=====] - 3s 2ms/step - loss: 0.7023 - accuracy: 0.755  
Epoch 7/10  
1563/1563 [=====] - 3s 2ms/step - loss: 0.6441 - accuracy: 0.775  
Epoch 8/10  
1563/1563 [=====] - 3s 2ms/step - loss: 0.5879 - accuracy: 0.793  
Epoch 9/10  
1563/1563 [=====] - 3s 2ms/step - loss: 0.5357 - accuracy: 0.812  
Epoch 10/10  
1563/1563 [=====] - 3s 2ms/step - loss: 0.4869 - accuracy: 0.830
```

Out[33]: <tensorflow.python.keras.callbacks.History at 0x252b0b1d080>

CRM-10 and CRM-100 dataset x Untitled - Jupyter Notebook x 18_cnn_cifar10_smali_image_classification x pygpu_performance_test_smali x pyhandling_imbalanced_data x +

localhost:8888/notebooks/18_cnn_cifar10_smali_image_classification/Untitled.ipynb?kernel_name=python3

jupyter Untitled Last Checkpoint: 32 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [23]:

```
ann = models.Sequential([
    layers.Flatten(input_shape=(32,32,3)),
    layers.Dense(1000, activation='relu'),
    layers.Dense(1000, activation='relu'),
    layers.Dense(10, activation='sigmoid')
])

ann.compile(optimizer='SGD',
            loss='sparse_categorical_crossentropy',
            metrics=['accuracy'])

ann.fit(X_train, y_train, epochs=5)
```

Epoch 1/5
1563/1563 [=====] - 3s 2ms/step - loss: 1.8591 - accuracy: 0.3366
Epoch 2/5
1563/1563 [=====] - 3s 2ms/step - loss: 1.6584 - accuracy: 0.4129
Epoch 3/5
1563/1563 [=====] - 3s 2ms/step - loss: 1.5704 - accuracy: 0.4474
Epoch 4/5
1563/1563 [=====] - 3s 2ms/step - loss: 1.5083 - accuracy: 0.4677
Epoch 5/5
1563/1563 [=====] - 3s 2ms/step - loss: 1.4576 - accuracy: 0.4857

Out[23]: <tensorflow.python.keras.callbacks.History at 0x24c0976edf0>

In [24]:

```
ann.evaluate(X_test, y_test)
```

313/313 [=====] - 0s 1ms/step - loss: 1.4823 - accuracy: 0.4759

Out[24]: 0.4759

Type here to search

jupyter Untitled Last Checkpoint: 33 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3

Run Cell Code

In [33]: `cnn.fit(x_train, y_train, epochs=10)`

```
Epoch 1/10
1563/1563 [*****] - 3s 2ms/step - loss: 1.4027 - accuracy: 0.4997
Epoch 2/10
1563/1563 [*****] - 3s 2ms/step - loss: 1.0595 - accuracy: 0.6280
Epoch 3/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.9334 - accuracy: 0.6757
Epoch 4/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.8446 - accuracy: 0.7062
Epoch 5/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.7665 - accuracy: 0.7355
Epoch 6/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.7023 - accuracy: 0.7551
Epoch 7/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.6441 - accuracy: 0.7759
Epoch 8/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.5879 - accuracy: 0.7938
Epoch 9/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.5357 - accuracy: 0.8124
Epoch 10/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.4869 - accuracy: 0.8382
```

In []:

jupyter Untitled Last Checkpoint: 33 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted Python 3

Run Cell Kernel Help

In [33]: `cnn.fit(X_train, y_train, epochs=10)`

```
Epoch 1/10
1563/1563 [*****] - 3s 2ms/step - loss: 1.4027 - accuracy: 0.4997
Epoch 2/10
1563/1563 [*****] - 3s 2ms/step - loss: 1.0595 - accuracy: 0.6280
Epoch 3/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.9334 - accuracy: 0.6757
Epoch 4/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.8446 - accuracy: 0.7062
Epoch 5/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.7665 - accuracy: 0.7355
Epoch 6/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.7023 - accuracy: 0.7551
Epoch 7/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.6441 - accuracy: 0.7759
Epoch 8/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.5879 - accuracy: 0.7938
Epoch 9/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.5357 - accuracy: 0.8124
Epoch 10/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.4869 - accuracy: 0.8382
```

In [34]: `cnn.evaluate(X_test,y_test)`

```
313/313 [*****] - 0s 1ms/step - loss: 1.0007 - accuracy: 0.6956
```

Out[34]: [1.0007039470672607, 0.6955999732017517]

CRM-10 and CRM-100 dataset x Untitled - Jupyter Notebook x 18_cnn_cifar10_small_image_classification x pygpu_performance_test_omni x pyhandling_imbalanced_data x +

localhost:8888/notebooks/18_cnn_cifar10_small_image_classification/Untitled.ipynb?kernel_name=python3

jupyter Untitled Last Checkpoint: 33 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

Code

```
layers.MaxPooling2D((2,2)),  
    layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),  
    layers.MaxPooling2D((2, 2)),  
  
    # dense  
    layers.Flatten(),  
    layers.Dense(64, activation='relu'),  
    layers.Dense(10, activation='softmax')  
])  
  
In [32]: cnn.compile(optimizer='adam',  
                    loss='sparse_categorical_crossentropy',  
                    metrics=['accuracy'])
```

In [33]: `cnn.fit(X_train, y_train, epochs=10)`

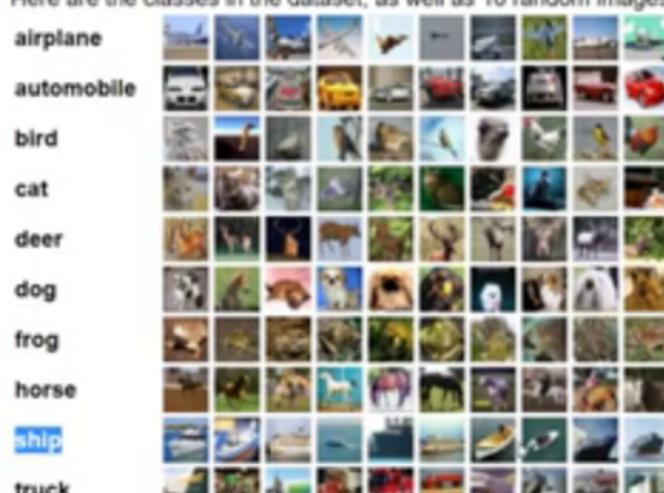
```
Epoch 1/10  
1563/1563 [*****] - 3s 2ms/step - loss: 1.4027 - accuracy: 0.4997  
Epoch 2/10  
1563/1563 [*****] - 3s 2ms/step - loss: 1.0595 - accuracy: 0.6280  
Epoch 3/10  
1563/1563 [*****] - 3s 2ms/step - loss: 0.9334 - accuracy: 0.6757  
Epoch 4/10  
1563/1563 [*****] - 3s 2ms/step - loss: 0.8446 - accuracy: 0.7062  
Epoch 5/10  
1563/1563 [*****] - 3s 2ms/step - loss: 0.7665 - accuracy: 0.7355  
Epoch 6/10  
1563/1563 [*****] - 3s 2ms/step - loss: 0.7023 - accuracy: 0.7551  
Epoch 7/10  
1563/1563 [*****] - 3s 2ms/step - loss: 0.6441 - accuracy: 0.7759
```

Type here to search

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset, as well as 10 random images from each:



The classes are completely mutually exclusive. There is no overlap between automobiles and trucks. "Automobile" includes sedans, SUVs, things of that sort. "Truck" includes only big trucks. Neither includes pickup trucks.

Download

If you're going to use this dataset, please cite the tech report at the bottom of this page.

Version	Size	md5sum
cifar-10-python.tar.gz	83.3 MB	1f37e04c0e44e3d8a56a71a93c93ecbd

Type here to search





Logout

jupyter Untitled Last Checkpoint: 33 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help

Trusted

Python 3

```
Epoch 2/10
1563/1563 [*****] - 3s 2ms/step - loss: 1.0595 - accuracy: 0.6280
Epoch 3/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.9334 - accuracy: 0.6757
Epoch 4/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.8446 - accuracy: 0.7062
Epoch 5/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.7665 - accuracy: 0.7355
Epoch 6/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.7023 - accuracy: 0.7551
Epoch 7/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.6441 - accuracy: 0.7759
Epoch 8/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.5879 - accuracy: 0.7938
Epoch 9/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.5357 - accuracy: 0.8124
Epoch 10/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.4869 - accuracy: 0.8302
```

In [34]: `cnn.evaluate(X_test,y_test)`

```
313/313 [*****] - 0s 1ms/step - loss: 1.0087 - accuracy: 0.6956
```

Out[34]: [1.0087039470672607, 0.6955999732017517]

jupyter Untitled Last Checkpoint: 33 minutes ago (autosaved)



Logout

File Edit View Insert Cell Kernel Help

Notebook saved Trusted Python 3

```
1563/1563 [*****] - 3s 2ms/step - loss: 0.9334 - accuracy: 0.6757
Epoch 4/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.8446 - accuracy: 0.7062
Epoch 5/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.7665 - accuracy: 0.7355
Epoch 6/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.7023 - accuracy: 0.7551
Epoch 7/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.6441 - accuracy: 0.7759
Epoch 8/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.5879 - accuracy: 0.7938
Epoch 9/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.5357 - accuracy: 0.8124
Epoch 10/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.4869 - accuracy: 0.8382
```

In [34]: `cnn.evaluate(X_test,y_test)`

```
313/313 [*****] - 0s 1ms/step - loss: 1.0087 - accuracy: 0.6956
```

Out[34]: [1.0087039470672607, 0.6955999732017517]

In []:

!

jupyter Untitled Last Checkpoint: 34 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

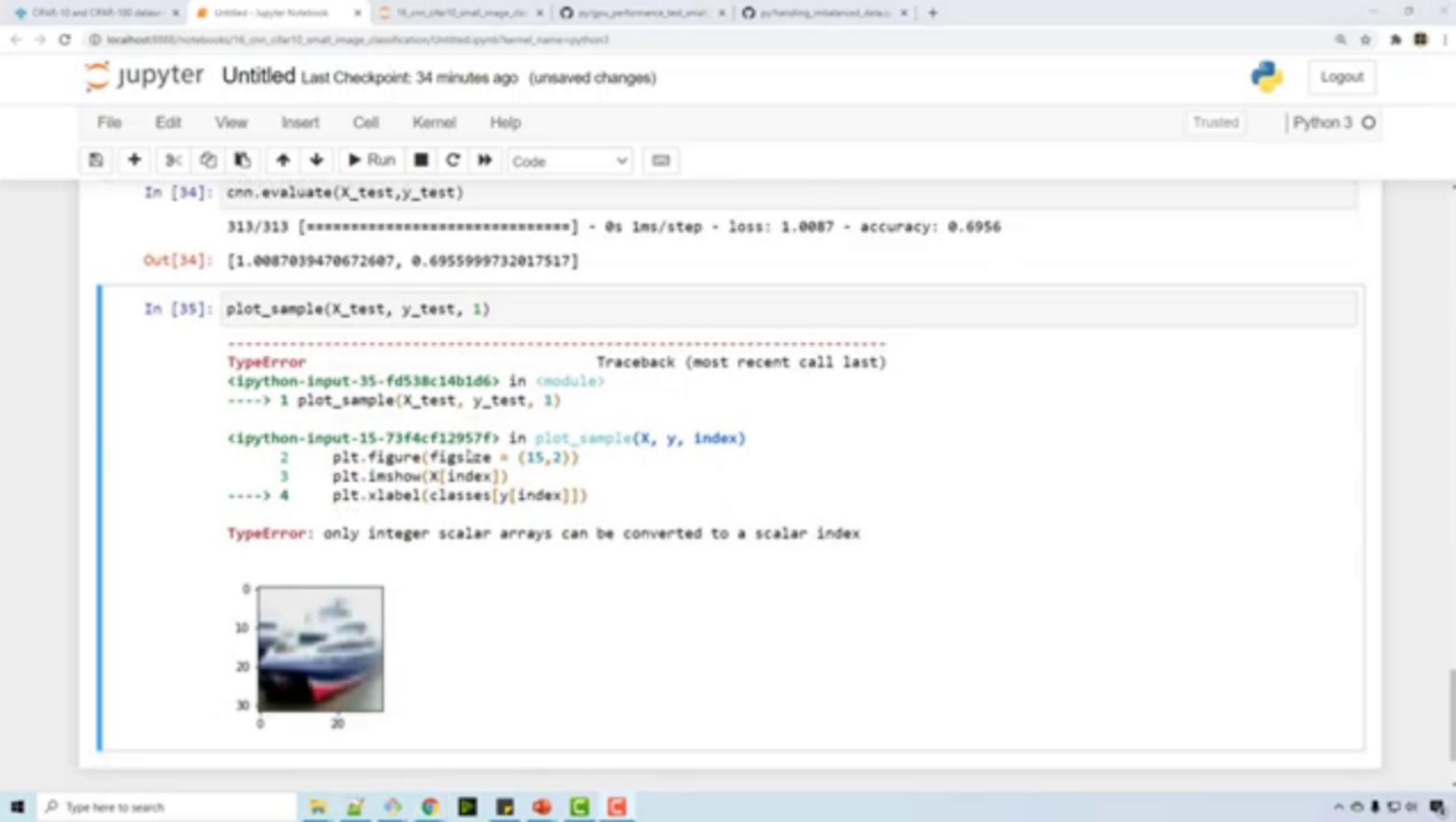
```
1563/1563 [*****] - 3s 2ms/step - loss: 0.9334 - accuracy: 0.6757
Epoch 4/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.8446 - accuracy: 0.7062
Epoch 5/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.7665 - accuracy: 0.7355
Epoch 6/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.7023 - accuracy: 0.7551
Epoch 7/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.6441 - accuracy: 0.7759
Epoch 8/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.5879 - accuracy: 0.7938
Epoch 9/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.5357 - accuracy: 0.8124
Epoch 10/10
1563/1563 [*****] - 3s 2ms/step - loss: 0.4869 - accuracy: 0.8382
```

In [34]: `cnn.evaluate(X_test,y_test)`

313/313 [*****] - 0s 1ms/step - loss: 1.0087 - accuracy: 0.6956

Out[34]: [1.0087039470672607, 0.6955999732017517]

In []: `plot_sample(X_test, y_test, 1)`



```
In [34]: cnn.evaluate(X_test,y_test)

313/313 [=====] - 0s 1ms/step - loss: 1.0087 - accuracy: 0.6956

Out[34]: [1.0087039470672607, 0.6955999732017517]
```

```
In [35]: plot_sample(X_test, y_test, 1)

-----
TypeError                                 Traceback (most recent call last)
<ipython-input-35-fd538c14b1d6> in <module>
----> 1 plot_sample(X_test, y_test, 1)

<ipython-input-15-73f4cf12957f> in plot_sample(X, y, index)
    2     plt.figure(figsize = (15,2))
    3     plt.imshow(X[index])
----> 4     plt.xlabel(classes[y[index]])

TypeError: only integer scalar arrays can be converted to a scalar index
```





Logout

jupyter Untitled Last Checkpoint: 34 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help

Trusted | Python 3



Out[34]: [1.00087039470672607, 0.6955999732017517]

In []: y| I

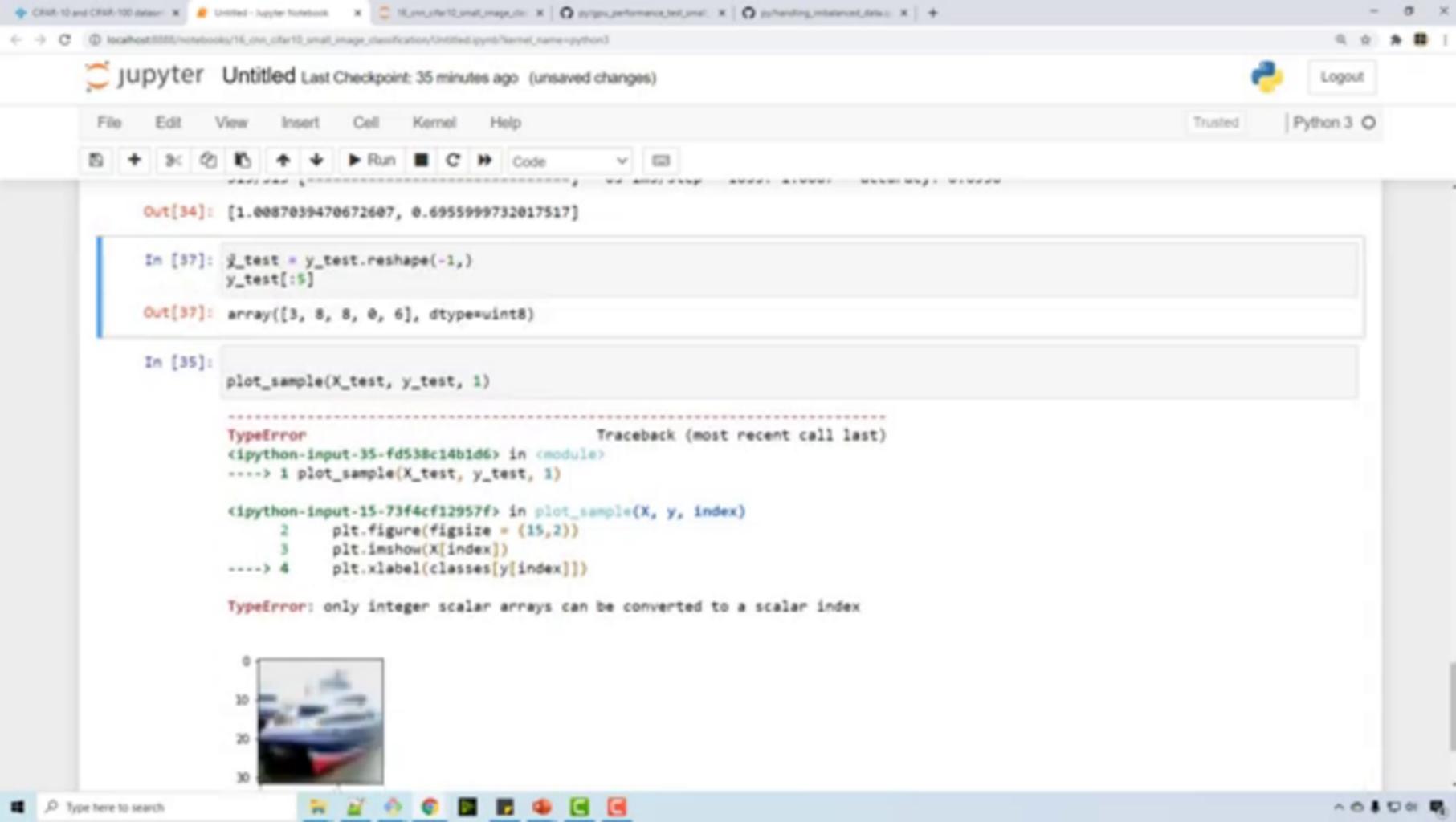
In [35]: y_test = y_test.reshape(-1,)
plot_sample(X_test, y_test, 1)

```
-----  
TypeError                                 Traceback (most recent call last)  
<ipython-input-35-fd538c14b1d6> in <module>  
----> 1 plot_sample(X_test, y_test, 1)
```

```
<ipython-input-15-73f4cf12957f> in plot_sample(X, y, index)  
    2     plt.figure(figsize = (15,2))  
    3     plt.imshow(X[index])  
----> 4     plt.xlabel(classes[y[index]])
```

TypeError: only integer scalar arrays can be converted to a scalar index





```
Out[34]: [1.00087039470672607, 0.6955999732017517]
```

```
In [37]: y_test = y_test.reshape(-1,  
y_test[:5]
```

```
Out[37]: array([3, 8, 8, 0, 6], dtype=uint8)
```

```
In [35]:  
plot_sample(X_test, y_test, 1)
```

```
-----  
TypeError                                 Traceback (most recent call last)  
<ipython-input-35-fd538c14b1d6> in <module>  
----> 1 plot_sample(X_test, y_test, 1)  
  
<ipython-input-15-73f4cf12957f> in plot_sample(X, y, index)  
    2     plt.figure(figsize = (15,2))  
    3     plt.imshow(X[index])  
----> 4     plt.xlabel(classes[y[index]])
```

```
TypeError: only integer scalar arrays can be converted to a scalar index
```



jupyter Untitled Last Checkpoint: 35 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3

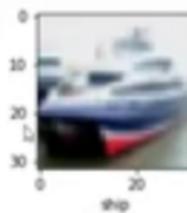
In [34]: `cnn.evaluate(X_test,y_test)`

313/313 [=====] - 0s 1ms/step - loss: 1.0087 - accuracy: 0.6956

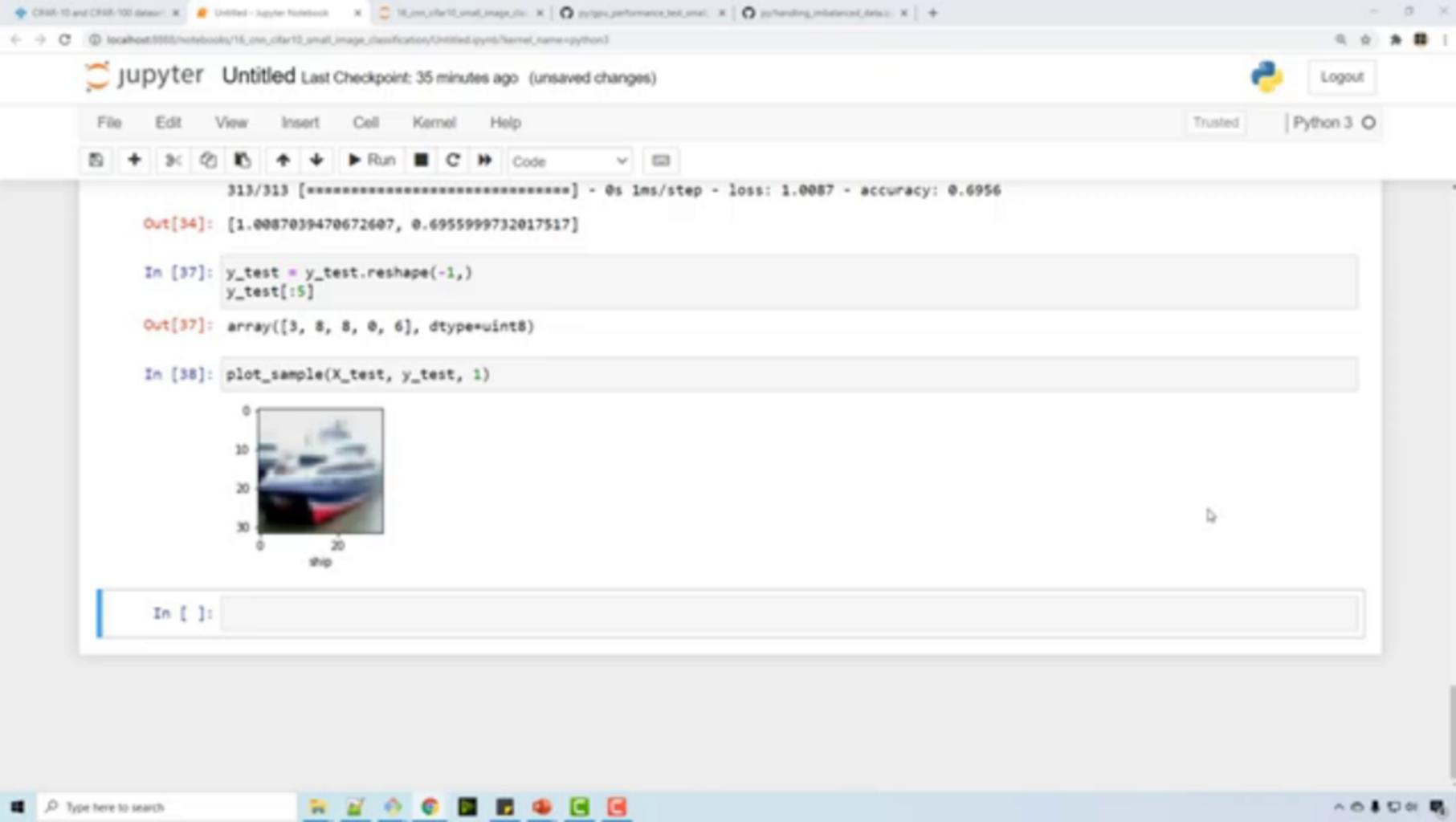
Out[34]: [1.0087039470672607, 0.6955999732017517]

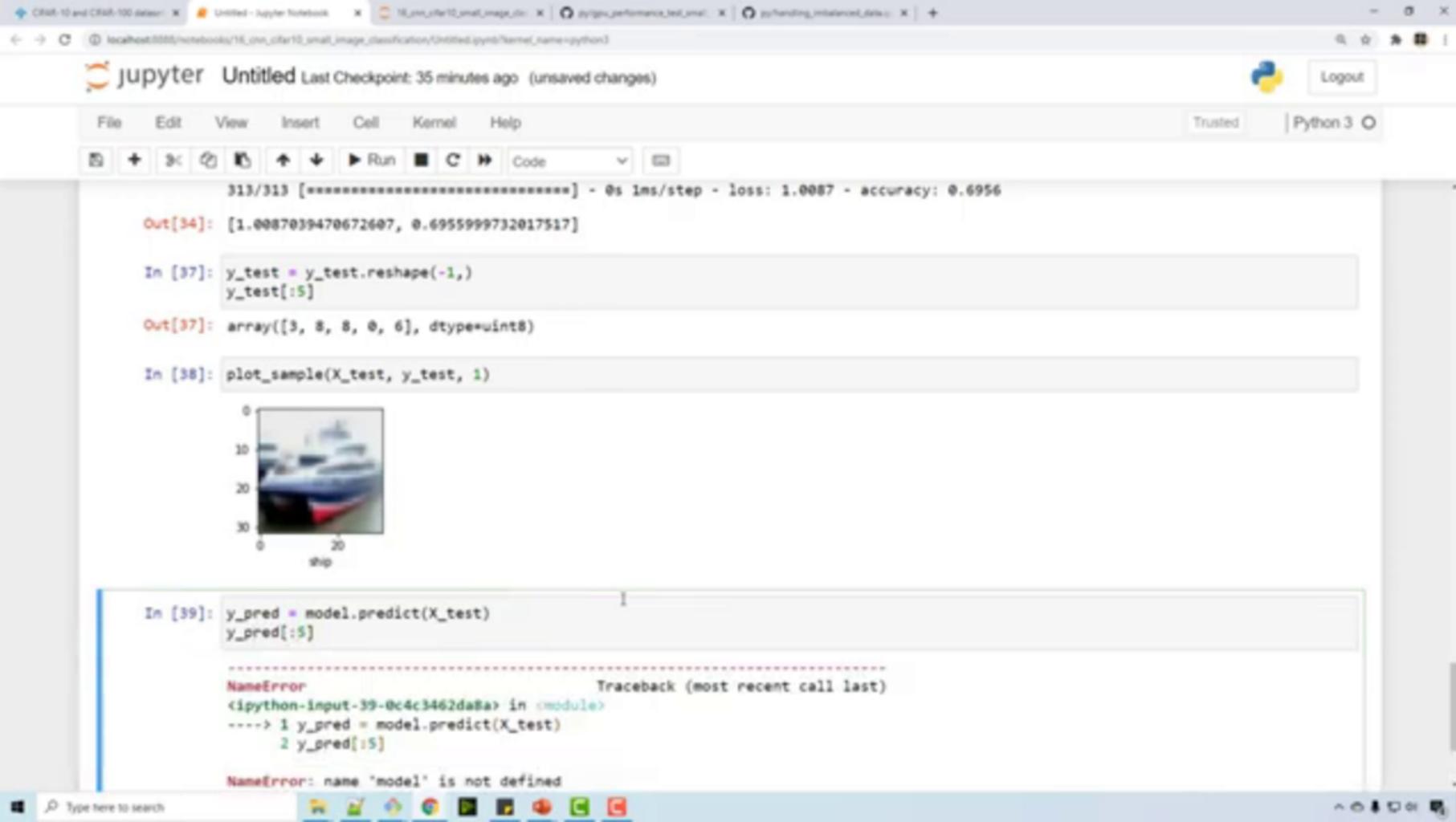
In [37]: `y_test = y_test.reshape(-1,)
y_test[:5]`

Out[37]: array([3, 8, 8, 0, 6], dtype=uint8)

In [38]: `plot_sample(X_test, y_test, 1)`

In []:





jupyter Untitled Last Checkpoint: 35 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3



313/313 [=====] - 0s 1ms/step - loss: 1.0087 - accuracy: 0.6956

Out[34]: [1.0087039470672607, 0.6955999732017517]

In [37]: y_test = y_test.reshape(-1)
y_test[:5]

Out[37]: array([3, 8, 8, 0, 6], dtype=uint8)

In [38]: plot_sample(X_test, y_test, 1)



In [39]: y_pred = model.predict(X_test)
y_pred[:5]

```
NameError                                 Traceback (most recent call last)
<ipython-input-39-0c4c3462da8a> in <module>
----> 1 y_pred = model.predict(X_test)
      2 y_pred[:5]
```

NameError: name 'model' is not defined

Type here to search



jupyter Untitled Last Checkpoint: 35 minutes ago (unsaved changes)

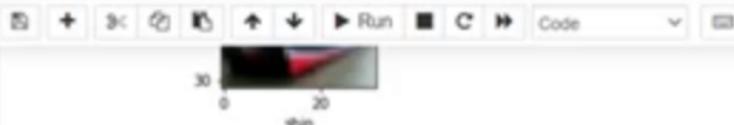


Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3



```
In [48]: y_pred = cnn.predict(X_test)  
y_pred[:5]
```

```
Out[48]: array([[ 3.91144975e-04,  4.33854322e-04,  1.43723926e-04,  3.99961412e-03,  
   1.47574965e-05,  5.83668640e-01,  8.97241412e-05,  3.79756239e-05,  
   5.23986659e-03,  3.95585485e-05],  
[ 1.05845742e-04,  4.75216005e-03,  1.78524562e-09,  4.15040696e-10,  
   4.30966957e-10,  5.29897237e-10,  9.62223190e-10,  1.35456984e-14,  
   9.95130420e-01,  1.16837355e-05],  
[ 1.01393769e-02,  1.31144105e-02,  1.32235728e-04,  3.10876094e-05,  
   4.81918505e-06,  4.86466633e-05,  3.23765053e-05,  4.00682353e-07,  
   9.75176334e-01,  1.32042286e-03],  
[ 9.41294730e-01,  2.95371632e-04,  4.70674075e-02,  4.77683637e-03,  
   4.87070996e-03,  3.40781081e-05,  2.73908488e-04,  2.67446762e-06,  
   1.18420972e-03,  2.00188282e-04],  
[ 1.18030968e-07,  1.14894130e-04,  7.65499286e-03,  8.58013406e-02,  
   5.15409887e-01,  4.64179739e-03,  3.86348605e-01,  6.14111229e-07,  
   2.77474282e-05,  2.12781082e-08]], dtype='float32')
```

jupyter Untitled Last Checkpoint: 35 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3



```
In [48]: y_pred = cnn.predict(X_test)
y_pred[:5]
```

```
Out[48]: array([[3.91144975e-04, 4.33854322e-04, 1.43723926e-04, 3.99961412e-01,
   1.47574965e-05, 5.93668640e-01, 6.97241412e-05, 3.79756239e-05,
   5.23908669e-03, 3.95585485e-05],
 [[1.05845742e-04, 4.7521605e-03, 1.78524562e-09, 4.15040696e-10,
   4.30966957e-10, 5.29897237e-10, 9.62223190e-10, 1.35456984e-14,
   9.95130420e-01, 1.16837355e-05],
 [[1.01393769e-02, 1.31144105e-02, 1.32235728e-04, 3.10876094e-05,
   4.81918505e-06, 4.86466633e-05, 3.23765053e-05, 4.00682353e-07,
   9.75176334e-01, 1.32042286e-03],
 [[9.41294730e-01, 2.95371632e-04, 4.70674075e-02, 4.77683637e-03,
   4.87070996e-03, 3.40781081e-05, 2.73908488e-04, 2.67446762e-06,
   1.18420972e-03, 2.00188282e-04],
 [[1.18030968e-07, 1.14894130e-04, 7.65499286e-03, 8.58013406e-02,
   5.15409887e-01, 4.64179739e-03, 3.86348605e-01, 6.14111229e-07,
   2.77474282e-05, 2.12781082e-08]], dtype='float32')
```

jupyter Untitled Last Checkpoint: 36 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

```
In [48]: y_pred = cnn.predict(X_test)
y_pred[:5]
```

```
Out[48]: array([[3.91144975e-04, 4.33854322e-04, 1.43723926e-04, 3.99961412e-01,
   1.47574965e-05, 5.93668640e-01, 6.97241412e-05, 3.79756239e-05,
   5.23908669e-03, 3.95585485e-05],
 [[1.05845742e-04, 4.75216005e-03, 1.78524562e-09, 4.15040696e-10,
   4.30966957e-10, 5.29897237e-10, 9.62223190e-10, 1.35456984e-14,
   9.95130420e-01, 1.16837355e-05],
 [[1.01393769e-02, 1.31144105e-02, 1.32235728e-04, 3.10876094e-05,
   4.81918505e-06, 4.86466633e-05, 3.23765053e-05, 4.00682353e-07,
   9.75176334e-01, 1.32042286e-03],
 [[9.41294730e-01, 2.95371632e-04, 4.70674075e-02, 4.77683637e-03,
   4.87070996e-03, 3.40781081e-05, 2.73908488e-04, 2.67446762e-06,
   1.18420972e-03, 2.00188282e-04],
 [[1.18030968e-07, 1.14894130e-04, 7.65499286e-03, 8.58013406e-02,
   5.15409887e-01, 4.64179739e-03, 3.86348605e-01, 6.14111229e-07,
   2.77474282e-05, 2.12781082e-08]], dtype=float32)
```

```
In [ ]: np.
```

jupyter Untitled Last Checkpoint: 36 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3

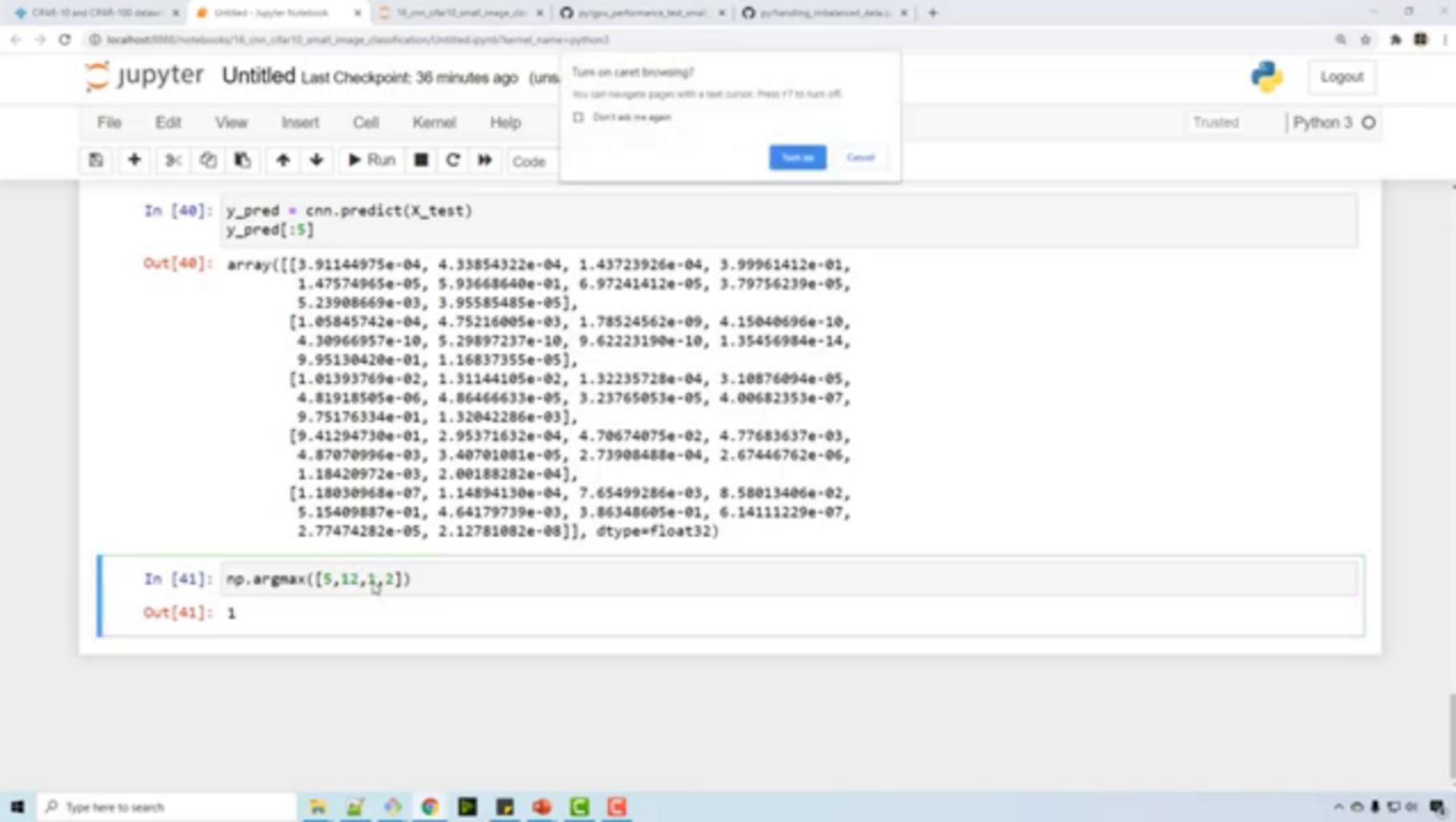


```
In [40]: y_pred = cnn.predict(X_test)
y_pred[:5]
```

```
Out[40]: array([[3.91144975e-04, 4.33854322e-04, 1.43723926e-04, 3.99961412e-01,
   1.47574965e-05, 5.93668640e-01, 6.97241412e-05, 3.79756239e-05,
   5.23908669e-03, 3.95585485e-05],
   [1.05845742e-04, 4.75216005e-03, 1.78524562e-09, 4.15040696e-10,
   4.30966957e-10, 5.29897237e-10, 9.62223190e-10, 1.35456984e-14,
   9.95130420e-01, 1.16837355e-05],
   [1.01393769e-02, 1.31144105e-02, 1.32235728e-04, 3.10876094e-05,
   4.81918505e-06, 4.86466633e-05, 3.23765053e-05, 4.00682353e-07,
   9.75176334e-01, 1.32042286e-03],
   [9.41294730e-01, 2.95371632e-04, 4.70674075e-02, 4.77683637e-03,
   4.87070996e-03, 3.40781081e-05, 2.73908488e-04, 2.67446762e-06,
   1.18420972e-03, 2.00188282e-04],
   [1.18030968e-07, 1.14894130e-04, 7.65499286e-03, 8.58013406e-02,
   5.15409887e-01, 4.64179739e-03, 3.86348605e-01, 6.14111229e-07,
   2.77474282e-05, 2.12781082e-08]], dtype=float32)
```

```
In [41]: np.argmax([5,12,1,2])
```

```
Out[41]: 1
```



jupyter Untitled Last Checkpoint: 36 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

File Edit View Insert Cell Kernel Help
Run Cell Code

ship

```
In [40]: y_pred = cnn.predict(X_test)
y_pred[:5]
```

```
Out[40]: array([[3.91144975e-04, 4.33854322e-04, 1.43723926e-04, 3.99961412e-01,
   1.47574965e-05, 5.93668640e-01, 6.97241412e-05, 3.79756239e-05,
   5.23908669e-03, 3.95585485e-05],
  [1.05845742e-04, 4.7521605e-03, 1.78524562e-09, 4.15040696e-10,
   4.30966957e-10, 5.29897237e-10, 9.62223190e-10, 1.35456984e-14,
   9.95130420e-01, 1.16837355e-05],
  [1.01393769e-02, 1.31144105e-02, 1.32235728e-04, 3.10876094e-05,
   4.81918505e-06, 4.86466633e-05, 3.23765053e-05, 4.00682353e-07,
   9.75176334e-01, 1.32042286e-03],
  [9.41294730e-01, 2.95371632e-04, 4.70674075e-02, 4.77683637e-03,
   4.87070996e-03, 3.40701081e-05, 2.73908488e-04, 2.67446762e-06,
   1.18420972e-03, 2.00188282e-04],
  [1.18030968e-07, 1.14894150e-04, 7.65499286e-03, 8.58013406e-02,
   5.15409887e-01, 4.64179739e-03, 3.86348605e-01, 6.14111229e-07,
   2.77474282e-05, 2.12781082e-08]], dtype=float32)
```

```
In [42]: np.argmax([5,12,167,2])
```

```
Out[42]: 2
```

jupyter Untitled Last Checkpoint: 36 minutes ago (unsaved changes)

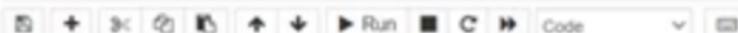


Long

File Edit View Insert Cell Kernel Help

Trusted

| Python 3.9



1

```
In [40]: y_pred = cnn.predict(X_test)
y_pred[:5]

Out[40]: array([[3.91144975e-04, 4.33854322e-04, 1.43723926e-04, 3.99961412e-01,
       1.47574965e-05, 5.93668640e-01, 6.97241412e-05, 3.79756239e-05,
       5.23908669e-03, 3.95585485e-05],
      [1.05845742e-04, 4.75216005e-03, 1.78524562e-09, 4.15040696e-10,
       4.30966957e-10, 5.29897237e-10, 9.62223190e-10, 1.35456984e-14,
       9.95130420e-01, 1.16837355e-05],
      [1.01393769e-02, 1.31144105e-02, 1.32235728e-04, 3.10876094e-05,
       4.81918505e-06, 4.86466633e-05, 3.23765053e-05, 4.00682353e-07,
       9.75176334e-01, 1.32042286e-03],
      [9.41294738e-01, 2.95371632e-04, 4.70874075e-02, 4.77683637e-03,
       4.87070996e-03, 3.40701081e-05, 2.73908488e-04, 2.67446762e-06,
       1.18420972e-03, 2.00188282e-04],
      [1.18030968e-07, 1.14894130e-04, 7.65499286e-03, 8.58013406e-02,
       5.15409887e-01, 4.64179739e-03, 3.86348605e-01, 6.14111229e-07,
       2.77474282e-05, 2.12781082e-01], ...])
```

```
In [43]: np.argmax(x_sced[0])
```

Out[43]:

jupyter Untitled Last Checkpoint: 36 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

File Edit View Insert Cell Kernel Help
Run Cell Code

ship

In [40]: `y_pred = cnn.predict(X_test)
y_pred[:5]`

Out[40]: `array([[3.91144975e-04, 4.33854322e-04, 1.43723926e-04, 3.99961412e-01,
1.47574965e-05, 5.93668640e-01, 6.97241412e-05, 3.79756239e-05,
5.23908669e-03, 3.95585485e-05],
[1.05845742e-04, 4.7521605e-03, 1.78524562e-09, 4.15040696e-10,
4.30966957e-10, 5.29897237e-10, 9.62223190e-10, 1.35456984e-14,
9.95130420e-01, 1.16837355e-05],
[1.01393769e-02, 1.31144105e-02, 1.32235728e-04, 3.10876094e-05,
4.81918505e-06, 4.86466633e-05, 3.23765053e-05, 4.00682353e-07,`

In [43]: `y_classes = np.argmax(y_pred[0])`

Out[43]: 5

jupyter Untitled Last Checkpoint: 37 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

File Edit View Insert Cell Kernel Help
Run Cell Code

ship

In [40]: `y_pred = cnn.predict(X_test)
y_pred[:5]`

Out[40]: `array([[3.91144975e-04, 4.33854322e-04, 1.43723926e-04, 3.99961412e-01,
1.47574965e-05, 5.93668640e-01, 6.97241412e-05, 3.79756239e-05,
5.23908669e-03, 3.95585485e-05],
[1.05845742e-04, 4.7521605e-03, 1.78524562e-09, 4.15040696e-10,
4.30966957e-10, 5.29897237e-10, 9.62223190e-10, 1.35456984e-14,
9.95130420e-01, 1.16837355e-05],
[1.01393769e-02, 1.31144105e-02, 1.32235728e-04, 3.10876094e-05,
4.81918505e-06, 4.86466633e-05, 3.23765053e-05, 4.00682353e-07,`

In [43]: `y_classes = [np.argmax(element) for element in y_pred]`

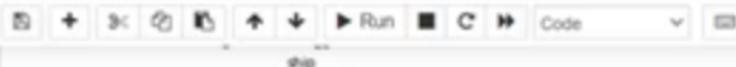
Out[43]: 5

 jupyter Untitled Last Checkpoint: 37 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3



ship

```
In [40]: y_pred = cnn.predict(X_test)
y_pred[:5]
```

```
Out[40]: array([[3.91144975e-04, 4.33854322e-04, 1.43723926e-04, 3.99961412e-01,
   1.47574965e-05, 5.93668640e-01, 6.97241412e-05, 3.79756239e-05,
   5.23908669e-03, 3.95585485e-05],
  [1.05845742e-04, 4.7521605e-03, 1.78524562e-09, 4.15040696e-10,
   4.30966957e-10, 5.29897237e-10, 9.62223190e-10, 1.35456984e-14,
   9.95130420e-01, 1.16837355e-05],
  [1.01393769e-02, 1.31144105e-02, 1.32235728e-04, 3.10876094e-05,
   4.81918505e-06, 4.86466633e-05, 3.23765053e-05, 4.00682353e-07,
   9.75176334e-01, 1.32042286e-03],
  [9.41294730e-01, 2.95371632e-04, 4.70674075e-02, 4.77683637e-03,
   4.87070996e-03, 3.40701081e-05, 2.73908488e-04, 2.67446762e-06,
   1.18420972e-03, 2.00188282e-04],
  [1.18030968e-07, 1.14894150e-04, 7.65499286e-03, 8.58013406e-02,
   5.15409887e-01, 4.64179739e-03, 3.86348605e-01, 6.14111229e-07,
   2.77474282e-05, 2.12781082e-08]], dtype=float32)
```

```
In [43]: y_classes = [np.argmax(element) for element in y_pred]
y_]
```

```
Out[43]: 5
```

jupyter Untitled Last Checkpoint: 37 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3



```
Out[40]: array([[3.91144975e-04, 4.33854322e-04, 1.43723926e-04, 3.99961412e-01,
   1.47574945e-05, 5.93668648e-01, 6.97241412e-05, 3.79756239e-05,
   5.23908669e-03, 3.95585485e-05],
  [1.05845742e-04, 4.75216005e-03, 1.78524562e-09, 4.15040696e-10,
   4.30966957e-10, 5.29897237e-10, 9.62223190e-10, 1.35456984e-14,
   9.95130420e-01, 1.16837355e-05],
  [1.01393769e-02, 1.31144105e-02, 1.32235728e-04, 3.10876094e-05,
   4.81918505e-06, 4.86466633e-05, 3.23765053e-05, 4.00682353e-07,
   9.75176334e-01, 1.32042286e-03],
  [9.41294730e-01, 2.95371632e-04, 4.70674075e-02, 4.77683637e-03,
   4.87070996e-03, 3.40701081e-05, 2.73908488e-04, 2.67446762e-06,
   1.18420972e-03, 2.00188282e-04],
  [1.18030968e-07, 1.14894130e-04, 7.65499286e-03, 8.58013406e-02,
   5.15409887e-01, 4.64179739e-03, 3.86348605e-01, 6.14111229e-07,
   2.77474282e-05, 2.12781082e-08]], dtype=float32)
```

```
In [44]: y_classes = [np.argmax(element) for element in y_pred]
y_classes[:5]
```

```
Out[44]: [5, 8, 8, 0, 4]
```

```
In [ ]: y_test[:]
```

jupyter Untitled Last Checkpoint: 37 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3



```
Out[40]: array([[3.91144975e-04, 4.33854322e-04, 1.43723926e-04, 3.99961412e-01,
   1.47574945e-05, 5.93668648e-01, 6.97241412e-05, 3.79756239e-05,
   5.23908669e-03, 3.95585485e-05],
  [1.05845742e-04, 4.75216005e-03, 1.78524562e-09, 4.15040696e-10,
   4.30966957e-10, 5.29897237e-10, 9.62223198e-10, 1.35456984e-14,
   9.95130420e-01, 1.16837355e-05],
  [1.01393769e-02, 1.31144105e-02, 1.32235728e-04, 3.10876094e-05,
   4.81918505e-06, 4.86466633e-05, 3.23765053e-05, 4.00682353e-07,
   9.75176334e-01, 1.32042286e-03],
  [9.41294730e-01, 2.95371632e-04, 4.70674075e-02, 4.77683637e-03,
   4.87070996e-03, 3.40701081e-05, 2.73908488e-04, 2.67446762e-06,
   1.18420972e-03, 2.00188282e-04],
  [1.18030968e-07, 1.14894130e-04, 7.65499286e-03, 8.58013406e-02,
   5.15409887e-01, 4.64179739e-03, 3.86348605e-01, 6.14111229e-07,
   2.77474282e-05, 2.12781082e-08]], dtype=float32)
```

```
In [44]: y_classes = [np.argmax(element) for element in y_pred]
y_classes[:5]
```

```
Out[44]: [5, 8, 8, 0, 4]
```

```
In [45]: y_test[:5]
```

```
Out[45]: array([3, 0, 8, 0, 6], dtype=uint8)
```

CRM-10 and CRM-100 dataset CRM-10 Untitled - Jupyter Notebook CRM-100 16_cnn_cifar10_smali_image_classification pygpu_performance_test_smali pyhandling_imbalanced_data +

jupyter Untitled Last Checkpoint: 37 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [40]: `y_pred = cnn.predict(X_test)`
`y_pred[:5]`

Out[40]: `array([[3.91144975e-04, 4.33854322e-04, 1.43723926e-04, 3.99961412e-01,`
`1.47574965e-05, 5.93668640e-01, 6.97241412e-05, 3.79756239e-05,`
`5.23908669e-03, 3.95585485e-05],`
`[1.05845742e-04, 4.75216005e-03, 1.78524562e-09, 4.15040696e-10,`
`4.30966957e-10, 5.29897237e-10, 9.62223190e-10, 1.35456984e-14,`
`9.95138420e-01, 1.16837355e-05],`
`[1.01393769e-02, 1.31144105e-02, 1.32235728e-04, 3.10876094e-05,`
`4.81918505e-06, 4.86466633e-05, 3.23765053e-05, 4.00682353e-07,`
`9.75176334e-01, 1.32042286e-03],`
`[9.41294730e-01, 2.95371632e-04, 4.78674075e-02, 4.77683637e-03,`
`4.87070996e-03, 3.40701081e-05, 2.73908488e-04, 2.67446762e-06,`
`1.18420972e-03, 2.00188282e-04],`
`[1.18030968e-07, 1.14894130e-04, 7.65499286e-03, 8.588013406e-02,`
`5.15409887e-01, 4.64179739e-03, 3.86348605e-01, 6.14111229e-07,`
`2.77474282e-05, 2.12781082e-08]], dtype=float32)`

In [44]: `y_classes = [np.argmax(element) for element in y_pred]`
`y_classes[:5]`

Out[44]: `[5, 8, 8, 0, 4]`

In [45]: `y_test[:5]`

Out[45]: `array([3, 8, 8, 0, 6], dtype=uint8)`

Type here to search       

CRNN-10 and CRNN-100 dataset... Untitled - Jupyter Notebook 16_cnn_cifar10_small_image_classification/Untitled.ipynb Kernel name: python3

jupyter Untitled Last Checkpoint: 37 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help Trusted Python 3 Logout

Code

```
4.30966957e-10, 5.29897237e-10, 9.62223190e-10, 1.35456984e-14,
9.95130420e-01, 1.16837355e-05],
[1.01393769e-02, 1.31144105e-02, 1.32235728e-04, 3.10876094e-05,
4.81918505e-06, 4.86466633e-05, 3.23765053e-05, 4.00682353e-07,
9.75176334e-01, 1.32042286e-03],
[9.41294730e-01, 2.95371632e-04, 4.70674075e-02, 4.77683637e-03,
4.87070996e-03, 3.40701081e-05, 2.73908488e-04, 2.67446762e-05,
1.18420972e-03, 2.00188282e-04],
[1.18030968e-07, 1.14894130e-04, 7.65499286e-03, 8.58013406e-02,
5.15409887e-01, 4.64179739e-03, 3.86348605e-01, 6.14111229e-07,
2.77474282e-05, 2.12781082e-08]], dtype=float32)
```

```
In [44]: y_classes = [np.argmax(element) for element in y_pred]
y_classes[:5]
```

```
Out[44]: [5, 8, 8, 0, 4]
```

```
In [45]: y_test[:5]
```

```
Out[45]: array([3, 8, 8, 0, 6], dtype=uint8)
```

```
In [ ]: plot_sample(X_test, y_test, 1)
```

jupyter Untitled Last Checkpoint: 38 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3

```
In [44]: y_classes = [np.argmax(element) for element in y_pred]
```

```
Out[44]: [5, 8, 8, 0, 4]
```

```
In [45]: y_test[:5]
```

```
Out[45]: array([3, 0, 8, 0, 6], dtype=uint8)
```

```
In [46]: plot_sample(X_test, y_test, 1)
```



```
In [ ]:
```



jupyter Untitled Last Checkpoint: 38 minutes ago (unsaved changes)



Layout

Trusted

```
In [44]: y_classes = [np.argmax(element) for element in y_pred]
y_classes[:5]
```

Out[44]: [5, 8, 8, 0, 4]

```
In [45]: y_test[:5]
```

```
Out[45]: array([3, 8, 8, 0, 6], dtype=uint8)
```

```
In [46]: plot_sample(X_test, y_test, 1)
```



```
In [ ]: classes[y_classes[1]]
```

jupyter Untitled Last Checkpoint: 38 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3

```
Run C Code   
[44]: array([4.4992929e-02, 4.4992929e-02, 7.924224000e-02, 9.200424950e-02,  
5.15409887e-01, 4.64179739e-03, 3.86348605e-01, 6.14111229e-07,  
2.77474282e-05, 2.127810882e-08]], dtype=float32)
```

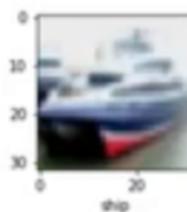
```
In [44]: y_classes = [np.argmax(element) for element in y_pred]  
y_classes[:5]
```

```
Out[44]: [5, 8, 8, 0, 4]
```

```
In [45]: y_test[:5]
```

```
Out[45]: array([3, 8, 8, 0, 6], dtype=uint8)
```

```
In [46]: plot_sample(X_test, y_test, 1)
```



```
In [48]: classes[y_classes[1]]
```

```
Out[48]: 'ship'
```



Logout

jupyter Untitled Last Checkpoint: 38 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help

Trusted

Python 3

```
[1.18030968e-07, 1.14894130e-04, 7.65499286e-03, 8.58813406e-02,
 5.15409887e-01, 4.64179739e-03, 3.86348605e-01, 6.14111229e-07,
 2.77474282e-05, 2.12781082e-08]], dtype=float32)
```

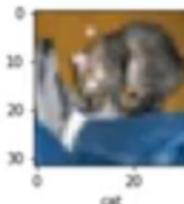
```
In [44]: y_classes = [np.argmax(element) for element in y_pred]
y_classes[:5]
```

```
Out[44]: [5, 8, 8, 0, 4]
```

```
In [45]: y_test[:5]
```

```
Out[45]: array([3, 8, 8, 0, 6], dtype=uint8)
```

```
In [49]: plot_sample(X_test, y_test, 0)
```



```
In [50]: classes[y_classes[0]]
```

```
Out[50]: 'dog'
```

jupyter Untitled Last Checkpoint: 38 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3

```
[1.18030968e-07, 1.14894130e-04, 7.65499286e-03, 8.58813406e-02,  
5.15409887e-01, 4.64179739e-03, 3.86348605e-01, 6.14111229e-07,  
2.77474282e-05, 2.12781082e-08]], dtype=float32)
```

```
In [44]: y_classes = [np.argmax(element) for element in y_pred]  
y_classes[:5]
```

```
Out[44]: [5, 8, 8, 0, 4]
```

```
In [45]: y_test[:5]
```

```
Out[45]: array([3, 8, 8, 0, 6], dtype=uint8)
```

```
In [49]: plot_sample(X_test, y_test, 3)
```



```
In [50]: classes[y_classes[0]]
```

```
Out[50]: 'dog'
```

jupyter Untitled Last Checkpoint: 38 minutes ago (unsaved changes)



Logout

File Edit View Insert Cell Kernel Help

Trusted

Python 3

Run Code

Out[44]: [5, 8, 8, 0, 4]

In [45]: y_test[:5]

Out[45]: array([3, 8, 8, 0, 6], dtype=uint8)

In [52]: plot_sample(X_test, y_test, 3)



In [50]: classes[y_classes[0]]

Out[50]: 'dog'



Logout

jupyter Untitled Last Checkpoint: 39 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help

Trusted | Python 3



In [53]: classes[y_classes[3]]

Out[53]: 'airplane'

In [54]: print("Classification Report: \n", classification_report(y_test, y_classes))

	precision	recall	f1-score	support
0	0.73	0.72	0.73	1000
1	0.78	0.85	0.81	1000
2	0.60	0.59	0.60	1000
3	0.62	0.48	0.49	1000
4	0.65	0.65	0.65	1000
5	0.52	0.67	0.59	1000
6	0.75	0.79	0.77	1000
7	0.78	0.71	0.74	1000
8	0.75	0.84	0.79	1000
9	0.80	0.73	0.76	1000
accuracy			0.70	10000
macro avg	0.70	0.70	0.69	10000
weighted avg	0.70	0.70	0.69	10000



Logout

jupyter Untitled Last Checkpoint: 39 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Help

Trusted

Python 3



In [53]: classes[y_classes[3]]

Out[53]: 'airplane'

In [54]: print("Classification Report: \n", classification_report(y_test, y_classes))

	precision	recall	f1-score	support
0	0.73	0.72	0.73	1000
1	0.78	0.85	0.81	1000
2	0.60	0.59	0.60	1000
3	0.62	0.40	0.49	1000
4	0.65	0.65	0.65	1000
5	0.52	0.67	0.59	1000
6	0.75	0.79	0.77	1000
7	0.78	0.71	0.74	1000
8	0.75	0.84	0.79	1000
9	0.80	0.73	0.76	1000
accuracy			0.70	10000
macro avg	0.70	0.70	0.69	10000
weighted avg	0.70	0.70	0.69	10000

CRM-10 and CRM-100 dataset x Untitled - Jupyter Notebook x 16_cnn_cifar10_small_image_classification x pygpu_performance_test_omni x pyhandling_imbalanced_data x +

localhost:8888/notebooks/16_cnn_cifar10_small_image_classification/Untitled.ipynb?kernel_name=python3

jupyter Untitled Last Checkpoint: 39 minutes ago (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

In [27]:

```
from sklearn.metrics import confusion_matrix, classification_report
import numpy as np
y_pred = ann.predict(X_test)
y_pred_classes = [np.argmax(element) for element in y_pred]

print("Classification Report: \n", classification_report(y_test, y_pred_classes))
```

Classification Report:

	precision	recall	f1-score	support
0	0.71	0.35	0.47	1000
1	0.55	0.71	0.62	1000
2	0.33	0.41	0.36	1000
3	0.44	0.15	0.23	1000
4	0.34	0.51	0.40	1000
5	0.43	0.33	0.37	1000
6	0.43	0.68	0.52	1000
7	0.65	0.45	0.53	1000
8	0.57	0.69	0.62	1000
9	0.59	0.48	0.53	1000
accuracy			0.48	10000
macro avg	0.50	0.48	0.47	10000
weighted avg	0.50	0.48	0.47	10000

In [30]:

```
cnn = models.Sequential([
    # CNN
    layers.Conv2D(filters=32, kernel_size=(3,3), activation='relu', input_shape=(32,32,3)),
    layers.MaxPooling2D((2,2)),
```

Type here to search

jupyter Untitled Last Checkpoint: 40 minutes ago (autosaved)



Logout

File Edit View Insert Cell Kernel Help

Trusted | Python 3



Out[53]: 'airplane'

In [54]: print("Classification Report: \n", classification_report(y_test, y_classes))

	precision	recall	f1-score	support
0	0.73	0.72	0.73	1000
1	0.78	0.85	0.81	1000
2	0.60	0.59	0.60	1000
3	0.62	0.40	0.49	1000
4	0.65	0.65	0.65	1000
5	0.52	0.67	0.59	1000
6	0.75	0.79	0.77	1000
7	0.78	0.71	0.74	1000
8	0.75	0.84	0.79	1000
9	0.80	0.73	0.76	1000
accuracy			0.78	10000
macro avg		0.70	0.70	10000
weighted avg		0.70	0.70	10000

CRM-10 and CRM-100 dataset X | Untitled - Jupyter Notebook X | 10_mnist_email_image_in X | pygpu_performance_test_email X | pyhandling_imbalanced_data X | python_mnist_exercise_solution X +

R 0 contributors

433 lines (433 sloc) 14.8 KB

<> Raw Blame

Handwritten digits classification using CNN

In this notebook we will classify handwritten digits using a simple neural network (ANN) first and than repeat same thing with convolutional neural network. We will see how accuracy improves clickly when you use convolutional neural network.

```
In [27]: import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
```

```
In [28]: (X_train, y_train), (X_test, y_test) = keras.datasets.mnist.load_data()
```

```
In [29]: X_train.shape
```

```
Out[29]: (60000, 28, 28)
```

```
In [30]: X_test.shape
```

```
Out[30]: (10000, 28, 28)
```

```
In [31]: X_train[0].shape
```

Type here to search

CRM-10 and CRM-100 dataset x Untitled - Jupyter Notebook x 16_mnist_10_email_image_dnn x pygpu_performance_test_email x pyhandling_imbalanced_data x python_mnist_exercise_solution x

File Edit Insert Cell Kernel Help

PR 0 contributors

433 lines (433 sloc) | 14.8 KB

<> Copy Raw Blame

Handwritten digits classification using CNN

In this notebook we will classify handwritten digits using a simple neural network (ANN) first and than repeat same thing with convolutional neural network. We will see how accuracy improves clickly when you use convolutional neural network.

```
In [27]: import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
```

```
In [28]: (X_train, y_train), (X_test, y_test) = keras.datasets.mnist.load_data()
```

```
In [29]: X_train.shape
```

```
Out[29]: (60000, 28, 28)
```

```
In [30]: X_test.shape
```

```
Out[30]: (10000, 28, 28)
```

```
In [31]: X_train[0].shape
```

Type here to search

$\mathcal{V}_{\text{master}}$ -

[py / DeepLearningML / 16_cnn_cifar10_small_image_classification / cnn_cifar10_dataset.ipynb](#)

Go to file ...

codebasics cifar10 image classification

Latest commit [594643a](#) 42 minutes ago



83,0 contributors

609 lines (609 sloc) | 45.7 KB

>

3

730

Blame

□

4

1



```
In [174]: y_classes = np.argmax(element) for element in y_pred  
y_classes[:5]
```

```
Out[173]: [3, 8, 8, 8, 6]
```

```
In [175]: classes[y_classes[1]]
```

```
Out[175]: 'ship'
```

```
In [176]: classes[y_classes[3]]
```

```
Out[176]: 'airplane'
```

Exercise

Use CNN to do handwritten digits classification using MNIST dataset. You can use this notebook as a reference: https://github.com/codebasics/py/blob/master/DeepLearningML/1_digits_recognition/digits_recognition_neural_network.ipynb

Above we used ANN for digits classification. You need to modify this code to use CNN instead. Check how accuracy improves fast with CNN and figure out how CNN can be a better choice for doing image classification compared to ANN. Once you have worked on this problem on your own, you can check my solution by clicking on this link: [Solution](#)



```
CNN-10 and CNN-100 dataset * Untitled - Jupyter Notebook * 10_mnist_10_email_image_d... * pygpu_performance_test_email * pyhandling_imbalanced_data * sysmnist_100_dataset.ipynb +
```

```
In [173]: y_classes = np.argmax(element) for element in y_pred
y_classes[:5]
```

```
Out[173]: [3, 8, 8, 0, 6]
```

```
In [175]: classes[y_classes[1]]
```

```
Out[175]: 'ship'
```

```
In [176]: classes[y_classes[3]]
```

```
Out[176]: 'airplane'
```

Exercise

Use CNN to do handwritten digits classification using MNIST dataset. You can use this notebook as a reference: https://github.com/codebasics/py/blob/master/DeepLearningML/1_digits_recognition/digits_recognition_neural_network.ipynb

Above we used ANN for digits classification. You need to modify this code to use CNN instead. Check how accuracy improves fast with CNN and figure out how CNN can be a better choice for doing image classification compared to ANN. Once you have worked on this problem on your own, you can check my solution by clicking on this link: [Solution](#)



© 2020 GitHub, Inc.

Terms

Privacy

Security

Status

Help

Contact GitHub

Pricing

API

Training

Blog

About

Type here to search



```
CRM-10 and CRM-100 dataset x Untitled - Jupyter Notebook x 10_mnist_10_email_image_d... x pygpu_performance_test_email x pyhandling_imbalanced_data x system_cifar10_dataset.ipynb x Loading... x + - * #
```

```
In [173]: y_classes = np.argmax(element) for element in y_pred
y_classes[:5]
```

```
Out[173]: [3, 8, 8, 0, 6]
```

```
In [175]: classes[y_classes[1]]
```

```
Out[175]: 'ship'
```

```
In [176]: classes[y_classes[3]]
```

```
Out[176]: 'airplane'
```

Exercise

Use CNN to do handwritten digits classification using MNIST dataset. You can use this notebook as a reference: https://github.com/codebasics/py/blob/master/DeepLearningML/1_digits_recognition/digits_recognition_neural_network.ipynb

Above we used ANN for digits classification. You need to modify this code to use CNN instead. Check how accuracy improves fast with CNN and figure out how CNN can be a better choice for doing image classification compared to ANN. Once you have worked on this problem on your own, you can check my solution by clicking on this link: [Solution](#)



© 2020 GitHub, Inc.

Terms

Privacy

Security

Status

Help

Contact GitHub

Pricing

API

Training

Blog

About

```
0.99215686, 0.99215686, 0.99215686, 0.98039216, 0.71372549,  
0., 0., 0., 0., 0.,  
0., 0., 0., ],  
[0., 0., 0., 0., 0.,  
0., 0., 0., 0., 0.,  
0.89411765, 0.44705882, 0.86666667, 0.99215686, 0.99215686,  
0.99215686, 0.99215686, 0.78823529, 0.30588235, 0.,  
0., 0., 0., 0., 0.,  
0., 0., 0., ],  
[0., 0., 0., 0., 0.,  
0., 0., 0., 0.09019608, 0.25882353,  
0.83529412, 0.99215686, 0.99215686, 0.99215686, 0.99215686,  
0.77647059, 0.31764706, 0.00784314, 0., 0.,  
0., 0., 0., 0., 0.,  
0., 0., 0., ],  
[0., 0., 0., 0., 0.,  
0., 0.07058824, 0.67058824, 0.85882353, 0.99215686,  
0.99215686, 0.99215686, 0.99215686, 0.76470588, 0.31372549,  
0.03529412, 0., 0., 0., 0.,  
0., 0., 0., 0., 0.,  
0., 0., 0., ],  
[0., 0., 0., 0., 0., 0.21568627,  
0.6745098, 0.88627451, 0.99215686, 0.99215686, 0.99215686,  
0.99215686, 0.95686275, 0.52156863, 0.04313725, 0.,  
0., 0., 0., 0., 0.,  
0., 0., 0., ],  
[0., 0., 0., 0., 0., 0.53333333,  
0.99215686, 0.99215686, 0.99215686, 0.83137255, 0.52941176,  
0.51764706, 0.0627451, 0., 0., 0.,  
0., 0., 0., 0., 0.,  
0., 0., 0., ],  
[0., 0., 0., 0., 0., 0.]
```

```
In [59]: model = keras.Sequential([
    keras.layers.Flatten(input_shape=(28, 28)),
    keras.layers.Dense(100, activation='relu'),
    keras.layers.Dense(10, activation='sigmoid')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(X_train, y_train, epochs=10)

Epoch 1/10
1875/1875 [=====] - 3s 2ms/step - loss: 0.2959 - accuracy: 0.9185
Epoch 2/10
1875/1875 [=====] - 3s 2ms/step - loss: 0.1368 - accuracy: 0.9603
Epoch 3/10
1875/1875 [=====] - 3s 2ms/step - loss: 0.0995 - accuracy: 0.9703
Epoch 4/10
1875/1875 [=====] - 3s 2ms/step - loss: 0.0771 - accuracy: 0.9772
Epoch 5/10
1875/1875 [=====] - 3s 2ms/step - loss: 0.0628 - accuracy: 0.9806
Epoch 6/10
1875/1875 [=====] - 3s 2ms/step - loss: 0.0519 - accuracy: 0.9841
Epoch 7/10
1875/1875 [=====] - 3s 2ms/step - loss: 0.0442 - accuracy: 0.9865
Epoch 8/10
1875/1875 [=====] - 3s 2ms/step - loss: 0.0369 - accuracy: 0.9886
Epoch 9/10
1875/1875 [=====] - 3s 2ms/step - loss: 0.0300 - accuracy: 0.9910
Epoch 10/10
1875/1875 [=====] - 3s 2ms/step - loss: 0.0264 - accuracy: 0.9917
```

```
Out[59]: <tensorflow.python.keras.callbacks.History at 0x1fe24629e80>
```



Type here to search



```
In [173]: y_classes = np.argmax(element) for element in y_pred  
y_classes[:5]
```

```
Out[173]: [3, 8, 8, 8, 6]
```

```
In [175]: classes[y_classes[1]]
```

```
Out[175]: 'ship'
```

```
In [176]: classes[y_classes[3]]
```

```
Out[176]: 'airplane'
```

Exercise

Use CNN to do handwritten digits classification using MNIST dataset. You can use this notebook as a reference: https://github.com/codebasics/py/blob/master/DeepLearningML/1_digits_recognition/digits_recognition_neural_network.ipynb

Above we used ANN for digits classification. You need to modify this code to use CNN instead. Check how accuracy improves fast with CNN and figure out how CNN can be a better choice for doing image classification compared to ANN. Once you have worked on this problem on your own, you can check my solution by clicking on this link: [Solution](#)

```
CNN-10 and CNN-100 dataset | Untitled - Jupyter Notebook | 10_mnist_10_email_image.pkl | pygpu_performance_test_email | pyhandling_imbalanced_data | system_cifar10_dataset.pywbd | digits_recognition_neural_net.ipynb - +
```

```
In [173]: y_classes = np.argmax(element) for element in y_pred
y_classes[:5]
```

```
Out[173]: [3, 8, 8, 0, 6]
```

```
In [175]: classes[y_classes[1]]
```

```
Out[175]: 'ship'
```

```
In [176]: classes[y_classes[3]]
```

```
Out[176]: 'airplane'
```

Exercise

Use CNN to do handwritten digits classification using MNIST dataset. You can use this notebook as a reference: https://github.com/codebasics/py/blob/master/DeepLearningML/1_digits_recognition/digits_recognition_neural_network.ipynb

Above we used ANN for digits classification. You need to modify this code to use CNN instead. Check how accuracy improves fast with CNN and figure out how CNN can be a better choice for doing image classification compared to ANN. Once you have worked on this problem on your own, you can check my solution by clicking on this link: [Solution](#)



© 2020 GitHub, Inc.

Terms

Privacy

Security

Status

Help

Contact GitHub

Pricing

API

Training

Blog

About



```
In [173]: y_classes = np.argmax(element) for element in y_pred  
y_classes[:5]
```

```
Out[173]: [3, 8, 8, 0, 6]
```

```
In [175]: classes[y_classes[1]]
```

```
Out[175]: 'ship'
```

```
In [176]: classes[y_classes[3]]
```

```
Out[176]: 'airplane'
```

Exercise

Use CNN to do handwritten digits classification using MNIST dataset. You can use this notebook as a reference: https://github.com/codebasics/py/blob/master/DeepLearningML/1_digits_recognition/digits_recognition_neural_network.ipynb

Above we used ANN for digits classification. You need to modify this code to use CNN instead. Check how accuracy improves fast with CNN and figure out how CNN can be a better choice for doing image classification compared to ANN. Once you have worked on this problem on your own, you can check my solution by clicking on this link: [Solution](#)



```
In [173]: y_classes = np.argmax(element) for element in y_pred  
y_classes[:5]
```

```
Out[173]: [3, 8, 8, 0, 6]
```

```
In [175]: classes[y_classes[1]]
```

```
Out[175]: 'ship'
```

```
In [176]: classes[y_classes[3]]
```

```
Out[176]: 'airplane'
```

Exercise

Use CNN to do handwritten digits classification using MNIST dataset. You can use this notebook as a reference: https://github.com/codebasics/py/blob/master/DeepLearningML/1_digits_recognition/digits_recognition_neural_network.ipynb

Above we used ANN for digits classification. You need to modify this code to use CNN instead. Check how accuracy improves fast with CNN and figure out how CNN can be a better choice for doing image classification compared to ANN. Once you have worked on this problem on your own, you can check my solution by clicking on this link: [Solution](#)



```
In [173]: y_classes = np.argmax(element) for element in y_pred  
y_classes[:5]
```

```
Out[173]: [3, 8, 8, 0, 6]
```

```
In [175]: classes[y_classes[1]]
```

```
Out[175]: 'ship'
```

```
In [176]: classes[y_classes[3]]
```

```
Out[176]: 'airplane'
```

Exercise

Use CNN to do handwritten digits classification using MNIST dataset. You can use this notebook as a reference: https://github.com/codebasics/py/blob/master/DeepLearningML/1_digits_recognition/digits_recognition_neural_network.ipynb

Above we used ANN for digits classification. You need to modify this code to use CNN instead. Check how accuracy improves fast with CNN and figure out how CNN can be a better choice for doing image classification compared to ANN. Once you have worked on this problem on your own, you can check my solution by clicking on this link: [Solution](#)

