

OUTPUT SCREENSHOTS

Jupyter PDE PROJECT Last Checkpoint: Last Wednesday at 10:31 PM (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

```
# IMPORT LIBRARIES/DATASETS

In [30]: import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import pandas as pd
import seaborn as sns
import pickle
import random
```

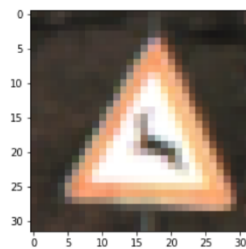
Jupyter PDE PROJECT Last Checkpoint: Last Wednesday at 10:31 PM (autosaved) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

```
# IMAGES VISUALIZATION

In [35]: i = np.random.randint(1, len(X_train))
plt.imshow(X_train[i])
y_train[i]

Out[35]: 31
```



Jupyter PDE PROJECT Last Checkpoint: Last Wednesday at 10:31 PM (autosaved) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

```
# CONVERT IMAGES TO GRAYSCALE AND PERFORM NORMALIZATION

In [39]: from sklearn.utils import shuffle
X_train, y_train = shuffle(X_train, y_train)

In [40]: X_train_gray = np.sum(X_train/3,axis = 3,keepdims = True)
X_test_gray = np.sum(X_test/3,axis = 3,keepdims = True)
X_valid_gray = np.sum(X_valid/3,axis = 3,keepdims = True)

In [41]: X_train_gray.shape
Out[41]: (34799, 32, 32, 1)

In [42]: X_train_gray_norm = (X_train_gray-128)/128
X_test_gray_norm = (X_test_gray-128)/128
X_valid_gray_norm = (X_valid_gray-128)/128

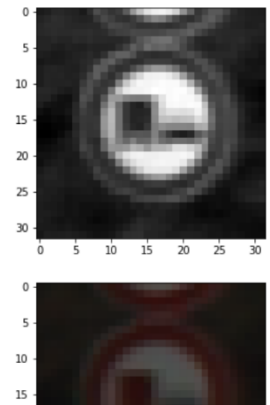
In [43]: X_train_gray_norm
Out[43]: array([[[-0.11197917],
               [-0.078125 ],
               [-0.03385417],
               ...,
               [-0.90885417],
```

jupyter PDE PROJECT Last Checkpoint: Last Wednesday at 10:31 PM (autosaved) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

```
plt.imshow(X_train[i])
plt.figure()
plt.imshow(X_train_gray_norm[i].squeeze(), cmap = 'gray')
```

Out[13]: <matplotlib.image.AxesImage at 0x278c09a82c8>



jupyter PDE PROJECT Last Checkpoint: Last Wednesday at 10:31 PM (autosaved) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

```
CNN.summary()
```

```
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 6)	156
average_pooling2d (AveragePo	(None, 14, 14, 6)	0
dropout (Dropout)	(None, 14, 14, 6)	0
conv2d_1 (Conv2D)	(None, 10, 10, 16)	2416
average_pooling2d_1 (Average	(None, 5, 5, 16)	0
flatten (Flatten)	(None, 400)	0
dense (Dense)	(None, 120)	48120
dense_1 (Dense)	(None, 84)	10164
dense_2 (Dense)	(None, 43)	3655

```
Total params: 64,511
Trainable params: 64,511
Non-trainable params: 0
```

jupyter PDE PROJECT Last Checkpoint: Last Wednesday at 10:31 PM (unsaved changes) Logout

File Edit View Insert Cell Kernel Help Trusted Python 3

COMPILE AND TRAIN DEEP CNN MODEL

```
In [15]: CNN.compile(optimizer = 'Adam', loss = 'sparse_categorical_crossentropy', metrics = ['accuracy'])
```

```
In [25]: history = CNN.fit(X_train_gray_norm,
                        y_train,
                        batch_size = 500,
                        epochs = 5,
                        verbose = 1,
                        validation_data = (X_valid_gray_norm, y_valid))
```

```
Train on 34799 samples, validate on 4410 samples
Epoch 1/5
34799/34799 [=====] - 14s 404us/sample - loss: 0.1667 - accuracy: 0.9516 - val_loss: 0.4397 - val_accu
racy: 0.8916
Epoch 2/5
34799/34799 [=====] - 14s 403us/sample - loss: 0.1531 - accuracy: 0.9569 - val_loss: 0.4216 - val_accu
racy: 0.8961
Epoch 3/5
34799/34799 [=====] - 14s 404us/sample - loss: 0.1426 - accuracy: 0.9604 - val_loss: 0.4287 - val_accu
racy: 0.8934
Epoch 4/5
34799/34799 [=====] - 14s 406us/sample - loss: 0.1331 - accuracy: 0.9626 - val_loss: 0.4252 - val_accu
racy: 0.8912
Epoch 5/5
34799/34799 [=====] - 14s 405us/sample - loss: 0.1282 - accuracy: 0.9648 - val_loss: 0.4208 - val_accu
racy: 0.8982
```

Jupyter PDE PROJECT Last Checkpoint: Last Saturday at 11:37 PM (unsaved changes)

File Edit View Insert Cell Kernel Help Trusted Python 3

```

validation_data = (X_valid_gray_norm, y_valid))

34799/34799 [=====] - 11s 330us/sample - loss: 0.0021 - accuracy: 0.9994 - val_loss: 0.5305 - val_ac
curacy: 0.9556
Epoch 745/750
34799/34799 [=====] - 11s 320us/sample - loss: 0.0016 - accuracy: 0.9994 - val_loss: 0.6542 - val_ac
curacy: 0.9442
Epoch 746/750
34799/34799 [=====] - 12s 340us/sample - loss: 0.0016 - accuracy: 0.9995 - val_loss: 0.6546 - val_ac
curacy: 0.9456
Epoch 747/750
34799/34799 [=====] - 12s 332us/sample - loss: 0.0018 - accuracy: 0.9994 - val_loss: 0.6054 - val_ac
curacy: 0.9506
Epoch 748/750
34799/34799 [=====] - 11s 323us/sample - loss: 0.0033 - accuracy: 0.9994 - val_loss: 0.5238 - val_ac
curacy: 0.9569
Epoch 749/750
34799/34799 [=====] - 11s 330us/sample - loss: 0.0010 - accuracy: 0.9997 - val_loss: 0.6362 - val_ac
curacy: 0.9476
Epoch 750/750
34799/34799 [=====] - 12s 335us/sample - loss: 8.2629e-04 - accuracy: 0.9997 - val_loss: 0.6426 - va
l_accuracy: 0.9512

In [25]: score = CNN.evaluate(X_test_gray_norm, y_test)
print('Test Accuracy: {}'.format(score[1]))

12630/12630 [=====] - 1s 112us/sample - loss: 1.1149 - accuracy: 0.9315
Test Accuracy: 0.9315122961997986

```

Jupyter PDE PROJECT Last Checkpoint: Last Wednesday at 10:31 PM (unsaved changes)

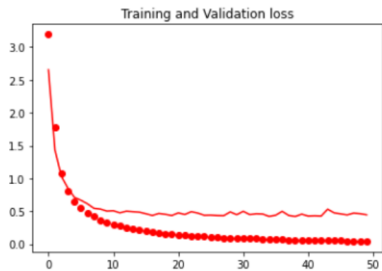
File Edit View Insert Cell Kernel Help Trusted Python 3

```

In [32]: epochs = range(len(accuracy))
plt.plot(epochs, loss, 'ro', label = 'Training loss')
plt.plot(epochs, val_loss, 'r', label = 'Validation loss')
plt.title('Training and Validation loss')

Out[32]: Text(0.5, 1.0, 'Training and Validation loss')

```



The plot shows training and validation loss over 50 epochs. The training loss (red circles) starts at approximately 3.0 and decreases rapidly to near 0.0 by epoch 10, remaining stable thereafter. The validation loss (red line) starts at approximately 2.5 and decreases to about 0.5 by epoch 10, then fluctuates slightly around 0.5 for the remainder of the training process.

Jupyter PDE PROJECT Last Checkpoint: Last Wednesday at 10:31 PM (autosaved)

File Edit View Insert Cell Kernel Help Trusted Python 3

```





In [29]: epochs = range(len(accuracy))
plt.plot(epochs, accuracy, 'ro', label = 'Training accuracy')
plt.plot(epochs, val_accuracy, 'r', label = 'Validation accuracy')
plt.title('Training and Validation accuracy')

Out[29]: Text(0.5, 1.0, 'Training and Validation accuracy')

```



The plot shows training and validation accuracy over 40 epochs. The training accuracy (red circles) starts at approximately 0.95 and increases to about 0.965 by epoch 40. The validation accuracy (red line) starts at approximately 0.89 and increases to about 0.90 by epoch 40, showing a slight dip around epoch 30.

<p>Prediction = 16 True = 16</p> 	<p>Prediction = 1 True = 1</p> 	<p>Prediction = 38 True = 38</p> 	<p>Prediction = 33 True = 33</p> 	<p>Prediction = 11 True = 11</p> 
<p>Prediction = 38 True = 38</p> 	<p>Prediction = 18 True = 18</p> 	<p>Prediction = 12 True = 12</p> 	<p>Prediction = 25 True = 25</p> 	<p>Prediction = 35 True = 35</p> 
<p>Prediction = 12 True = 12</p> 	<p>Prediction = 7 True = 7</p> 	<p>Prediction = 19 True = 23</p> 	<p>Prediction = 7 True = 7</p> 	<p>Prediction = 1 True = 4</p> 
<p>Prediction = 9 True = 9</p> 	<p>Prediction = 21 True = 21</p> 	<p>Prediction = 20 True = 20</p> 	<p>Prediction = 27 True = 27</p> 	<p>Prediction = 38 True = 38</p> 