Traffic sign classification using CNN report

# CNN

CNN’s are very powerful tools for identifying objects within images. It is inspired by how our brain identifies objects using signals received from our retina.

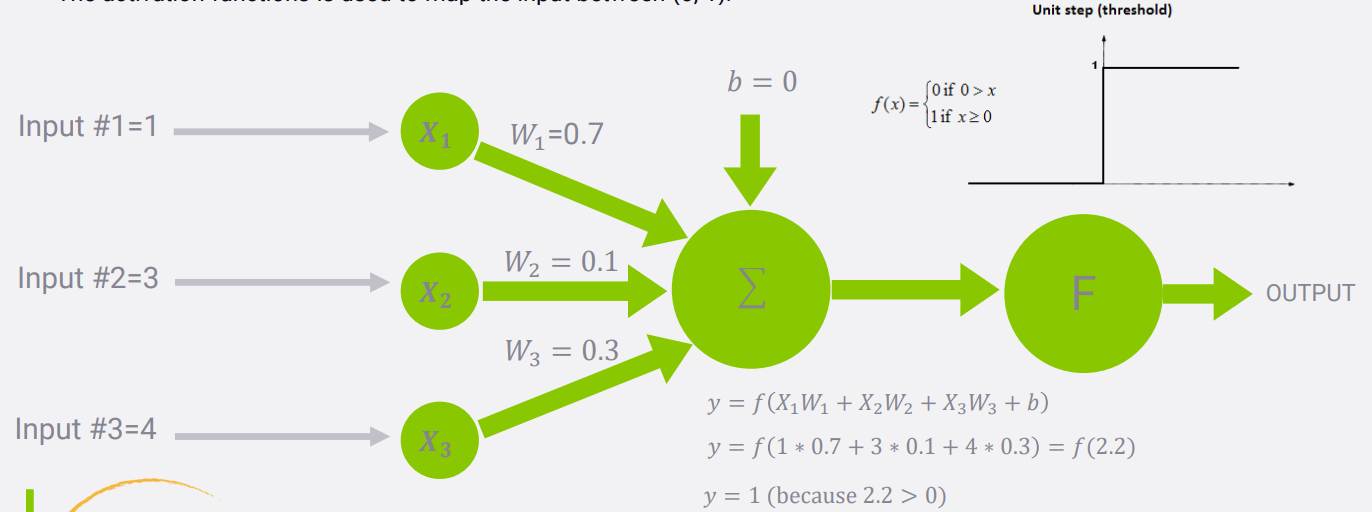
## what are they for?

• when you have data that doesn’t neatly align into columns   
• Images that you want to find features within   
• Machine translation   
• Sentence classification   
• Sentiment analysis   
  
  
• they can find features that aren’t in a specific spot   
• Like a stop sign in a picture   
• Or words within a sentence   
  
• they are “feature-location invariant”

## how do they work?

• local receptive fields are groups of neurons that only respond to a part of what your eyes see (subsampling)   
• they overlap each other to cover the entire visual field (convolutions)   
• they feed into higher layers that identify increasingly complex images   
• some receptive fields identify horizontal lines, lines at different angles, etc. (filters)   
• these would feed into a layer that identifies shapes   
• which might feed into a layer that identifies objects

# SINGLE NEURON MODEL

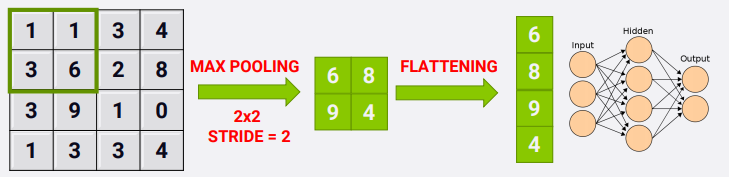
• The neuron collects signals from input channels named dendrites, processes information in its nucleus, and then generates an output in a long thin branch called the axon.   
• Weights shows the strength of the particular node  
• bias allows to shift the activation function curve up or down.   
• Number of adjustable parameters = 4 (3 weights and 1 bias).   
• Activation function “F”.  
• Let’s assume that the activation function is a Unit Step Activation Function.   
• The activation functions is used to map the input between (0, 1).  
  


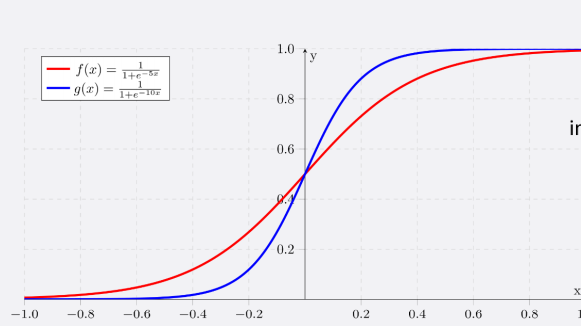
# CNN’s with keras

• source data must be of appropriate dimensions   
 ie width x length x color channels   
• conv2D layer type does the actual convolution on a 2D image   
 Conv1D and Conv3D also available – doesn’t have to be image data   
• flatten layers will convert the 2D layer to a 1D layer for passing into a flat hidden layer of neurons   
• typical usage: Convolve -> Dropout -> Flatten -> Dense -> Dropout -> Softmax

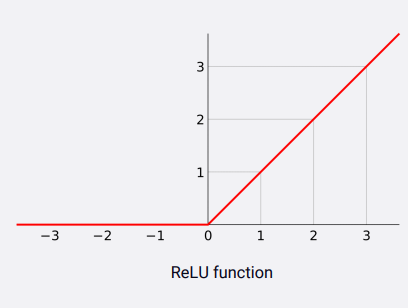
# Max pooling/Flattening

It is used to reduce an image by looking at some groups of pixels in it and choose the highest value within that group to reduce feature map dimensionality. We actually capture the most important information in the image.

By flattening the image we feed it to the normal deep neural network.S  
  
  
  
Activation Functions  
• Step functions don’t work with gradient descent – there is no gradient!   
 Mathematically, they have no useful derivative.   
• Alternatives:   
• Logistic (sigmoid) function   
• Hyperbolic tangent function   
• Exponential linear unit (ELU)

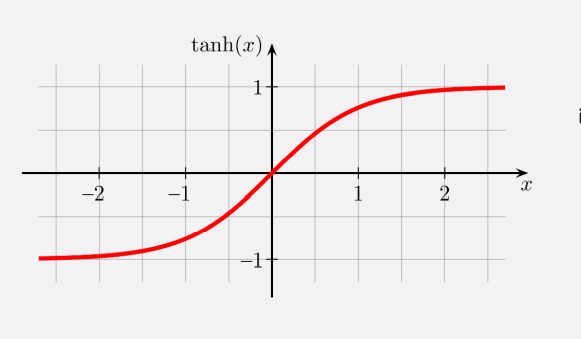
SIGMOID:   
• Takes a number and sets it between 0 and 1   
• Converts large negative numbers to 0 and large positive numbers to 1.   
• Generally used in output layer.   
 

## ReLU function (Rectified Linear Unit)

• if input x < 0, output is 0 and if x > 0 the output is x.   
• RELU does not saturate so it avoids vanishing gradient problem.   
• It uses simple thresholding so it is computationally efficient.   
• Generally used in hidden layers  


## HYPERBOLIC TANGENT ACTIVATION FUNCTION

• “Tanh” is similar to sigmoid, converts number between -1 and 1.   
• Unlike sigmoid, tanh outputs are zero-centered (range: -1 and 1).   
• Tanh suffers from vanishing gradient problem so it kills gradients when saturated.   
• In practice, tanh is preferable over sigmoid.



# Avoiding overfitting with regularization

• With thousands of weights to tune, overfitting is a problem   
• Early stopping (when performance starts dropping)   
• Regularization terms added to cost function during training   
• Dropout – ignore say 50% of all neurons randomly at each training step   
• Works surprisingly well! • Forces your model to spread out its learning

# Classify traffic sign with CNN CODE

# our classes(43 classes)

* ( 0, b'Speed limit (20km/h)') ( 1, b'Speed limit (30km/h)')
* ( 2, b'Speed limit (50km/h)') ( 3, b'Speed limit (60km/h)')
* ( 4, b'Speed limit (70km/h)') ( 5, b'Speed limit (80km/h)')
* ( 6, b'End of speed limit (80km/h)') ( 7, b'Speed limit (100km/h)')
* ( 8, b'Speed limit (120km/h)') ( 9, b'No passing')
* (10, b'No passing for vehicles over 3.5 metric tons')
* (11, b'Right-of-way at the next intersection') (12, b'Priority road')
* (13, b'Yield') (14, b'Stop') (15, b'No vehicles')
* (16, b'Vehicles over 3.5 metric tons prohibited') (17, b'No entry')
* (18, b'General caution') (19, b'Dangerous curve to the left')
* (20, b'Dangerous curve to the right') (21, b'Double curve')
* (22, b'Bumpy road') (23, b'Slippery road')
* (24, b'Road narrows on the right') (25, b'Road work')
* (26, b'Traffic signals') (27, b'Pedestrians') (28, b'Children crossing')
* (29, b'Bicycles crossing') (30, b'Beware of ice/snow')
* (31, b'Wild animals crossing')
* (32, b'End of all speed and passing limits') (33, b'Turn right ahead')
* (34, b'Turn left ahead') (35, b'Ahead only') (36, b'Go straight or right')
* (37, b'Go straight or left') (38, b'Keep right') (39, b'Keep left')
* (40, b'Roundabout mandatory') (41, b'End of no passing')
* (42, b'End of no passing by vehicles over 3.5 metric tons')

## In this model we have used german traffic sign dataset

## 

## Building our model

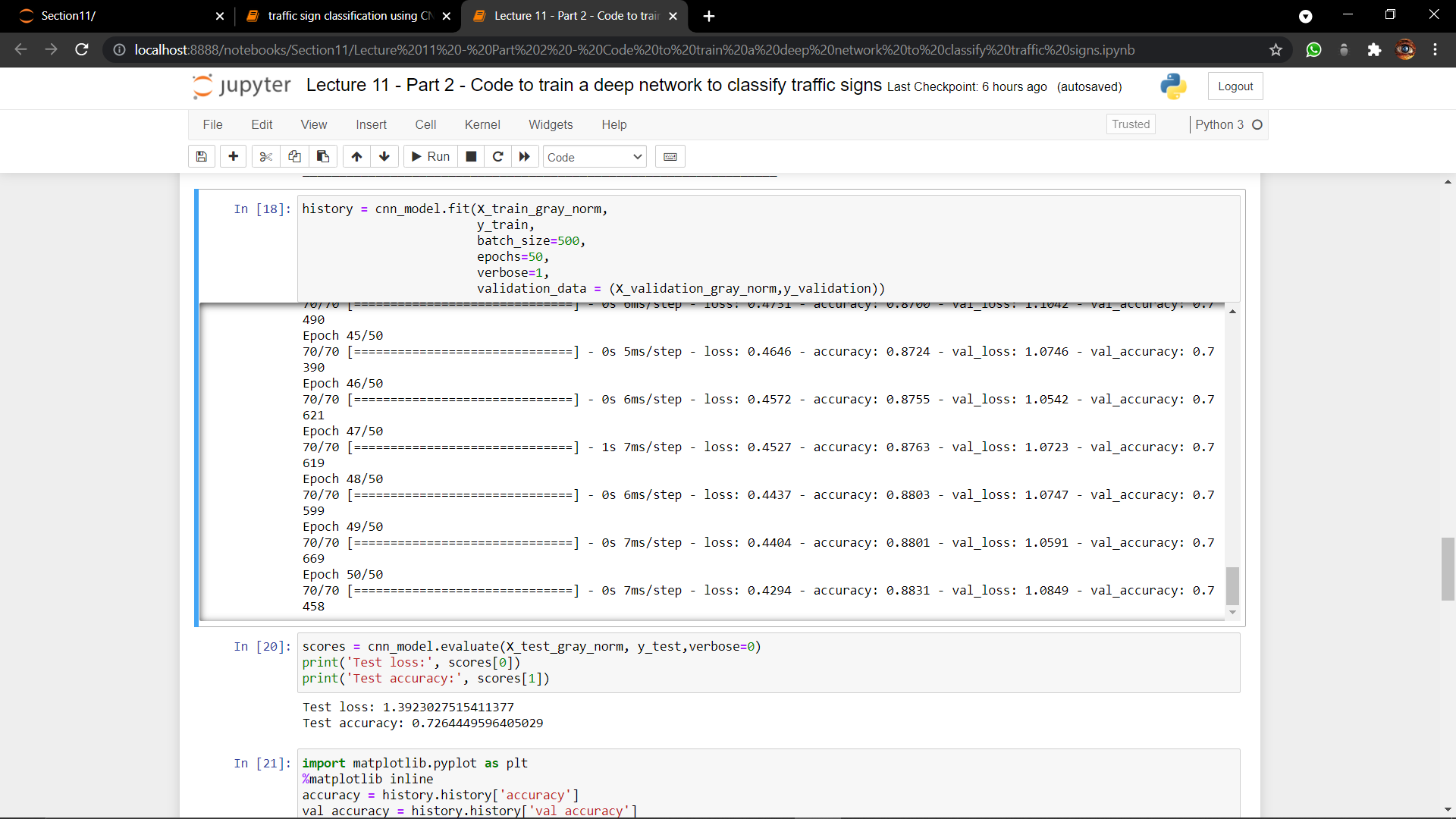
## 

## Model summary

# 

# Train our Model

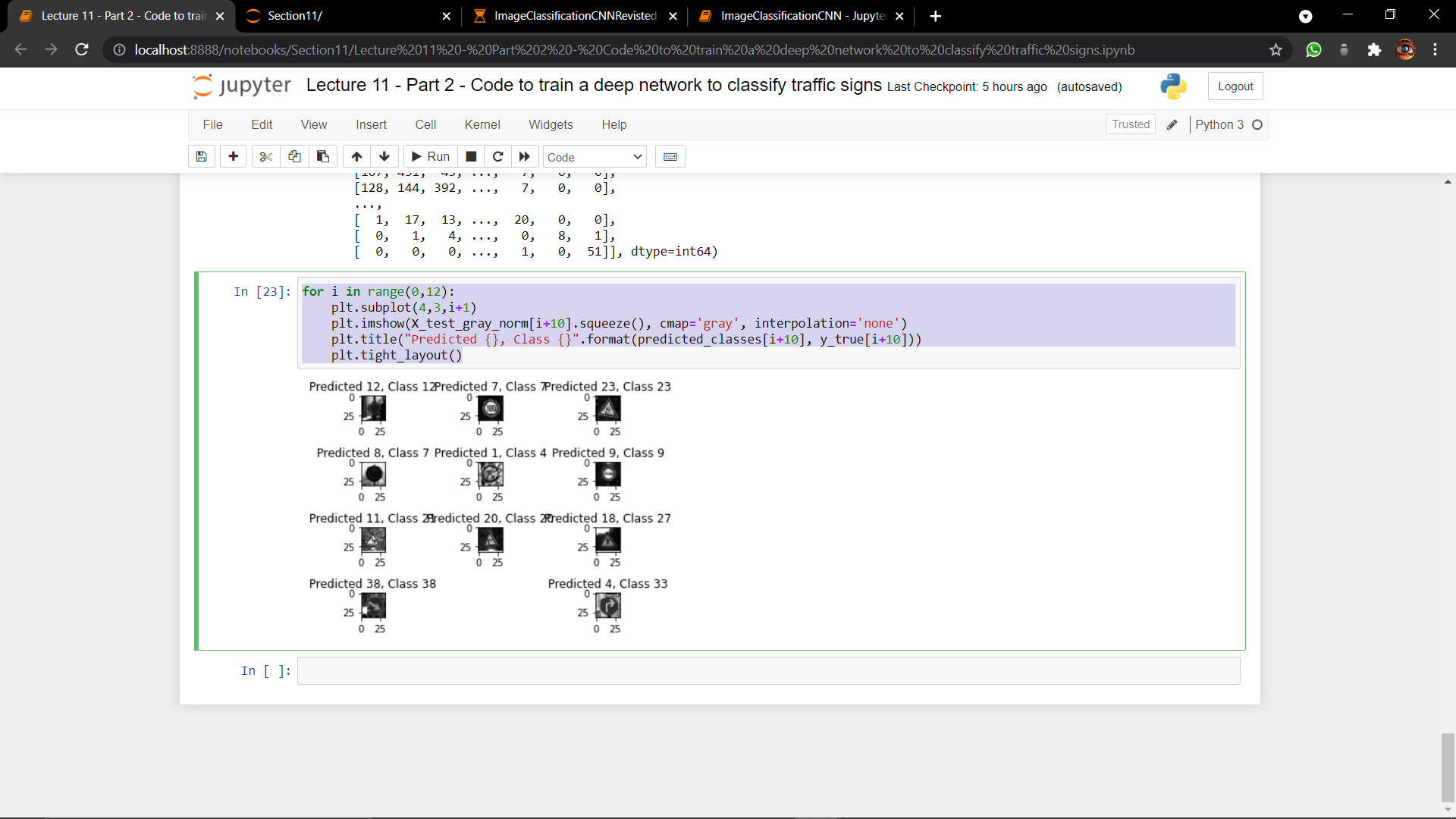
There is less test accuracy.



# Plotting train and validation accuracy

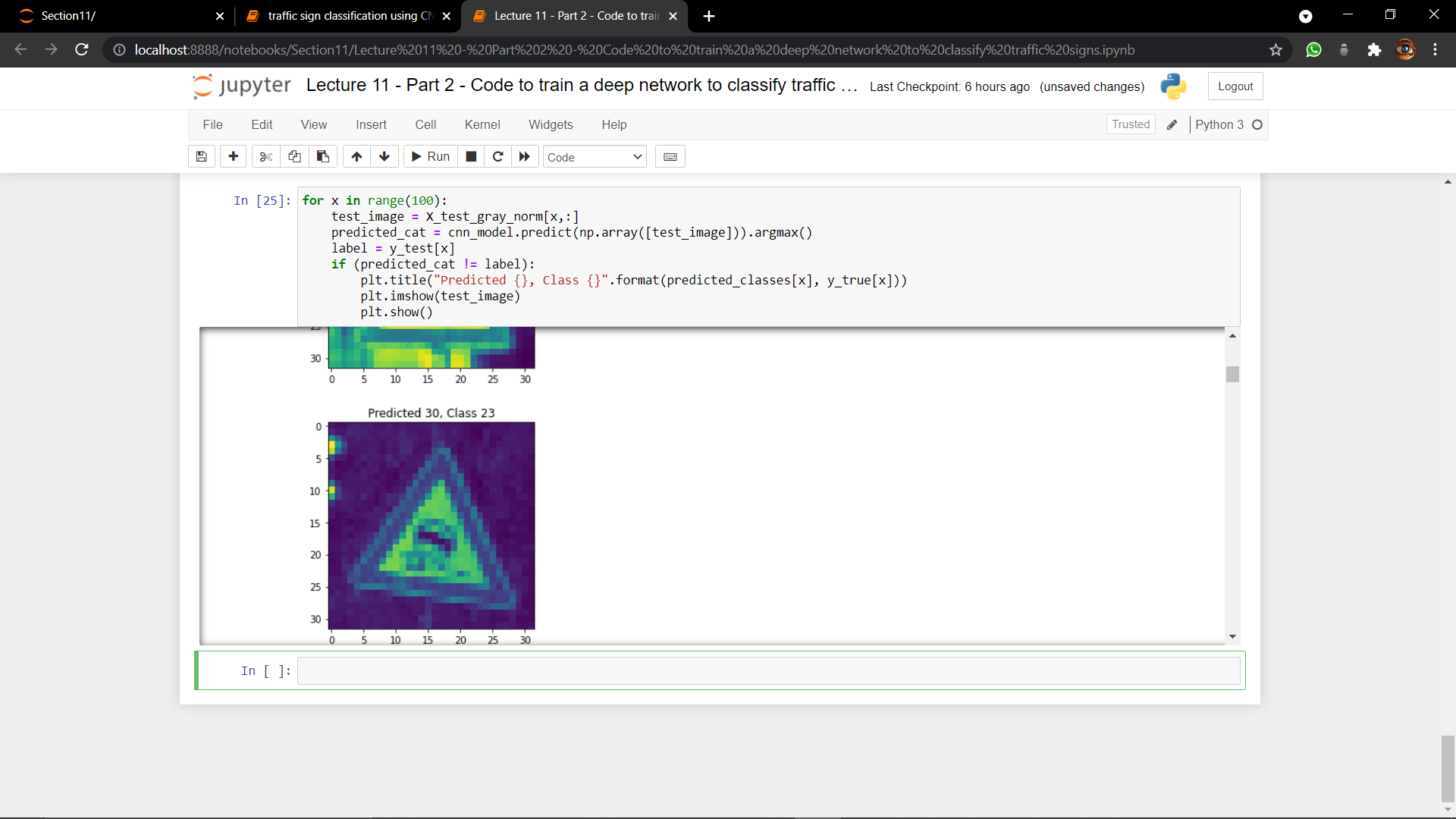
## Take a look on first 12 random pictures with their Predicted classes/Labels vs Random classes

Many wrong prediction.



## Lets find out wrong prediction in first 100 images

As you can see there are many wrong predictions.

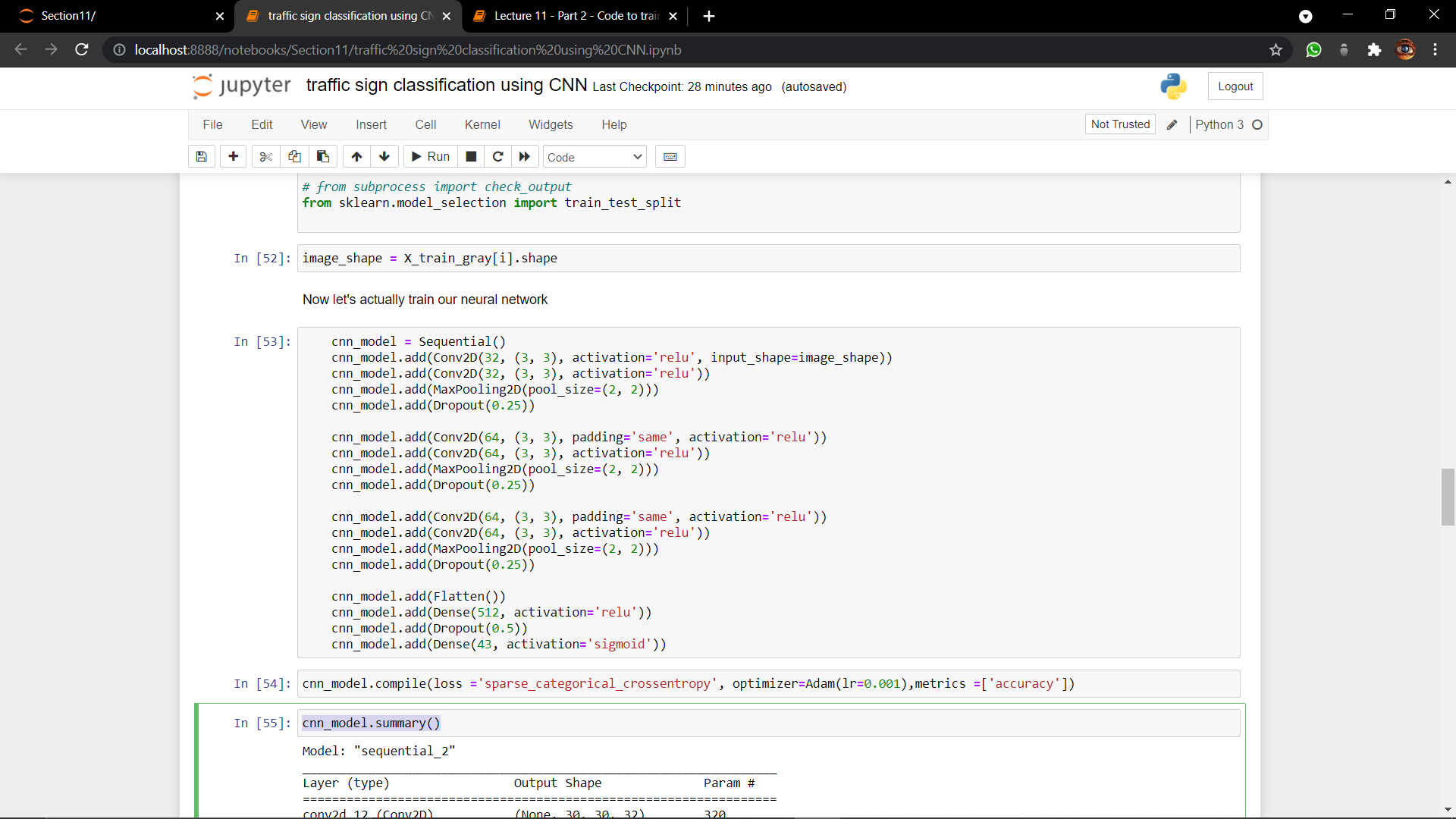


# We improve our model

# AFTER INCREASEING FILTERS/DROPOUT

• Improve accuracy by adding more feature detectors/filters or adding a dropout.   
• Dropout refers to dropping out units in a neural network.  
 • Neurons develop co-dependency amongst each other during training   
• Dropout is a regularization technique for reducing overfitting in neural networks.   
• It enables training to occur on several architectures of the neural network

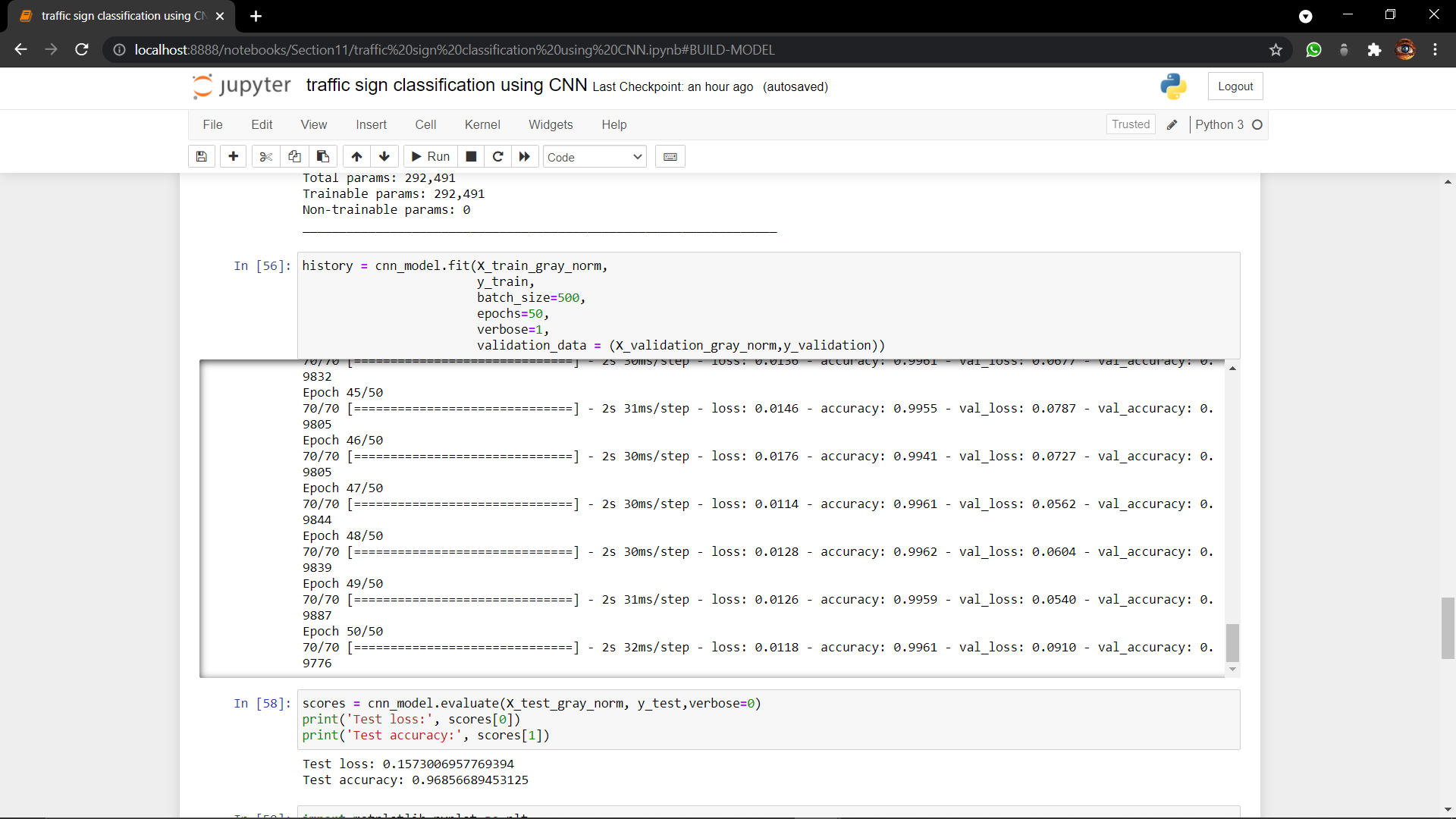
## Build improved model



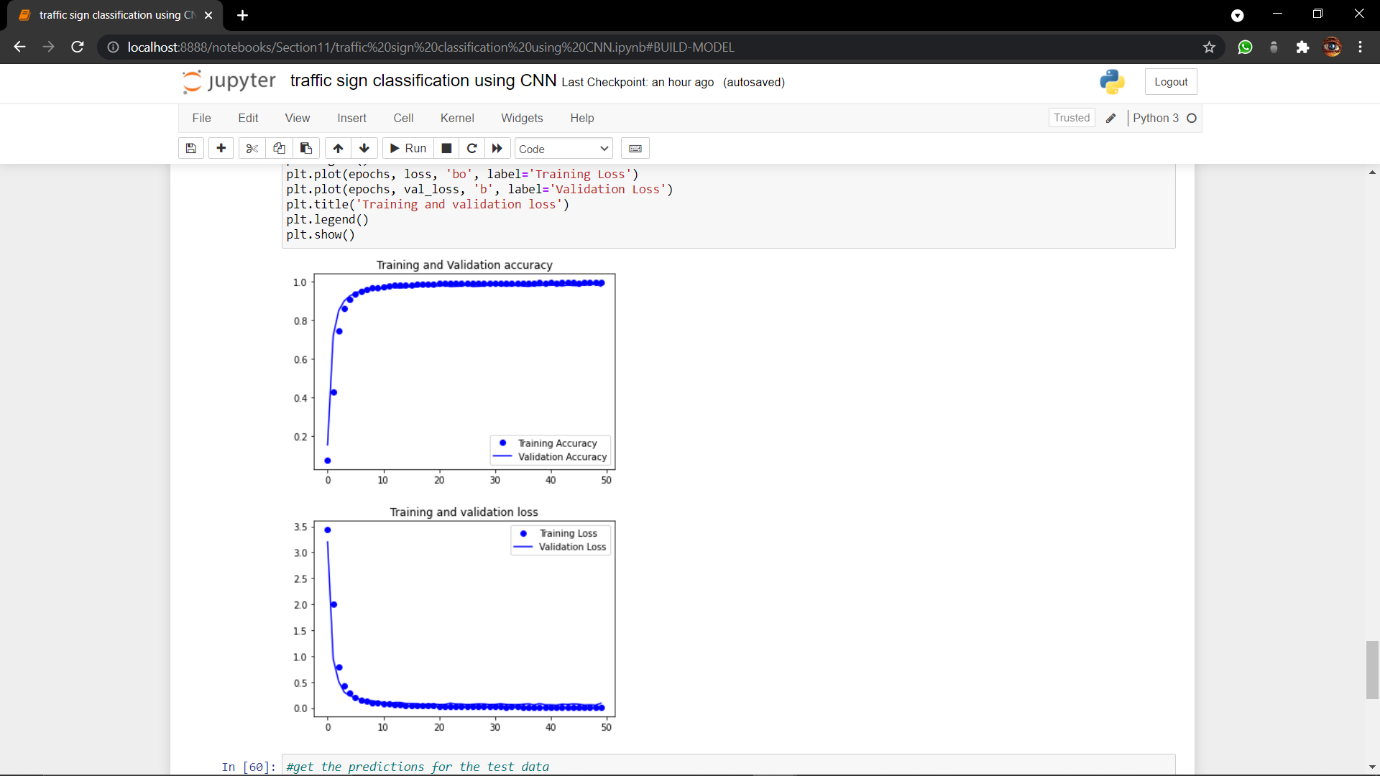
## Model summary

# 

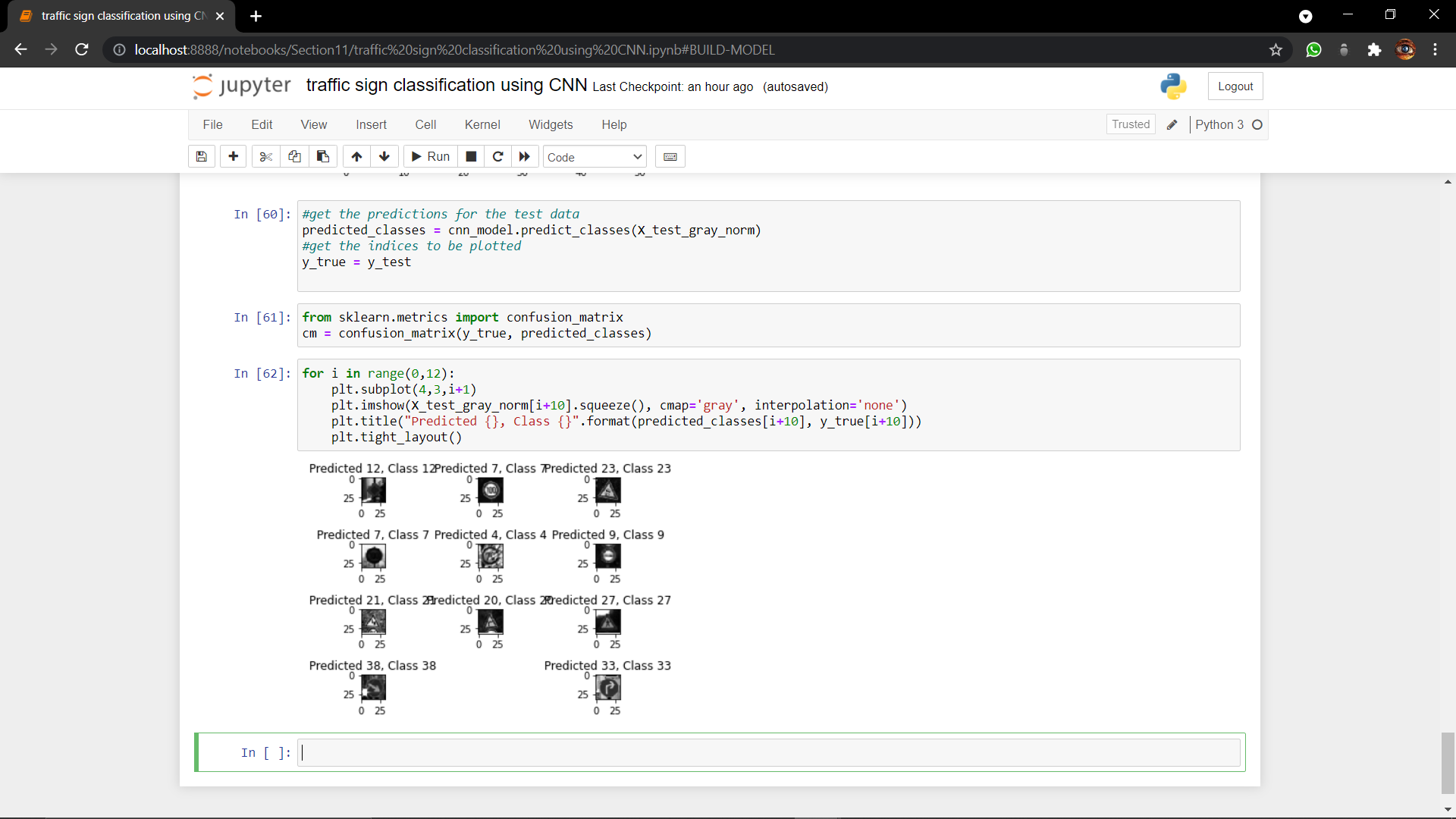
## Train our improved model

As you can see now the test accuracy has been improed.  


# Plotting train and validation accuracy of improved model



## Take a look on first 12 random pictures with their Predicted classes/Labels vs Random classes

Almost no wrong prediction.  


## Lets find out wrong prediction in first 100 images

As you can see there are only few wrong predictions.

