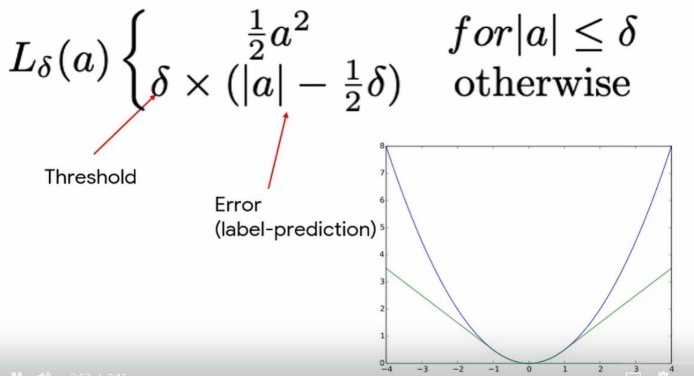
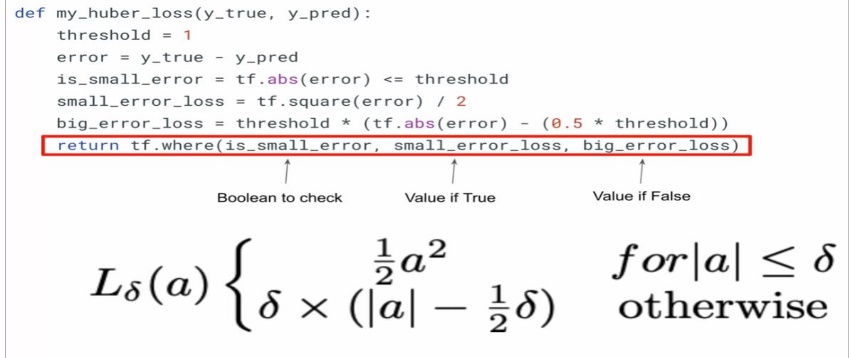
Loss functions help measure how well a model is doing

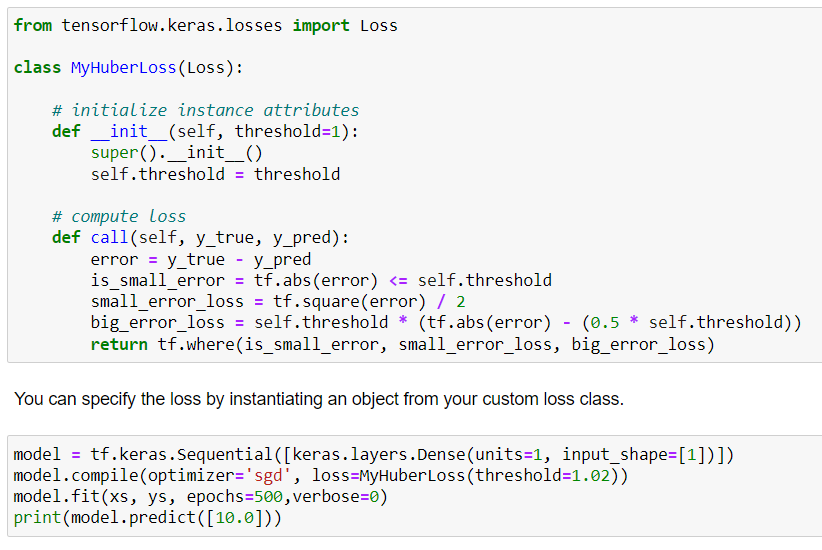
Huber Loss:



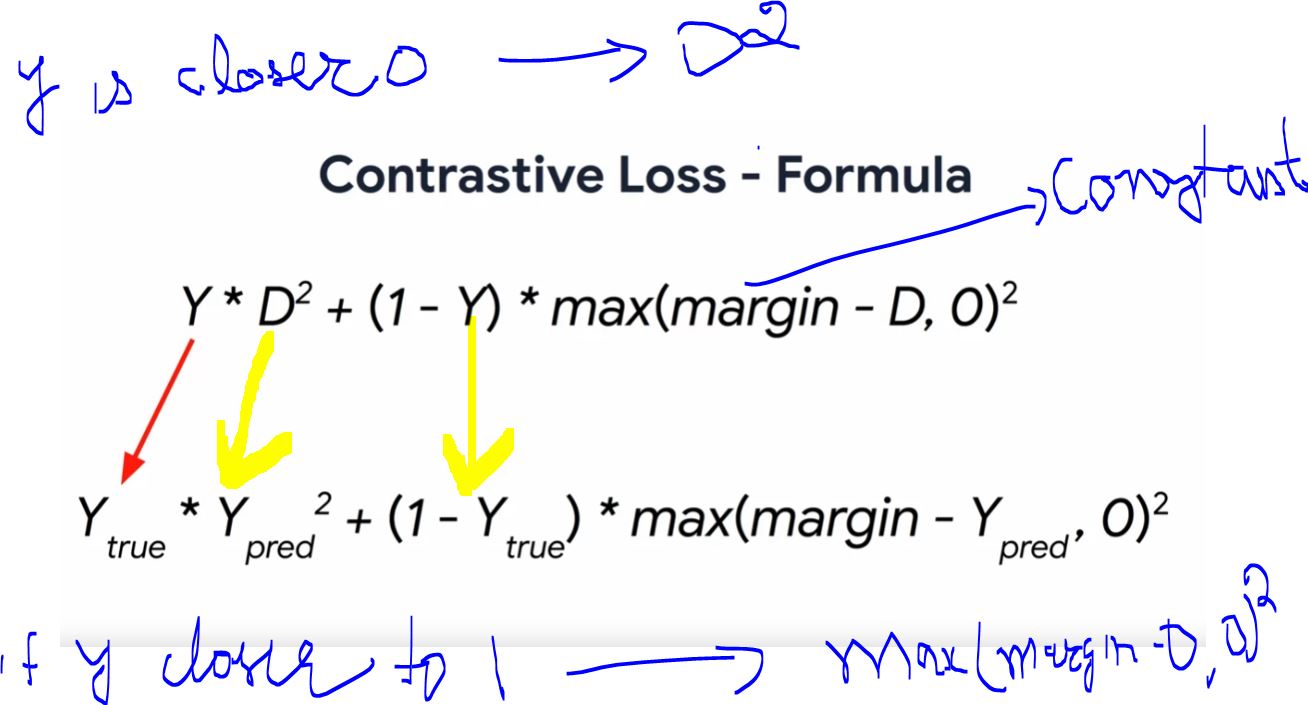
Huber loss implemention:



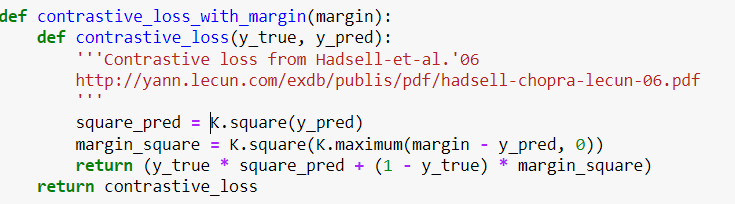
Class level implementation:-



Contrastive loss: Used to find similarity between two vactors.



Contrastive loss implementaions:



# measure the similarity of the two vector outputs

output = Lambda(euclidean\_distance, name="output\_layer", output\_shape=eucl\_dist\_output\_shape)([vect\_output\_a, vect\_output\_b])

# specify the inputs and output of the model

model = Model([input\_a, input\_b], output)

**Custom Dense Layer** will contain weights that can be updated during training.

 Requires three functions: \_\_init\_\_(), build() and call()

class SimpleDense(Layer):

# add an activation parameter

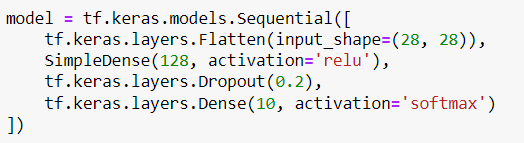
def \_\_init\_\_(self, units=32, activation=None):

def build(self, input\_shape):

def call(self, inputs):

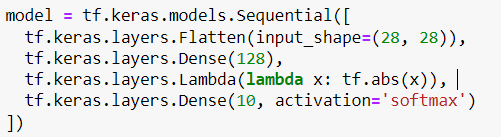
# pass the computation to the activation layer

return self.activation(tf.matmul(inputs, self.w) + self.b)



<https://www.coursera.org/learn/custom-models-layers-loss-functions-with-tensorflow/ungradedLab/laUcE/custom-dense-layer/lab?path=%2Fnotebooks%2FC1_W3_Lab_2_custom-dense-layer.ipynb%23Ungraded-Lab%3A-Building-a-Custom-Dense-Layer>

**Lambda layer** define a custom function that the Lambda layer will call



<https://www.coursera.org/learn/custom-models-layers-loss-functions-with-tensorflow/ungradedLab/AFDgQ/lambda-layer/lab?path=%2Fnotebooks%2FC1_W3_Lab_1_lambda-layer.ipynb>

**Test Implementation of calls from Utils layers:-**

import utils

utils.test\_simple\_quadratic(SimpleQuadratic)

# Coding a Wide and Deep Model

# inherit from the Model base class

class WideAndDeepModel(Model):

def \_\_init\_\_(self, units=30, activation='relu', \*\*kwargs): # Initialize the instance attributes.

def call(self, inputs): ##build the network and return the output layers

# create an instance of the model

model = WideAndDeepModel()

<https://www.coursera.org/learn/custom-models-layers-loss-functions-with-tensorflow/ungradedLab/JB1Zr/build-a-basic-model/lab?path=%2Fnotebooks%2FC1_W4_Lab_1_basic-model.ipynb>

Residual Networks make use of skip connections to make deep models easier to train.

## Implement Model subclasses

class IdentityBlock(tf.keras.Model):

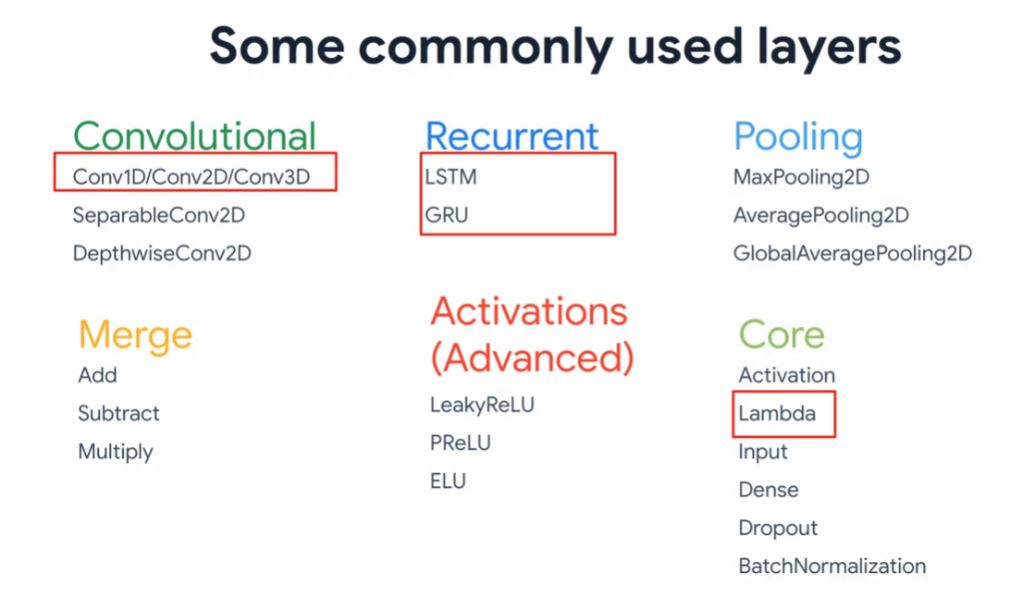
def \_\_init\_\_(self, filters, kernel\_size):

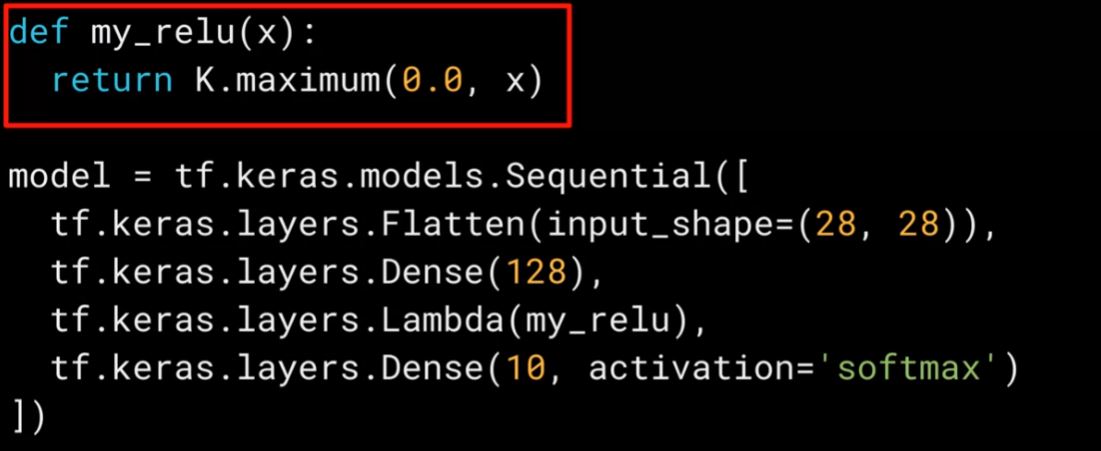
super(IdentityBlock, self).\_\_init\_\_(name='')

def call(self, input\_tensor):

<https://www.coursera.org/learn/custom-models-layers-loss-functions-with-tensorflow/ungradedLab/hStfq/build-a-resnet-model/lab?path=%2Fnotebooks%2FC1_W4_Lab_2_resnet-example.ipynb>

Sequential and Functional APIs have their limitations?





Example for implementing multiple loss and metrics:

model.compile(optimizer=rms,

loss = {'wine\_type' : 'binary\_crossentropy',

'wine\_quality' : 'mean\_squared\_error'

},

metrics = {'wine\_type' : 'accuracy',

'wine\_quality': tf.keras.metrics.RootMeanSquaredError()

}

)