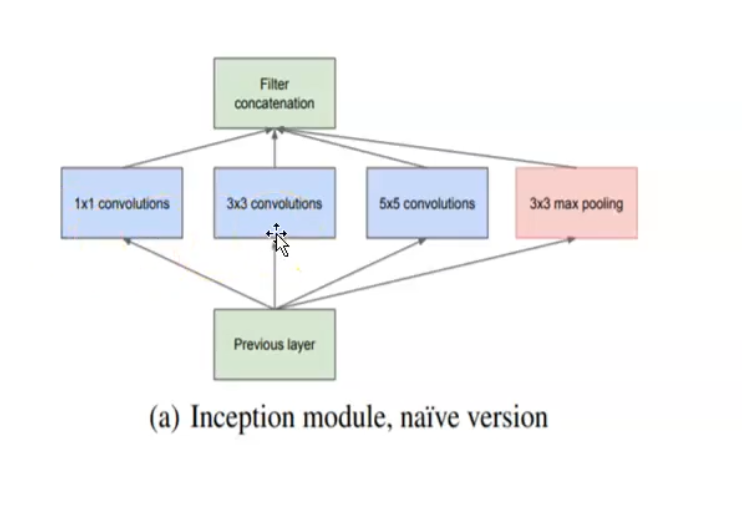
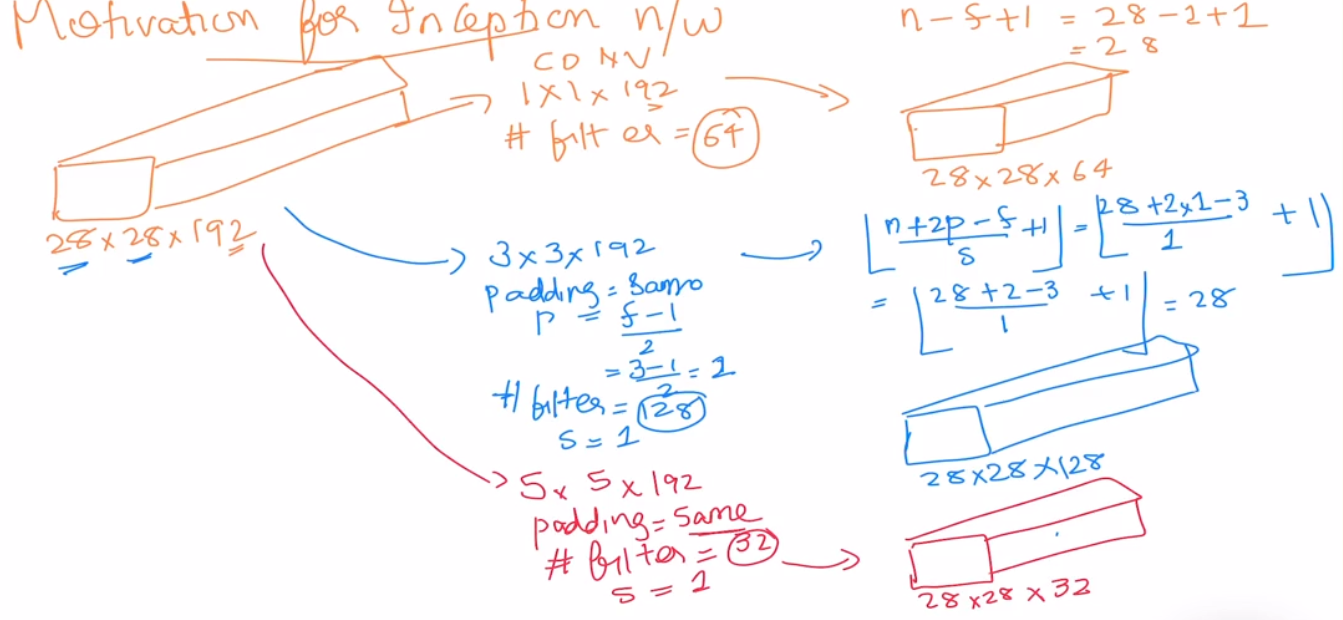
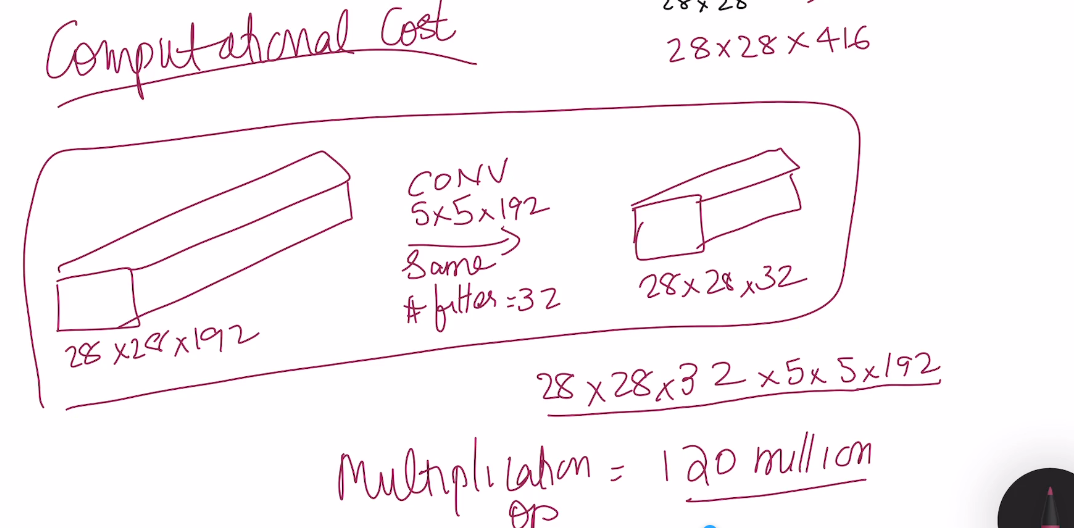
**Inception** : For bigger network it is difficult to set filter becasuse larger filter work better for the image with information distributed globally while for image with smaller portion we requireed smaller filter.

Padding = same

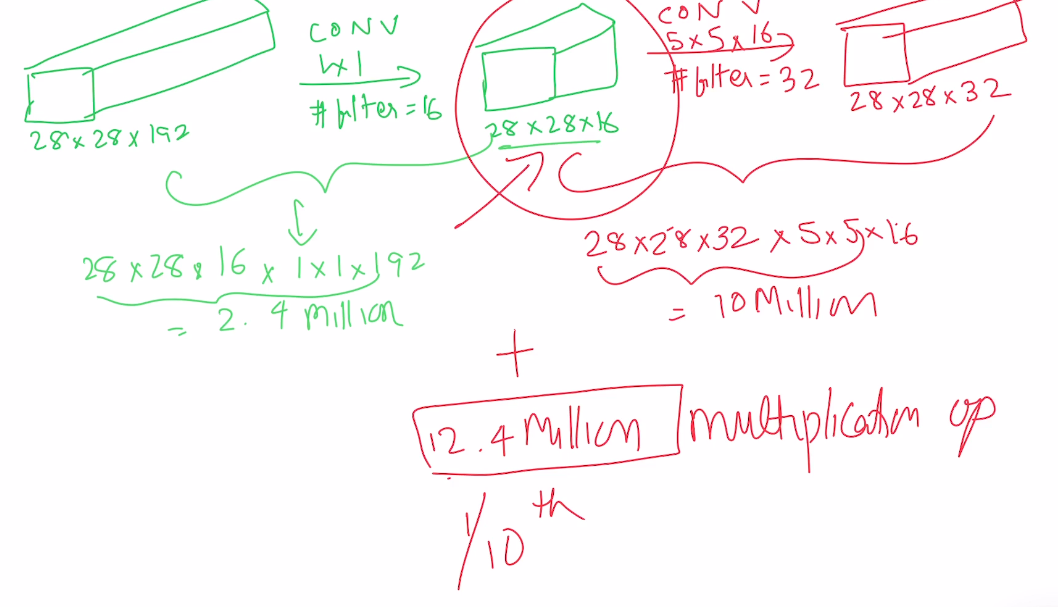
= (f-1)/2

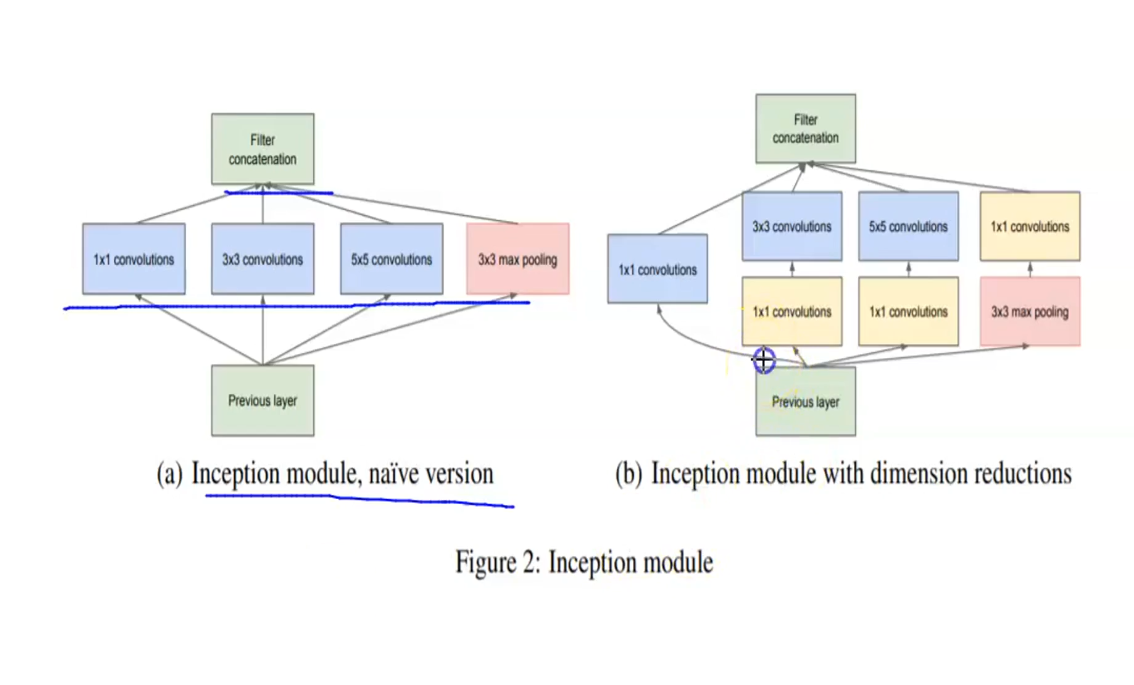


Computational cosat is very high:



Computational cosat is very low as compared to previous version:



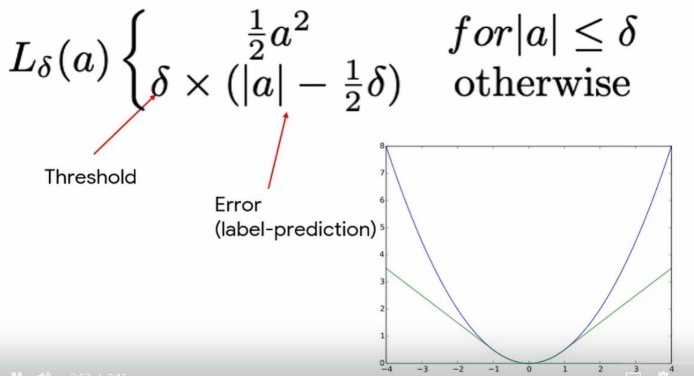


Train is 60000,28, 28

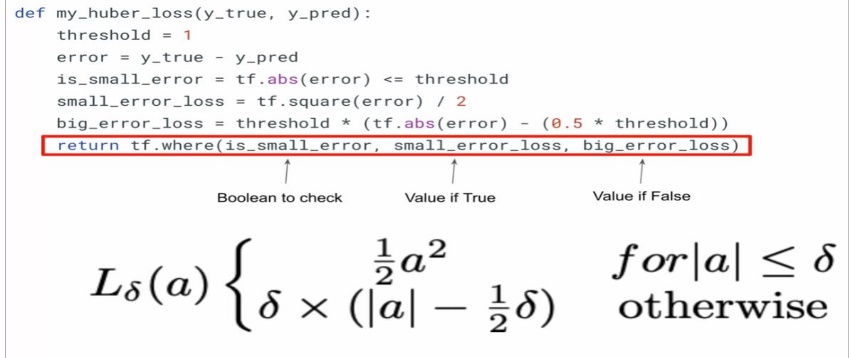
Keras take data in this format X\_train.reshape(60000,28,28,1)

Loss functions help measure how well a model is doing

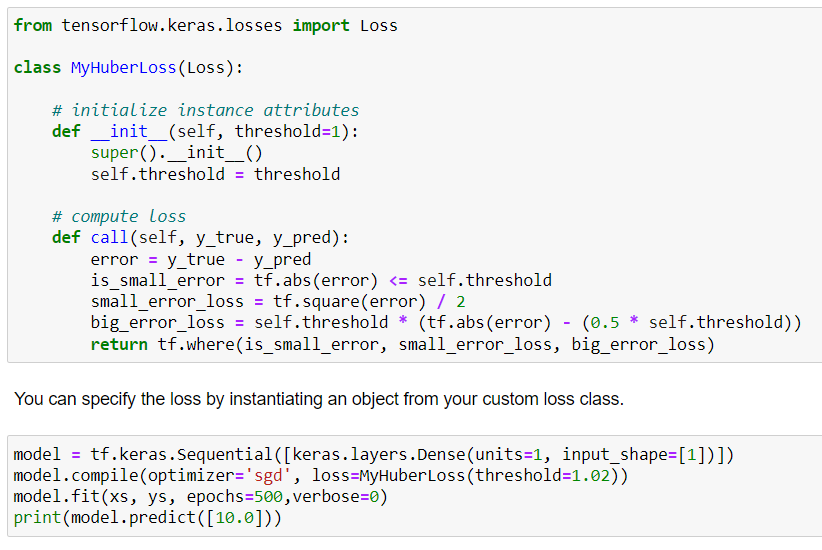
Huber Loss: The Huber loss function can be used **to balance between the Mean Absolute Error, or MAE, and the Mean Squared Error, MSE**. It is therefore a good loss function for when you have varied data or only a few outliers



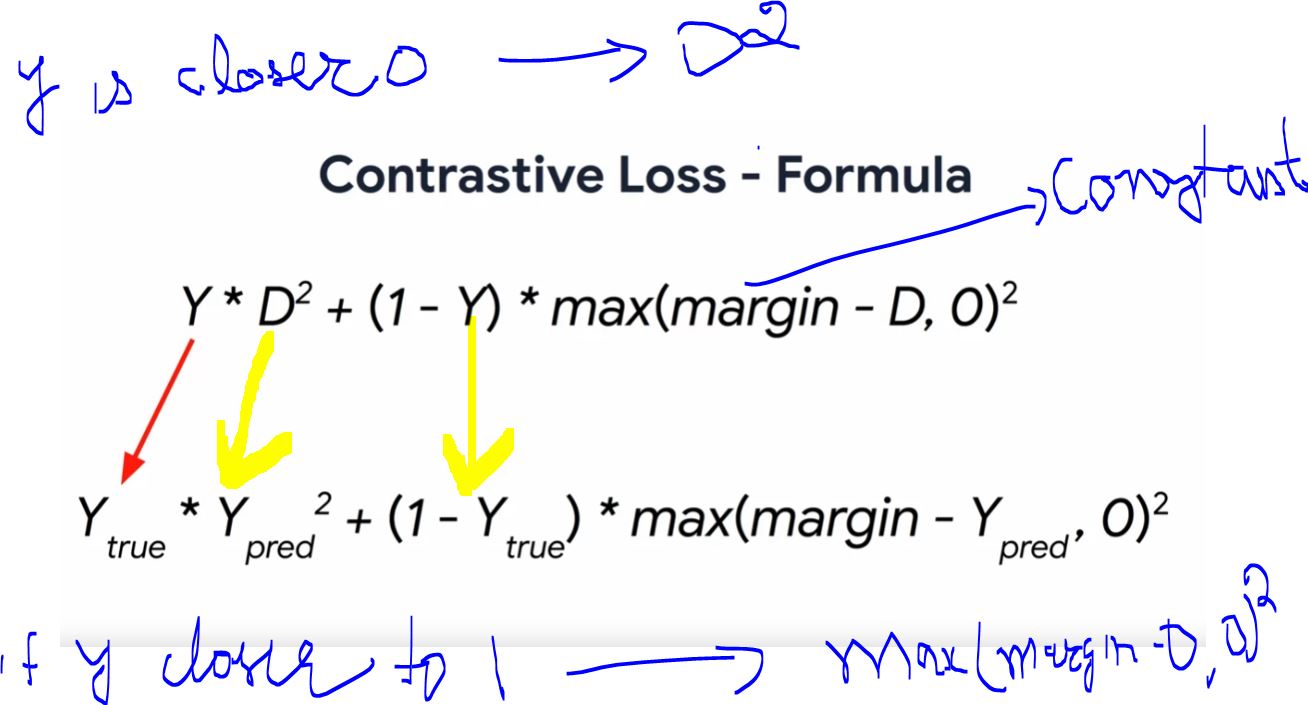
**Huber loss implemention:**



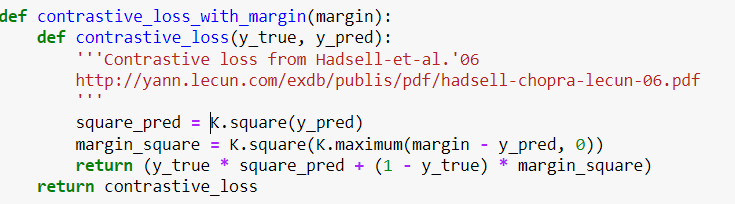
Class level implementation:-



**Contrastive loss:** Used to find similarity between two vactors. Contrastive loss, like triplet and magnet loss, is used **to map vectors that model the similarity of input items**



**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

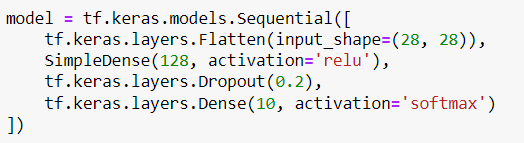
def \_\_init\_\_(self, units=32, activation=None):

def build(self, input\_shape):

def call(self, inputs):

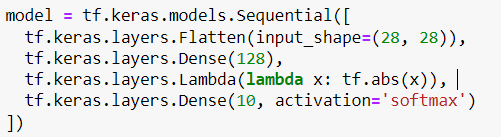
# pass the computation to the activation layer

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?

**Basic Tensor and Python points:-**

**\_\_dict\_\_ is a Python dictionary that contains the object's instance variables and values as key value pairs.**

If you call vars() and pass in an object, it will call the object's \_\_dict\_\_ attribute

# Format a string using f-string notation

i=1

print(f"var{i}")

# Format a string using .format notation

i=2

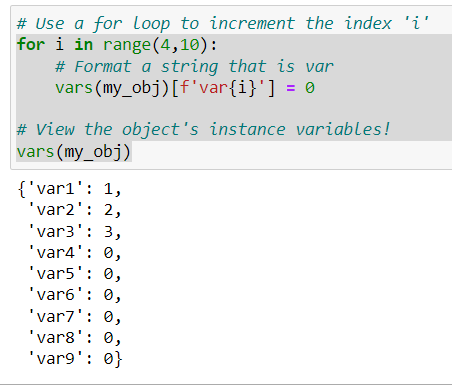
print("var{}".format(i))

**my\_obj = MyClass()**

my\_obj = MyClass()

my\_obj.\_\_dict\_\_ # both will give same output

vars(my\_obj) # both will give same output



Important : vars can be shared across the different methods in the pyhton.

<https://www.coursera.org/learn/custom-distributed-training-with-tensorflow/ungradedLab/mDT5h/basic-tensors/lab>

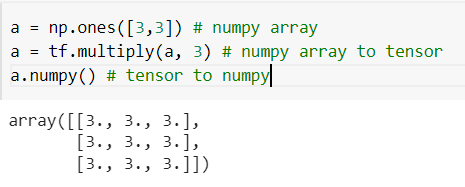
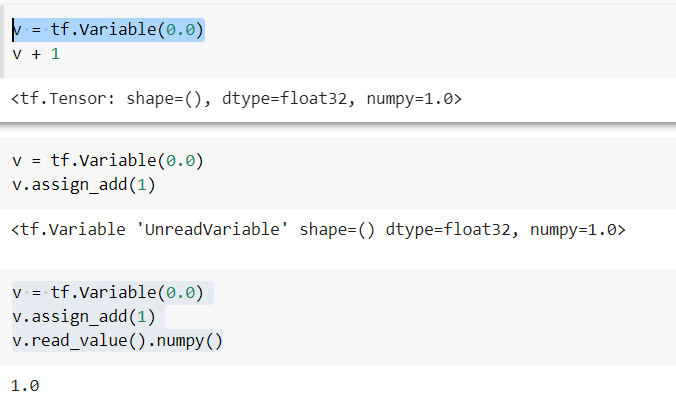
a = tf.constant([[1 , 2],

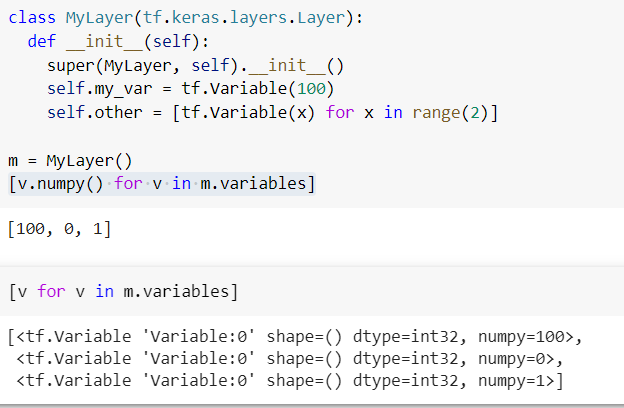
[3, 4]])

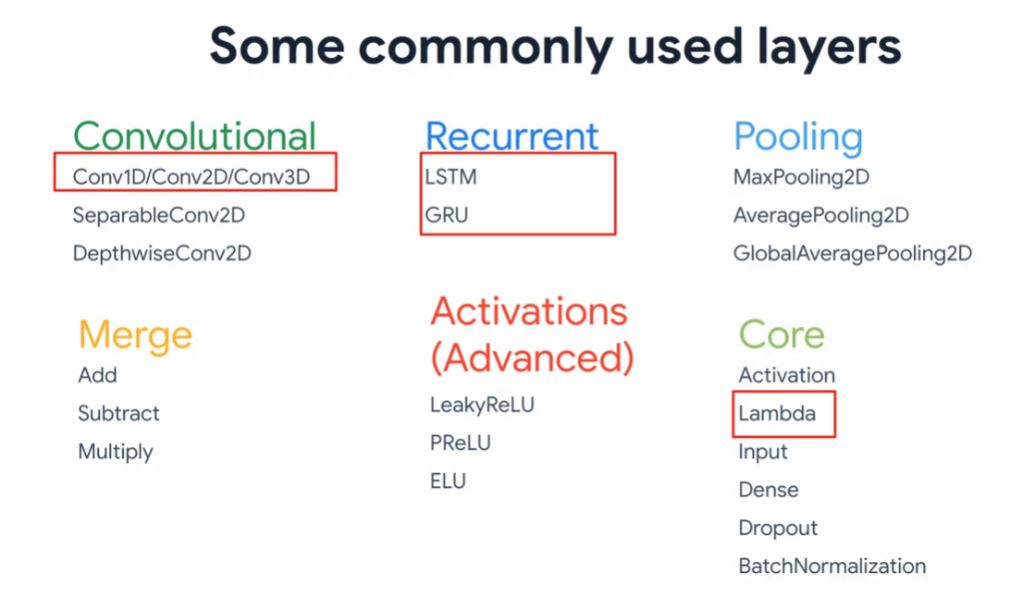
tf.add(a, 1)

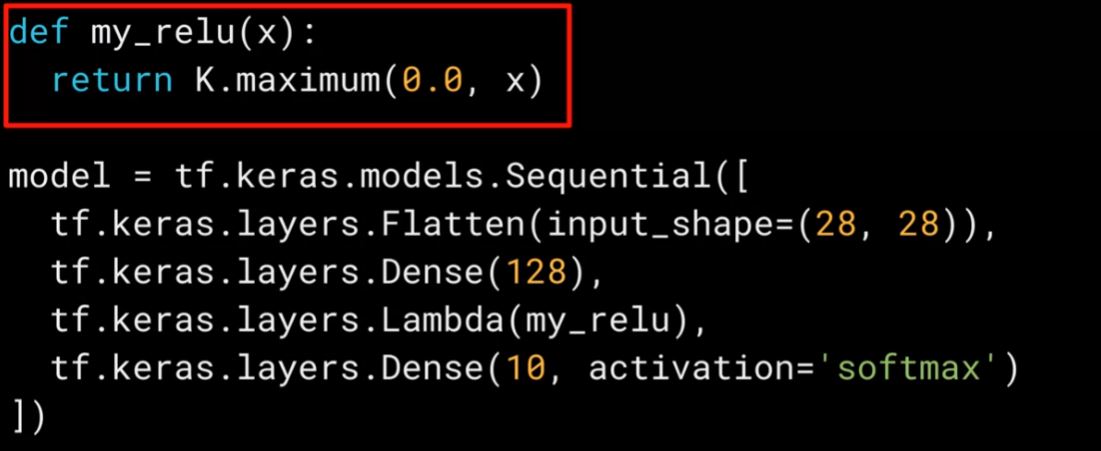
<tf.Tensor: shape=(2, 2), dtype=int32, numpy=array([[2, 3], [4, 5]], dtype=int32)>

Numpy interoperability







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() } )

**VGG-16** : 16 layers, 3 \* 3 kernel, 2 \* 2 max pooling size.

ImageDataGenerator class supports a number of pixel scaling methods, as well as a range of data augmentation techniques.

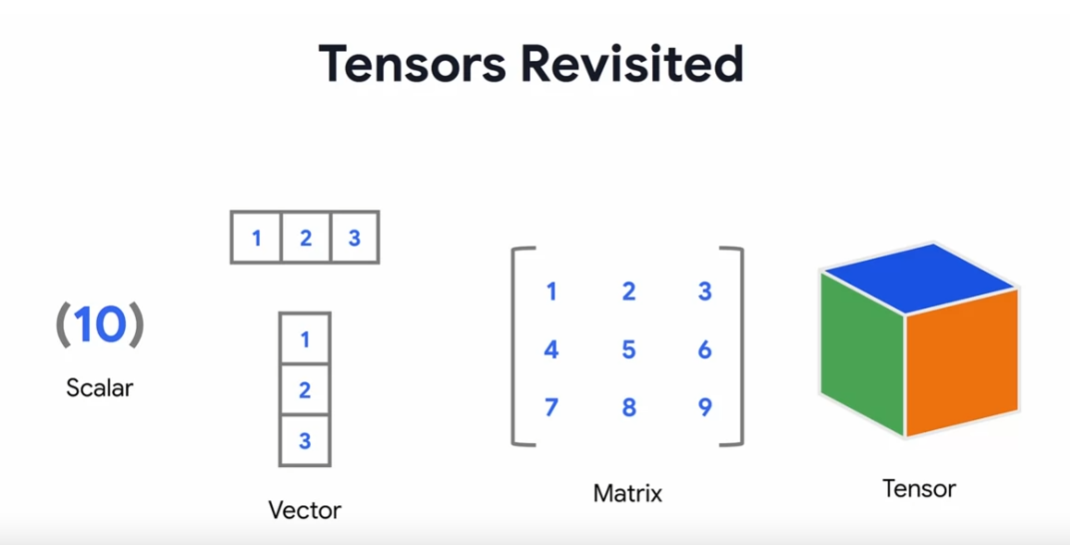
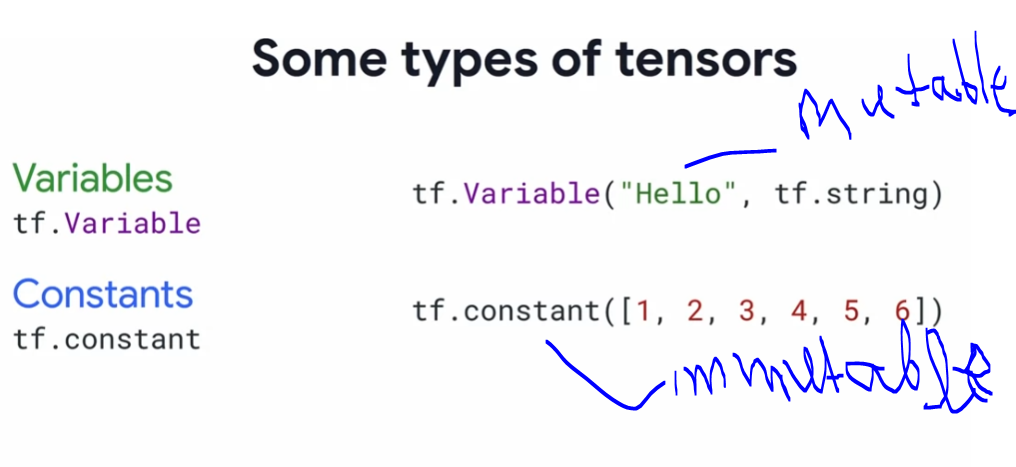
ModelCheckpoint : model will be saved to disk only if the validation accuracy of the model in current epoch is greater than what it was in the last epoch.

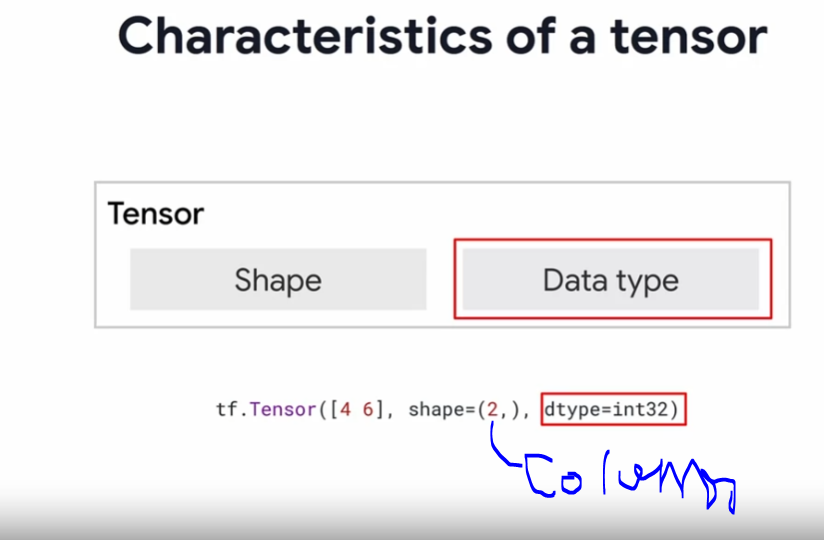
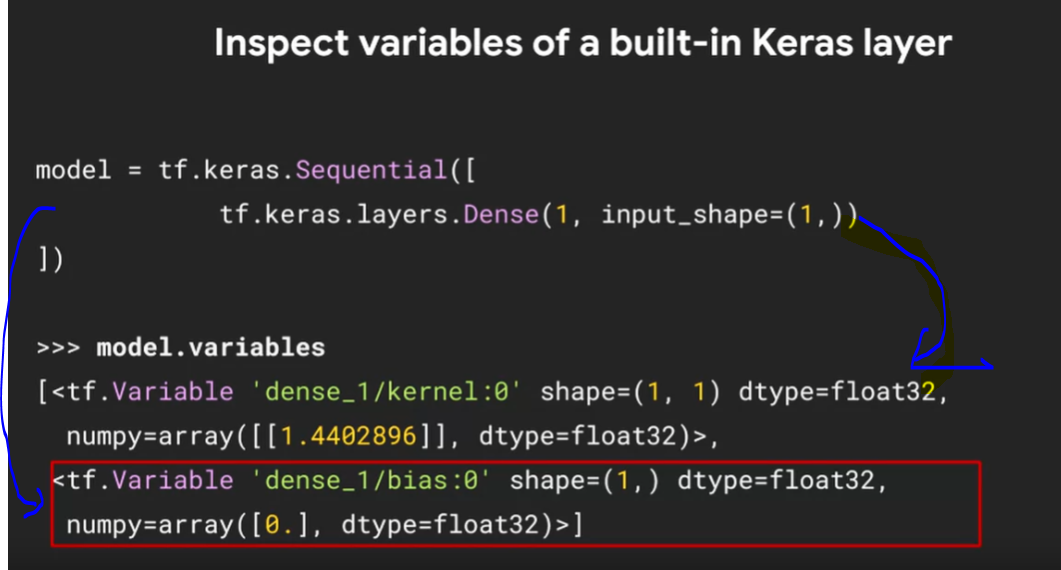
Early Stopping : set patience to 5 which means that the model will stop to train if it doesn’t see any rise in Validation accuracy in 5 consecutive epochs.

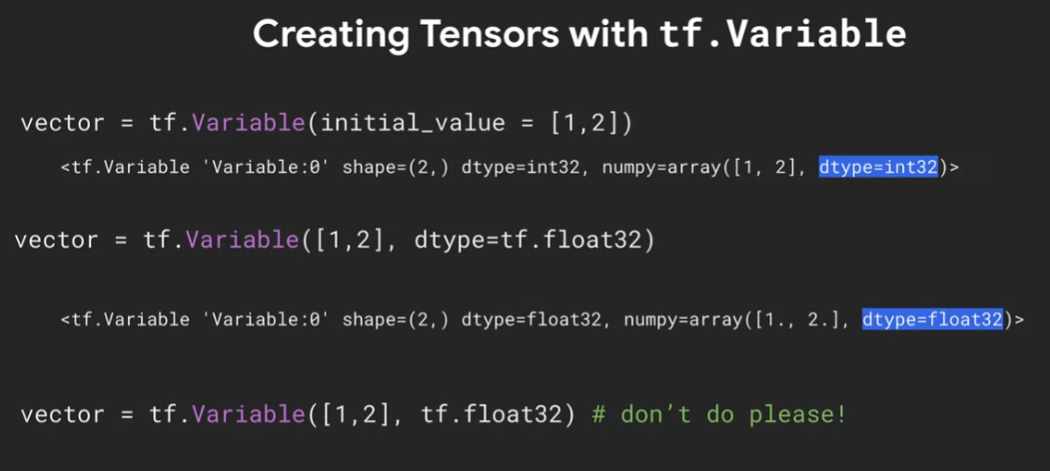
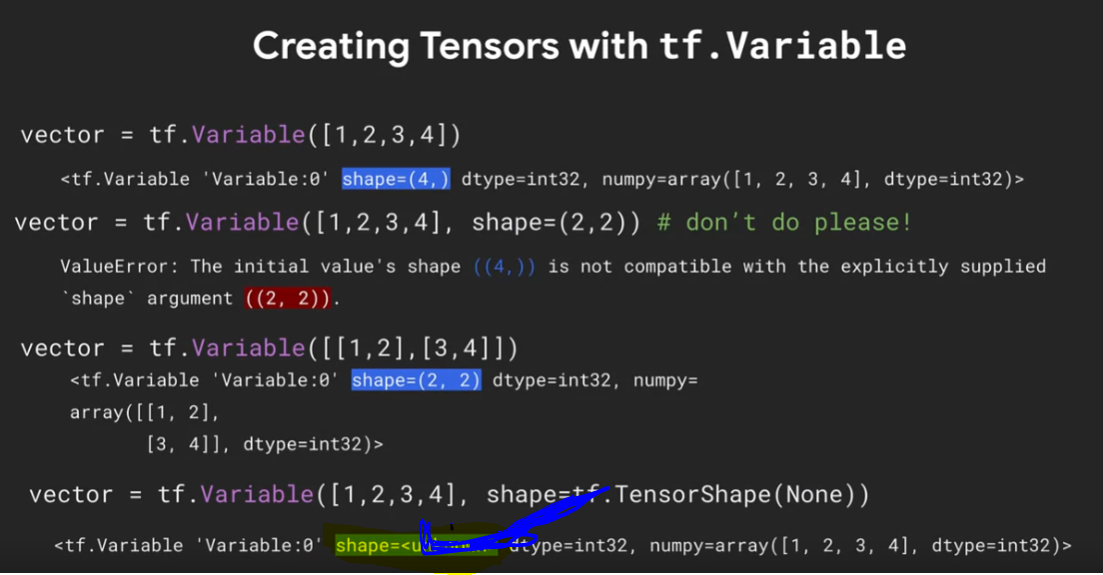
**Eager vs Graph based Execution:**

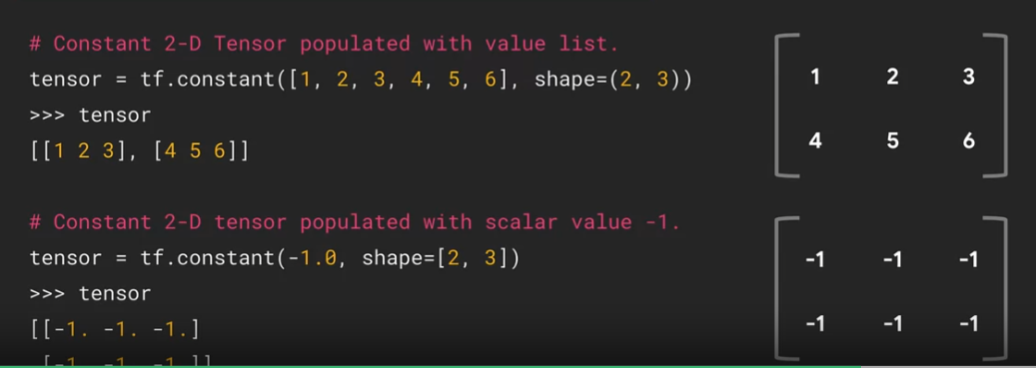
Difference between  graph-based execution and eager execution in TensorFlow?

eager is user friendly



## An overview of callback methods:

tf.keras.callbacks.Callback – call back main class

Common methods for training/testing/predicting

For training, testing, and predicting, following methods are provided to be overridden.

### on\_(train|test|predict)\_begin(self, logs=None)

### on\_(train|test|predict)\_end(self, logs=None)

### on\_(train|test|predict)\_batch\_begin(self, batch, logs=None)

### on\_(train|test|predict)\_batch\_end(self, batch, logs=None)

### Training specific methods

on\_epoch\_begin(self, epoch, logs=None)

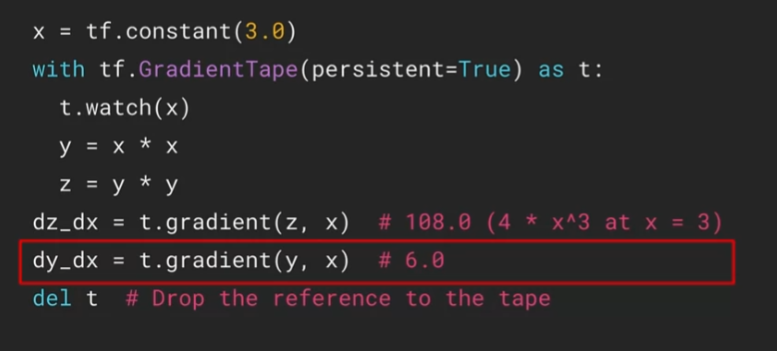
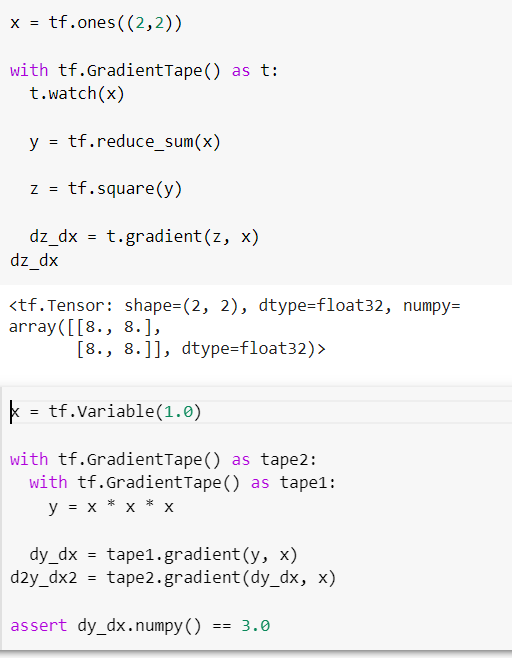
on\_epoch\_end(self, epoch, logs=None)

https://www.coursera.org/learn/custom-models-layers-loss-functions-with-tensorflow/ungradedLab/UaFHl/custom-callbacks/lab?path=%2Fnotebooks%2FC1\_W5\_Lab\_2\_custom-callbacks.ipynb

**Eager mode** : One type ovpaf mode in TensorFlow allows for immediate evaluation of values

**Gradient Tape**: Intensive flow optimizers are implemented using TensorFlow automatic differentiation API call **Gradient Tape**

**Basic gradient eg:- Basic gradient presistant = true eg:-**



[**https://www.coursera.org/learn/custom-distributed-training-with-tensorflow/ungradedLab/jKn7w/gradient-tape-basics/lab?path=%2Fnotebooks%2FC2\_W1\_Lab\_2\_gradient-tape-basics.ipynb**](https://www.coursera.org/learn/custom-distributed-training-with-tensorflow/ungradedLab/jKn7w/gradient-tape-basics/lab?path=%2Fnotebooks%2FC2_W1_Lab_2_gradient-tape-basics.ipynb)

[**https://www.coursera.org/learn/custom-distributed-training-with-tensorflow/programming/TwjF2/basic-tensor-operations/lab?path=%2Fnotebooks%2FC2W1\_Assignment.ipynb**](https://www.coursera.org/learn/custom-distributed-training-with-tensorflow/programming/TwjF2/basic-tensor-operations/lab?path=%2Fnotebooks%2FC2W1_Assignment.ipynb)

<https://www.youtube.com/watch?v=TudQZtgpoHk&t=1409s>

https://www.geeksforgeeks.org/optimizers-in-tensorflow/#:~:text=Optimizers%20are%20techniques%20or%20algorithms,better%20accuracy%20of%20model%20faster.