EASY COOK
LITTLE TIPS IN Keras

BatchNormalization

- Why BN?
- How does it work?
- Do you really need it?

```
from keras.layers import BatchNormalization
-----------------------------------------------------------------------------------------

input = Input(shape=(20, ), name='Input_Layer')
dense_1 = Dense(32, activation='relu',
                name='dense_layer_1')(input)
bn_1 = BatchNormalization(name='batchNorm_1')(dense_1)
dense_2 = Dense(16, activation='relu',
                name='dense_layer_2')(bn_1)
```

Custom Layer

- When do we need this?
- 4 steps to build your layer
- One more thing you need to know!

```
from keras.layers import Lambda
--------------------------------------------------------------------------------
lambda_layer1 =
Lambda(lambda x: K.xxxxx)(‘keraslayer’)
```

```
from keras import backend as K
from keras.layers import Layer
--------------------------------------------------------------------------------

class MyLayer(Layer):
    def __init__(self, output_dim, **kwargs):
        self.output_dim = output_dim
        super(MyLayer, self).__init__(**kwargs)

    def build(self, input_shape):
        self.kernel = self.add_weight(name=‘cLayer’,
             shape=(input_shape[1], self.output_dim),
             initializer=‘uniform’, trainable=True)
        super(MyLayer, self).build(input_shape)

    def call(self, x):
        return K.dot(x, self.kernel)

    def compute_output_shape(self, input_shape):
        return (input_shape[0], self.output_dim)
```

Custom Loss

- When do we need this?
- Another way, *Custom Layer*
- Same as custom metrics

```
from keras import backend as K
```

```
def custom_loss_1(y, pred_y):
    # identity loss
    loss = K.mean((pred_y - 0 * y), axis=-1)
    return loss

def custom_loss_2(y, pred_y):
    # mse loss
    loss = K.mean(K.square(pred_y - y), axis=-1)
    return loss
```

Generator **

- Why we need it?
- One of the most important skills in Keras

```
from keras.utils import Sequence
-------------------------------------------------------------------------------

class Generator(Sequence):
    def __init__(self):
        # setting the initial parameters

    def __len__(self):
        # Denotes the number of batches per epoch

    def __getitem__(self):
        # Generate one batch of data

    def on_epoch_end(self):
        # Updates or do something after each epoch

    def __data_generation(self):
        # Generates data containing batch_size samples
```