Keras is a powerful Python library for deep learning. We need to design and configure the deep learning model in order to make the performance best. Most decisions must be resolved through experiments and evaluated against actual results. Therefore, it is essential to have a reliable metric to evaluate the performance of neural networks and deep learning models.

A metric is a function that is used to judge the performance of your model. Metric functions are to be supplied in the metrics parameter when a model is compiled.

Metrics usually has two arguments, ‘y\_true’ and ‘y\_pred’.

The evaluation indicators it receives are loss function(mse, etc.) and accuracy, both of which act on the training set and verification set (eg: loss:..). Of course, The results of this performance evaluation will not really be used for training, but only for display.

**binary\_accuracy**

keras.metrics.binary\_accuracy(y\_true, y\_pred)

This metric simply defines the accuracy binarily. It rounds y\_pred and implies that the threshold is 0.5, everything above 0.5 will be considered as correct.

**categorical\_accuracy**

keras.metrics.categorical\_accuracy(y\_true, y\_pred)

This metric uses K.argmax(y\_true), which takes the highest value to be the prediction and matches against the comparative set. It checks to see if the index of the maximal true value is equal to the index of the maximal predicted value.

**sparse\_categorical\_accuracy**

keras.metrics.sparse\_categorical\_accuracy(y\_true, y\_pred)

This metric check to see if the maximal true value is equal to the index of the maximal predicted value. It might be a better metric than categorical\_accuracy in some cases depending on your data.

**top\_k\_categorical\_accuracy**

def top\_k\_categorical\_accuracy(y\_true, y\_pred, k=5):

return K.mean(K.in\_top\_k(y\_pred, K.argmax(y\_true, axis=-1), k), axis=-1)

The metric ‘top\_k\_categorical\_accuracy’ is used to calculate the top-k accuracy. It means that the prediction is considered correct when there is a target category in the first k values of the predicted values.

**sparse\_top\_k\_categorical\_accuracy**

def sparse\_top\_k\_categorical\_accuracy(y\_true, y\_pred, k=5):

# If the shape of y\_true is (num\_samples, 1), flatten to (num\_samples,)

return K.mean(K.in\_top\_k(y\_pred, K.cast(K.flatten(y\_true), 'int32'), k),

axis=-1)

The function of metric ‘sparse\_top\_k\_categorical\_accuracy’ is same as ‘op\_k\_categorical\_accuracy’. The difference is that ‘sparse\_top\_k\_categorical\_accuracy’ can be applied to sparse cases.

**Custom Metrics**

**IoU metric**

In order to test intersection over union which can be transferred to union of mask vs background

The basic expression of iou is IoU = (|X & Y|)/ (|X or Y|)

Then we found pred = the greater one between y\_pred and 0.5, which is random pick chance

Union = y\_true + pred

Intersection over union should be y\_true\*pred

Then we apply parameters to IoU function

We have iou = K.sum(intersec) / (K.sum(union) + K.epsilon())

Implementation as below:

import keras.backend as K

def iouMetric(y\_true, y\_pred):

pred = K.cast(K.greater(y\_pred, 0.5), K.floatx())

union = K.cast(K.greater(y\_true + pred, 0), K.floatx())

intersec = y\_true \* pred

iou = K.sum(intersec) / (K.sum(union) + K.epsilon())

return iou