A Beginners Guide to TensorFlow2.X

Official Documentation at:
Module: tf | TensorFlow v2.12.0

Installation

```
# Requires the latest pip
! pip install --upgrade pip
# Current stable release for CPU and GPU
! pip install tensorflow
# Or try the preview build (unstable)
! pip install tf-nightly
```

Creating Tensors

```
# Import Tensorflow
import tensorflow as tf
# Create Tensor from numpy array or Python list
# Create Constant Tensor
c = tf.constant([1, 2, 3, 4, 5, 6])
# Create a Variable Tensor
x = tf.Variable([[1.], [2.]])
# Convert Tensor, Python list, scalar or
# numpy array to Tensor
tf.convert_to_tensor(c, dtype=tf.float32)
# Create random Tensor from a distribution
# Normal Distribution
tf.random.normal((10,32,32), mean=1, stddev=1.0)
# Uniform distribution
tf.random.uniform((10,32,32), minval=0, maxval=1)
# Gamma Distribution
tf.random.gamma((10,32,32))
```

Reshaping Tensors

```
# Reshape
tf.reshape(c, [2,3])
# Use -1 to infer shape
tf.reshape(c, [2,-1])
# Pass '[-1]' to flatten
tf.reshape(c, [-1])
# Expand dims, adds new axis at 'axis' position
tf.expand_dims(c, axis=0)
# Create Rank R+1 Tensor from list of Rank R
x = tf.constant([1, 4]) # (2,) Tensor
y = tf.constant([2, 5]) # (2,) Tensor
z = tf.constant([3, 6]) # (2,) Tensor
tf.stack([x, y, z], axis = 1) # (2,3) Tensor
# Concatenates tensors along specified axis
t1 = [[1, 2, 3], [4, 5, 6]] # (2,3) Tensor
t2 = [[7, 8, 9], [10, 11, 12]] # (2,3) Tensor
tf.concat([t1, t2], 0) # (4,3) Tensor
```

Activation Functions

Name	Usage
relu	Default activation
sigmoid	Binary classification
tanh	Faster convergence than sigmoid
softmax	Multiclass classification

Layers

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
# A usual dense layer
layers.Dense(units, activation, input_shape)
# A layer that flattens the input
layers.Flatten(input_shape)
# 2D convolution for images
layers.Conv2D(filters, kernel_size, activation, input_shape)
# Maxpooling layer to reduce input dimension, has 0 trainable parameters
layers.MaxPool2D(pool_size)
# Dropout layer to prevent overfitting
layers.Dropout(rate)
# Embedding layer to learn embeddings for words
layers.Embedding(input_dim, output_dim, input_length)
# Global average pooling for temporal data
layers.GlobalAveragePooling1D()
# Bidirectional Long Short-Term Memory layer
layers.Bidirectional(layers.LSTM(units, return_sequence))
# 1D convolution for signals
layers.Conv1D(filters, kernel_size, activation, input_shape)
# Bidirectional Gated Recurrent Unit
layers.Bidirectional(layers.GRU(units))
# Fully-connected RNN where the output is to be fed back to input
layers.SimpleRNN(units, activation, return sequences, input_shape)
# Wraps arbitrary expressions as a Layer object
layers.Lambda(function)
```

Models

```
# Sequentially adds a stack of layers into a tf.ker-as.M-odel
model = tf.keras.Sequential(layers)
# Configures the model for training
model.compile(optimizer, loss, metrics)
# Trains the model for a fixed number of epochs
history = model.fit(x, y, epoch)
# Fits the model on data yielded batch--by--batch by a Python generator
history = model.fit_generator(train_generator, steps_per_epoch, epochs,
                              validation_data, validation_steps)
# Returns the loss value & metrics values for the model in test mode
model.evaluate(x, y)
# Generates output predic-tions for the input samples
model.predict(x)
# Prints the summary of the model
model.summary()
# Saves a model as a TensorFlow SavedModel or HDF5 file
model.save("path/to/model")
# Stops training when true
model.stop_training
# Load a saved model from path
new_model = models.load_model("path/to/model")
```

Optimizers

Name	Usage
Adam	Adam combines the good properties of Adadelta and RMSprop and hence tend to do better for most of the problems.
SGD	Stochastic gradient descent is very basic and works well for shallow networks.
AdaGrad	Adagrad can be useful for sparse data such as tf-idf.
AdaDelta	Extension of AdaGrad which tends to remove the decaying learning Rate problem of it.
RMSprop	Very similar to AdaDelta.

Read Dataset

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
# Load data set
mnist = datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

Build a model

```
# Initialize a model
model = models.Sequential()
# Add a layer to flatten the input
model.add(layers.Flatten(input_shape=(28, 28)))
# Add a dense layer
model.add(layers.Dense(512, activation=tf.nn.relu))
# Add a dropout layer
model.add(layers.Dropout(0.2))
# Add output layer with softmax activation
model.add(layers.Dense(10, activation=tf.nn.softmax))
```

Compile the Model

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Train and Evaluate the Model

```
model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

Loss Functions

Name	Usage
MeanSquaredError	Default for regression problems.
MeanSquaredLogarithmicError	For regression problems with large spread.
MeanAbsoluteError	More robust to outliers.
BinaryCrossEntropy	Default for binary classification problems.
Hinge	Intended for binary classification when the target values are in the set {-1, 1}.
SquaredHinge	If using a hinge loss does result in better performance on a given binary classification problem, is likely that a squared hinge loss may be appropriate.
CategoricalCrossEntropy	Default for multi-class classification problems.
SparseCategoricalCrossEntropy	When target variable is catergorical and is not one-hot encoded for training.
KLD	KL divergence is more commonly used when using models that learn to approx-imate a more complex function than simply multi--class classi-fic-ation, such as in the case of an autoen-coder used for learning a dense feature repres-ent-ation under a model that must recons-truct the original input.
Huber	Less sensitive to outliers

References

Module: tf | TensorFlow v2.12.0
19-04-11-Cheat-Sheet-TensorFlow-2-0.pdf (storage.googleapis.com)

https://github.com/ryanxjhan/TensorFlow-2.x-Cheat-Sheet

https://github.com/patrickphat/Tensorflow-2-cheatsheet/blob/master/tf-cheatsheet.md

https://www.datacamp.com/cheat-sheet/working-with-dates-and-times-in-python-cheat-sheet

pratik-tan10 (Pratik Dhungana) · GitHub