TensorFlow

TensorFlow API r1.4

tf.contrib.layers.batch_norm

```
batch_norm(
    inputs,
    decay=0.999,
    center=True,
    scale=False,
    epsilon=0.001,
    activation_fn=None,
    param_initializers=None,
    param_regularizers=None,
    updates_collections=tf.GraphKeys.UPDATE_OPS,
    is_training=True,
    reuse=None,
    variables_collections=None,
    outputs_collections=None,
    trainable=True,
    batch_weights=None,
    fused=None,
    data_format=DATA_FORMAT_NHWC,
    zero_debias_moving_mean=False,
    scope=None,
    renorm=False,
    renorm_clipping=None,
    renorm_decay=0.99
)
```

Defined in `tensorflow/contrib/layers/python/layers/layers.py`.

See the guide: Layers (contrib) > Higher level ops for building neural network layers

Adds a Batch Normalization layer from http://arxiv.org/abs/1502.03167.

"Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift"

Sergey Ioffe, Christian Szegedy

Can be used as a normalizer function for conv2d and fully_connected.

★ **Note:** when training, the moving_mean and moving_variance need to be updated. By default the update ops are placed in `tf.GraphKeys.UPDATE_OPS`, so they need to be added as a dependency to the `train_op`. For example:

```
update_ops = tf.get_collection(tf.GraphKeys.UPDATE_OPS)
with tf.control_dependencies(update_ops):
  train_op = optimizer.minimize(loss)
```

One can set updates_collections=None to force the updates in place, but that can have a speed penalty, especially in distributed settings.

Args:

- `inputs`: A tensor with 2 or more dimensions, where the first dimension has `batch_size`. The normalization is over all but the last dimension if `data_format` is `NHWC` and the second dimension if `data_format` is `NCHW`.

- `decay`: Decay for the moving average. Reasonable values for `decay` are close to 1.0, typically in the multiple-nines range: 0.999, 0.99, 0.9, etc. Lower `decay` value (recommend trying `decay`=0.9) if model experiences reasonably good training performance but poor validation and/or test performance. Try zero_debias_moving_mean=True for improved stability.
- `center`: If True, add offset of `beta` to normalized tensor. If False, `beta` is ignored.
- `scale`: If True, multiply by `gamma`. If False, `gamma` is not used. When the next layer is linear (also e.g. `nn.relu`), this can be disabled since the scaling can be done by the next layer.
- `epsilon`: Small float added to variance to avoid dividing by zero.
- `activation_fn`: Activation function, default set to None to skip it and maintain a linear activation.
- `param_initializers`: Optional initializers for beta, gamma, moving mean and moving variance.
- `param_regularizers`: Optional regularizer for beta and gamma.
- `updates_collections`: Collections to collect the update ops for computation. The updates_ops need to be executed with the train_op. If None, a control dependency would be added to make sure the updates are computed in place.
- `is_training`: Whether or not the layer is in training mode. In training mode it would accumulate the statistics of the moments into `moving_mean` and `moving_variance` using an exponential moving average with the given `decay`. When it is not in training mode then it would use the values of the `moving_mean` and the `moving_variance`.
- `reuse`: Whether or not the layer and its variables should be reused. To be able to reuse the layer scope must be given.
- `variables_collections`: Optional collections for the variables.
- `outputs_collections`: Collections to add the outputs.
- `trainable`: If `True` also add variables to the graph collection `GraphKeys.TRAINABLE_VARIABLES` (see `tf.Variable`).
- `batch_weights`: An optional tensor of shape `[batch_size]`, containing a frequency weight for each batch item. If present, then the batch normalization uses weighted mean and variance. (This can be used to correct for bias in training example selection.)
- `fused`: if `True`, use a faster, fused implementation if possible. If `None`, use the system recommended implementation.
- `data_format`: A string. `NHWC` (default) and `NCHW` are supported.
- `zero_debias_moving_mean`: Use zero_debias for moving_mean. It creates a new pair of variables 'moving_mean/biased' and 'moving_mean/local_step'.
- `scope`: Optional scope for `variable_scope`.
- `renorm`: Whether to use Batch Renormalization (https://arxiv.org/abs/1702.03275). This adds extra variables during training. The inference is the same for either value of this parameter.
- `renorm_clipping`: A dictionary that may map keys 'rmax', 'rmin', 'dmax' to scalar `Tensors` used to clip the renorm correction. The correction `(r, d)` is used as `corrected_value = normalized_value * r + d`, with `r` clipped to [rmin, rmax], and `d` to [-dmax, dmax]. Missing rmax, rmin, dmax are set to inf, 0, inf, respectively.
- `renorm_decay`: Momentum used to update the moving means and standard deviations with renorm. Unlike `momentum`, this affects training and should be neither too small (which would add noise) nor too large (which would give stale estimates). Note that `decay` is still applied to get the means and variances for inference.

Returns:

A `Tensor` representing the output of the operation.

Raises:

- `ValueError`: If `data_format` is neither `NHWC` nor `NCHW`.

- `ValueError` : If the rank of `inputs` is undefined.
- `ValueError` : If rank or channels dimension of `inputs` is undefined.

Last updated November 2, 2017.

Stay Connected

Blog

GitHub

Twitter

Support

Issue Tracker

Release Notes

Stack Overflow

English

Terms | **Privacy**