tf.contrib.gan.features.VBN

Contents

Class VBN

Methods

__init__

__call__

reference_batch_normalization

Class **VBN**

Defined in `tensorflow/contrib/gan/python/features/python/virtual_batchnorm_impl.py`.

A class to perform virtual batch normalization.

This technique was first introduced in `Improved Techniques for Training GANs` (Salimans et al, https://arxiv.org/abs/1606.03498). Instead of using batch normalization on a minibatch, it fixes a reference subset of the data to use for calculating normalization statistics.

To do this, we calculate the reference batch mean and mean square, and modify those statistics for each example. We use mean square instead of variance, since it is linear.

Note that if `center` or `scale` variables are created, they are shared between all calls to this object.

The `__init__` API is intended to mimic `tf.layers.batch_normalization` as closely as possible.

Methods

`__init__`

```
__init__(
    reference_batch,
    axis=-1,
    epsilon=0.001,
    center=True,
    scale=True,
    beta_initializer=tf.zeros_initializer(),
    gamma_initializer=tf.ones_initializer(),
    beta_regularizer=None,
    gamma_regularizer=None,
    trainable=True,
    name=None,
    batch_axis=0
)
```

Initialize virtual batch normalization object.

We precompute the 'mean' and 'mean squared' of the reference batch, so that `__call__` is efficient. This means that the axis must be supplied when the object is created, not when it is called.

We precompute 'square mean' instead of 'variance', because the square mean can be easily adjusted on a per-example

basis.

Args:

- `reference_batch`: A minibatch tensors. This will form the reference data from which the normalization statistics are calculated. See https://arxiv.org/abs/1606.03498 for more details.
- `axis`: Integer, the axis that should be normalized (typically the features axis). For instance, after a `Convolution2D` layer with `data_format="channels_first"`, set `axis=1` in `BatchNormalization`.
- `epsilon`: Small float added to variance to avoid dividing by zero.
- `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.
- `beta_initializer`: Initializer for the beta weight.
- `gamma_initializer`: Initializer for the gamma weight.
- `beta_regularizer`: Optional regularizer for the beta weight.
- `gamma_regularizer`: Optional regularizer for the gamma weight.
- `trainable`: Boolean, if `True` also add variables to the graph collection `GraphKeys.TRAINABLE_VARIABLES` (see tf.Variable).
- `name`: String, the name of the ops.
- `batch_axis`: The axis of the batch dimension. This dimension is treated differently in `virtual batch normalization` vs `batch normalization`.

Raises:

- `ValueError`: If `reference_batch` has unknown dimensions at graph construction.
- `ValueError`: If `batch_axis` is the same as `axis`.

`__call__`

```
__call__(inputs)
```

Run virtual batch normalization on inputs.

Args:

- `inputs`: Tensor input.

Returns:

A virtual batch normalized version of `inputs`.

Raises:

- `ValueError`: If `inputs` shape isn't compatible with the reference batch.

`reference_batch_normalization`

```
reference_batch_normalization()
```

Return the reference batch, but batch normalized.

Last updated November 2, 2017.