Can not use both bias and batch normalization in convolution layers

Asked 2 years, 7 months ago Active 2 years, 7 months ago Viewed 9k times

16

3

I use slim framework for tensorflow, because of its simplicity. But I want to have convolutional layer with both biases and batch normalization. In vanilla tensorflow, I have:

```
def conv2d(input_, output_dim, k_h=5, k_w=5, d_h=2, d_w=2, name="conv2d"):
    with tf.variable_scope(name):
        w = tf.get_variable('w', [k_h, k_w, input_.get_shape()[-1], output_dim],

    initializer=tf.contrib.layers.xavier_initializer(uniform=False))
    conv = tf.nn.conv2d(input_, w, strides=[1, d_h, d_w, 1], padding='SAME')

    biases = tf.get_variable('biases', [output_dim],
initializer=tf.constant_initializer(0.0))
    conv = tf.reshape(tf.nn.bias_add(conv, biases), conv.get_shape())

    tf.summary.histogram("weights", w)
    tf.summary.histogram("biases", biases)

    return conv

d_bn1 = BatchNorm(name='d_bn1')
h1 = lrelu(d_bn1(conv2d(h0, df_dim + y_dim, name='d_h1_conv')))
```

and I rewrote it to slim by this:

```
h1 = slim.conv2d(h0,
                 num_outputs=self.df_dim + self.y_dim,
                 scope='d_h1_conv',
                 kernel_size=[5, 5],
                 stride=[2, 2],
                 activation_fn=lrelu,
                 normalizer_fn=layers.batch_norm,
                 normalizer_params=batch_norm_params,
                 weights_initializer=layers.xavier_initializer(uniform=False),
                 biases_initializer=tf.constant_initializer(0.0)
                 )
```

But this code does not add bias to conv layer. That is because of https://github.com/tensorflow/tensorflow/blob/master/tensorflow/contrib/layers/python/layers/layers.py#L1025 where is

```
    layer = layer_class(filters=num_outputs,
                        kernel_size=kernel_size,
                        strides=stride,
                        padding=padding,
                        data_format=df,
                        dilation_rate=rate,
                        activation=None,
                        use_bias=not normalizer_fn and biases_initializer,
                        kernel_initializer=weights_initializer,
```

```
                    activity_regularizer=None,
                    trainable=trainable,
                    name=sc.name,
                    dtype=inputs.dtype.base_dtype,
                    _scope=sc,
                    _reuse=reuse)
    outputs = layer.apply(inputs)
```

in the construction of layer, which results in not having bias when using batch normalization. Does that mean that I can not have both biases and batch normalization using slim and layers library? Or is there another way to achieve having both bias and batch normalization in layer when using slim?

python tensorflow

asked Sep 16 '17 at 17:43

Matěj Račinský
944 9 20

1 Answer

Active | Oldest | Votes

25

Batchnormalization already includes the addition of the bias term. Recap that BatchNorm is already:

```
gamma * normalized(x) + bias
```

So there is no need (and it makes no sense) to add another bias term in the convolution layer. Simply speaking BatchNorm shifts the activation by their mean values. Hence, any constant will be canceled out.

If you still want to do this, you need to remove the `normalizer_fn` argument and add BatchNorm as a single layer. Like I said, this makes no sense.

But the solution would be something like

```
net = slim.conv2d(net, normalizer_fn=None, ...)
net = tf.nn.batch_normalization(net)
```

Note, the BatchNorm relies on *non*-gradient updates. So you either need to use an optimizer which is compatible with the `UPDATE_OPS` collection. Or you need to manually add `tf.control_dependencies` .

Long story short: Even if you implement the ConvWithBias+BatchNorm, it will behave like ConvWithoutBias+BatchNorm. It is the same as multiple fully-connected layers without activation function will behave like a single one.

edited Sep 16 '17 at 18:01

answered Sep 16 '17 at 17:54

Thanks! I forgot that batch normalization already includes bias. I was confused, because in DCGAN, they use both trainable bias and batch normalization github.com/carpedm20/DCGAN-tensorflow/blob/master/model.py#L331 and I just tried to rewrite it into slim before I start tinkering with it. – Matěj Račinský Sep 16 '17 at 18:52