How to correctly use batch normalization with U-net in keras?

Asked 3 years ago Modified 5 months ago Viewed 4k times

11

1

I am trying to use batch normalization layers whith U-net for the segmentation task. Same layers works fine for res-net, vgg, xception etc., and I'm curious if it is an architecture dependent problem? During the training everything is fine, metrics grow losses dpor, but once I'm trying to evaluate the model or predict the mask it generates garbage. Seems like learned weights for those layers keep updating even during test and prediction. How to solve this problem in keras? keras version = 2.2.2

I was trying to use Batch norm layers only in encoder part, doesn't help. I was also trying to set layers parameter: trainable=False, doesn't help.

```
from keras.models import Input, Model
from keras.layers import Conv2D, Concatenate, MaxPooling2D
from keras.layers import UpSampling2D, Dropout, BatchNormalization

def conv_block(m, dim, res, do=0):
    n = Conv2D(dim, 3, padding='same')(m)
    n = BatchNormalization()(n)
    n = keras.layers.LeakyReLU(0)(n)
    n = Dropout(do)(n) if do else n
    n = Conv2D(dim, 3, padding='same')(n)
    n = BatchNormalization()(n)
    n = keras.layers.LeakyReLU(0)(n)
    return Concatenate()([m, n]) if res else n


def conv_block_bn(m, dim, res, do=0):
    n = Conv2D(dim, 3, padding='same')(m)
    n = BatchNormalization()(n)
    n = keras.layers.LeakyReLU(0)(n)
    n = Dropout(do)(n) if do else n
    n = Conv2D(dim, 3, padding='same')(n)
    n = BatchNormalization()(n)
    n = keras.layers.LeakyReLU(0)(n)
    return Concatenate()([m, n]) if res else n

def level_block(m, dim, depth, inc,  do,  mp, up, res):
    if depth > 0:
        n = conv_block_bn(m, dim, res)#(m, dim, acti, bn, res)
        m = MaxPooling2D()(n) if mp else Conv2D(dim, 3, strides=2,
padding='same')(n)
        m = level_block(m, int(inc*dim), depth-1, inc,  do,  mp, up, res)
        if up:
            m = UpSampling2D()(m)
            m = Conv2D(dim, 2, padding='same')(m)
            m = BatchNormalization()(m)
            m = keras.layers.LeakyReLU(0)(m)
        else:
            m = Conv2DTranspose(dim, 3, strides=2, activation='relu',
padding='same')(m)
        n = Concatenate()([n, m])
```

```
            m = conv_block_bn(n, dim, res)#(n, dim, acti, bn, res)
        else:
            m = conv_block_bn(m, dim, res,do)#(m, dim, acti, bn, res, do)
        return m

    def UNet(img_shape, out_ch=1, start_ch=64, depth=4, inc_rate=2.,
    activation='relu',
             dropout=0.5, batchnorm=False, maxpool=True, upconv=True,
    residual=False):
        i = Input(shape=img_shape)
        o = level_block(i, start_ch, depth, inc_rate,dropout, maxpool, upconv,
    residual)
        o = Conv2D(out_ch, 1, activation='sigmoid')(o)
        return Model(inputs=i, outputs=o)


    model1 = UNet((512,512,1), out_ch=1, start_ch=64, depth=4, inc_rate=2.,
             dropout=0.5, maxpool=True, upconv=True, residual=False)
    model1 = multi_gpu_model(model1,gpus=6)
    model1.compile(Adam(lr = 3.5e-6), loss = custom_losses, metrics = [dice_coef])
    ```
```

python keras deep-learning batch-normalization

Share Edit Follow

asked May 20, 2019 at 20:41

Primakov
**256** 2 9

Check this out for a nice Keras implementation that uses batch normalization – user14675723 Jul 8, 2021 at 4:50

## 2 Answers

Sorted by: Highest score (default)

0

try bach normalization on each layer and also eliminate dropout layer. See what you get. In prediction also set the flag of training=False.

Share Edit Follow

answered Aug 7, 2020 at 18:39

pedroska
**1** 1

0

When predicting outputs after training you must call your model with: Option1:

```
prediction = trained_model(input, training=False)
```

Option2:
```

```
prediction = trained_model.call(input, training=False)
```

Option3:

```
prediction = trained_model.predict(input)
```

The reason is that layers such as Normalization and dropout behave differently uring training. Hence, you must tell the model that you are running inference, not training.

Share Edit Follow

answered Jan 14 at 14:24

Sascha Kirch
366 2 2 17