CS5542 Big Data Apps and Analytics

In Class Programming –11 Report
(Jongkook Son)

Project Overview:

Make 4 changes(for example changing number of layers, changing hyper parameters, adding more cell blocks to bottle neck layer etc) to U-NET architecture given in the source code and train it on the same (or different) data set. Justify your changes by providing explanations in the report for each of the change you made. Visualize model performance and compare any difference you noticed in model performance.

Requirements/Task(s):

1) Designing a modified U-Net architecture and successfully running the code (70 points)

2) Explanation/ justification of 4 changes that you made and observations (5 points)

3) Visualization of model performance like one in source code (5 points)

4) overall code quality (10 points)

5) Pdf Report quality, video explanation (10 points)

What I learned in ICP:

Overall, I could have learned The basic structure of Unet in this ICP. Basically, Convolutional networks make the assumption of locality, and hence are more powerful From this aspect we use convolution neural network in Unet. In Unet, A class label is supposed to be assigned to each pixel. U-net learns segmentation in and end-to-end setting. And also Unet structure can be divided in to 3 structure. We can divide inti encoder and bottle neck and decoder. Unet is also very Flexible and can be used for any rational image masking task

ICP description what was the task you were performing and Screen shots that shows the successful execution of each required step of your code

Use a different data and use the model provided in icp 6 source code

Changing 4 hyper parameters

1. **Added more layer to encoder**

```
# added layer to encoder
c41 = tf.keras.layers.Conv2D(256, (3, 3), activation=tf.keras.activations.elu, kernel_initializer='he_normal',
                             padding='same')(p4)
c41 = tf.keras.layers.Dropout(0.2)(c41)
c41 = tf.keras.layers.Conv2D(256, (3, 3), activation=tf.keras.activations.elu, kernel_initializer='he_normal',
                             padding='same')(c41)
p41 = tf.keras.layers.MaxPooling2D(pool_size=(2, 2))(c41)
```

⇨ **adding more hidden layers, it will give more accuracy. This is true for larger datasets, as more layers with less stride factor will extract more features for your input data. For this reason, In this architecture, I have added first layer c41 with filter of size 256 and kernal of (3,3), elu activation function with dropout rate 20%.**

2. **More layers to the bottleneck**

```
#added layers to bottleneck
c5 = tf.keras.layers.Conv2D(512, (3, 3), activation=tf.keras.activations.elu, kernel_initializer='he_normal',
                            padding='same')(p41)
c5 = tf.keras.layers.Dropout(0.2)(c5)
c5 = tf.keras.layers.Conv2D(512, (3, 3), activation=tf.keras.activations.elu, kernel_initializer='he_normal',
                            padding='same')(c5)
c5 = tf.keras.layers.Dropout(0.2)(c5)
c5 = tf.keras.layers.Conv2D(512, (3, 3), activation=tf.keras.activations.elu, kernel_initializer='he_normal',
                            padding='same')(c5)
c5 = tf.keras.layers.Dropout(0.2)(c5)
c5 = tf.keras.layers.Conv2D(512, (3, 3), activation=tf.keras.activations.elu, kernel_initializer='he_normal',
                            padding='same')(c5)
```

⇨ **I added two more layers to the bottle neck for get uniform training and slow training for the whole training process. Basically bottleneck work as to help the model to learn a compression of the input data. The idea is that this compressed view should only contain the "useful" information to be able to reconstruct the input. So adding layer to bottle neck will help this function.**

3. Added more layer to decoder

```
# added layer to decoder
u60 = tf.keras.layers.Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same')(c5)
u60 = tf.keras.layers.concatenate([u60, c41])
c60 = tf.keras.layers.Conv2D(256, (3, 3), activation=tf.keras.activations.elu, kernel_initializer='he_normal',
                             padding='same')(u60)
c60 = tf.keras.layers.Dropout(0.2)(c60)
c60 = tf.keras.layers.Conv2D(128, (3, 3), activation=tf.keras.activations.elu, kernel_initializer='he_normal',
                             padding='same')(c60)
```

⇨ **adding more hidden layers, it will give more accuracy. This is true for larger datasets, as more layers with less stride factor will extract more features for your input data. And I already added a layer to encoder for this reason, And I should add a layer to decoder to encounter that. So Conv2D transpose with filter size 256 is added.**

4. Change the hyperparameters

```
checkpoint_path = "training_1/cp.ckpt"
checkpoint_dir = os.path.dirname(checkpoint_path)

# Create checkpoint callback
cp_callback = tf.keras.callbacks.ModelCheckpoint(checkpoint_path,
                                                 save_weights_only=True,
                                                 verbose=1)

callbacks = [
  tf.keras.callbacks.EarlyStopping(patience=2, monitor='val_loss', verbose=1),
  tf.keras.callbacks.TensorBoard(log_dir='./logs'),
  cp_callback
]

# change in validation_split
results = model.fit(x_train, y_train, validation_split=0.2, batch_size=16, epochs=50 ,callbacks=callbacks)
```

⇨ **Increase the epoch number to 50 incase for deeper training. Also changed the train test data split ratio to 0.2 to get more accurate result**

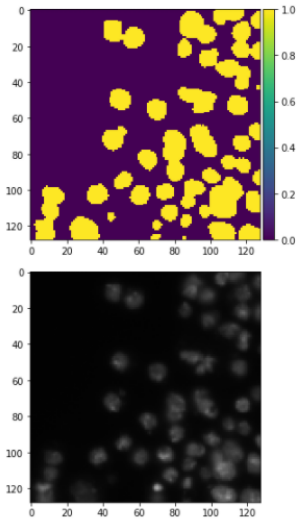
Result comparison

```
Epoch 1/20
 1/38 [..............................] - ETA: 0s - loss: 0.8638 - dice_acc: 0.2264WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/ten
Instructions for updating:
use `tf.profiler.experimental.stop` instead.
 2/38 [>.............................] - ETA: 2s - loss: 0.7633 - dice_acc: 0.2077WARNING:tensorflow:Callbacks method `on_train_batch_end` is slow c
38/38 [==============================] - ETA: 0s - loss: 0.3303 - dice_acc: 0.4482
Epoch 00001: saving model to training_1/cp.ckpt
38/38 [==============================] - 3s 70ms/step - loss: 0.3303 - dice_acc: 0.4482 - val_loss: 0.2034 - val_dice_acc: 0.6700
Epoch 2/20
37/38 [============================>.] - ETA: 0s - loss: 0.1600 - dice_acc: 0.7156
Epoch 00002: saving model to training_1/cp.ckpt
38/38 [==============================] - 2s 43ms/step - loss: 0.1604 - dice_acc: 0.7177 - val_loss: 0.1856 - val_dice_acc: 0.7207
Epoch 3/20
37/38 [============================>.] - ETA: 0s - loss: 0.1278 - dice_acc: 0.7758
Epoch 00003: saving model to training_1/cp.ckpt
38/38 [==============================] - 2s 43ms/step - loss: 0.1280 - dice_acc: 0.7751 - val_loss: 0.1725 - val_dice_acc: 0.7529
Epoch 4/20
37/38 [============================>.] - ETA: 0s - loss: 0.1102 - dice_acc: 0.8087
Epoch 00004: saving model to training_1/cp.ckpt
38/38 [==============================] - 2s 43ms/step - loss: 0.1094 - dice_acc: 0.8097 - val_loss: 0.2098 - val_dice_acc: 0.7368
Epoch 5/20
37/38 [============================>.] - ETA: 0s - loss: 0.1062 - dice_acc: 0.8140
Epoch 00005: saving model to training_1/cp.ckpt
38/38 [==============================] - 2s 44ms/step - loss: 0.1068 - dice_acc: 0.8137 - val_loss: 0.1252 - val_dice_acc: 0.8064
Epoch 6/20
37/38 [============================>.] - ETA: 0s - loss: 0.1012 - dice_acc: 0.8241
Epoch 00006: saving model to training_1/cp.ckpt
38/38 [==============================] - 2s 44ms/step - loss: 0.1016 - dice_acc: 0.8235 - val_loss: 0.1927 - val_dice_acc: 0.7646
Epoch 7/20
37/38 [============================>.] - ETA: 0s - loss: 0.097[illegible] - dice_acc: 0.8288
Epoch 00007: saving model to training_1/cp.ckpt
38/38 [==============================] - 2s 43ms/step - loss: 0.0999 - dice_acc: 0.8275 - val_loss: 0.1306 - val_dice_acc: 0.8077
Epoch 00007: early stopping
```

```
34/34 [==============================] - 4s 110ms/step - loss: 0.1142 - dice_acc: 0.3574 - val_loss: 0.0739 - val_dice_acc: 0.5778
Epoch 2/50
33/34 [============================>.] - ETA: 0s - loss: 0.0530 - dice_acc: 0.6681
Epoch 00002: saving model to training_1/cp.ckpt
34/34 [==============================] - 2s 70ms/step - loss: 0.0528 - dice_acc: 0.6714 - val_loss: 0.0480 - val_dice_acc: 0.7203
Epoch 3/50
33/34 [============================>.] - ETA: 0s - loss: 0.0393 - dice_acc: 0.7587
Epoch 00003: saving model to training_1/cp.ckpt
34/34 [==============================] - 2s 71ms/step - loss: 0.0390 - dice_acc: 0.7621 - val_loss: 0.0484 - val_dice_acc: 0.7588
Epoch 4/50
33/34 [============================>.] - ETA: 0s - loss: 0.0347 - dice_acc: 0.7929
Epoch 00004: saving model to training_1/cp.ckpt
34/34 [==============================] - 2s 71ms/step - loss: 0.0347 - dice_acc: 0.7949 - val_loss: 0.0409 - val_dice_acc: 0.7915
Epoch 5/50
33/34 [============================>.] - ETA: 0s - loss: 0.0312 - dice_acc: 0.8128
Epoch 00005: saving model to training_1/cp.ckpt
34/34 [==============================] - 2s 71ms/step - loss: 0.0312 - dice_acc: 0.8129 - val_loss: 0.0377 - val_dice_acc: 0.8082
Epoch 6/50
33/34 [============================>.] - ETA: 0s - loss: 0.0314 - dice_acc: 0.8180
Epoch 00006: saving model to training_1/cp.ckpt
34/34 [==============================] - 2s 70ms/step - loss: 0.0313 - dice_acc: 0.8187 - val_loss: 0.0362 - val_dice_acc: 0.8078
Epoch 7/50
33/34 [============================>.] - ETA: 0s - loss: 0.0293 - dice_acc: 0.8260
Epoch 00007: saving model to training_1/cp.ckpt
34/34 [==============================] - 3s 81ms/step - loss: 0.0291 - dice_acc: 0.8266 - val_loss: 0.0311 - val_dice_acc: 0.8363
Epoch 8/50
33/34 [============================>.] - ETA: 0s - loss: 0.0279 - dice_acc: 0.8365
Epoch 00008: saving model to training_1/cp.ckpt
34/34 [==============================] - 3s 76ms/step - loss: 0.0277 - dice_acc: 0.8351 - val_loss: 0.0293 - val_dice_acc: 0.8410
Epoch 9/50
33/34 [============================>.] - ETA: 0s - loss: 0.0256 - dice_acc: 0.8464
Epoch 00009: saving model to training_1/cp.ckpt
34/34 [==============================] - 2s 73ms/step - loss: 0.0257 - dice_acc: 0.8458 - val_loss: 0.0316 - val_dice_acc: 0.8318
Epoch 10/50
33/34 [============================>.] - ETA: 0s - loss: 0.[illegible] - dice_acc: 0.8404
Epoch 00010: saving model to training_1/cp.ckpt
34/34 [==============================] - 2s 73ms/ste[illegible] - loss: 0.0267 - dice_acc: 0.8399 - val_loss: 0.0295 - val_dice_acc: 0.8432
Epoch 00010: early stopping
```

⇨ **You can find out that the loss is much more better and the overall accuracy is also good.**

```
1/1 [==============================] - 0s 2ms/step
/usr/local/lib/python3.6/dist-packages/skimage/io/_plugins/matplotlib_plugin.py:150: UserWarning: Low image data range; displaying image with stretched contrast.
  lo, hi, cmap = _get_display_range(image)
```



Challenges that I faced:

The most difficult challenge that I faced was it was hard to understand the structure of Unet architecture. But thanks to the lecture and material I could finally grasp the big picture of the Unet. I still feel like need further training on this.

Video link

https://www.youtube.com/watch?v=OJaITIk1VBM