Jun (Junxian) Chen

Professor Chang

Final Summary

Methods

Model 1

The input size is (256, 256, 1) pixels with a weighted mask value of one on the lung and pneumonia. Every convolution has LeakyRelu activation and batch normalization. The inception network is used with same padding. The learning rate is $2 * 10^{-4}$ with Adam optimizers with a 20% reduction rate if every two epoch validation accuracy did not increase. The training batch size is 12, 75 steps per epoch for 75 epochs. The amount of total parameter is 853,312. The loss function and metric are SparseCategoricalCrossentropy. The test evaluation is based on classification.

```
a_1 = 1

c1_1 = inception(int(8*a_1), inputs['dat'])
c2_1 = inception(int(16*a_1), inception2(int(16*a_1),c1_1))
c2_1_1 = inception(int(16*a_1), c2_1)
c3_1 = inception(int(32*a_1), inception2(int(32*a_1),c2_1_1))
c3_1_1 = inception(int(32*a_1), c3_1)
c4_1 = inception(int(48*a_1), inception2(int(48*a_1),c3_1_1))
c4_1_1 = inception(int(48*a_1), c4_1)
c5_1 = inception(int(64*a_1), inception(int(64*a_1),c4_1_1))
c5_1_1 = inception(int(64*a_1), c5_1)
c6_1 = inception(int(64*a_1), inception(int(64*a_1),c5_1_1))
c6_1_1 = inception(int(64*a_1), c6_1)
c7_1 = inception(int(32*a_1), inception(int(32*a_1),c6_1_1)) #bottleneck
c8_1_1 = inception(int(16*a_1), c7_1)
c9_1_1 = inception(int(32*a_1), c8_1_1)
c10_1_1 = inception(int(48*a_1), c9_1_1)
c11_1_1 = inception(int(48*a_1), inception(int(48*a_1),c10_1_1))
c12_1_1 = inception(int(48*a_1), c11_1_1)

r1_1 = layers.Reshape((-1,1,1,32*16*int(48*a_1)))(c12_1_1)
r2_1 = leakyrelu(norm(layers.Conv3D(int(24*a_1), kernel_size = (1,1,1))(r1_1)))
r3_1 = leakyrelu(norm(layers.Conv3D(int(8*a_1), kernel_size = (1,1,1))(r2_1)))

logits_1 = {}

logits_1['pna-cls'] = layers.Conv3D(2, name = 'pna-cls', kernel_size = (1,1,1))(r3_1)

model1 = Model(inputs = inputs, outputs = logits_1)

model1.summary()
```

Model 2

The second model is based on a U-Net architecture. The input size is (256, 256, 1) with a weighted mask value of one on the lung and pneumonia. Every convolution has LeakyRelu activation and batch normalization. The inception network is used with same padding. The learning rate is $2 * 10^{-4}$ with Adam optimizers with a 20% reduction rate if every two epochs val dice score did not increase. The training batch size is 12, 75 steps per epoch for 75 epochs, validation step is 75, and validation frequency per epoch. The total amount of parameter is 16,276,346. The loss function is custom.sce and metric is custom.dsc. The test evaluation is based on the classification and the segmentation threshold of 90.

```
a_2 = 5

c1_2 = inception(int(8*a_2), inputs['dat'])
c2_2 = inception(int(16*a_2), conv2_2(int(16*a_2),c1_2))
c3_2 = inception(int(16*a_2), c2_2)
c4_2 = inception(int(32*a_2), conv2_2(int(32*a_2),c3_2))
c5_2 = inception(int(32*a_2), c4_2)
c6_2 = inception(int(64*a_2), conv2_2(int(64*a_2),c5_2))
c7_2 = inception(int(64*a_2), c6_2)
c8_2 = inception(int(64*a_2), conv2_2(int(64*a_2),c7_2))
c9_2 = inception(int(64*a_2), c8_2)
c10_2 = inception(int(64*a_2), conv2_2(int(64*a_2),c9_2))
c11_2 = inception(int(64*a_2), c10_2)
c12_2 = inception(int(64*a_2), conv2_2(int(64*a_2),c11_2))
c13_2 = inception(int(64*a_2), c12_2)

e1_2 = tran_2(int(64*a_2), c13_2)
e2_2 = inception(int(64*a_2), e1_2)
e3_2 = tran_2(int(64*a_2), e2_2 + inception(int(64*a_2), c11_2))
e4_2 = inception(int(64*a_2), e3_2)
e5_2 = tran_2(int(64*a_2), e4_2 + inception(int(64*a_2), c9_2))
e6_2 = inception(int(64*a_2), e5_2)
e7_2 = tran_2(int(64*a_2), e6_2 + inception(int(64*a_2), c7_2))
e8_2 = inception(int(32*a_2), e7_2)
e9_2 = tran_2(int(32*a_2), e8_2 + inception(int(32*a_2), c5_2))
e10_2 = inception(int(16*a_2), e9_2)
e11_2 = tran_2(int(16*a_2), e10_2 + inception(int(16*a_2), c3_2))
e12_2 = inception(int(8*a_2), e11_2)

logits_2 = {}

logits_2['pna-seg'] = layers.Conv3D(filters = 2, name = 'pna-seg', **kwargs)(e12_2 + c1_2)


model2 = Model(inputs = inputs, outputs = logits_2)

model2.summary()
```

Model 3

Model 3 is a combination of Model 1 and Model 2. The input size is (256, 256, 1) pixels with a weighted mask value of one on the lung and pneumonia. Every convolution has LeakyRelu activation and batch normalization. The inception network is used with same padding. The learning rate is $2 * 10^{-4}$ with Adam optimizers with a 20% reduction rate if every two epochs val dice score did not increase. The training batch size is 12, 100 steps per epoch for 75 epochs. The total amount of parameter is 384,000. The loss functions are SparseCategoricalCrossentropy and custom.sce. The metrics are SparseCategoricalCrossentropy and custom.dsc. The test evaluation is based on the classification and the segmentation threshold of 90.

```
a_3 = 1

c1_3 = inception(int(8*a_3), inputs['dat'])
c2_3 = inception(int(16*a_3), inception2(int(16*a_3),c1_3))
c2_3_3 = inception(int(16*a_3), c2_3)
c3_3 = inception(int(32*a_3), inception2(int(32*a_3),c2_3_3))
c3_3_3 = inception(int(32*a_3), c3_3)
c4_3 = inception(int(48*a_3), inception2(int(48*a_3),c3_3_3))
c4_3_3 = inception(int(48*a_3), c4_3)
c5_3 = inception(int(64*a_3), inception2(int(64*a_3),c4_3_3))
c5_3_3 = inception(int(64*a_3), c5_3)
c6_3 = inception(int(64*a_3), inception2(int(64*a_3),c5_3_3))
c6_3_3 = inception(int(64*a_3), c6_3)
c7_3 = inception(int(64*a_3), inception2(int(64*a_3),c6_3_3))

c8_3_3 = inception(int(32*a_3), c7_d_3)
c9_3_3 = inception(int(16*a_3), c8_3_3)
c10_3_3 = inception(int(32*a_3), c9_3_3)
c11_3_3 = inception(int(64*a_3), c10_3_3)

r1_3 = layers.Reshape((-1,1,1,2*4*int(16*a_3)))(c9_3_3)
r2_3 = layers.Conv3D(int(24*a_3), kernel_size = (1,1,1))(r1_3)
r3_3 = layers.Conv3D(int(8*a_3), kernel_size = (1,1,1))(r2_3)

e3_3 = tran_3(int(64*a_3), c6_3_3)
e4_3 = tran_3(int(48*a_3), inception(int(64*a_3), e3_3 + c5_3_3))
e5_3 = tran_3(int(32*a_3), inception(int(48*a_3), e4_3 + c4_3_3))
e6_3 = tran_3(int(16*a_3), inception(int(32*a_3), e5_3 + c3_3_3))
e7_3 = tran_3(int(8*a_3), inception(int(16*a_3), e6_3 + c2_3))
e8_3 = inception(int(8*a_3), e7_3 + c1_3)

logits_3 = {}

logits_3['pna-seg'] = layers.Conv3D(filters = 2, name = 'pna-seg', **kwargs)(e8_3)
logits_3['pna-cls'] = layers.Conv3D(2, name = 'pna-cls', kernel_size = (1,1,1))(r3_3)

model3 = Model(inputs = inputs, outputs = logits_3)
```

Results

	ACC	SEN	SPE	PPV	NPV
Model 1	88.30%	87.38%	88.98%	85.58%	90.40%
Model 2	92.80%	92.05%	93.35%	91.20%	94.01%
Model 3	89.50%	89.01%	89.86%	86.78%	91.62%

The results are based on 1000 samples on the test data set. Model 1 accuracy, sensitivity, specificity, positive predictive value, and negative predicted value is 88.30%, 87.38%, 89.98%, 85.58%, 90.40% respectively. Model 2 accuracy, sensitivity, specificity, positive predictive value, and negative predicted value is 92.80%, 92.02%, 93.35%, 91.20%, 94.01% respectively. Model 3 accuracy, sensitivity, specificity, positive predictive value, and negative predicted value is 89.50%, 89.01%, 89.86%, 86.89%, 86.78% respectively.

Discussion

Based on the results, Model 2 (segmentation) produced the highest accuracy. Model 1 (classification) and 3 (classification and segmentation) are relatively smaller compared to Model 2. After doing different testing on model 1 and model 3, the accuracy is below 90%. Increasing the amount of parameters in model 1 and 3 did not increase the accuracy. In addition, Google Colab has a tendency to crash after running over five hours; therefore, increasing the parameter is a last result. However, increasing the parameters of Model 2 produced a noticeable increase in accuracy. Model 2 accuracy saturated around 16 million parameters at 92.80%.