CSC413 Assignment 2

Chen Liang

February, 2020

1. **Part 1.**

   (b) The images generated don't look good to me. Reason: For training data, some of the images are not clear enough and some of them are not correctly coloured. The images in the testing data don't have clear boundaries and they are not blurred. Therefore the images generated don't look good to me.

   (c) **Question:** Compute the number of weights, outputs, and connections in the model, as a function of NF and NC. Compute these values when each input dimension (width/height) is doubled. Report all 6 values.
   Solution: Since ReLu doesn't change the size of inputs, so these two layers do not contribute anything to weights, outputs and connections, so we don't need to consider these two layers.

      i. **DownConv 1:**

         A. **MyConv2D-NF:** Because the kernel size is 3×3, MyConv2D-NF layer has NF output channels, 3 inputs RGB channel, so the number of total weights would be $3 \times 3 \times NF$, and the total number of bias would be NF. The total number of connections is $32 \times 32 \times 3 \times 3 \times 3 \times NF$ The output would have size $32 \times 32 \times NF$.
         weights: $3 \times 3 \times NF = 9NF$,
         connections: $32 \times 32 \times 3 \times 3 \times NF = 9216NF$,
         outputs: $32 \times 32 \times NF = 1024NF$

         B. **MaxPool:** After maxpooling the input, the output would have size $16 \times 16 \times NF$
         outputs: $16 \times 16 \times NF = 256NF$.

         C. **BatchNorm:** For each of beta, gamma, sigma, and mu there exists a weight.
         weight: 4NF

      ii. **DownConv 2:**

         A. **MyConv2D-2NF:**Because the kernel size is 3×3, MyConv2D-2NF layer has 2NF output channels, input has NF channels, so the number of total weights would be $3 \times 3 \times NF \times 2NF$, and the total number of bias would be $2NF$. The total number of connections is $16 \times 16 \times 3 \times 3 \times NF \times 2NF$ The output would have size $16 \times 16 \times 2NF$.
         weights: $3 \times 3 \times NF \times 2NF = 18NF^2$,
         connections: $16 \times 16 \times 3 \times 3 \times NF \times 2NF = 4608NF^2$,
         outputs: $16 \times 16 \times 2NF = 512NF$

         B. **MaxPool:**After maxpooling the input, the output would have size $8 \times 8 \times 2NF$
         outputs: $8 \times 8 \times 2NF = 128NF$.

         C. **BatchNorm:** For each of beta, gamma, sigma, and mu there exists a weight.
         weight: 8NF

      iii. **RFConv:**

         A. **MyConv2D-2NF:**Because the kernel size is 3×3, MyConv2D-2NF layer has 2NF output channels, input has 2NF channels, so the number of total weights would be $3 \times 3 \times 2NF \times 2NF$, and the total number of bias would be $2NF$. The total number of connections is

$8 \times 8 \times 3 \times 3 \times 2NF \times 2NF$ The output would have size $8 \times 8 \times 2NF$.
weights: $3 \times 3 \times 2NF \times 2NF = 36NF^2$,
connections: $8 \times 8 \times 3 \times 3 \times 2NF \times 2NF = 2304NF^2$,
outputs: $8 \times 8 \times 2NF = 128NF$

B. **BatchNorm:** For each of beta, gamma, sigma, and mu there exists a weight.
weight: 8NF

iv. **UpConv1:**

A. **MyConv2D-NF:** Because the kernel size is 3×3, MyConv2D-2NF layer has NF output channels, input has 2NF channels, so the number of total weights would be $3 \times 3 \times 2NF \times NF$, and the total number of bias would be NF. The total number of connections is $8 \times 8 \times 3 \times 3 \times NF \times 2NF$ The output would have size $8 \times 8 \times NF$.
weights: $3 \times 3 \times 2NF \times NF = 18NF^2$,
connections: $8 \times 8 \times 3 \times 3 \times 2NF \times NF = 1152NF^2$,
outputs: $8 \times 8 \times NF = 64NF$

B. **Upsample:** After upsampling the input, the output would have size $16 \times 16 \times NF$
outputs: $16 \times 16 \times NF = 256NF$.

C. **BatchNorm:** For each of beta, gamma, sigma, and mu there exists a weight.
weight: 4NF

v. **UpConv2:**

A. **MyConv2D-NC:** Because the kernel size is 3×3, MyConv2D-2NF layer has NC output channels, input has NF channels, so the number of total weights would be $3 \times 3 \times NF \times NC$, and the total number of bias would be NC. The total number of connections is $16 \times 16 \times 3 \times 3 \times NF \times NC$ The output would have size $16 \times 16 \times NC$.
weights: $3 \times 3 \times NF \times NC = 9NF \cdot NC$,
connections: $16 \times 16 \times 3 \times 3 \times NF \times NC = 2304NF \cdot NC$,
outputs: $16 \times 16 \times NC = 256NC$

B. **Upsample:** After upsampling the input, the output would have size $32 \times 32 \times NC$
outputs: $32 \times 32 \times NC = 1204NC$.

C. **BatchNorm:** For each of beta, gamma, sigma, and mu there exists a weight.
weight: 4NC

vi. **FinalConv:**

A. **MyConv2D-NC:** Because the kernel size is 3×3, MyConv2D-2NF layer has NC output channels, input has NC channels, so the number of total weights would be $3 \times 3 \times NC \times NC$, and the total number of bias would be NC. The total number of connections is $32 \times 32 \times 3 \times 3 \times NC \times NC$ The output would have size $32 \times 32 \times NC$.
weights: $3 \times 3 \times NC \times NC = 9NC^2$,
connections: $32 \times 32 \times 3 \times 3 \times NC \times NC = 9216NC^2$,
outputs: $32 \times 32 \times NC = 1204NC$

total weights $= 33NF + 18NF^2 + 36NF^2 + 18NF^2 + 9NF \cdot NC + 9NC^2 + 4NC = 33NF + 72NF^2 + 9NF \cdot NC + 9NC^2 + 4NC$
total connections $= 9216NF + 4608NF^2 + 2304NF^2 + 1152NF^2 + 2304NF \cdot NC + 9216NC^2 = 9216NF + 8064NF^2 + 2304NF \cdot NC + 9216NC^2$
total outputs $= 1024NF + 256NF + 512NF + 128NF + 128NF + 64NF + 256NF + 256NC + 1204NC + 32 * 32NC = 2368NF + 2304NC$

If height and width are doubled, the total weights would remain unchanged, while the total connections and total outputs increase to four times of its original size. In this way,

total weights' = **total weights**
total connections' = $4\times$ **total connections**
total outputs' = $4\times$ **total outputs**

(d) **Question:** Consider an pre-processing step where each input pixel is multiplied elementwise by scalara,and is shifted by some scalarb. That is, where the original pixel value is denotedx, the newvalue is calculatedy=ax+b. Assume this operation does not result in any overflows. How does this pre-processing step affect the output of the conv net from Question 1 and 2?
Answer: The pre-processing step will not affect the ouput of the conv net from Question 1. Reason: 1)The relative distances among pixels do not change. 2) After batch normalization, the normalized results would be the same. 3) As we use max-pooling, the prediction should not change.

2. **Part 2**

(c) **Question:** How does the result compare to the previous model? Did skip connections improve the validation loss and accuracy? Did the skip connections improve the output qualitatively? How? Give at least two reasons why skip connections might improve the performance of our CNN models.
Answer: UNet has a better performance than CNN. Skip connections improve the validation loss and accuracy, and improve the output qualitatively.
Reason One: Passing gradient to deeper layers could raise the problem of vanishing gradients. Skipping connections would make gradients to flow through the network without passing through the nonlinear activation functions, so in this way we could effectively prevent the gradient vanishing problem.
Reason Two: Information could be lost during the max-pooling process, and by combing the deep feature maps with shallow, low-level feature maps could prevent the information loss caused by max-pooling, leading to a more accurate result compared to CNN results.

(d) **Question:** Re-train a few more "UNet" models using different mini batch sizes with a fixed number of epochs. Describe the effect of batch sizes on the training/validation loss, and the final image output.
Answer: Fix epoch number as 25. In this case, change the mini batch size from 5 to 500 and the result is listed as below:
Mini-Bacth Size: 5, Val Loss: 1.2344, Val Acc: 52.7%, Time(s): 176.75,
Mini-Batch Size: 10, Val Loss: 1.2575, Val Acc: 52.4%, Time(s): 94.68
Mini-Batch Size: 20, Val Loss: 1.2859, Val Acc: 51.2%, Time(s): 54.65
Mini-Batch Size: 50, Val Loss: 1.3803, Val Acc: 48.4%, Time(s): 36.50
Mini-Batch Size: 100, Val Loss: 1.3236, Val Acc: 49.5%, Time(s): 36.29
Mini-Batch Size: 200, Val Loss: 1.4076, Val Acc: 47.2%, Time(s): 33.29
Mini-Batch Size: 500, Val Loss: 1.5182, Val Acc: 43.2%, Time(s): 31.91

If the batch size becomes smaller, the loss on both testing set and validation set becomes smaller and the accuracy rate becomes larger. But noticeably, when the batch size is way too smaller, such as the batch size is equal to 5, the overfitting occurs and the loss curve of the tesing set becomes slightly unstable, and some colours of the images are wrongly assigned.
When the batch size increases and becomes relatively large such as 100, or 200, the accuracy for both the test and validation set decreases as the batch size increases. When the batch size becomes too large, the situation of under-fitting occurs and the boundaries of some images are blurred as well as the color of some pixels in images are wrongly allocated.

3. **Part 3**

(d) **Question:** Consider a case of fine-tuning a pre-trained model with n number of layers. Each of the layers have a similar number of parameters, so the total number of parameters for the model is proportional to n. Describe the difference in memory complexity in terms of n between fine-tuning an entire pre-trained model versus fine-tuning only the last layer (freezing all the other layers).

What about the computational complexity?
Answer: The memory complexity for fine-tuning an entire pre-trained model is $O(n)$ because there are n layers of weight and bias to be stored.
The memory complexity for fine-tuning the last layer is $O(1)$ because we only need to store the weight and bias of the last layer.
The computational complexity for fine-tuning an entire model is also $O(n)$. Freezing all layers except for the last one will not decrease the complexity for the forward process, so the computational complexity is also $O(n)$.

(e) **Question:** If we increase the height and the width of the input image by a factor of 2, how does this affect the memory complexity of fine-tuning? What about the number of parameters?
Answer: If the height and weight both increases by 2 the total number of pixels becomes 4 times of the original size, but the complexity remains to be $O(1)$. The total number of parameters remains to be the same as conv layer's weight is not affected by the image size.