Part A

Question 2

The results do not look good to me because the pictures look like a grayscale image with a bit of added colors. Also, the accuracy is very low, about 30%.

Question 3

Let K be the kernel size and W, H be the width and height of the input image

Weights:

$$= (NF \cdot K^2) + (NF) + (2NF \cdot NF \cdot K^2) + (2NF) + (2NF \cdot 2NF \cdot K^2) +$$
$$(2NF) + (NF \cdot 2NF \cdot K^2) + (NF) + (NC \cdot NF \cdot K^2) + (NC) + (NC \cdot NC \cdot K^2)$$

Outputs:

$$= (W \cdot H \cdot NF) + \left(\frac{W}{2} \cdot \frac{H}{2} \cdot NF\right) + \left(\frac{W}{2} \cdot \frac{H}{2} \cdot 2NF\right) + \left(\frac{W}{4} \cdot \frac{H}{4} \cdot 2NF\right) +$$
$$\left(\frac{W}{4} \cdot \frac{H}{4} \cdot NF\right) + \left(\frac{W}{2} \cdot \frac{H}{2} \cdot NC\right) + \left(\frac{W}{2} \cdot \frac{H}{2} \cdot NC\right) + (W \cdot H \cdot NC) + (W \cdot H \cdot NC)$$

Connections:

$$= (W \cdot H \cdot K^2 \cdot NF) + (W \cdot H \cdot NF) + \left(\frac{W}{2} \cdot \frac{H}{2} \cdot K^2 \cdot 2NF \cdot NF\right) + \left(\frac{W}{2} \cdot \frac{H}{2} \cdot 2NF\right) +$$
$$\left(\frac{W}{4} \cdot \frac{H}{4} \cdot K^2 \cdot 2NF \cdot 2NF\right) + \left(\frac{W}{4} \cdot \frac{H}{4} \cdot 2NF\right) + \left(\frac{W}{4} \cdot \frac{H}{4} \cdot K^2 \cdot 2NF \cdot NF\right) + \left(\frac{W}{4} \cdot \frac{H}{4} \cdot NF\right) +$$
$$\left(\frac{W}{2} \cdot \frac{H}{2} \cdot K^2 \cdot NF \cdot NC\right) + \left(\frac{W}{2} \cdot \frac{H}{2} \cdot NC\right) + (W \cdot H \cdot K^2 \cdot NC \cdot NC) + (W \cdot H \cdot NC)$$

When each input dimension is doubled, the values are as follow:

Weights:

$$= (NF \cdot K^2) + (NF) + (2NF \cdot NF \cdot K^2) + (2NF) + (2NF \cdot 2NF \cdot K^2) +$$
$$(2NF) + (NF \cdot 2NF \cdot K^2) + (NF) + (NC \cdot NF \cdot K^2) + (NC) + (NC \cdot NC \cdot K^2)$$

Outputs:

$$= (2W \cdot 2H \cdot NF) + (W \cdot H \cdot NF) + (W \cdot H \cdot 2NF) + \left(\frac{W}{2} \cdot \frac{H}{2} \cdot 2NF\right) + \left(\frac{W}{2} \cdot \frac{H}{2} \cdot 2NF\right) +$$
$$\left(\frac{W}{2} \cdot \frac{H}{2} \cdot NF\right) + (W \cdot H \cdot NC) + (W \cdot H \cdot NC) + (2W \cdot 2H \cdot NC) + (2W \cdot 2H \cdot NC)$$

Connections:

$$= (2W \cdot 2H \cdot K^2 \cdot NF) + (2W \cdot 2H \cdot NF) + (W \cdot H \cdot K^2 \cdot 2NF \cdot NF) + (W \cdot H \cdot 2NF) +$$
$$\left(\frac{W}{2} \cdot \frac{H}{2} \cdot K^2 \cdot 2NF \cdot 2NF\right) + \left(\frac{W}{2} \cdot \frac{H}{2} \cdot 2NF\right) + \left(\frac{W}{2} \cdot \frac{H}{2} \cdot K^2 \cdot 2NF \cdot NF\right) + \left(\frac{W}{2} \cdot \frac{H}{2} \cdot NF\right) +$$
$$(W \cdot H \cdot K^2 \cdot NF \cdot NC) + (W \cdot H \cdot NC) + (2W \cdot 2H \cdot K^2 \cdot NC \cdot NC) + (2W \cdot 2H \cdot NC)$$

Question 4

Given the pre-processing step, the scalar a will change the standard deviation and the scalar b will change the mean of the input data. Therefore, the BatchNorm layers will output different values, where the values are relatively closer to each other. Thus, the final output of the model would have big patches of similar colors since the values are more similar to each other and the model is less likely to classify these similar pixels.

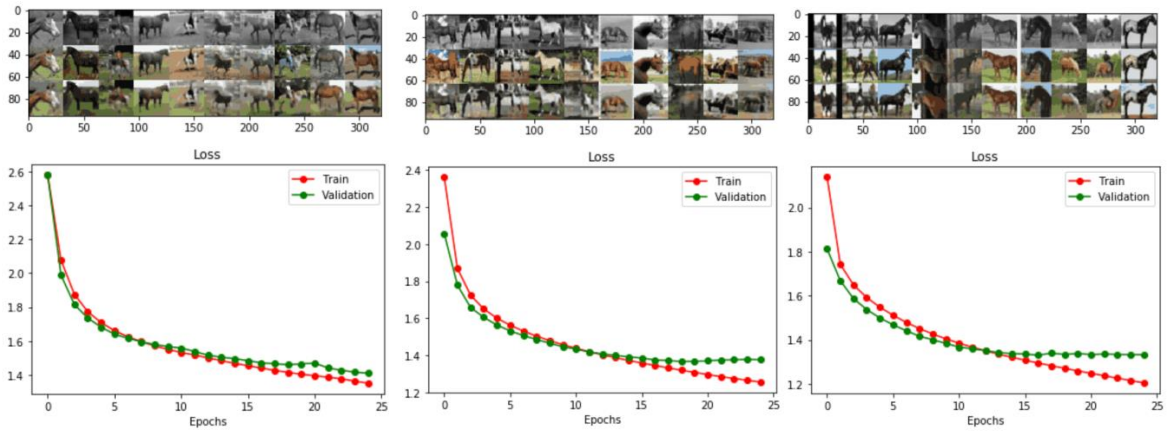
Part B

Question 3

The results are much better compare to the previous model, the skip connections improved the validation loss and accuracy and the overall output qualitatively, i.e. more colors can be seen from the output image. The first reason to why this might improve the performance is because concatenating the layers expands the input and output dimension for the 4th to 6th convolutional layers. Thus, increasing the number of trainable parameters and making our model more expressive. Secondly, since information is lost during max pooling and up sampling, adding the input and following layers will be able to add back the information lost during the process. This can also be described as combining the learning after information is lost with the reference input that conserved all the information. Thus, it makes our model able to combine the learning from different scale and achieve a better result.

Question 4

I ran the model in batch sizes of 50, 100, 200, here are the results I got:



Left: 200 Middle: 100 Right: 50

We can see the loss for the model with batch size of 200 has not reach a convergence yet since it is still constantly decreasing, and the difference between the training and validation loss is smaller than the other two. For the model with batch size of 100, we can see that the validation loss has reached a convergence since it is relatively constant after epoch 20. On the other hand, the model with batch size of 50 converged much fast, the validation is relatively constant after epoch 14. In addition, the different between the training and validation loss for the model with 50 batch size is also larger. Therefore, we can say that smaller batch size would reach convergence faster for the validation loss but not the training loss since it is very similar across all 3 models.

For the output, although not highly distinguishable, the model with higher batch size is slightly better which agrees with the lowest validation loss across the 3 models. However, it is not actually very visually differentiable.

Part C

Question 4

In terms of memory complexity, the entire pre-trained model will need $O(n)$ space because it needs to store all the parameters in order to update with gradient. On the other hand, fine-tuning only the last layer will require $O(1)$ space because only the last layer's weights are needed to change which requires $O(1)$ memory. This is also proved by question 3 where we need to disable grad for all layers except the last layer in order to avoid memory problem.

In terms of computational complexity, the entire pre-trained model will need $O(n)$ time because it needs to modify all the parameters and it is proportional to n. On the other hand, fine-tuning only the last layer will only require $O(1)$ time because we only need to modify the last layer's parameters which is constant in terms of n.

Question 5

If we increase the height and the width of the input image by a factor of 2, assuming that fully connected layers does not get affected, the memory complexity would not change because the memory complexity is equal to the number of parameters since we need to store all of it for gradient update. From part A question 3, we know that the dimension does not affect the number of weights or parameters.