Question 1 Colourization as Classification.

(a) Complete the model CNN.

(b) *Run main training loop of CNN. This will train the CNN for a few epochs using the cross-entropy objective. It will generate some images showing the trained result at the end. Do these results look good to you? Why or why not?*

I don't think the results look good. For the training dataset, although the boundary of objects is basically detected, the objects are still not coloured as the true labels. For the test dataset, the boundary becomes even blurred.

(c) *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.*

Layers	Units	Weights	Connections
Image	32*32*1	-	-
MyConv2D-NF	32*32*NF	3*3*1*NF+NF=10NF	9NF*32*32=9216NF
MaxPool	16*16*NF	-	-
BatchNorm-NF	16*16*NF	16*16*NF*4=1024NF	-
MyConv2D-2NF	16*16*2NF	3*3*NF*2NF+2NF=2NF+18NF^2	18NF^2*16*16=4608NF^2
MaxPool	8*8*2NF	-	-
BatchNorm-NF	8*8*2NF	8*8*2NF*4=512NF	-
MyConv2D-2NF	8*8*2NF	3*3*2NF*2NF+2NF=2NF+36NF^2	36NF^2*8*8=2304NF^2
BatchNorm-NF	8*8*2NF	8*8*2NF*4=512NF	-
MyConv2D-NF	8*8*NF	3*3*2NF*NF+NF=NF+18NF^2	18NF^2*8*8=1152NF^2
UpSample	16*16*NF	-	-
BatchNorm-NF	16*16*NF	16*16*NF*4=1024NF	-
MyConv2D-NC	16*16*NC	3*3*NF*NC+NC=NC+9NF*NC	9NF*NC*16*16=2304NF*NC
UpSample	32*32*NC	-	-
BatchNorm-NF	32*32*NC	32*32*NC*4=4096NC	-
MyConv2D-NC	32*32*NC	3*3*NC*NC+NC=NC+9NC^2	9NC^2*32*32=9216NC^2
Total	-	3086NF+72NF^2+9NF·NC+9NC^2+4098NC	9216NF+8064NF^2+2304NF·NC+9216NC^2

We assume the convolutional kernels are all 3 × 3.
The number of weights, outputs, and connections in the model are (3086NF + 72NF^2 + 9NF·NC + 9NC^2 + 4098NC), 1024NC, and (9216NF + 8064NF^2 + 2304NF·NC + 9216NC^2) respectively.

when each input dimension (width/height) is doubled, these values are weights = (12302NF + 72NF^2 + 9NF·NC + 9NC^2 + 16386NC), outputs = (4096NC), and connections=(36864NF + 32256NF^2 + 9216NF·NC + 36864NC^2).

(d) *Consider an pre-processing step where each input pixel is multiplied element-wise by scalar a, and is shifted by some scalar b. That is, where the original pixel value is denoted x, the new value is calculated y = 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?*

The network has no need to change, i.e., it won't influence the results of Question 2.
Multiplying the input pixels element-wise by scalar a and then shifting them by scalar b will increase the variance of the input data of both training and test set. However, the Batch Normalization layer should fix the difference.

$$\hat{x} = \frac{x - E[x]}{\sqrt{Var[x]}}$$

$$\hat{x^*} = \frac{x^* - E[x^*]}{\sqrt{Var[x^*]}} = \frac{ax + b - (aE[x] + b)}{\sqrt{Var(ax+b)}} = \frac{a(x - aE[x])}{a\sqrt{Var(x)}} = \frac{x - E[x]}{\sqrt{Var[x]}}$$

Question 1 [(d)] continued on next page. . .

I did experiments on the image dataset, i.e., I multiplied the training data and validation data by 10 and added them by 10. The loss curves of training dataset and validation set are almost the same.

Question 2 Skip Connections.

(a) Add a skip connection from the first layer to the last, second layer to the second last, etc.

(b) *Train the model for at least 25 epochs and plot the training curve using a batch size of 100.*

The figure of the training curve of U-Net is shown in Figure 1.

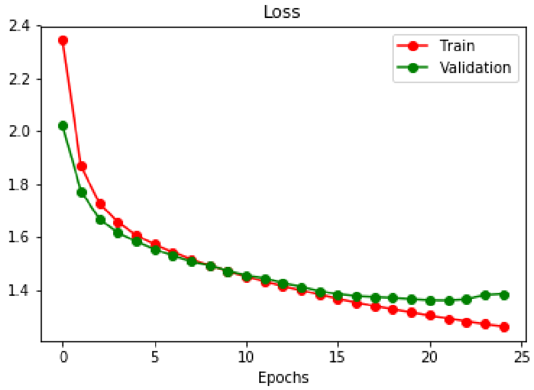


Figure 1: Training curve of U-Net.

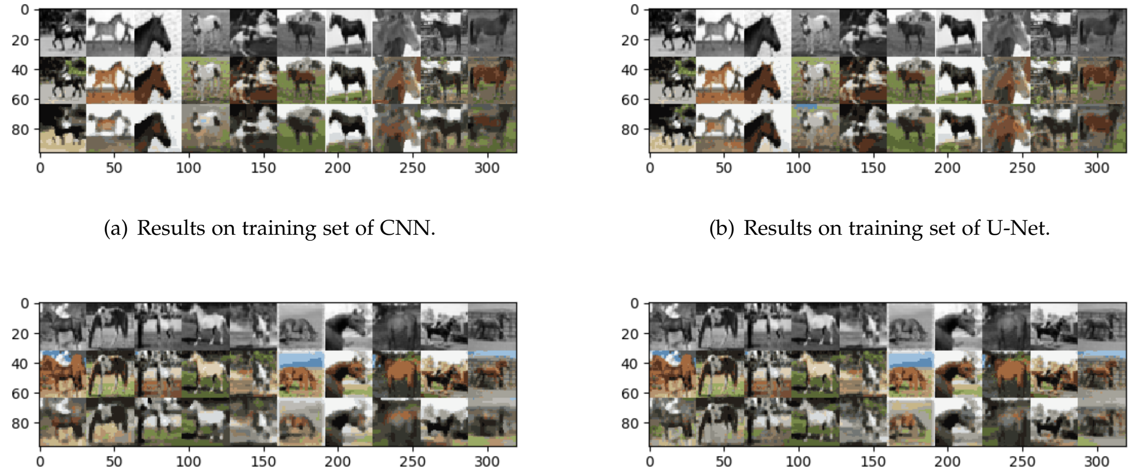
(c) *How does the result compare to the previous model? Did skip connections improve the validation loss and accuracy? Did the skip connections improve the output qualitatively? How? Give at least two reasons why skip connections might improve the performance of our CNN models.*

The U-Net outperforms the previous network.
The skip connections improve the validation loss and accuracy. I trained the both networks with exactly same parameters, and the validation loss and accuracy of the previous network are 1.598 and 40.9% respectively, while for U-Net they are 1.383 and 48.5%.
The skip connections also improve the output qualitatively (as Figure 2). The contour of these horses are clearer, and some of the original colors that were not right are corrected by U-Net.
Two reasons why skip connections might improve the performance of our CNN models: 1) The original CNN reduces the picture size (or no longer corresponds to the input picture size) due to pooling when getting deeper. Compared with the original CNN, the structure of U-Net makes it more accurate for the positioning of each pixel of the picture, and it outperforms CNN to give each pixel its corresponding color value. 2) The deeper layer of U-Net can fully combine the simple representations of the shallow layer, which outperforms on small training set, by avoid underfitting.

(d) *Re-train a few more "U-Net" 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.*

I tested with batch size = 10, 50, 100, 200.
The smaller the batch size, the longer the run time.
In general, the smaller the batch size, the smaller the loss of the training set, as **smaller batch size will compute the gradient at a fresher value of** θ (see Slides Lec04).

Question 2 [(d)] continued on next page. . .



(a) Results on training set of CNN.

(b) Results on training set of U-Net.

(c) Results on test set of CNN.

(d) Results on test set of U-Net.

Figure 2: Results of CNN and U-Net networks.

The loss of the test set should also conform to this trend as training set; however, when the batch size becomes small enough (=10), overfitting occurs, as the loss of the test set is not stable.
For the final image outputs of different batch size on the test set, when batch size is large, some underfitting occurred, e.g., the contour of the object is not found clearly; however, there are also some deviations when the batch size is too small, such as separating objects (sky, grassland) that should be a whole.

Question 3 Fine-tune Semantic Segmentation Model.

(a) Complete the 'train' function to fine-tune only the last layer in our downloaded deeplabv3.

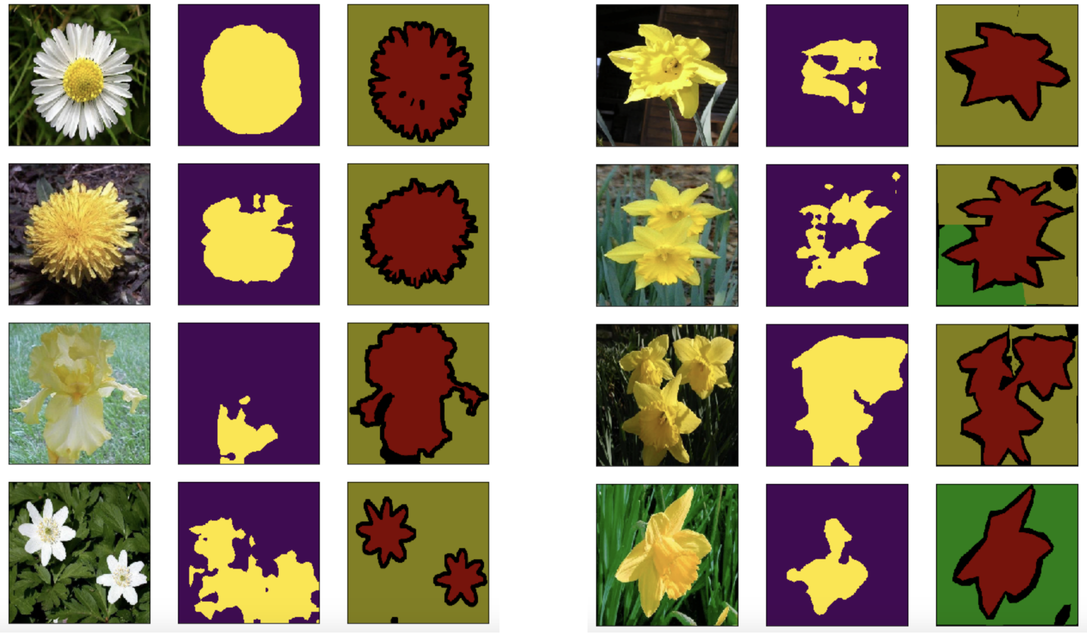
(b) Complete the script in Question 2 of Part C by adding around 2 lines of code and train the model.

(c) *Visualize the predictions by running the helper code provided.*
The visualization of the predictions of the fine-tuned semantic segmentation model is shown in Figure 3.

(d) *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?*

The memory complexity of fine-tuning the entire model is **n times** that of fine-tuning of the last layer. We freeze the other layers, so back-propagation does not enter these layers. It should be noted that in addition to saving parameters, fine-tuning the entire network also needs to save the gradient of the intermediate layer results.
The back-propagation computational complexity of fine-tuning the entire model is also n times that of fine-tuning of the last layer. However, freezing the other layers will not decrease the computational complexity of the forward process, which is almost the same as the computational complexity of the backward process. As a result, the computational complexity of fine-tuning the entire model (O(2n), i.e., O(n)) is **equal** to that of fine-tuning of the last layer (O(n+1), i.e., O(n)), which are both O(n).

Question 3 continued on next page...



(a) Predictions on training set.

(b) Predictions on validation set.

Figure 3: Visualization of the predictions of the fine-tuned Semantic Segmentation Model.

(e) *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?*

The height and the width of the input image will affect the memory complexity of fine-tuning. All the intermediate results are stored for computing the next intermediate results. The memory complexity of the intermediate units (such as feature maps) will be increased by the factor of 4, while the memory complexity of the parameters will not change.
The number of parameters of convolutional layers will not be influenced by the height and the width of the input image, while the number of parameters of Batch Normalization layers will be increased by the factor of 4.