# CSC421 Assignment 2

Zhihong Wang 1002095207

2019-02

Part A: Colourization as Regression

Q1. Describe the model RegressionCNN. How many convolution layers does it have? What are the filter sizes and number of filters at each layer? Construct a table or draw a diagram.

Six convolution layers: downconv1, downconv2, rfconv, upconv1, upconv2, finalconv

| Layer | Filter size | Number of filters |
|---|---|---|
| downconv1 | 3 x 3 | 32 |
| Downconv2 | 3 x 3 | 64 |
| rfconv | 3 x 3 | 64 |
| upconv1 | 3 x 3 | 32 |
| upconv2 | 3 x 3 | 3 |
| finalconv | 3 x 3 | 3 |

Q2. Run all the notebook cells in colour-regression.ipynb on Colab (No coding involved). You will train a CNN, and generate some images showing validation outputs. How many epochs are we training the CNN model in the given setting?

25 epochs.

Q3. Re-train a couple of new models using a different number of training epochs. You may train each new models in a new code cell by copying and modifying the code from the last notebook cell. Comment on how the results (output images, training loss) change as we increase or decrease the number of epochs.

Number of training epochs: 5, 25, 50, 100.

When increasing the number of epochs, the output images become clearer, and the training loss becomes less and stable (i.e. the loss decreased faster at first, and then become slower with the increment of epochs).

In general, with the increment of epochs, we get better results, but the improvement is decreasing.

Epochs 5:


Epochs 25:


Epochs 50:


Epochs 100:

| Epoch 5: | Epoch 25: | Epoch 50: | Epoch 100: |
|---|---|---|---|
| Epoch [1/5], Loss: 0.0666<br>Epoch [1/5], Val Loss: 0.0703<br>Epoch [2/5], Loss: 0.0162<br>Epoch [2/5], Val Loss: 0.0163<br>Epoch [3/5], Loss: 0.0138<br>Epoch [3/5], Val Loss: 0.0136<br>Epoch [4/5], Loss: 0.0128<br>Epoch [4/5], Val Loss: 0.0126<br>Epoch [5/5], Loss: 0.0122<br>Epoch [5/5], Val Loss: 0.0119 | Epoch [1/25], Loss: 0.1070<br>Epoch [1/25], Val Loss: 0.1196<br>...<br>Epoch [21/25], Loss: 0.0085<br>Epoch [21/25], Val Loss: 0.0084<br>Epoch [22/25], Loss: 0.0083<br>Epoch [22/25], Val Loss: 0.0083<br>Epoch [23/25], Loss: 0.0082<br>Epoch [23/25], Val Loss: 0.0082<br>Epoch [24/25], Loss: 0.0081<br>Epoch [24/25], Val Loss: 0.0081<br>Epoch [25/25], Loss: 0.0080<br>Epoch [25/25], Val Loss: 0.0080 | Epoch [1/50], Loss: 0.0501<br>Epoch [1/50], Val Loss: 0.0508<br>...<br>Epoch [45/50], Loss: 0.0067<br>Epoch [45/50], Val Loss: 0.0067<br>Epoch [46/50], Loss: 0.0067<br>Epoch [46/50], Val Loss: 0.0066<br>Epoch [47/50], Loss: 0.0066<br>Epoch [47/50], Val Loss: 0.0066<br>Epoch [48/50], Loss: 0.0066<br>Epoch [48/50], Val Loss: 0.0065<br>Epoch [49/50], Loss: 0.0066<br>Epoch [49/50], Val Loss: 0.0066<br>Epoch [50/50], Loss: 0.0066<br>Epoch [50/50], Val Loss: 0.0066 | Epoch [1/100], Loss: 0.0745<br>Epoch [1/100], Val Loss: 0.0924<br>...<br>Epoch [71/100], Loss: 0.0060<br>Epoch [71/100], Val Loss: 0.0063<br>...<br>Epoch [81/100], Loss: 0.0059<br>Epoch [81/100], Val Loss: 0.0062<br>...<br>Epoch [91/100], Loss: 0.0058<br>Epoch [91/100], Val Loss: 0.0062<br>Epoch [92/100], Loss: 0.0058<br>Epoch [92/100], Val Loss: 0.0062<br>...<br>Epoch [99/100], Loss: 0.0057<br>Epoch [99/100], Val Loss: 0.0062<br>Epoch [100/100], Loss: 0.0056<br>Epoch [100/100], Val Loss: 0.0062 |

Q4. A colour space3 is a choice of mapping of colours into three-dimensional coordinates. Some colours could be close together in one colour space, but further apart in others. The RGB colour space is probably the most familiar to you, the model used in colour-regression.py computes squared error in RGB colour space. But, most state of the art colourization models do not use RGB colour space. How could using the RGB colour space be problematic? Your answer should relate how human perception of color is different than the squared distance. You may use the Wikipedia article on color space to help you answer the question.

Wiki: A color space in which the perceptual difference between colors is directly related to distances between colors as represented by points in the color space.

In the RGB colour space, some colours are close to each other can help the model to make a good prediction by computing the distance; some colours are far from each other will show less relevance based on the model to make a bad prediction. The squared error could remain the same even if we use different

color intensities combinations. (i.e. the output of image maybe inaccurately)

Q5. How does framing colourization as a classification problem alleviate the above problem?

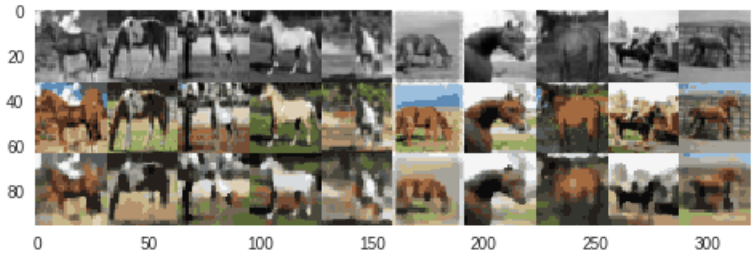
At the regression model, we try to minimize the squared error to get the prediction, which is inefficient, may cause overfitting, and generate additional colors for the output.

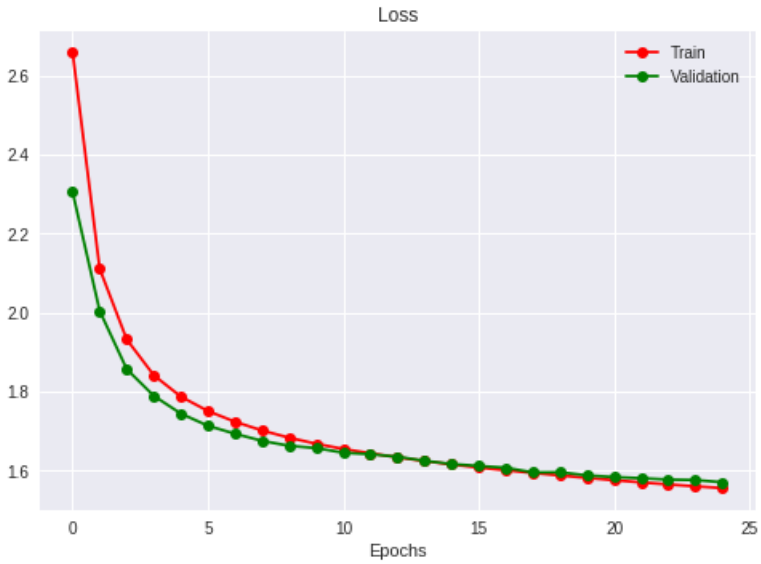
By using classification which is discrete, we can increase the efficiency, reduce overfitting, generate colors based on training data, get more color combinations even they are far from each other on the colour space.

Part B: Colourization as Classification

Q2. Run main training loop of CNN in colourization.ipynb on Colab. This will train a CNN for a few epochs using the cross-entropy objective. It will generate some images showing the trained result at the end. How do the results compare to the previous regression model?

The result is better than the previous regression model. We can see the CNN predicted more correct colours than the previous one. However, the results of CNN is not predicted every parts correctly. There still exists some wrong colours.



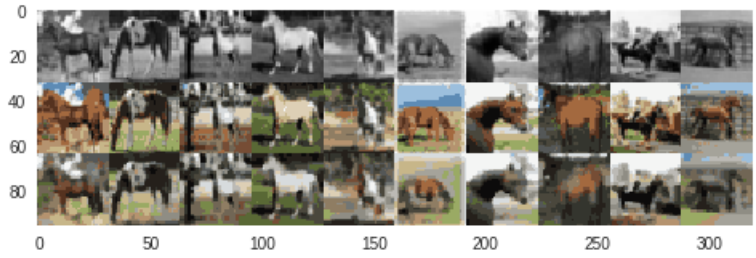



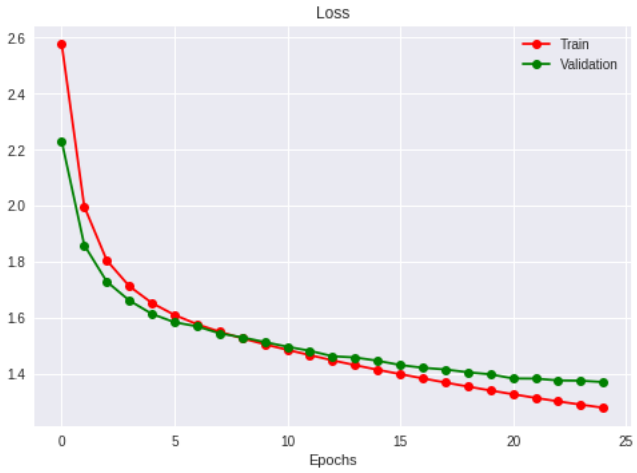
Part C: Skip Connections

Q2. Train the "UNet" model for the same amount of epochs as the previous CNN and plot the training curve using a batch size of 100. 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.

It is better the CNN model before with the same epochs. Both training and validation loss got reduced based on the graphs (from 1.6064 to 1.3894), and the accuracy got increased around 8% (from 40 to 48 %). The output images are better and sharper too. The colours are more correct.

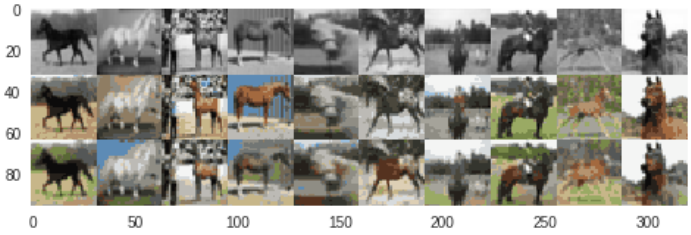
Two reasons of skip connections improves the performance:
1. Image details may lost when the network contains too many layers. Skip connections help to recover those details in deeper layers to improve the output.
2. Skip connections input the initial greyscale input image to the final convolution, which provides more image details into the final layer, and helps to improve the validation accuracy and the output qualitatively.



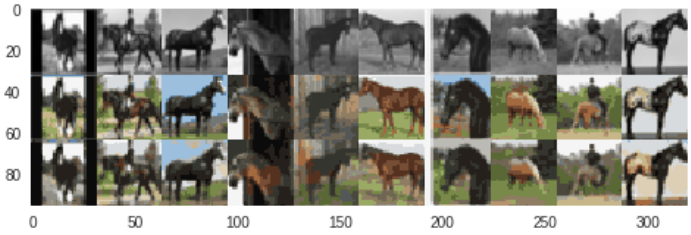



Q3. Re-train a few more "UNet" models using different mini batch sizes with a fixed number of epochs. Describe the effect of batch sizes on the training/validation loss, and the final image output.

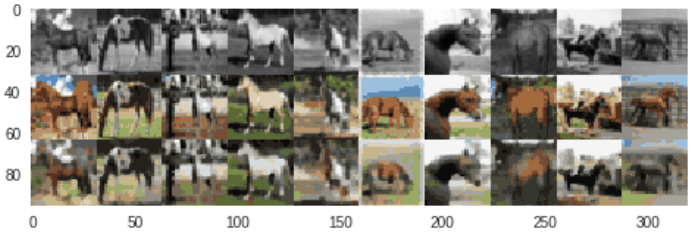
Batch 10:



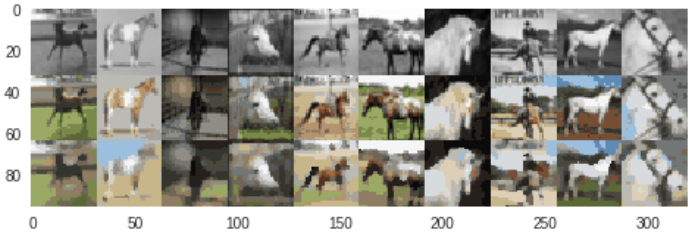

Batch 50:



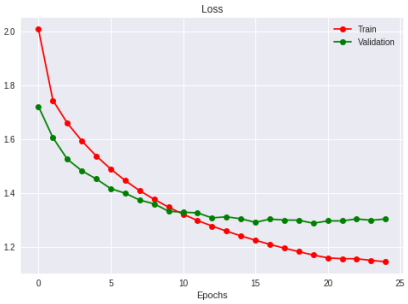

Batch 100:



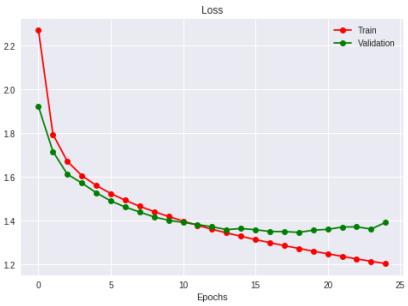

Batch 500:



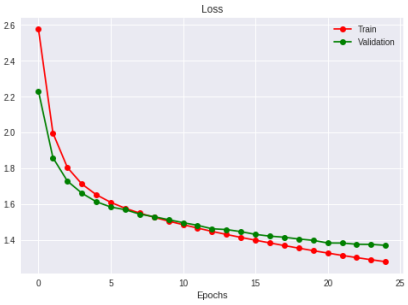
Batch 10:



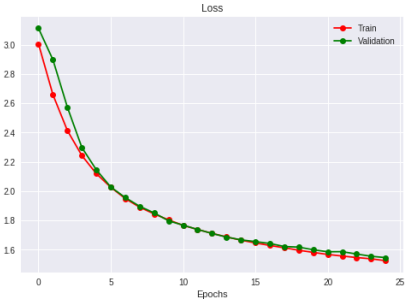

Batch 50:




Batch 100:




Batch 500:



With the increment of batch size,

the loss becomes higher ($1.2845 \rightarrow 1.3467 \rightarrow 1.3894 \rightarrow 1.5616$);

the accuracy becomes lower ($52\% \rightarrow 48\% \rightarrow 48\% \rightarrow 42\%$);

the training time is decreasing ($142 \rightarrow 82 \rightarrow 77 \rightarrow 71$);

the quality of image outputs becomes lower too (check previous images).

If the batch size is too large, the validation loss will become greater than the training loss.

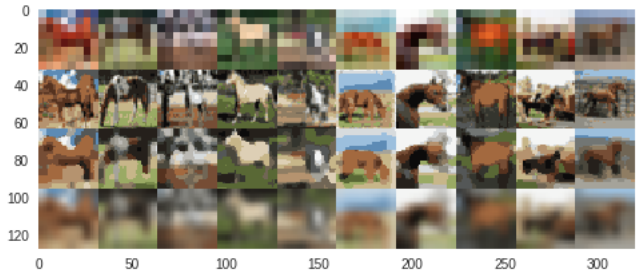
Part D: Super-Resolution

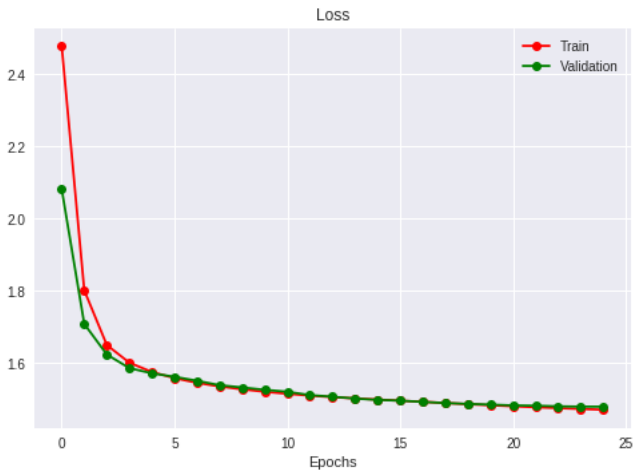
Q1. Take a look at the data process function *process*. What is the resolution difference between the downsized input image and output image?

The resolution of the inputs is less than the outputs.

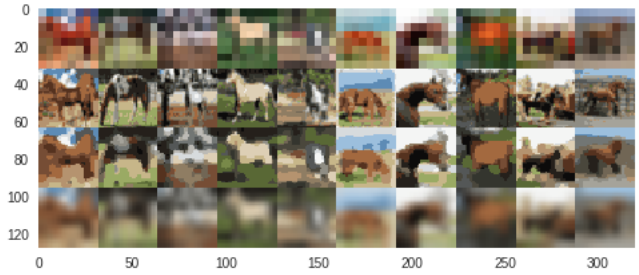
Q2. Bilinear interpolation is one of the basic but widely used resampling techniques in image processing. Run super-resolution with both CNN and UNet. Are there any difference in the model outputs? Also, comment on how the neural network results (images from the third row) differ from the bilinear interpolation results (images from the fourth row). Give at least two reasons why conv nets are better than bilinear interpolation.

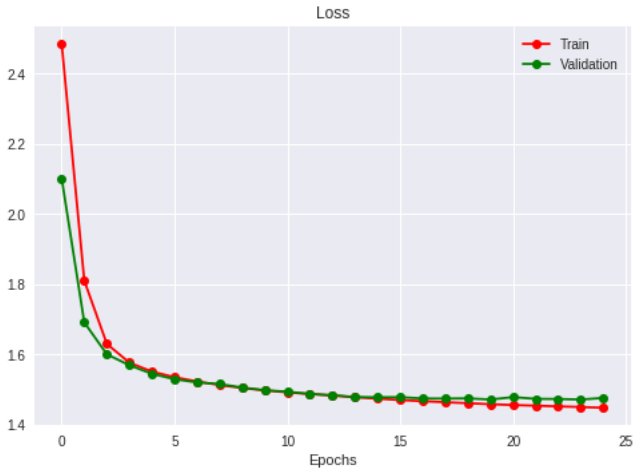
CNN:




UNet:




The outputs from CNN or UNet model are similar.

The neural network results (images from the third row) are better, clearer, more recognizable than the bilinear interpolation results (images from the fourth row).

Reasons of conv nets are better than bilinear interpolation:
1. Bilinear interpolation will lose image quality because it will blur the sharpness during the prediction.
2. Since bilinear uses a weighted average of the four nearest cell centers. The closer an input cell center is to the output cell center, the higher the influence of its value is on the output cell value. The output value could be different than the nearest input, but is always within the same range of values as the input. Since the values can change, bilinear is not recommended for categorical data. Instead, it should be used for continuous data like elevation and raw slope values.

Part E: Visualizing Intermediate Activations

Q1. Visualize the activations of the CNN for a few test examples. How are the activation in the first few layers different from the later layers? You do not need to attach the output images to your writeup, only descriptions of what you see.

The image becomes clearer and clearer after each layer. The first few layers focus on the specific parts, e.g. the edges, distinguish points like eyes, mouth of the horse.

For the later layers, it focus on the whole image, like the horse and the background. The last layer is the final prediction.

Q2. Visualize the activations of the colourization UNet for a few test examples. How do the activations differ from the CNN activations?

The UNet has more colourful, clearer, brighter outputs than CNN, since it has more pixels.

For activations in UNet, the first few layers still focus on the specific parts. Further more, due to the skip connections which connects the output of first layers to the last layers, the last few layers not only focus on the whole image, but also focus on the details too. The last layers become similar to the original input (i.e. the output images have more details and sharpness).

Q3. Visualize the activations of the super-resolution UNet for a few test examples. Describe how the activations differ from the colourization models?

Compare to the colourization UNet, the super-resolution UNet has less details. The activations are similar to the previous one: the first layers focus on the specific parts, the last layers focus on the whole image and the details. However, since super-resolution is aim to recover the high resolution image from a low resolution input, the predictions that it made are blurry, which are not as good as colourization UNet.

Part F: Conceptional Problems

Q1. We also did not tune any hyperparameters for this assignment other than the number of epochs and batch size. What are some hyperparameters that could be tuned? List five.

kernel size, number of colors, number of filters, learning rate, activation function, loss function, weights, etc.

Q2. In the RegressionCNN model, nn.MaxPool2d layers are applied after nn.ReLU activations, comment on how the output of CNN changes if we switch the order of the max-pooling and ReLU?

Both orders will have the same outputs. But the running times are different. Relu layer doesn't change the size of the input, but max pooling can reduce it. We can save a lot of running time if we put max pooling before the non-linear layers like relu.

Q3. The loss functions and the evaluation metrics in this assignment are defined at pixel-level. In general, these pixel-level measures correlate poorly with human assessment of visual quality. How can we improve the evaluation to match with human assessment better? You may find useful for answering this question.

Based on the "Perceptual losses for real-time style transfer and super-resolution.", we can use the perceptual loss to improve the evaluation to match human assessment better. Normally, we use the per-pixel loss, which correlate poorly with human assessment. However, if we use a loss network to generate the perceptual loss, it computes the predictions which match human assessment better. Perceptual loss is able to better reconstruct fine details compared to methods trained with per-pixel loss, since the per-pixel loss does not capture perceptual differences between output and ground-truth images. Perceptual losses measure image similarities more robustly than per-pixel losses, and at test-time the transformation networks run in real-time. We can define two perceptual loss functions that measure high-level perceptual and semantic differences between images. They make use of a loss network pre-trained for image classification, meaning that these perceptual loss functions are themselves deep convolutional neural networks.

Q4. In colourization.ipynb, we have trained a few different image processing convolutional neural networks on input and output image size of 32x32. In the test time, the desired output size is often different than the one used in training. Describe how we can modify the trained models in this assignment to colourize test images that are larger than 32x32.

If we can only modify the trained models, then changing the number of filters to the size we want is a good way.

If it is asking how do we modify the CNN so that it can train on 32x32 images but be tested with larger than 32x32 images, then we can downsample the test input by adding a pooling layer to the test image to decrease the size.