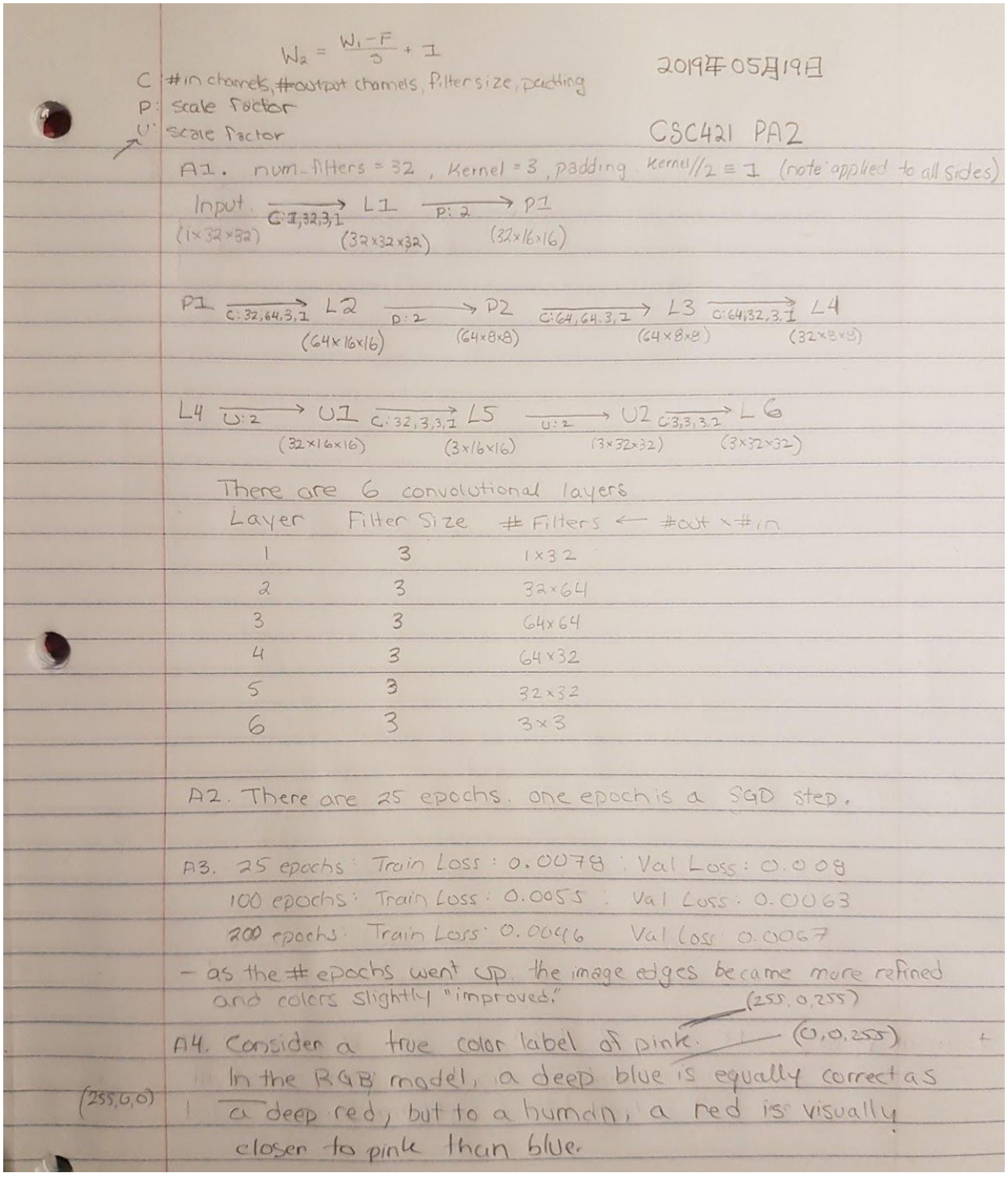


A5. The labels for classification problems comes from humans which introduces human categories, biases and preferences. Humans will label ambiguously red/pink things as red/pink and therefore the algorithm will learn to associate red and pink together, but not blue. During classification, it will output red/pink objects as highly probable of being red or pink, but again, not blue. For unambiguously pink or red objects, e.g. pigs, humans will only label as pink, and never blue.

B1. Note that we only had to do a convolution with 24 output channels and we did not have a softmax layer. This is because the criterion actually includes a softmax procedure automatically.

B2. The categorization method used a higher range of colors when recoloring, and perhaps unambiguously better colors too. For example, it used green to color grass where the regression model did not. In another picture however; it colored part of a horse green, and it colored the ground green where it was actually not grass.

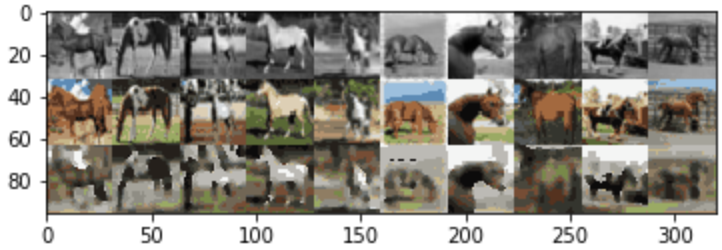


Figure: CNN 25 epochs

C2.

	Train Loss	Val Loss	Val Acc	Epoch	Batch Size
UNet	1.6706	1.6558	40.0%	25	100
CNN	1.7424	1.7284	37.0%	25	100
UNet	1.3907	1.5418	42.0%	100	100
CNN	1.5441	1.5807	41.0%	100	100

As can be seen in the graph above, the skip connection neural network trained faster and achieved a better accuracy. However; it was also very prone to overfitting. The training loss after 100 epoch was 1.39 while the validation loss was 1.54. One can also see this in the training graphs below.

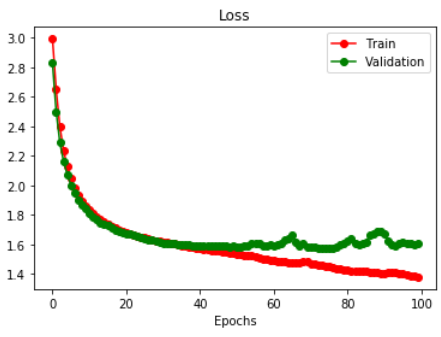


Figure: UNET,Batch: 100

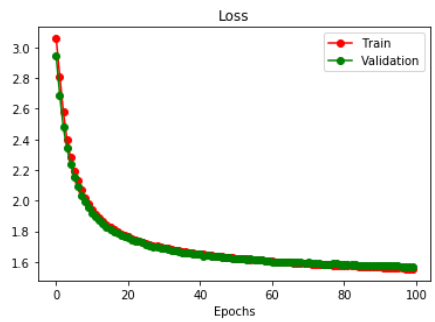


Figure: CNN,Batch: 100

Qualitatively, the output after 100 epoches was not much better. Both had similar performance. After 25 epochs though, the skip connection NN was better. This can be seen by looking at the picture in the third-from-last column where the skip connection NN colored the horse less green.

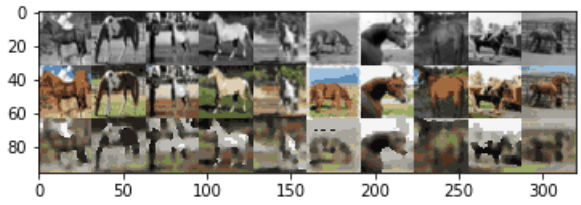


Figure: Results after 25 epochs for CNN

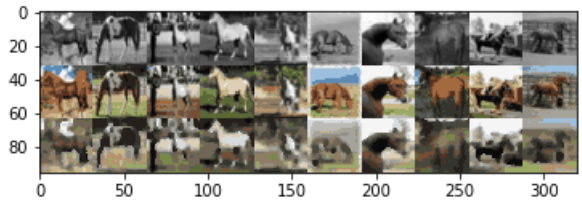
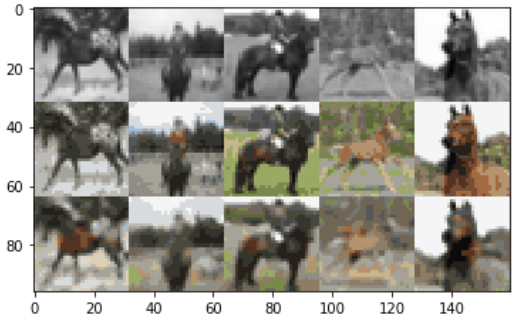


Figure: Results after 25 epochs for UNet

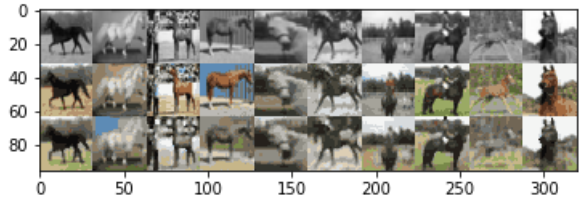
Skip connections can be better because:
- pass stronger gradient information from early layers to the loss function (whereas gradient from early layers gets diluted if it is passed through all the layers)
- prevents NN from learning identity functions if they are necessary, which means it can use the same weights differently and represent a wider set of functions

C3. The number of epochs was held constant at 25.

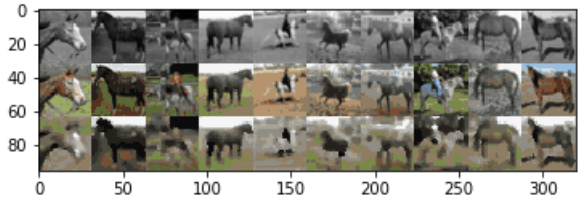
Batch Size	Train Loss	Validation Loss	Val Accuracy	Time
5	1.5197	1.5055	44.0%	157
10	1.4873	1.5404	42.0%	82
50	1.5652	1.5622	42.0%	31
100	1.6620	1.6462	39.0%	33
200	1.8023	1.7982	36.0%	31
500	2.0682	2.0602	30.0%	31
1000	2.4509	2.4562	22.0%	31



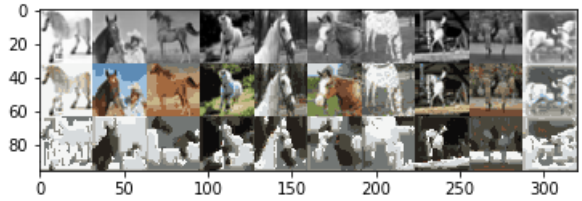
Batch Size: 5



Batch Size: 10



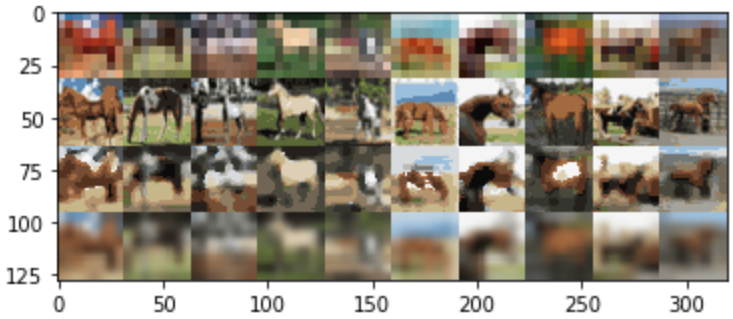
Batch Size: 200



Batch Size: 1000

As can be seen, as batch size increases, the losses increase and accuracy decrease. Qualitatively, the recolorization also degrades. Also, due to some computation quark, counter-intuitively the time to train also decreases. These are shocking and counter-intuitive results.

D1. The input image is averaged using a 2x2 filter twice and then upscaled twice. Therefore, it's resolution is 16 times lower. Below is the (in row order) input image, target image, predicted image and finally, a classical approach to improving resolution: bilinear interpolation. It is clear that the neural network's performance is far superior.



E1. Most of the activations in the first layer have a distance boundary between the 'ground' and 'the sky' and there are shapes which could plausibly be horses, or parts of them. The later layers are more abstract and blurry. In the last layer, the original image can be seen but it is blurry. It seems to separate ground, sky, rider and horse.

E2. The activations of the later layers of the network have more clearly refined edges and shapes belonging to the original image.

E3. The shapes in the last layer resemble horses much less than the previous applications where we were converting from greyscale to color.

F1.

1. learning rate
2. number of hidden layers
3. pooling filter size
4. convolution filter size
5. (for regression problem) squared norm penalization constant

F2. Consider a 2 x 2 example [[x1, x2], [x3, x4]]
Applying ReLu, and then pool:
$r = \frac{max(0, x1) + max(0, x2) + max(0, x3) + max(0, x4)}{4}$
Appling pool, then ReLu:
$r = max(0, \frac{x1 + x2 + x3 + x4}{4})$

Applying a pool after ReLu averages only the activated neurons, where activated means their value is greater than zero.
Applying a pool before ReLu averages the local group before it, as if treating it as a single neuron, and then activates it if on average the value is greater than zero.
Intuitively, we expect that applying a ReLu *after* a pool will result in more deactivated neuron.

F3. The key idea of [4] and the way we can improve on a pixel-based loss function is to define a loss function that uses a pre-trained image classifying neural network because it is already trained to recognize perceptual and semantic information in images.

Suppose there is an image classifier network ϕ .Then a 'feature reconstruction loss' function is defined that attempts to make the activations of ϕ equal for both the label image y and the output image $\hat{y}$.

F4. One possible option is to break the image up into 32x32 pieces (zero padding remainders) and recolor portions of it, stitching the output at the end.

Another possibility is to downscale the image to 32x32, recolor it, and then use the super resolution neural network to upscale it.