CS5785 HW4

Andrew Palmer (ajp294), Kushal Singh (ks2377)

4 December 2019

NOTE: We used our one and only slip day on this assignment.

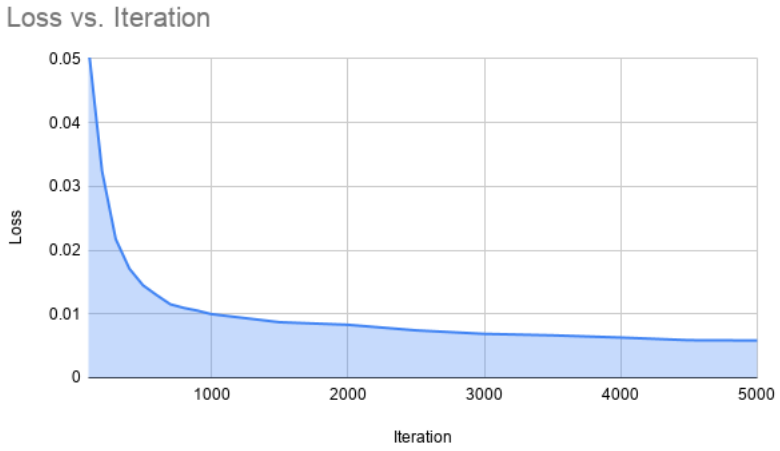
1 Image Approximation with Neural Networks

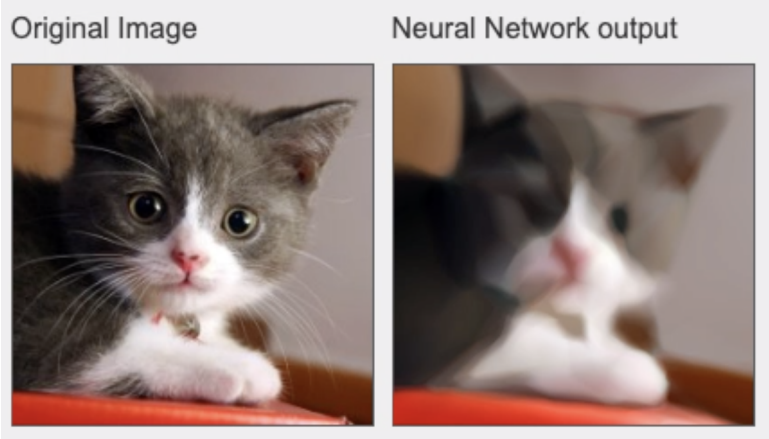
(a) This neural network is set up with 8 layers. Out of these 8 layers, the 7 hidden layers are performing the RELU algorithm with 20 neurons in each of those layers. Finally, there is an output layer with the L2 regression algorithm consisting of 3 neurons.

(b) The loss described in this demo comes from the regression output layer described earlier, which is the difference in pixel values between the original image and neural network output image. By inspecting the code, specifically in the *covnet_layers_loss.js* file, we see that the loss function used in this network is the L2 distance function.

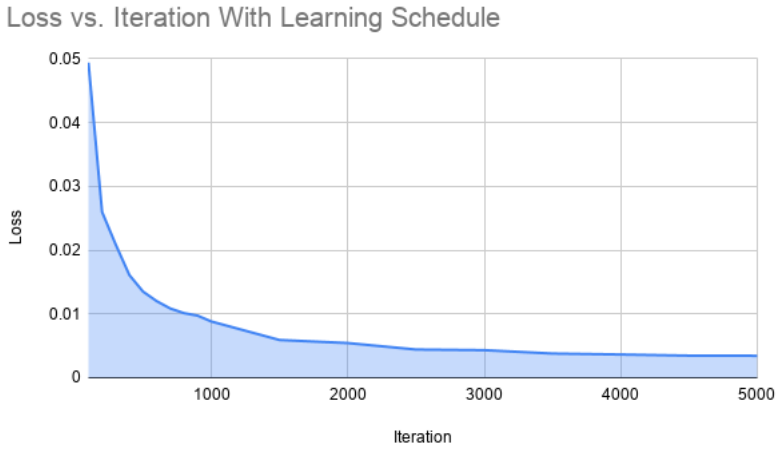
```
// implements an L2 regression cost layer,
// so penalizes \sum_i(||x_i - y_i||^2), where x is its input
// and y is the user-provided array of "correct" values.
var RegressionLayer = function(opt)
```

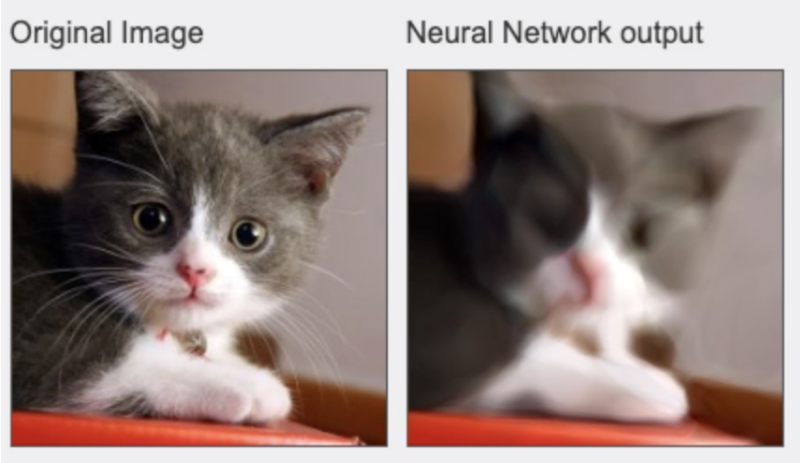
(c) Plotting the loss over time, we see that the loss gets to about 0.00584227. The graph of loss per iteration is shown below along with the generated image.



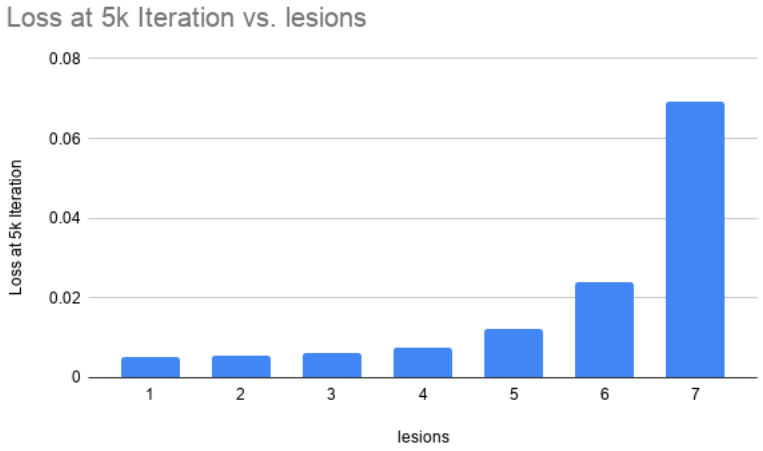


(d) To make the network converge to a lower loss function by the time it reached 5000 iterations, we lowered the learning rate every 1000 iterations. As suggested, we halved the learning rate every 1000 iterations. As shown in the graph below, this procedure allowed us to converge to a lower loss function of about 0.003446425 compared to 0.005842270, which is about an improvement of about 41%. The output image is also shown below, which you can see has slightly more accuracy to the original image.

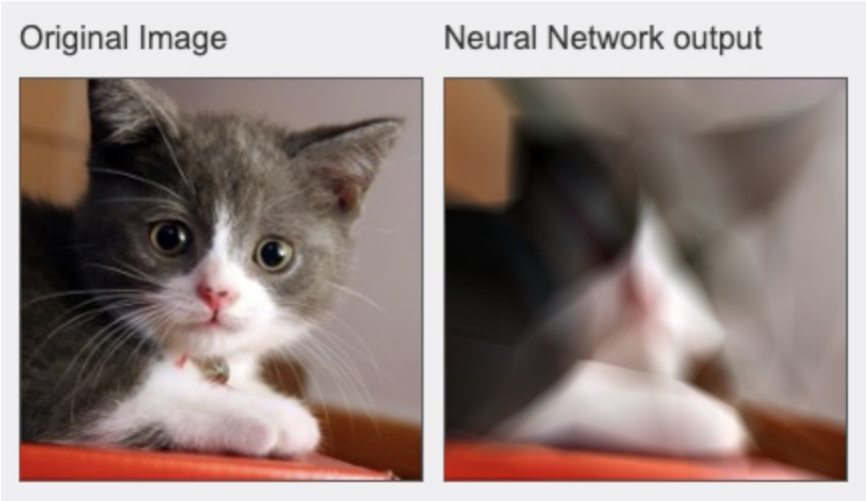


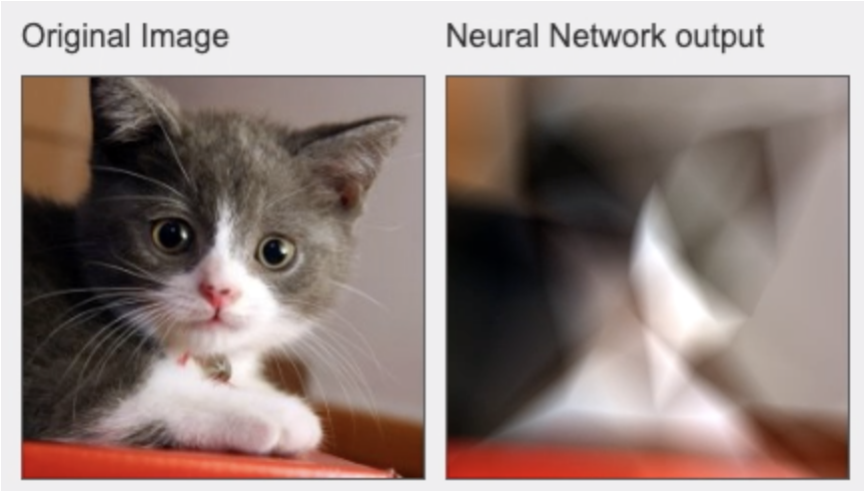


(e) To perform our lesion study, we removed one hidden layer at a time and noted the loss at 5000 iterations for each layer. The results are shown in the graph below. As illustrated, we noticed that we can drop about 4 layers before the loss becomes incredibly noticeable. At the 4th lesion, we are still under a loss of 0.01. Thus, we concur that at least 3 hidden units are needed before quality drops noticeably.



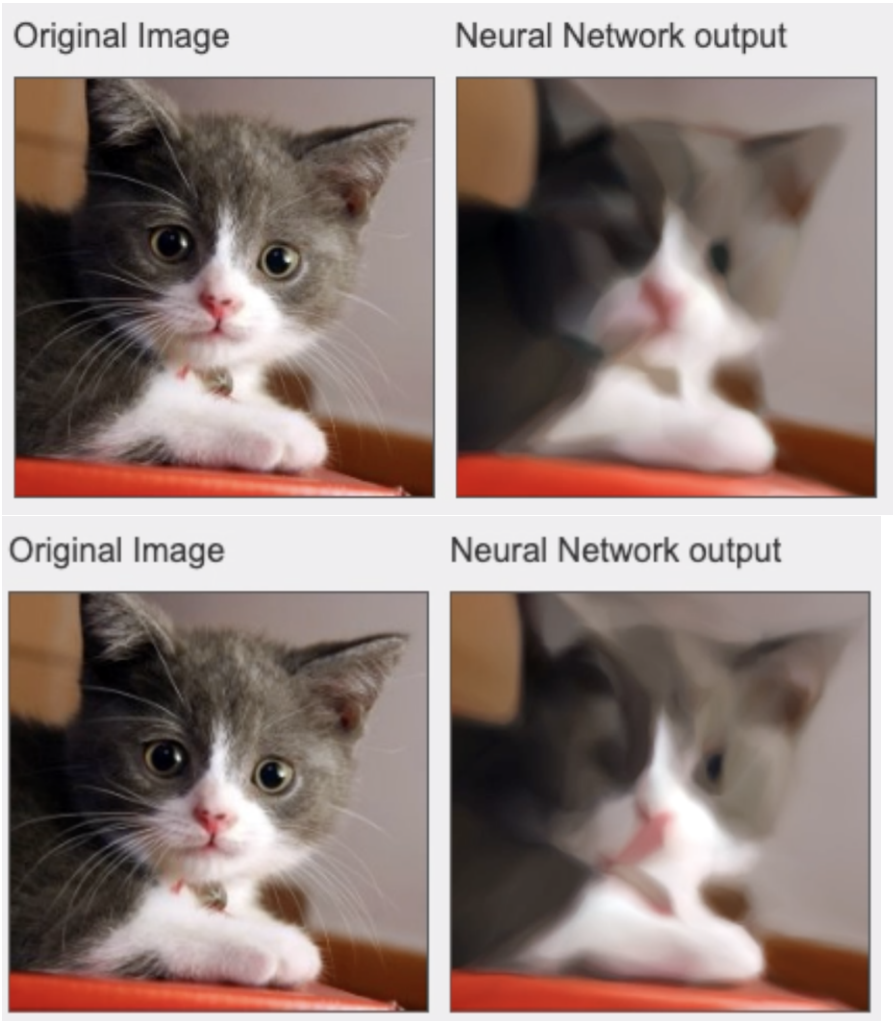
Below we have an image comparing the 4th lesion (1st image) and 5th lesion (2nd image). Even from the output images, we can see the large loss of detail in the 5th lesion image.





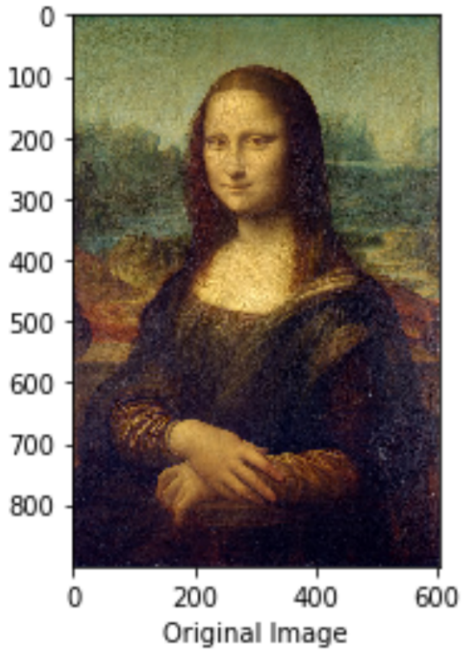
(f) For this experiment, we added 3 additional hidden layers to the network. Following a similar procedure to before, we let the network run for 5000 iterations. After 5 iterations, we saw that the loss was about 0.005288290905. Compared to the original network, which had a loss of about 0.00584227093, this is about a 9% improvement. However, this is not a very noticeable improvement in the accuracy in terms of the trade off in network complexity and compute time. The original network took about $2:33$ to reach 5000 iterations and the new network took about $3:34$ to reach 5000 iterations. The additional 30% in processing time does not yield enough of an accuracy improvement to be worthwhile for this use case.

The first image is the original network, and the second image is the generated image from the network with 3 additional layers. From the images, we cannot see noticeable improvement from the network with more layers.

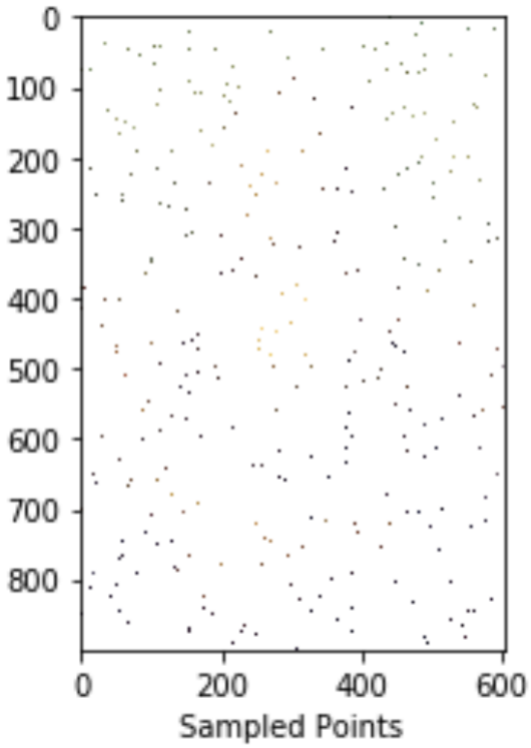


2 Random forests for image approximation

(a) We did end up using an image of the Mona Lisa for this assignment. The image is pictured below.



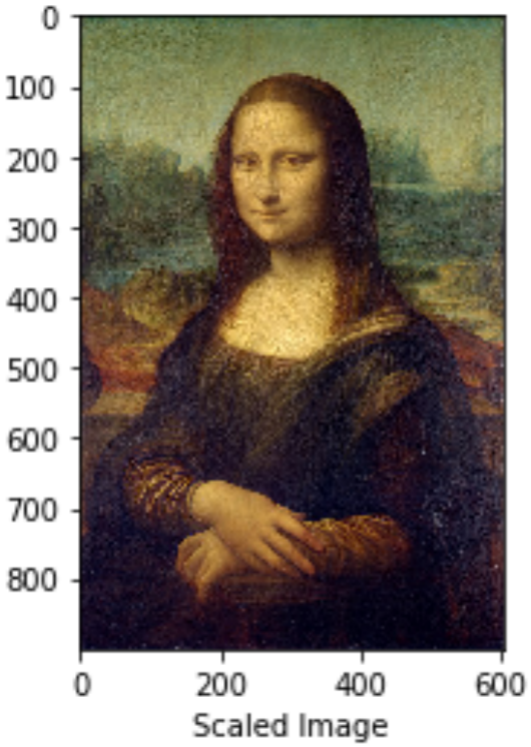
(b) For this part, we uniformly sampled 5000 random (x, y) coordinate locations from the Mona Lisa, and collected the r, g, b label values of those coordinates. We then plotted the 5000 random coordinate locations along with the r, g, b vector and named the graph Sampled Points.



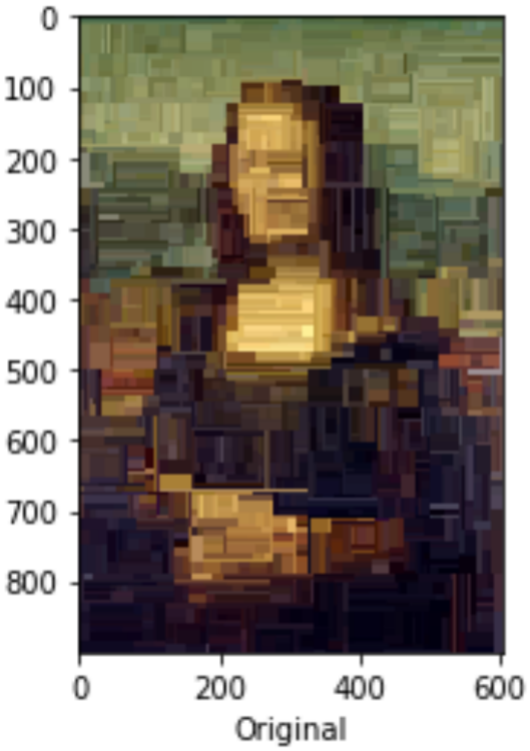
Since decision trees are not affected by the structure of the data, mean subtraction, standardization, or unit normalization are not necessary to perform. Therefore, we did not use any other preprocessing steps for random forest inputs.

(c) We decided to use the second approach to sample pixel values at each of the given coordinate locations, where our function maps (x, y) coordinates to (r, g, b) values.

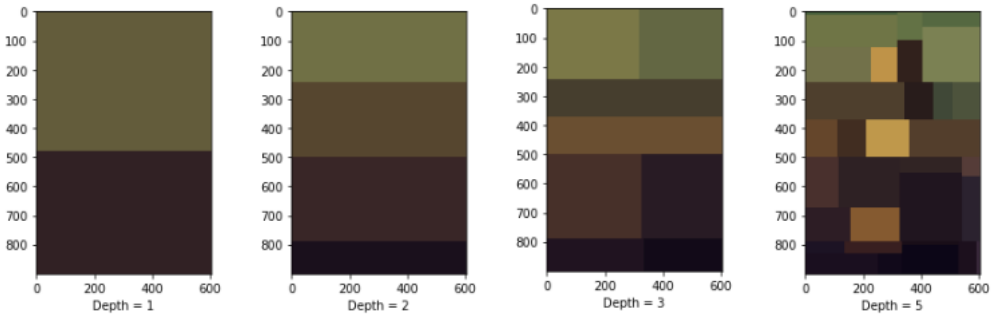
In order to rescale the pixel intensities to lie between 0.0 and 1.0, we divided each of the r, g, b pixel values at each of the coordinates in the original image by the total number of pixels = 256. A picture of the sampled image is shown below.

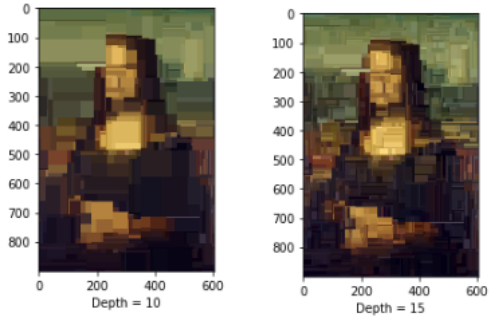


(d) Attached is a picture of the predicted image.



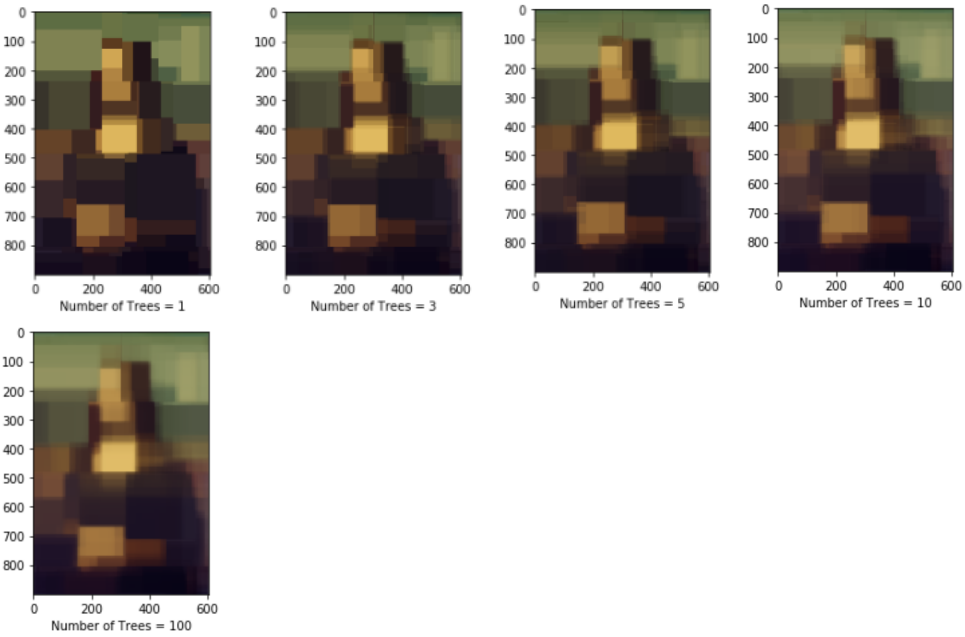
(e) (i) Pictured below are the graphs for depths = 1, 2, 3, 5, 10, 15, for a single decision tree.





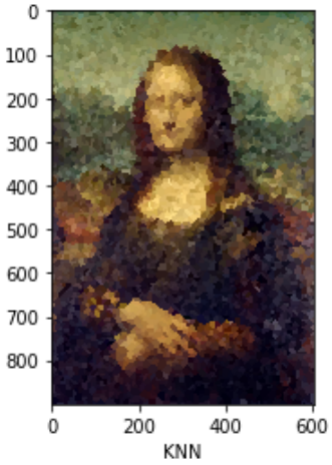
As depth increases, the image has more blocks of color and therefore is similar to the original. A deeper tree means more splits, with more information about the data being captured. This results in more differentiation.

(ii) Pictured below are the graphs for number of trees = 1, 3, 5, 10, 100, with a depth = 1.



As number of trees increases, the image becomes smoother, looking more similar to the original. One reason for this could be that errors get normalized with a larger number of uncorrelated trees, leading to smoother gradients in different blocks.

(iii) Picture below is the image generated from KNN regressor with $k = 1$.



In random forest, since we split the image based on the pixel position of the image compared with the threshold of each subtree in the forest, our image ended up being divided into small rectangles of similar colors.

In KNN, each pixel color is assigned to the closest one in the sample, which makes sense because the image we ended up getting was small pieces of color, with non-linear boundary between them.

(iv) We decided not to prune the tree since the sample size itself is fairly large. More specifically, the intention of pruning is to prevent overfitting. In random forests, however, each tree has already been through bagging, which already accounts for overfitting of the data.

(f)

(i) The decision rule is to split the r, g, b vector based on a particular input coordinate (x, y). If we assume that each split is based on the x coordinate and a threshold T, then the formula would be:

If $X \geq T$, then output left leaf node.
Otherwise, output right leaf node.

(ii) The resulting image is made up of patches of color since we have a limited number of colors as our samples.

In random forest, each block of color has a rectangular shape and there are overlaps. Since different segments of the images are being assigned into these colors, we observe patches of colors.

In KNN, since each pixel color is assigned to its closest neighbor, the image is comprised of small pieces of color, with a nonlinear boundary between them.

(iii) 2^{depth}

We know that each leaf of the tree results in a patch of color. Furthermore, since there is only one tree, number of leaves = 2^{depth}.

(iv)

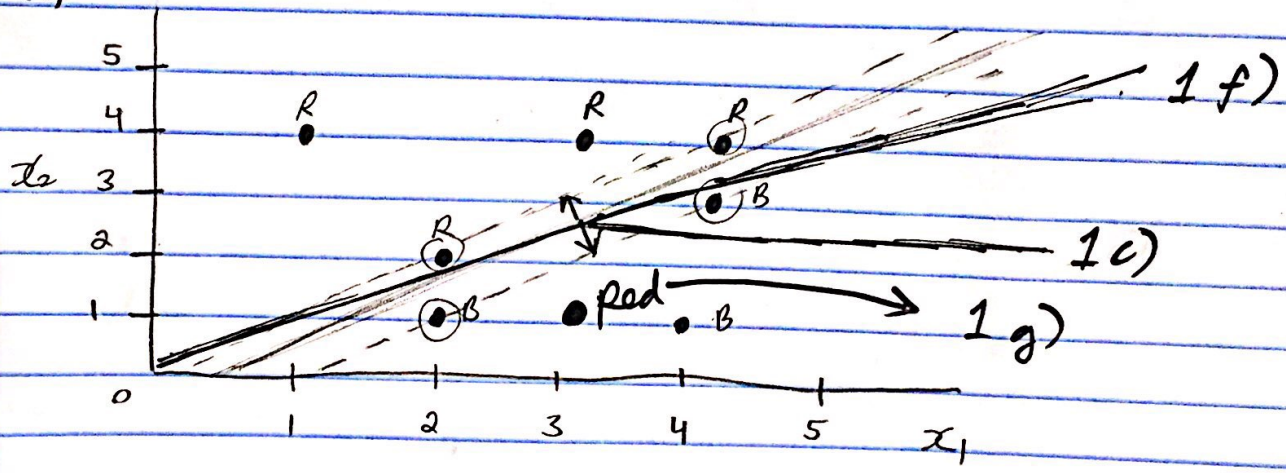
$$n * 2^{depth} \text{ choose } n$$

For a single tree, the number of leaves = 2^{depth}. Therefore, by similar logic, for n trees, the number of leaves $\leq n * 2^{depth}$. There are $n * 2^{depth}$ choose n ways of combinations to choose n subtrees from $n * 2^{depth}$ decisions.

1.

X_1	X_2	Y
3	4	Red
2	2	Red
4	4	Red
1	4	Red
2	1	Blue
4	3	Blue
4	1	Blue

a)



b) $2x_2 - 2x_1 + 1 > 0$

$\beta_0 = 1, \beta_1 = -2, \beta_2 = 2$

c) Margin is distance between dotted lines on the graph. It is marked as 1c).

d) Support vectors are marked in the graph:
(2, 1), (2, 2), (4, 3), (4, 4)

e) Since the 7th observation (4, 1) is <u>not</u> a support vector, it would <u>not</u> affect the maximal margin hyperplane.

f) The line is sketched in the graph.

$$\frac{1}{4} + x_1 + \frac{3}{4}x_2 = 0;$$

$$\beta_0 = \frac{1}{4},\ \beta_1 = 1,\ \beta_2 = \frac{3}{4}$$

g) Marked in graph as 1 g), @ (3, 1) with a red dot.

2. Neural Networks as function approximates

Fn. from graph :

$$y = \begin{cases} 0, & x \in [0, 1) \\ 2x-2, & x \in [1, 2) \\ \frac{1}{3}x + \frac{4}{3}, & x \in [2, 5) \\ 2x-7, & x \in [5, 6) \\ -\frac{5}{3}x + 15, & x \in [6, 9) \\ 0, & x \in [9, 10] \end{cases}$$

$a_1(b_1x + c_1) = 2x - 2, \quad a_1 = 1, \; b_1 = 2$

$2x - 2 + \underbrace{a_2(b_2x + c_2)}_{= -\frac{5}{3}x + \frac{10}{3}} = \frac{1}{3}x + \frac{4}{3}$

$a_2 = -1, \; b_2 = \frac{5}{3}$

$\frac{1}{3}x + \frac{4}{3} + a_3(b_3x + c_3) = 2x - 7, \quad a_3 = 1, \; b_3 = \frac{5}{3}$

$2x - 7 + a_4(b_4x + c_4) = -\frac{5}{3}x + 15, \; a_4 = -1, \; b_4 = \frac{11}{3}$

From this, we can see that the neural network has 1 hidden layer with 4 units.