Notes for Evaluation

* User studies
  + Def talk about how users had to back up?
  + My results for one user study are in the Excel file inside the folder \_usertests/
* Check how many tiles we actually use and try to figure out what images work well with what kinda datbases: I wrote a method, and on average we’re using around 100 of a 500 image database, ranging from 10-25% - could we better utilize them in the future?
* Check how many different tiles appear in a collage, and take that as a ratio of total number from database, to see how many images we actually use:

**print** "Percent of possible tiles used: **%.3f**, **%d** out **%d** images from tile library used" %(round((float(n)/len(tiles)), 3), n, len(tiles))

Rainbow Image shows the lack of green in our tile database:

Also, I think the randomness of dominant color for the first cat looks good… I REALLY WANT TO ADD THESE CATS BECAUSE I FEEL LIKE SO MUCH OF OUR STRUGGLES WAS BASED ON THEM AND THE REALIZATION THAT 8 BIT NO BUENO FOR FINDING SIMILARITIES BY COLOR HISTOGRAMS

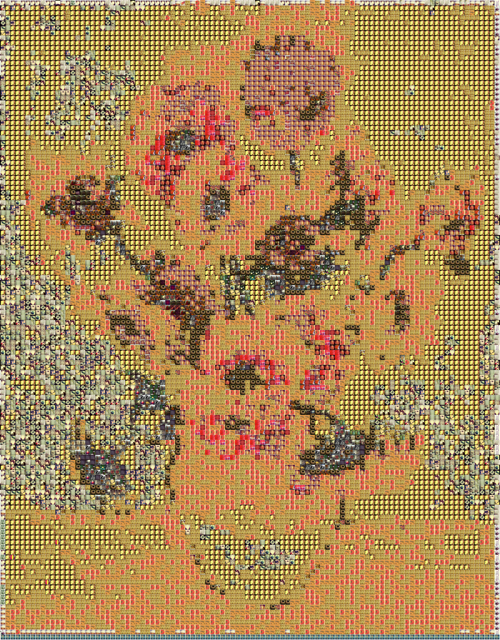
|  |  |
| --- | --- |
|  |  |
| Percent of possible tiles used: 0.236, 118 out 500 images from tile library used  Expensive operations: 237 of 4100 : 0.0578048780488  Dominant operations: 908 of 4100 : 0.221463414634  History operations: 2955 of 4100 : 0.720731707317  Percent of possible tiles used: 0.176, 88 out 500 images from tile library used | The lack of effective green in our database becomes apparent when we test a rainbow image. |

* If we don’t actually use all the images, then we don’t need to load PIL display images for all of them. we could add a “cached” property to each tile… if it’s cached, that means it has a tile.display image and you can just use it, else you can load a new image with PIL and then cache it
* If we can increase efficiency and diversification, we can use a bigger database
  + Instead of loading only first 500 images for our database, we could load more images, or load RANDOM images from the database, not just 1-500!
* Try grayscale method and see effects
  + One way we can do a fast algorithm is to sort all the tile images in the database based on their brightness, then we can select the right image to match the cell in the source image based on itsposition, i.e. if we need a full black image, grab ones at the front of the sorted list, if needing full white, grab one at the end, and so forth for everything to begin
  + CAN WE DO SOMETHING SIMILAR WITH COLOR?
  + If we do blacmk and white, we’ll find that they work best with relatively HIGH CONTRAST source images

FUTURE EXTENSIONS TO SYSTEM

* A graphical user interface
  + Let users choose their tile database and base images
  + If we had more time, could we have written a method that can see how well an image database would perform with a given base image? i.e. see the range of colors represented in a database (for example, we’re missing green)
* Ghosting – placing a layer of original image on new image – but actually pretty proud we have such good results without ghosting!
* Use tile pictures that are conceptually related to the source image
  + Fix instagram zipper file not to grab images by user, but instead grab images by hash tags!
* Vary the size of the individual images in the grid
  + While the uniformity of the grid is nice for making the image as clear as possible, it would be compositionally more interesting and collage-like to have the sizes of the images vary more
  + Could try breaking up the rectangular shape of each of the source images and using masking
  + Using Laplacian or Canny edge dtection to detect edges of the picture
* Would have liked to play with Jigsaw method or explore Voronoi stippling. This is a technique for converting a grayscale image into a series of dogs of different weights to represent the darkness of each region in a natural way, much like stipped drawings created by hand
* Build a histogram of the number of tiles for each grayscale intensity. Rescale the intensity of images so that each grayscale level has several photos to choose from.
* Come up with a fast way of selecting image tiles that does not require comparing each pixel to each tile's mean intensity.

Bad:

Deep Representations

* Could we tie in deep representations? Kender emphasized deep representation and neural networks quite a bit in the last class
  + If we had tile pictures conceptually related to source image
  + Better dominant color matching
  + Varied tile sizes so some of the tiles were larger and you could immediately see them
  + Would the human brain look and immediately make the various symbolic patterns and connections?

From Internet:

* Imagine you are trying to recognize someone's handwriting - whether they drew a '7' or a '9'. From years of seeing handwritten digits, you automatically notice the vertical line with a horizontal top section. If you see a closed loop in the top section of the digit, you think it is a '9'. If it is more like a horizontal line, you think of it as a '7'. Easy enough. What it took for you to correctly recognize the digit, however, is an impressive display of fitting smaller features together to make the whole - noticing contrasted edges to make lines, seeing a horizontal vs. vertical line, noticing the positioning of the vertical section underneath the horizontal section, noticing a loop in the horizontal section, etc.
* Ultimately, this is what deep learning or representation learning is meant to do: discover multiple levels of features that work together to define increasingly more abstract aspects of the data (in our case, initial image pixels to lines to full-blown numbers).
* Deep learning is about creating an abstract hierarchical representation of the input data to create useful features for traditional machine learning algorithms. Each layer in the hierarchy learns a more abstract and complex feature of the data, such as edges to eyes to faces.
* This representation gets its power of abstraction by stacking nonlinear functions, where the output of one layer becomes the input to the next.
* The two main schools of thought for analyzing deep architectures are probabilistic vs. direct encoding.
* The probabilistic interpretation means that each layer defines a distribution of hidden units given the observed input, P(h | x).