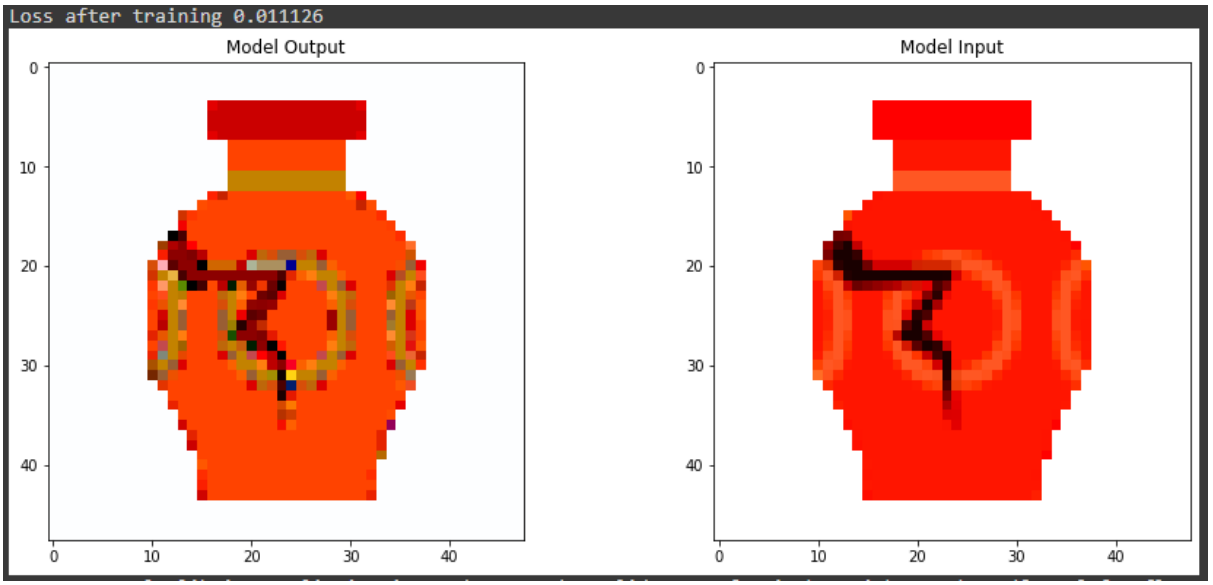
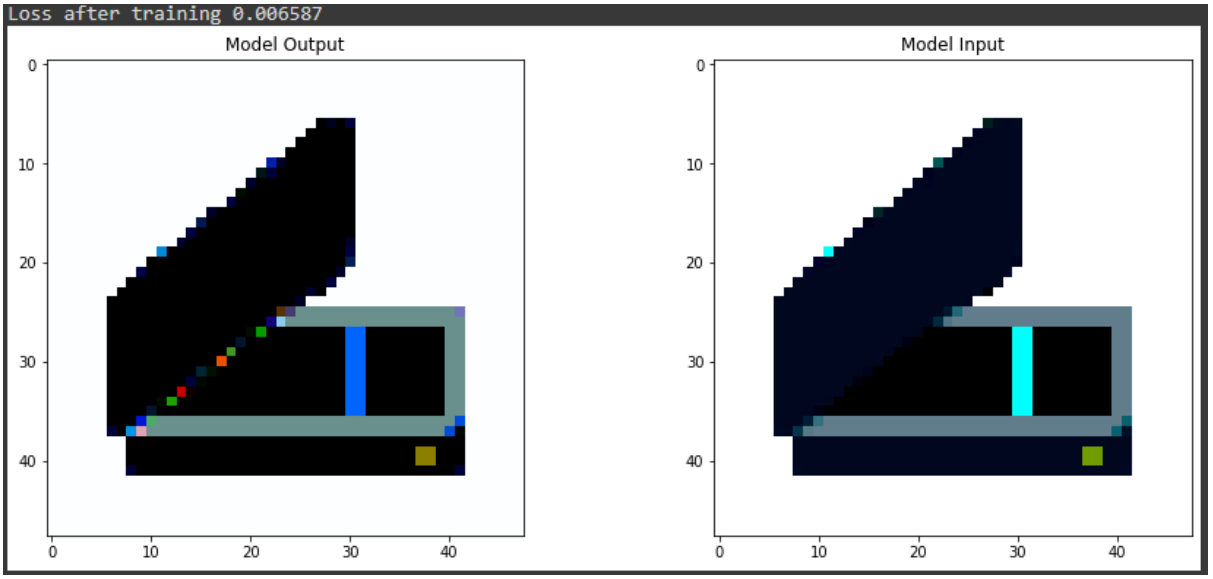
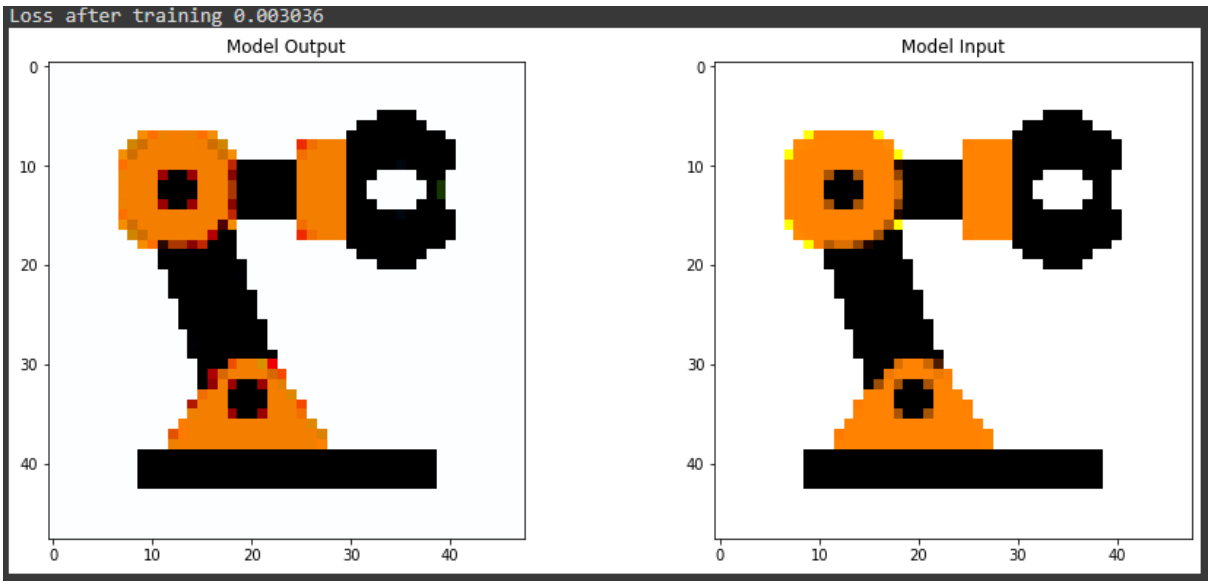
Cogntiv DL Research - Idan Vidra

Image Representation



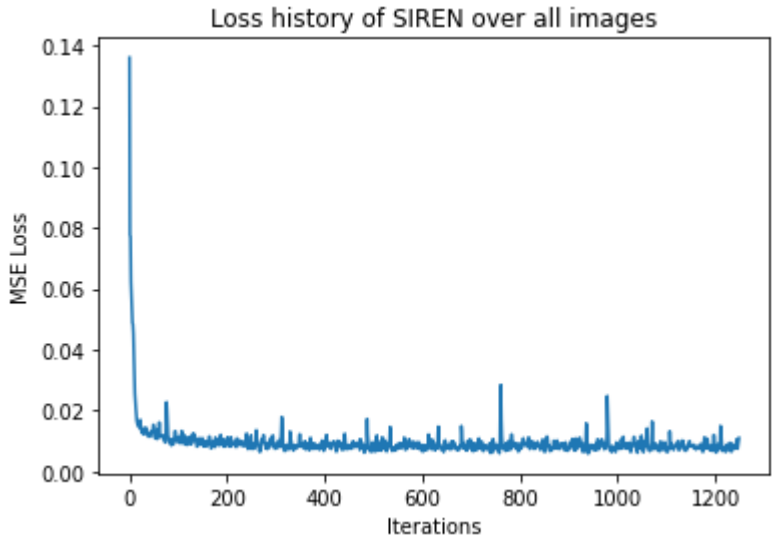




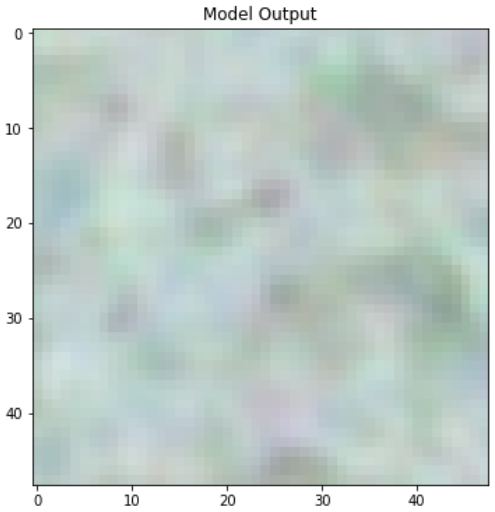
1. The SIREN model is a bit different then standard DL models we are accustomed to. It can be imagined as a way to represent an “natural” object (in our case an image) using a NN. The model's input is an (x,y) coordinate grid of an image and the model's output is a (r,g,b) value for each coordinate. Meaning that using the NN you can represent ONE image.

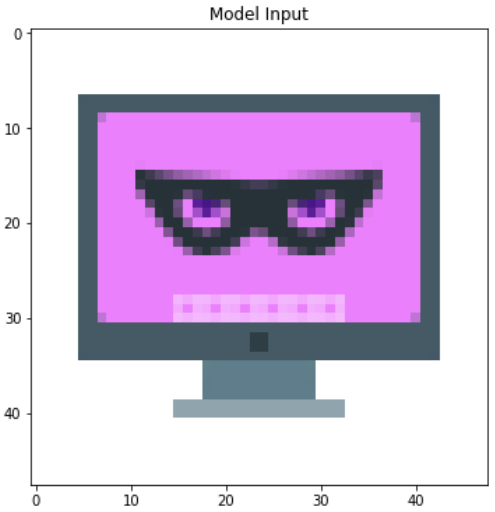
   One form of generalizing this task is to train the NN on several images - as the first task requires - meaning that one NN can represent several images.

   There are other forms of generalization in this task - another one could be using the continuous nature of the approximated function given by the trained NN - meaning, instead of asking the trained network to give the color value of a pixel location (for example (4,3)) we could get the color value of each location between the the pixels (for example (4.6,3.9)).
2. To track the network generalization during training I’ll show two measurments:
   a. Loss history over training periods:

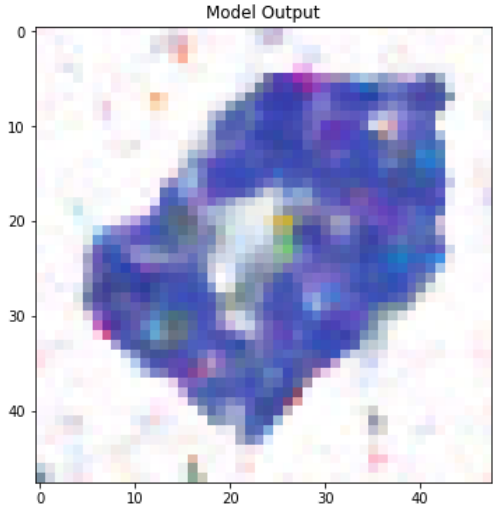


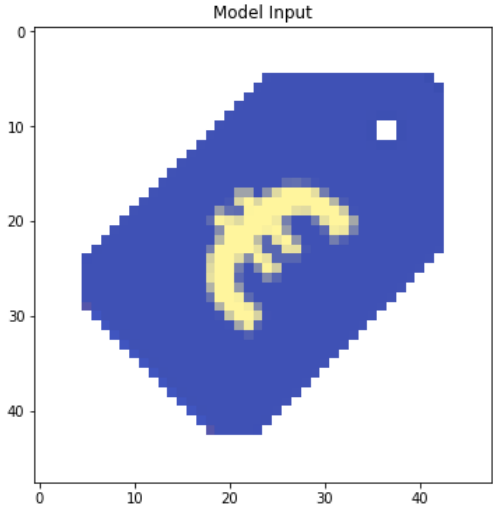
b. Printing evaluation images in different stages of the training process:
First training step:



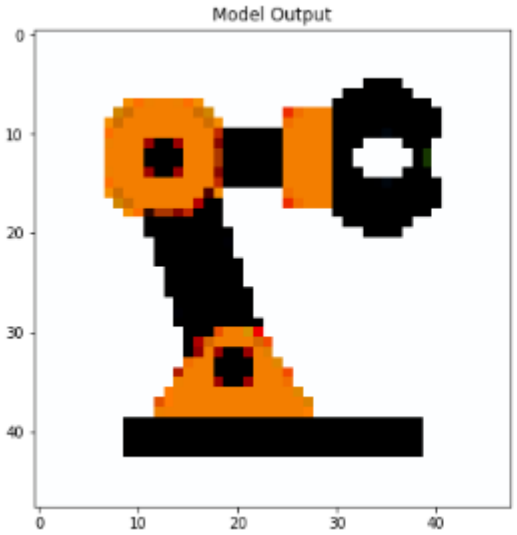


500th training step:





1000th training step:



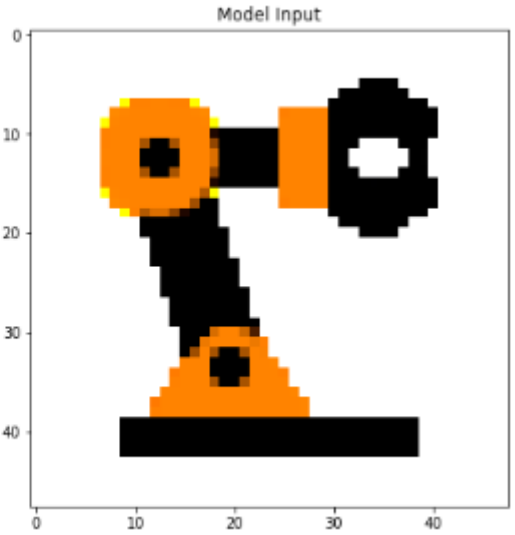
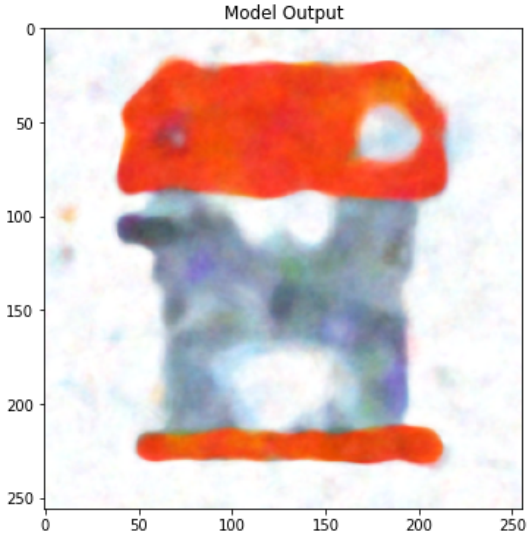
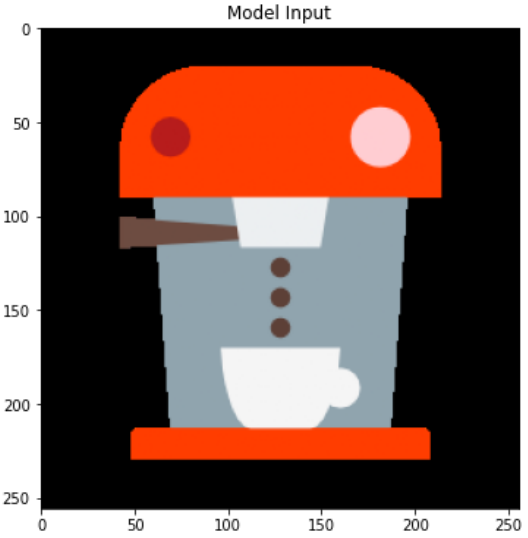
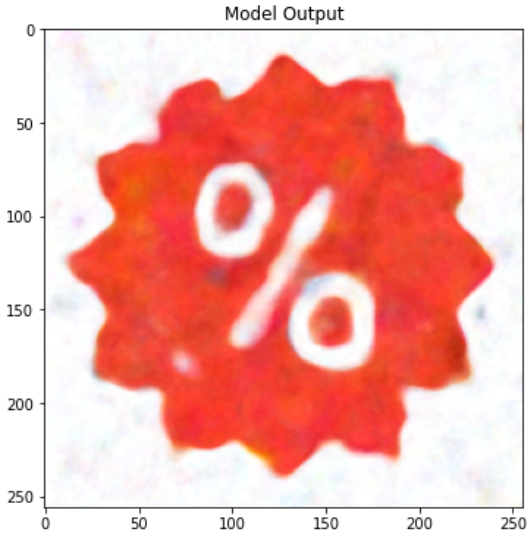


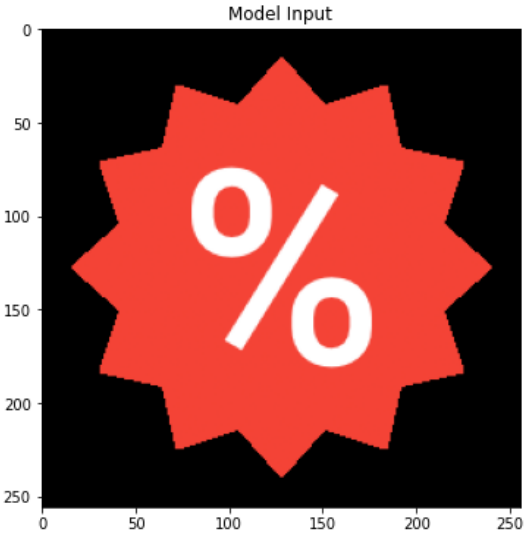
Image interpolation

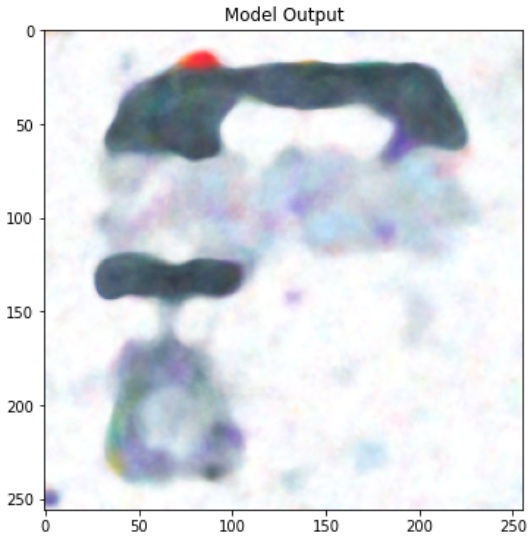
a. Three examples of upsampling:

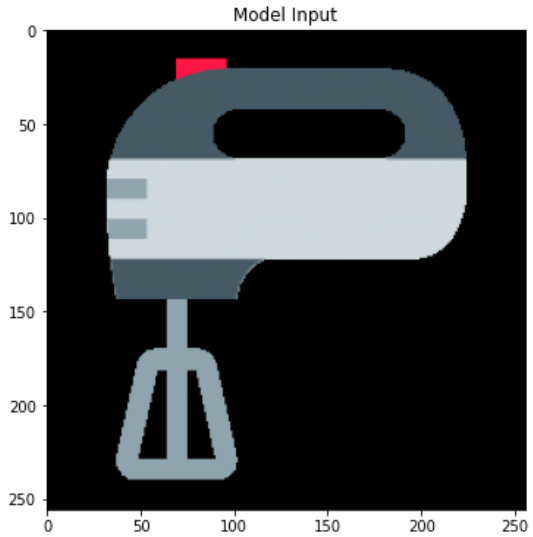










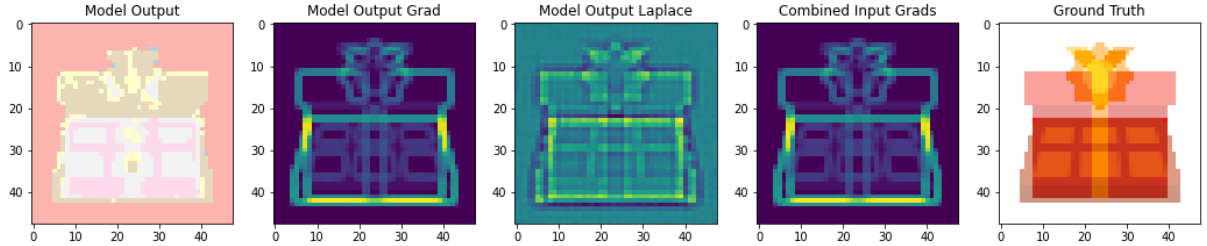


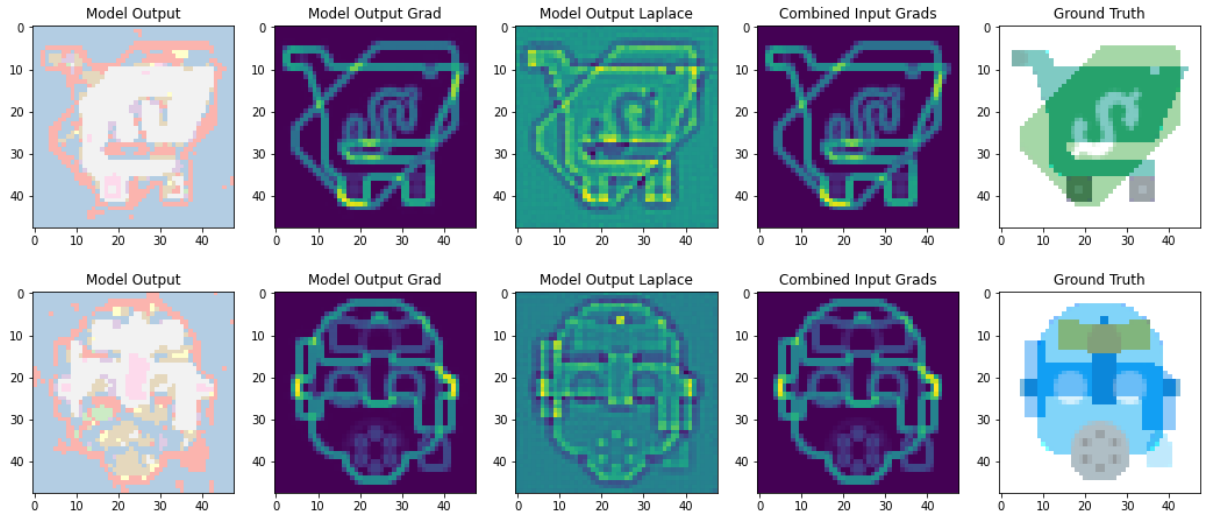
b. 1. Here are some similarity measure between representation:

- Cosine similarity: used to measure the “similarity” of two vectors - excellent measure when the representation is a vector. For example, when using LSTM’s the representation of a sentence could end up being a vector, and so you could check how ‘similar’ two sentences are (‘similar’ depends on what the network learned to into its representation context vector. it might be high level like semantic (queen is close to king) or it might be low level like similar letters (dog is close to god)).
- The measure I’ll be using is the mean squared error between the gradient of the learned representation and the actual combined gradient of the two images.
  MSE measures the average squared difference between the estimated values and the actual value - making sure the network tries to mimic the ground truth as closely as possible.
  The reason I fit the network to the **gradient** of the combined images is that the gradient “holds the most information” of both images. Linearly interpolating two images would return a “wash” of the two images with artifacts and color mixes.
  Thus, adding the gradients of the two images we want to combine, saves the areas that are richest with information in both images and allows the network to combine the images more coherently.

2. The best choice for images to merge would be images that don’t overlap where there is the most concentration of information - since the data is all logo situated in the middle of the screen I didn’t find any pairs that are better for this test then others.

3.





4. The shortcomings of my results are the following:
    - The coloring of the image does not match that of the original images - this is because of the method I chose to use as loss - MSE over the gradients.
      Since the network fits the output of the network to the gradient of the combined images it loses the bias of the original images (original coloring) and instead fits to the edges. Basically we get the same result as fitting to a grayscale combined image.
    - The shape is not intact and there are artifacts - this is due to the lower accuracy of the model.

Improved image interpolation

To achieve better results I would use image synthesis using conditional GAN's such as Odena et al. Since we have a very limited amount of data I would use an adaptive discriminator augmentation mechanism to ensure the discriminator won't overfit.

1. I would create a dataset of tuples for each combination of two images in the data and their naive combination (image 1, image 2, naive combination). (Naive combination is adding the images values pointwise and dividing by 2).
2. The adversarial network will receive (image 1, image 2) as input and generate a "fake" combination of the two images, and try to fool the discriminator with the tuple (image 1, image 2, combination).
3. The discriminator network would be trained to guess if a tuple is real or fake.

I think GANs can achieve more refined results because they offer more control than the SIRENs. If we consider architectures like Style GANs (Karras et al) that utilize:

- Ada-IN to control the learning at different resolutions, and so affect the end result on different levels (coarse, intermediate, fine).
- Adding stochastic variation so that the end result seems more natural (usually more relevant for natural data, moreso then icons).

These features of the different GAN architectures allow higher fidelity in the task of merging two images than the basic SIREN I've used.