Assignment 6 Part 1 - Adversarial Learning

Eitamar Saraf, Moshe Nasletashvili

* Net Architecture

The net architecture is similar to preceding assignments:

Conv2d->Conv2d->MaxPool->Dropout->Flatten>Linear>Dropout->Linear->Softmax

We used Cross Entropy as our loss and Accuracy as our metric.

Hyperparameters:

batch\_size = 128

#of epochs = 12

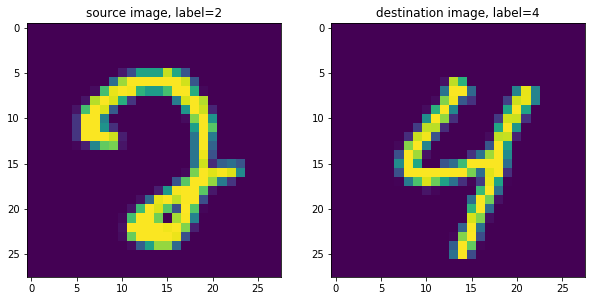
learning rate =0.1

lr\_decay = 1e-6

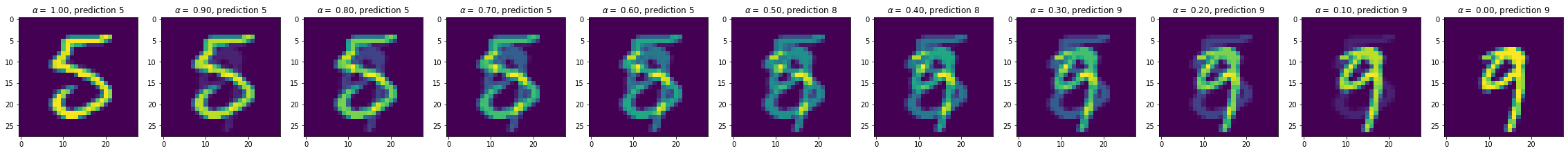
lr\_drop = 20

* Interpolating images

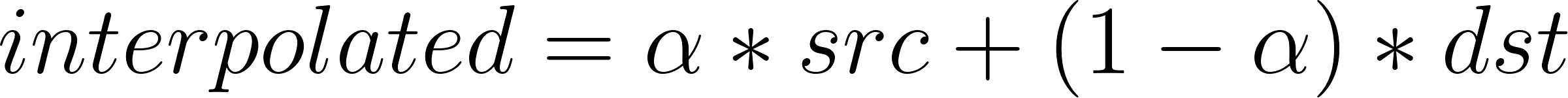
We randomly chose 2 images from the dataset with different labels:



Than, we interpolated the 2 images and checked the model prediction on the interpolated images. On every image title we can see how much of the image is of the source image (the constant [](https://www.codecogs.com/eqnedit.php?latex=%5Calpha#0) and the model prediction on the interpolated image.

:

The meaning of interpolating the 2 images is to go through the hyper-line that connects these 2 images. By means of formula:

[](https://www.codecogs.com/eqnedit.php?latex=%20interpolated%20%3D%20%5Calpha%20*%20src%20%2B%20(1-%5Calpha)%20*%20dst%20#0)

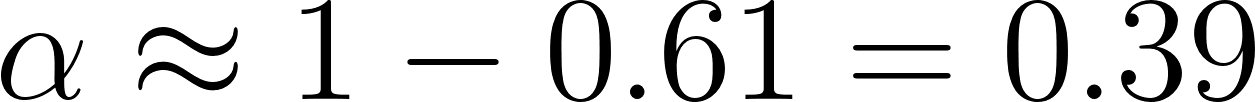
Where src is the source image and dst is the destination image.

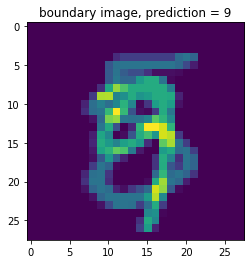
The interpolation operation is related to the binary search we perform on the Hop Skip Jump algorithm. When searching for the decision boundary we actually take a look at the interpolated version of the source and destination images, with different coefficients.

From the image above, it seems like somewhere around the middle of the hyper-line there is a change of label to the 8 zone, and then in the 30% near to the destination image there is a change of the label to the 9 zone.

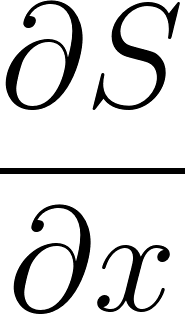
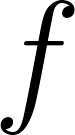
* Binary Search

We implemented binary search in order to find the decision boundary. We found that the decision boundary is got for:

[](https://www.codecogs.com/eqnedit.php?latex=%20%5Calpha%20%5Capprox%201-0.61%20%3D%200.39#0)

i.e, The boundary is where we have about 61% of the destination image and 39% of the source image.

* Calculating gradients

For the gradient calculation we recall that max operator uses as a “gradient router”. Hence, for calculating , we calculate the gradient of [](https://www.codecogs.com/eqnedit.php?latex=f#0) WRT [](https://www.codecogs.com/eqnedit.php?latex=x#0) and then subtract the gradient of the target label and the gradient of the label that gave maximum probability.