Denis Deronjic 1231829

# HOMEWORK 2

**INTRODUCTION**

In this homework we have to implement and test neural network models for solving unsupervised problems.
In this homework MNIST dataset is used.
The 1st task of this homework is the implementation of an autoencoder for image reconstruction. In this homework a convolutional autoencoder is implemented to solve this task with some regularizations and hyperparameters optimization.
The 2nd task of the homework is a denoising autoencoder. MNIST images for solving this task were noised with a gaussian noise with mean 0 and variance of 1.
The 3rd task asks to use the trained autoencoder for supervised classification task. This can be achieved by adding at the end of the decoder a fully connected layer with 10 neurons.
The 4th task is to explore a latent space structure using t-SNE and generate new samples from latent codes.
The 5th and last task is about implementation of a generative model. A GAN is implemented to achieve this task.

**CONVOLUTIONAL AUTOENCODER**

The autoencoder is implemented with encoder and decoder separately. The encoder's architecture is the following:

- Convolutional layer with 24 filters with the kernel size 3x3, stride 2 and padding 1
- Convolutional layer with 48 filters with the kernel size 3x3, stride 2 and padding 1
- Convolutional layer with 96 filters with the kernel size 3x3, stride 2 and padding 0
- Linear layer with 128 neurons
- And the latest layer with the encoding space dim number of neurons

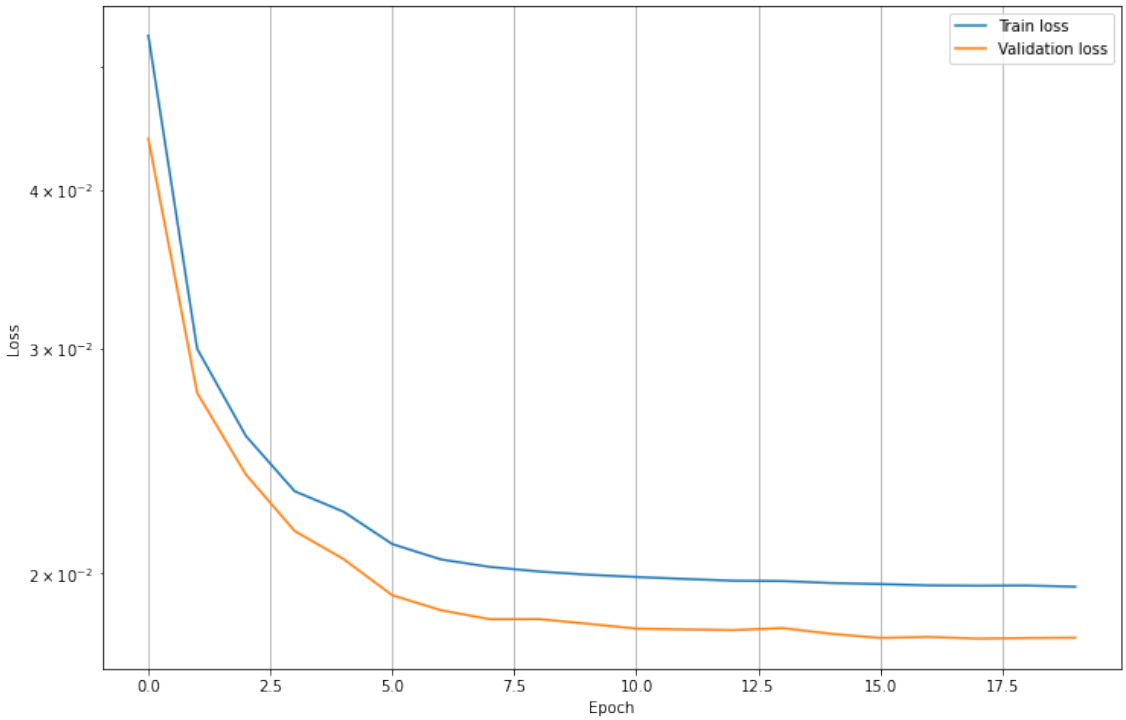
Dropout layers are also added in the network for regularization.
The encoding space dimension is an hyperparameter to optimize.
The activation function used for encoder and decoder is ReLU and Sigmoid in the output of the decoder.
Adam optimizer is chosen.
The decoder's architecture is similar to encoder's one, the order of the layers is reverted.
To tune this model, a grid search was run for the following hyperparameters:

- Learning rate: 0.001, 0.02
- Weight Decay: 0.0001, 0.005
- Epochs: 20
- Encoded space dim: 2, 10, 100

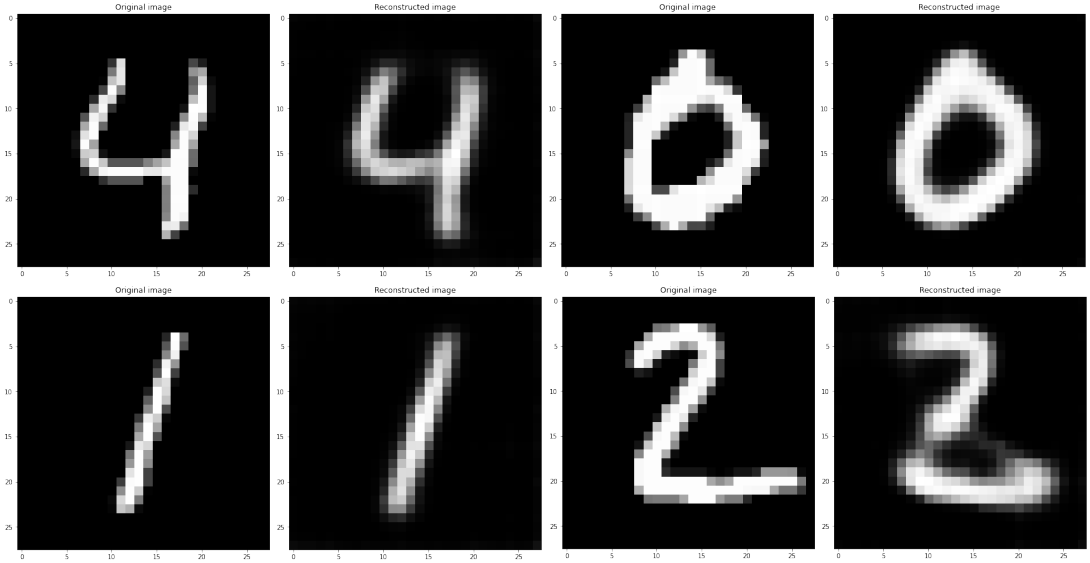
The best hyperparameters found are:

- Learning rate: 0.001
- Weight Decay: 0.0001
- Epochs: 20
- Encoded Space dim: 10

The train-val loss can be seen in the following image.



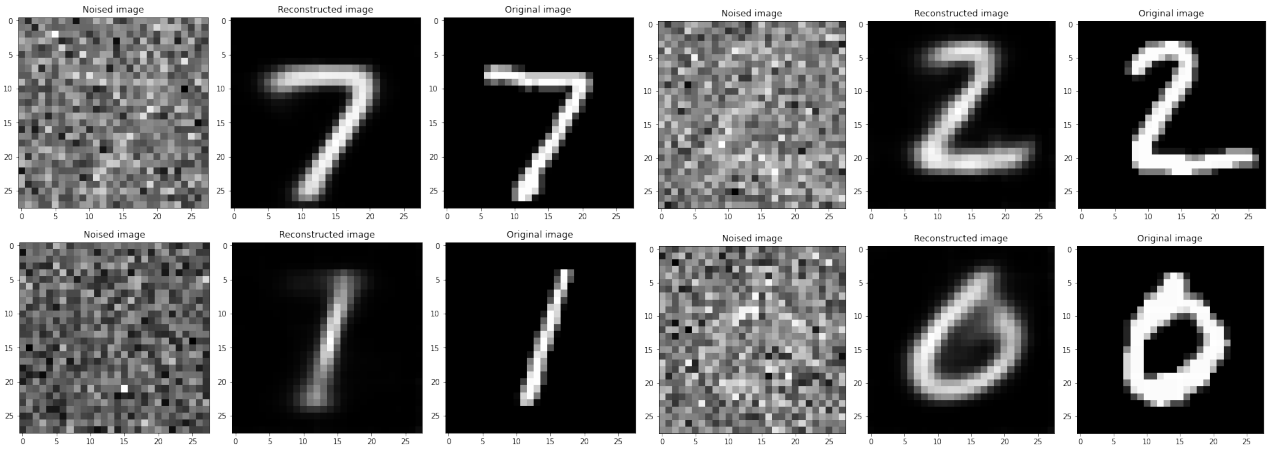

In the following some examples of reconstruction images.




**DENOISING AUTOENCODER**

The denoising autoencoder is trained with training images with an addition of gaussian noise of mean 0 and variance 1. The autoencoder's output is then compared with the original version of the image and the loss is computed with a mean squared error. The architecture of the denoising autoencoder is the same as the previous one. The only difference here is that the autoencoder is trained with noised images.

The following images shows some examples of denoised images. At the left the noised image, in the center the denoised image and on the left the original image.



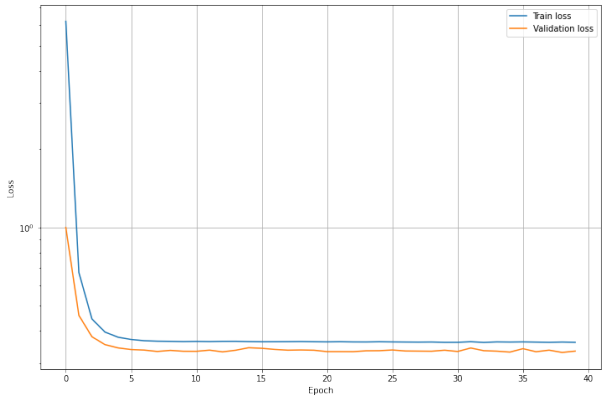

**SUPERVISED CLASSIFICATION TASK WITH A TRAINED ENCODER**

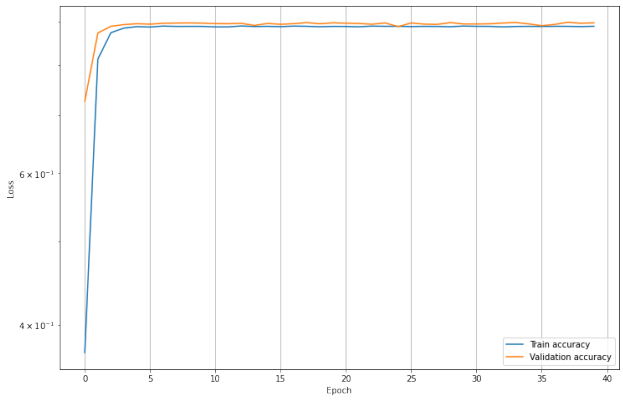
A supervised classification task with unsupervised pretraining can be very useful when we have a large dataset but only some of them are labeled. In this case we can train an autoencoder that stores the important features of the images learned. The decoder is removed and in place of this, a fully connected layer is added as output. The fully connected layer should learn values stored in the encoed space given a label of the image.

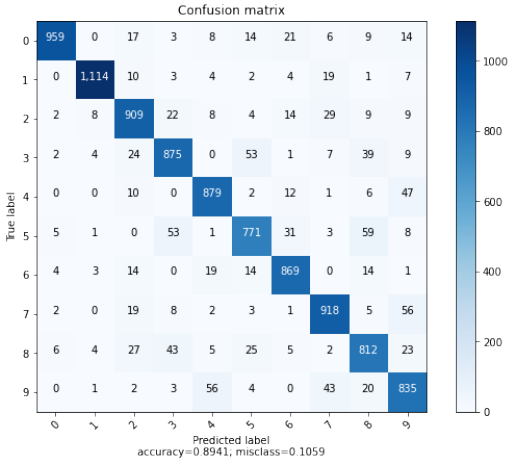
In this case, we have the entirely labeled dataset, so we are going to learn only the fully connected layer with 10 neurons one per class, where the encoder is maintained as it is without any further training.

The model achieved 89.41% of accuracy where in the first homework the best model achieved an accuracy of 99.12%. This accuracy can be improved with some hyperparameters optimization and/or adding more linear layers.

In the following you can see the train-val loss and accuracy plot and the confusion matrix



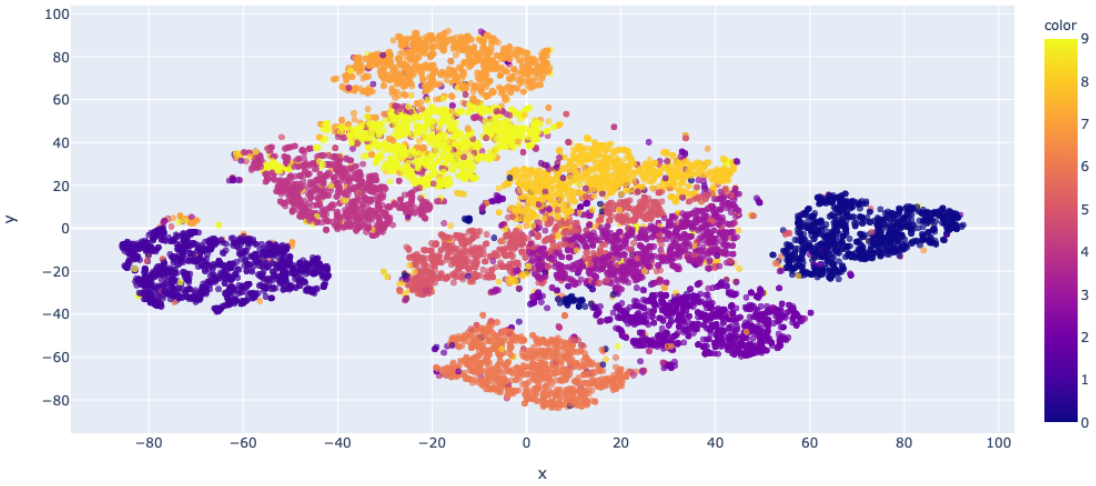



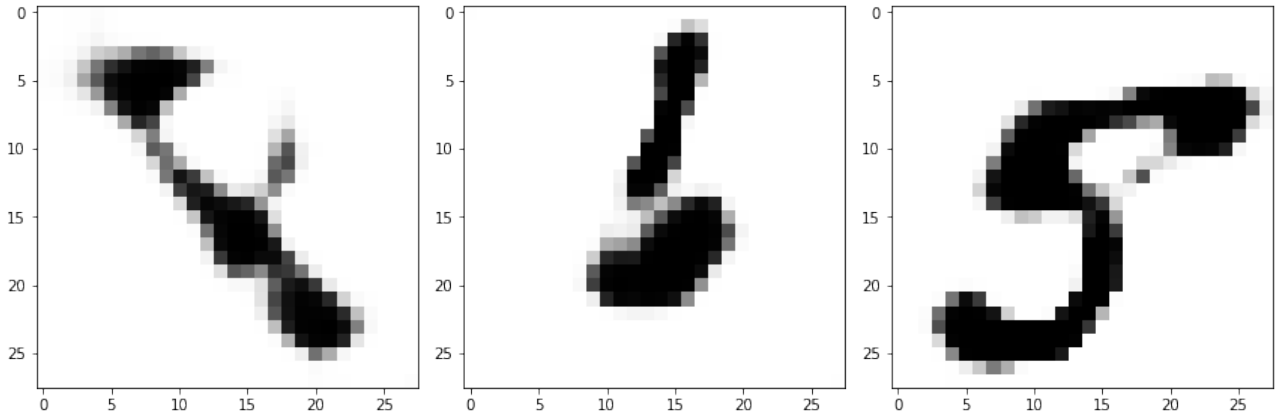
**LATENT SPACE AND SAMPLE GENERATION**

In this part of the homework we are asked to plot the encoded space dim in a 2d plane. The dimension of the encoded space dim of 10 dimensions is reduced in 2 dimensions using a PCA. The latent space representation can help us to show whether or not there's a good clustering of our encoded samples.

In the image below you can see the encoded clustering. The clustering is good, we can see distinct and distinguishable regions for every label. Some clusters in the center though overlap, there could be an uncertainty for some values.




This image representation can help us to easily see which values are required for the encoder space to generate the image samples. The following images show an example of generated images through random values given to the latent representation vector. Unfortunately giving random values to the latent representation vector doesn't guarantee that good images are generated. As you can see only the third image is recognizable as a number.



**GENERATIVE ADVERSIAL NETWORK**

A GAN network is implemented to achieve the sample generation task.

To train this network, train set and test set are merged together in order to have as many samples as possible. A test set could be helpful to test the discriminator's capacity to distinguish fake images vs real images in a dataset that it never seen. However we're not interested on test this, so we can avoid splitting the train-test set.  The GAN is composed of 2 networks, a generator that

generates an image from a latent vector code of dimension 100 and a discriminator. Both are convolutional networks.
The architecture of the generator is the following:

- Linear input layer of 100 neurons
- A unflatten layer that maps a linear layer of 128*3*3 neurons to a tensor image of (128, 3, 3).
- A convolutional transpose layer with 64 filters with a 3x3 kernel, stride 2 and output padding 0
- A convolutional transpose layer with 32 filters with a 3x3 kernel, stride 2, padding 1 and output padding 1
- A convolutional transpose layer with 1 filter with a 3x3 kernel, stride 2, padding 1 and output padding 1
- The activation function between there layers is SELU and a Sigmoid is used for the output

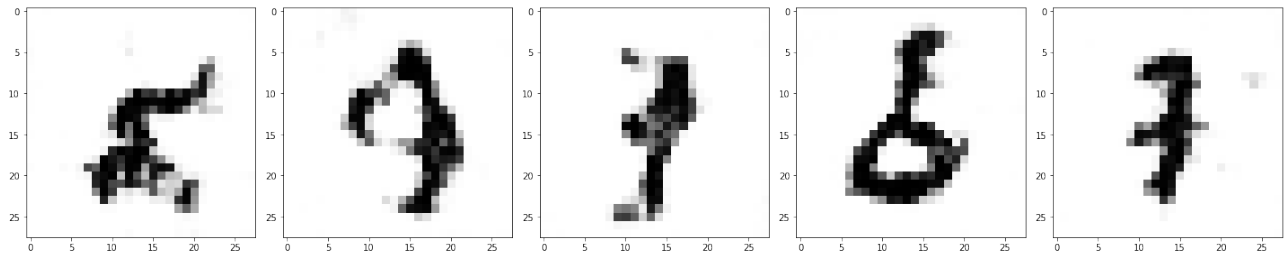
The architecture of the discriminator is the following:

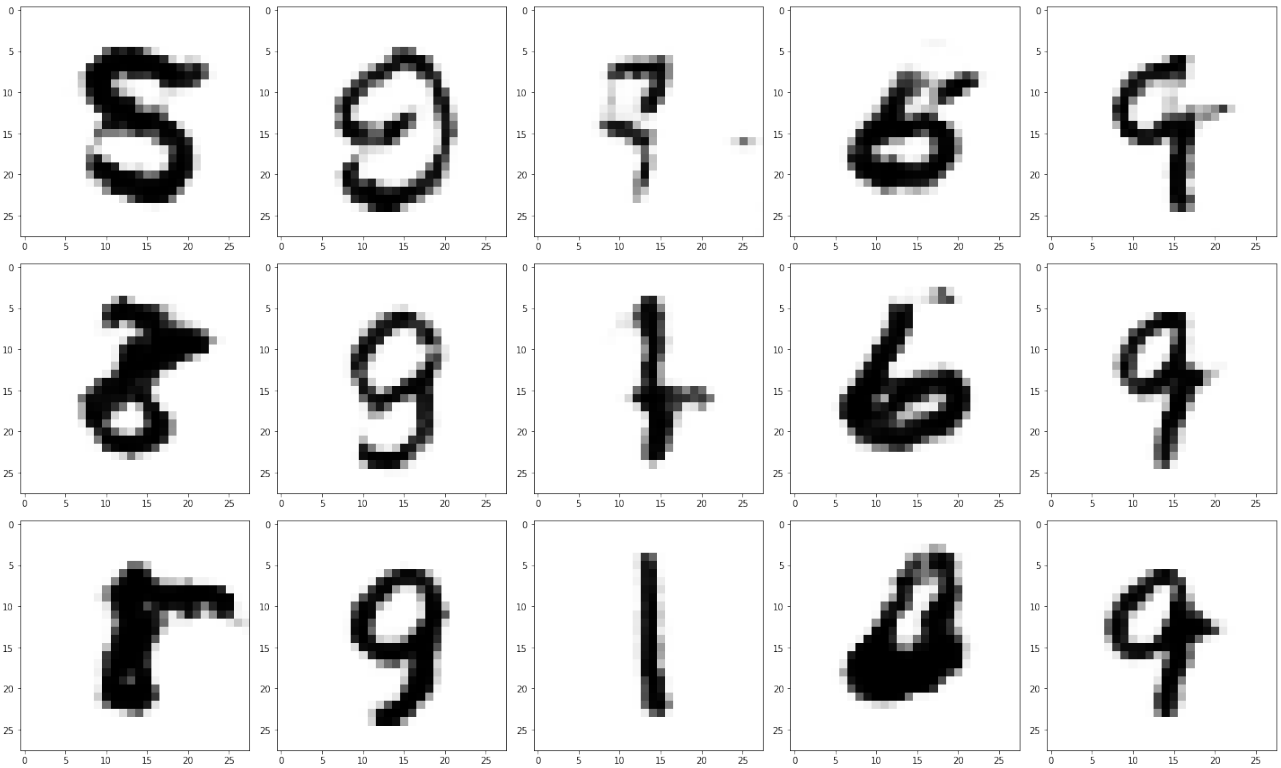
- A convolutional layer with 64 filters with a 3x3 kernel, stride 2 and padding 1
- A convolutional layer with 128 filters with a 3x3 kernel, stride 2 and padding 1
- A convolutional layer with 256 filters with a 3x3 kernel, stride 2 and padding 0
- A linear layer with output of dimension 1
- The activation function is the LeakyRelu with negative slope of 0.2 and a Sigmoid is used for the output

The training process was a bit difficult since it was not a classical training such as in the other implemented networks. Each network needs a separate backpropagation phase.
The loss used is the BinaryCrossEntropy.
The optimizer used for discriminator and generator is Adam. Two instances of Adam optimizer are implemented for each network. In this case both optimizers have the same learning rate (0.02) and beta parameters (0.5, 0.999).
A batch size of 200 is used and the number of epochs for training is 15.
The function to minimize is the following

$$\min_{\theta_g} \max_{\theta_d} V(D, G) = \mathbb{E}_{x \sim p_{data}(x)}[log D_{\theta_d}(x)] + \mathbb{E}_{z \sim p_z(z)}[log(1 - D_{\theta_d}(G_{\theta_g}(z)))]$$

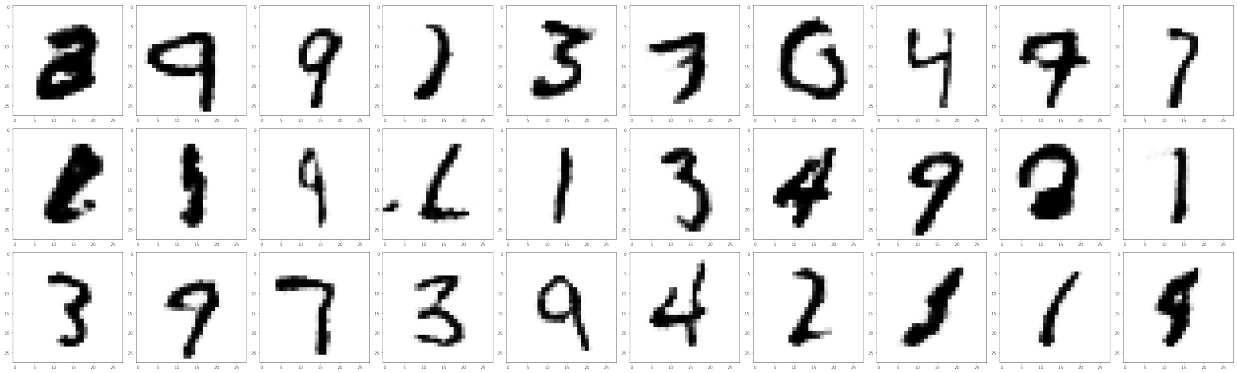
In the training phase the discriminator and the generator have opposite goals, the discriminator tries to distinguish fake images vs real images and the generator tries to fool the discriminator generating images that look real.
Unfortunately to have a good trained model, more than one training phase should be done.
Some examples of generated images during 1st epoch, 5th epoch, 10th epoch and 15th epoch





As we can see, some images tend to degrade duting the training where in the latest epoch, two digits are not recognizable.

In the following you can see some examples of images generated after the training phase. As we can see, the GAN learned pretty well on generating some kind of digits (1, 3, 4, 7, 9), but other digits don't have a clear shape.



One of the reasons why is difficult to train a GAN is caused by the capacity of the network to generate pretty well images of some kind of labels where other labels have barely good and clear shapes.