**Generative Adversarial Networks**

**Group Members:**

➢ Huang Xie, 281685, huang.xie@tuni.fi

➢Liang Fang, 281684, liang.fang@tuni.fi

1. *Understanding and topologies of GANs and DCGANs.*

* *GANs*

The key components of GANs are generator and discriminator that are responsible for generating images which are like real ones and discriminating if an image is real or fake respectively. Moreover, a generator which can generate "real" enough images and a discriminator which has strong discrimination capability would be obtained through the adversarial training, in which training is finished when the discriminator is not capable to discriminate fake images.

In the script of vanilla GANs implementation, the networks are straightforward shown in Fig. 1: for the discriminator model, images from MNIST dataset are reshaped into 784-dimension vector and input into discriminator model, the binary classification prediction (probability) is obtained through two fully connected layers with Relu and a sigmoid after the last fully connected layer; While for the generator model, a 100-dimension noise vector is as the input into the model, the image vector which is 784-dimension would be obtained through two fully connected layers and sigmoid activation function as well, the final image is shown by reshaping in to 28\*28.

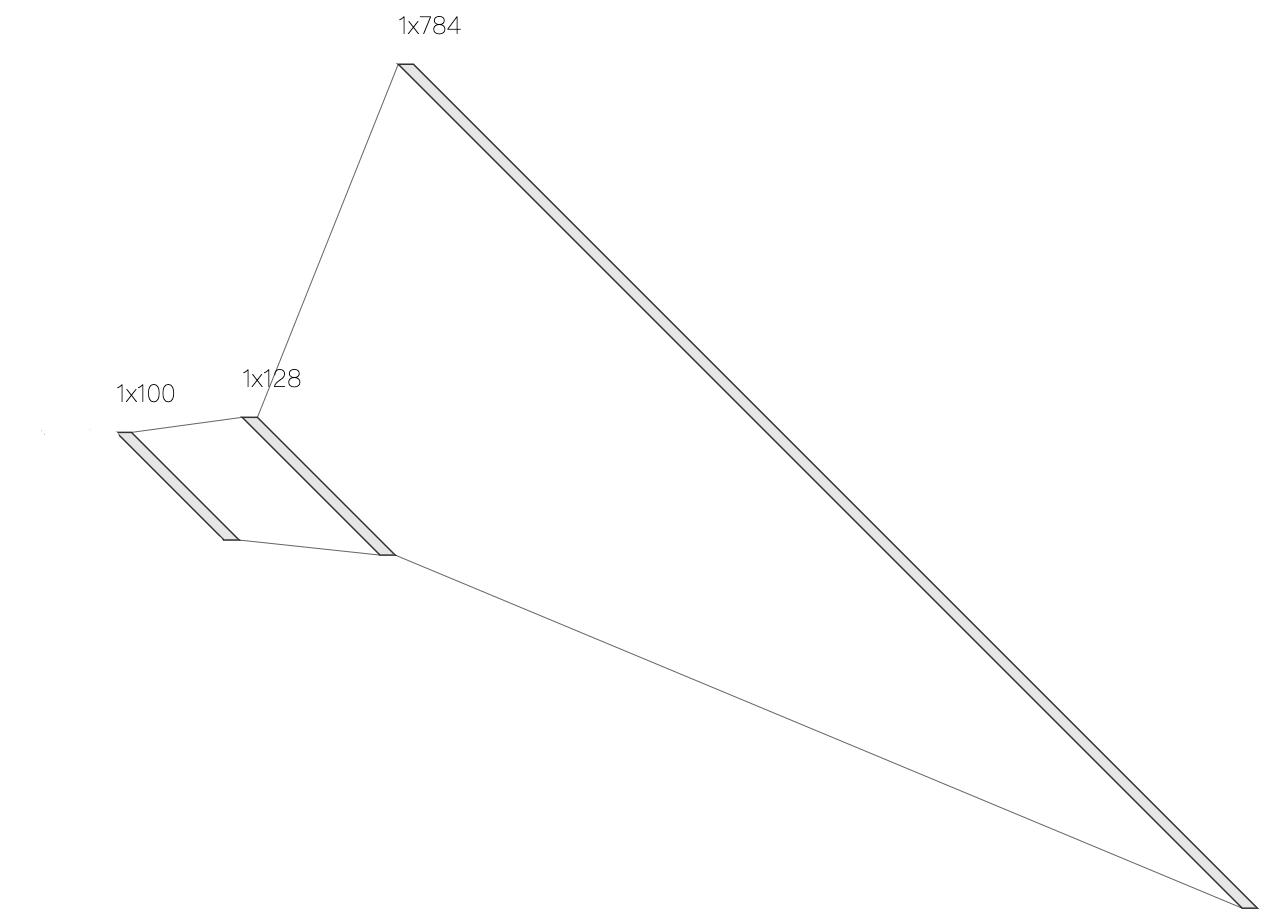
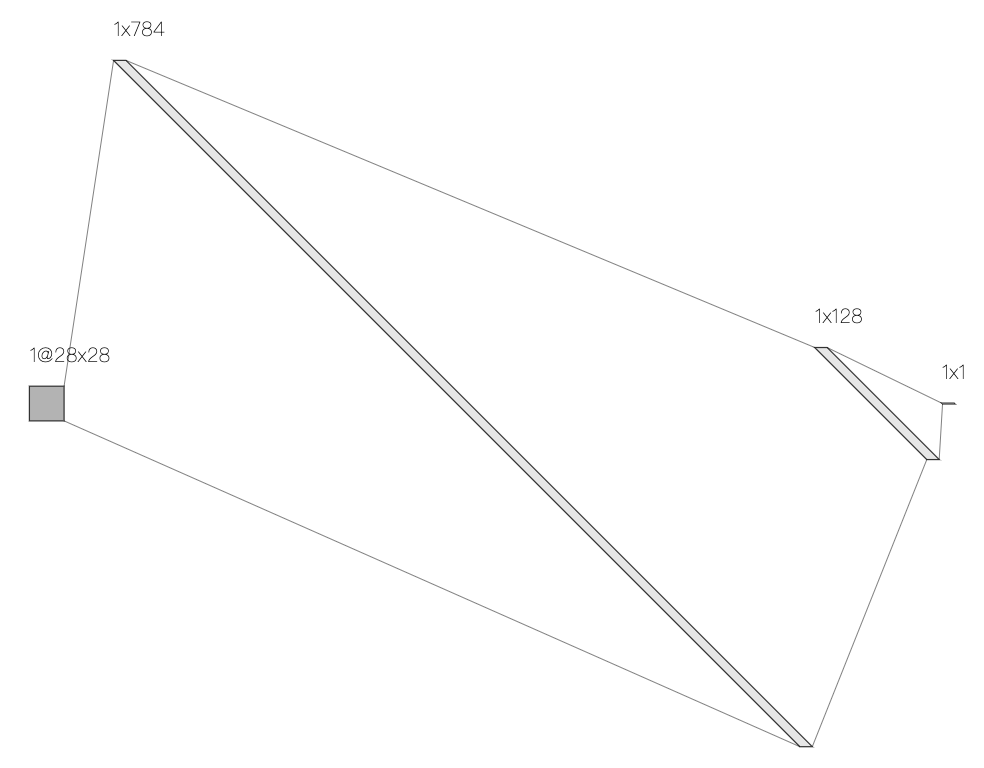


Fig. 1 network structures of discriminator and generator for GAN from left to right

* *DCGANs*

DCGANs is an extension of vanilla GANs. The main idea is the same as GANs, that is, DCGANs includes a generator and a discriminator model as well, but it replaces max pooling and fully connected layers with convolutional layers in these two models, and the down-sampling and up-sampling is processed with convolutional stride and transposed convolution instead. Besides, batch normalization layer is applied for every feature map to normalize the value.

In the script of DCGANs implementation, the networks are shown in Fig. 2: for the discriminator model, the original image from MNIST dataset is down-sampled gradually through 3 convolutional layers with 5\*5 filter size and stride 2, then reshape the feature maps 4\*4\*256 into a 4096-dimension vector and pass to the last fully connected layer with sigmoid to obtain the binary classification prediction (probability); While, for the generator model, 7\*7\*256 feature maps are reshaped from a 12544-dimension vector, then up-sample it gradually with transpose convolution operation and obtain the original size of image with 1\*1 filter size.

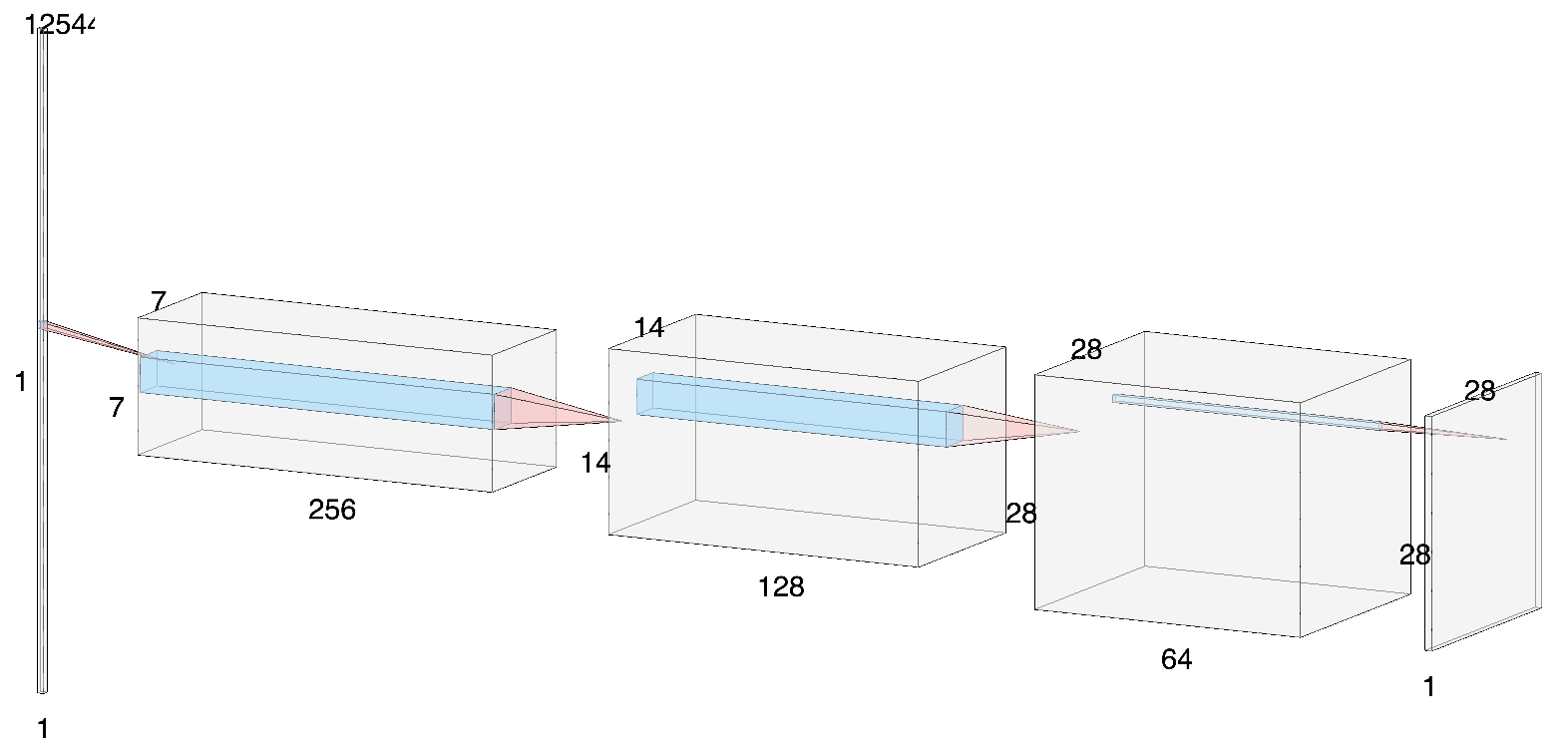
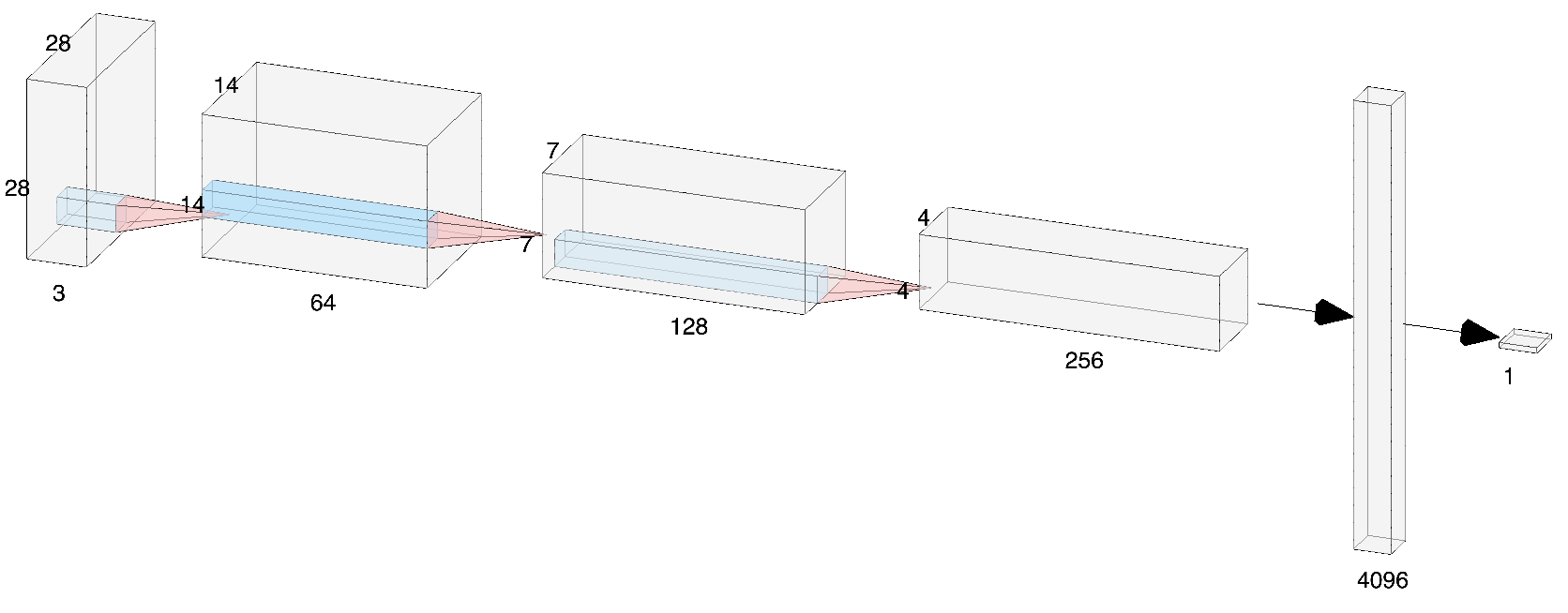


Fig. 2 network structures of discriminator and generator for GCGAN from up to down

1. *Pros and cons of GANs and DCGANs.*

* *GANs*

Pros:

1. The structure of vanilla GANs is simple and easy to implement, and it is not necessary to focus on the shape of intermediate components since feature is processed with vector and fully connected layer.

Cons:

1. Vanilla GANs ignores the spatial information since all operation is done with feature vector so that vanilla GANs is not good at handling complex images or video frames.
2. Computation complexity would be a challengeable problem when the size of target image is large since fully connected layer is the basic component of vanilla GANs, weights of neurons are not shared with each other like convolutional layer.

* *DCGANs*

Pros:

1. Convolutional layers find spatial relation within an image so that DCGANs would likely be suitable for images or video frames. Extracted feature from discriminator can be more representative with convolutional layers, and the classification and image generation would be easier to achieve a good result.
2. Computation complexity is much lower compared to fully connected layers with the same size of input and output since weights of filter kernels are shared within an image.

Cons:

1. Structure of network should be implemented with caution since the shape of output after every layer depends on previous up-sampling or down-sampling operation.

Fig. 3 shows two generated images from vanilla GANs and DCGANs: as we can see, the image from DCGANs has higher quality with less noisy point compared to the one from vanilla GANs.

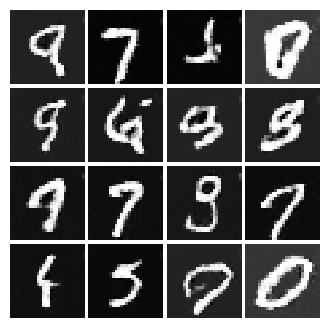
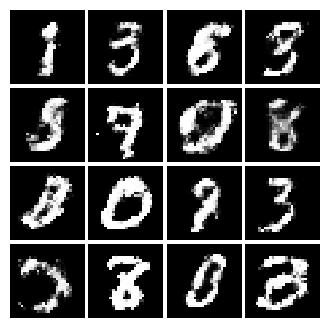


Fig. 3 Generated images with vanilla GANs and DCGANs from left to right

1. *Experiment processing.*

* *vanilla\_gan.py*

1. Generator and discriminator model are built explained previously.
2. Fake image **self.img\_fake** is generated by generator, probability of fake and real images **self.D\_fake** and **self.D** are yielded by discriminator.
3. Compute the losses of fake and real images from discriminator **self.D\_loss\_real** and **self.D\_loss\_fake** and combine them as the final loss **self.D\_loss** for discriminator; Compute the loss for generator **self.G\_loss**.
4. Optimize generator and discriminator with Adam optimizer, and update them iteratively.
5. Run the training process for 100000 epochs and yield a generated image every 1000 epochs.

* *dcgan.py*

1. Generator and discriminator model are built explained previously. Specifically, add batch normalization layers for feature maps; Use LeakyRelu to replace Relu for generator and discriminator. Use tanh as the final activation function for generator instead of sigmoid.
2. Following steps are the same as that in vanilla\_gan.py.