MŰEGYETEM 1782

DEPARTMENT OF TELECOMMUNICATION AND MEDIA INFORMATICS

BUDAPEST UNIVERSITY OF TECHNOLOGY AND ECONOMICS

Realistic ultrasound tongue image synthesis using Generative Adversarial Networks

Written by: Nadia HAJJEJ

Supervised by : Dr. Csapo Tamas Gabor

29/10/2018

Absztrakt

Napjainkban sokféle nyelvészeti kutatásban alkalmazzák a 2D ultrahang képeket és az artikulációs adatokat. Ahhoz, hogy a kidolgozott módszereket ellenőrizni lehessen, sok képből álló nagyméretű adatbázisokra van szükség. Emiatt az ultrahang képek generálása hozzájárulhat a beszédkutatás számos ágához [1].

Ahhoz, hogy egy ilyen adatbázist létrehozzunk, a generatív versengő hálózatokat (Generative Adversarial Network, GAN) választottuk, melyek a gépi tanulás nemellenőrzött tanítás családjába tartoznak, és képesek bármilyen adat eloszlást modellezni, illetve ahhoz hasonló adatokat létrehozni. A GAN folyamata lényegében két játékossal modellezhető. A generátor az a játékos, amely a tanító adat eloszlása alapján új mintákat generál. A másik játékos a diszkriminátor, amely a valódi és hamis képeket próbálja megkülönböztetni. A diszkriminátor tanítási folyamata általában hagyományos ellenőrzött tanulás, melynek végeredményeként meg tudja különböztetni a két osztályt (valódi vagy hamis). A generátort úgy tanítjuk, hogy be tudja csapni a diszkriminátort.

Az adatbázis nyelvultrahang képekből áll. Ezeket egy ultrahangos szoftverrel gyűjtöttük [3]. Ezek a fájlok nyers ultrahang képeket tartalmaznak egymás után. Minden pixel 1 bájtos előjel nélküli egész számként tárolódik, ami valójában a szürkeárnyalatos pixel intenzitás. A nyers ultrahang képekből előállíthatjuk a nyelvultrahang videót, amely a nyelv mozgását mutatja. Esetünkben a nyers, szkennelési vonalakból álló képekkel dolgozunk. A felvételek után egy adatbázist készítettünk, amely 27925 nyers, 64*842 dimenziójú képből áll. A projekthez mély konvolúciós hálózatot választottunk a generátorhoz és diszkriminátorhoz is. A modellben a rejtett réteget számát kilencnek választottuk a generátor és diszkriminátor esetén is. A hiperparaméterek közül a batch méret 64, valamint az epochok száma 25. Minden 100 iteráció után 64 ultrahang képet generáltunk.

Ahhoz, hogy a generált nyers képeket értékeljük, egy internetes tesztet készítettünk. A tesztben 100 minta volt, valós és generált képekből. A résztvevők feladata az volt, hogy mindegyik képhez egy számot rendeljenek (0-100%), anélkül, hogy tudnák, melyik volt eredeti és melyik generált. Az eredmények szerint a ’rossz’ generált képeket 36.1%-ra, a ’jó’ generált képeket (amelyek a teljes tanításból származnak), 70.73%-ra, míg az eredeti nyelvultrahang képeket 90.1%-ra értékelték. Az eredmények elemzése alapján azt mondhatjuk, hogy a GAN-okkal hatékonyan tudtunk képeket generálni, és sikerült becsapni az embereket.

Az eredmények hasznosak lehetnek az ultrahang képsorozatban a következő képkocka becsléséhez, vagy a képeken a nyelv kontúrjának detektálásához [1].

Abstract

Nowadays, a large amount of linguistic studies is applied for analyzing the 2D ultrasound images and especially the articulatory data, which requires a huge dataset large number of images to objectively assess the quality of developed methods. Thus, the generation of ultrasound images can facilitate the exploitation of higher-level representation of the tongue in a variety of applications in speech research [1].

In order to build the dataset, we have chosen the Generative Adversarial Networks (GANs) [2] as a branch of unsupervised learning techniques in machine learning which are able to mimic any data distribution and generate data like it.

. For our project, we have chosen to use the deep convolutional Generative Adversarial Networks. In our model, we have fixed the number of hidden layers to nine for both discriminator and generator. As hyperparamater, we have fixed 64 as batch size and 25 for the number of epochs. After every 100 iterations, we generated 64 ultrasound images.

In order to assess the generated raw images, we have developed an internet-based test. In this test, we have 100 samples made of real and generated images. The participants task was to accord a number (between 0-100%) for each image without any knowledge about their origin whether they are real or generated ones. According to the results, the 'bad' generated images resulted in 36.1%, the 'good' generated images (from a full training) resulted in 70.73%, whereas real ultrasound images achieved 90.1%. Analyzing the result, we can say that the GANs are very efficient in image generation and deceive humans.

The results can be useful for predicting the next frame in an ultrasound image sequence or for motion detection of tongue contours within images [1]

Contents

I. Introduction:

Ever since computers were developed, scientists and engineers thought of artificially intelligent systems that work and react like humans. In the past decades, the increase of generally available computational power provided a helping hand for developing fast learning machines, whereas the internet supplied an enormous amount of data for training. These two developments boosted the research on smart self-learning systems, with neural networks among the most promising techniques.

Ultrasound tongue imaging (UTI) is a technique suitable for the acquisition of articulatory data. Although a large number of linguistic studies are applying 2D ultrasound, there is a lack of freely available databases with a large number of images. The realistic synthesize of ultrasound images can be a starting point for exploiting the higher-level representation of the tongue in a variety of applications in speech research [1].

In our research, we are interested in generative adversarial networks as a branch of unsupervised learning techniques in machine learning which are able to mimic any data distribution and generate data like it. Actually, since their introduction in 2014 [2], they have proven a huge potential to automatically learn the natural features of a dataset, whether categories or dimensions or something else entirely. As a task we have chosen to adapt a generative adversarial network to synthesis tongue ultrasound images.

In this paper we are going to detail our project. Thus, we are going to start with an overview of the history of the generative model. Second, we present in section 2 the concept of generative adversarial network and its different applications. Then, in section 3, we will detail our adopted model to present, in section 4 our experimental result and our evaluation method. Finally, we will finish this report by the conclusion and possible future direction

II. Generative models

Generative model can be defined as networks which are able to learn data distribution using unsupervised learning and they have achieved a great success in just few years. In general, the generative model goal is to estimate or to learn the data distribution of the training data and generate new data points with some variations by modelling a distribution which is as much as possible close to the real data distribution. Most of generative model use the maximum likelihood to define the model that estimates the parameterized probability distribution [3]. Not all the generative networks use the maximum likelihood but we will restrict our attention to those using it as it is adopted by the GANs.

The maximum likelihood goal is to define a model that present the estimation of a probability distribution which is parameterized by parameters θ. We can define then the likelihood as the probability that the model assigns to the training data:

$$\prod_i^N p_{model}(x^{(i)},\theta) \text{ , for a dataset containing N training examples } x^{(i)}. \quad (1)$$

As we have said we will focus on generative networks using the ML estimation. Those models differ mainly in the approximation of maximum likelihood. Hence, we have two types:

- Models that aim to represent the probability distribution over the space where the data lies explicitly
- Models that interact implicitly with the probability distribution and try to generate samples from it.

In figure (1) we illustrate the classification of the generative models:

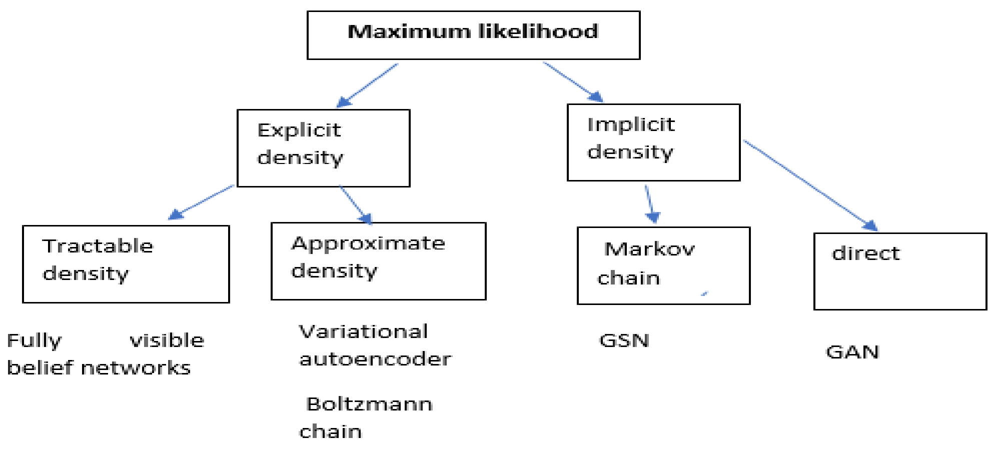


Figure 1 generative model classification

1. Explicit density estimation

For the models that define an explicit density function, the maximization of the likelihood is straightforward. We define the model's density function and connect it to the expression for the likelihood, and follow the gradient uphill. The main difficulty present in explicit density models is designing a model that can capture all the complexity of the data to be generated while still maintaining computational tractability. There are two different strategies used to confront this challenge:

- careful construction of models whose structure guarantees their tractability
- models that admit tractable approximations to the likelihood and its gradients

1.1. The fully visible belief networks (FVBN)

The first type can be defined as the fully visible belief networks [4] which use the chain rule of probability to decompose a probability distribution over an n-dimensional vector x into a product of one-dimensional probability distributions:

$$p_{model} = \prod_i^N p_{model}(x_i \,|x_1,x_2,x_3 \ldots .x_{i-1})$$

We can easily, notice that the FVBNs generate samples by treating one input at a time. For example, in case of an image, the latter will be generated pixel by pixel so the cost of generating a sample is O(n). In modern FVBNs, such as WaveNet [5], the distribution over each x_i is computed by a deep neural network, so each of these n steps involves a nontrivial amount of computation. Moreover, these steps cannot be parallelized which consist the main drawback of this method.

In figure (2) we illustrate example of generated images using pixel RNN [6]

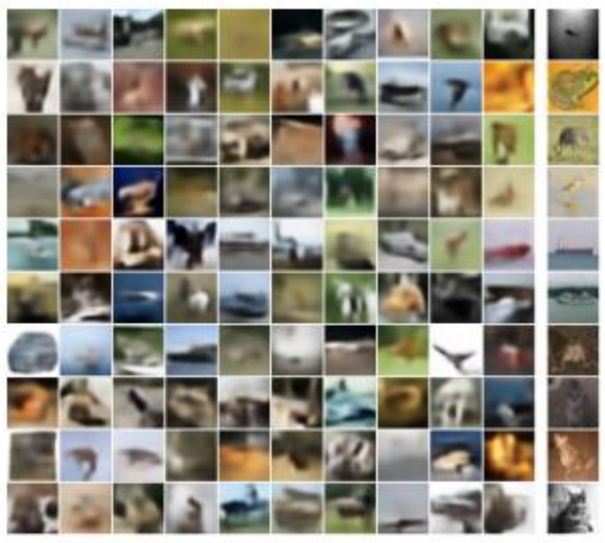


Figure 2 generated CIFAR images by pixel RNN

1.2. Explicit networks

In order to remedy the FVBNs disadvantages, the explicit density approximation network was built where we find the markov chain that uses a stochastic approximation and the variational methods which uses deterministic one

1.2.1. Variational approximation

The variational approximation tends to maximise the log of the model. The most known model is the variational autoencoder (VAE) [7]. The VAE learns to reconstruct the image using hidden variable z which should have a lower dimension than the original image x and we general set their prior probability to a nice gaussian. So; the training process is starts by computing the mean and the covariance of z with respect to x and according to estimated z we compute the covariance and mean of the new images x’. This process can be summarized in the following diagram.

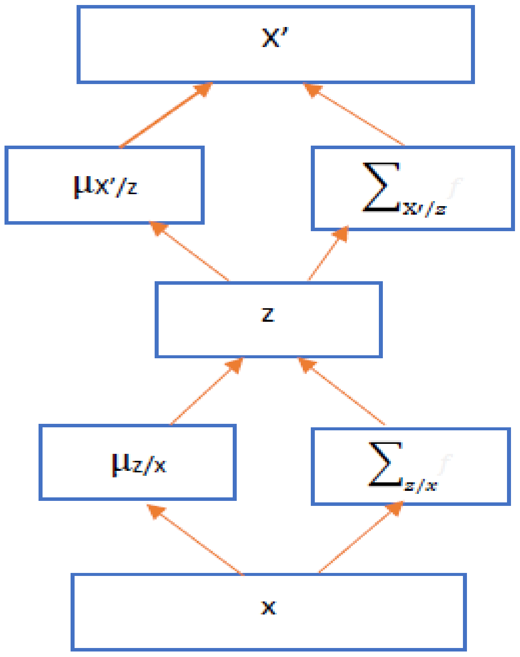


Figure 3 VAE process

The VAE are much faster than the FVBNs and they show a great performance in obtaining a good likelihood. Thus, they are later used to generate samples but the latter are very bad. The following figure (4) illustrate some images generated by VAE

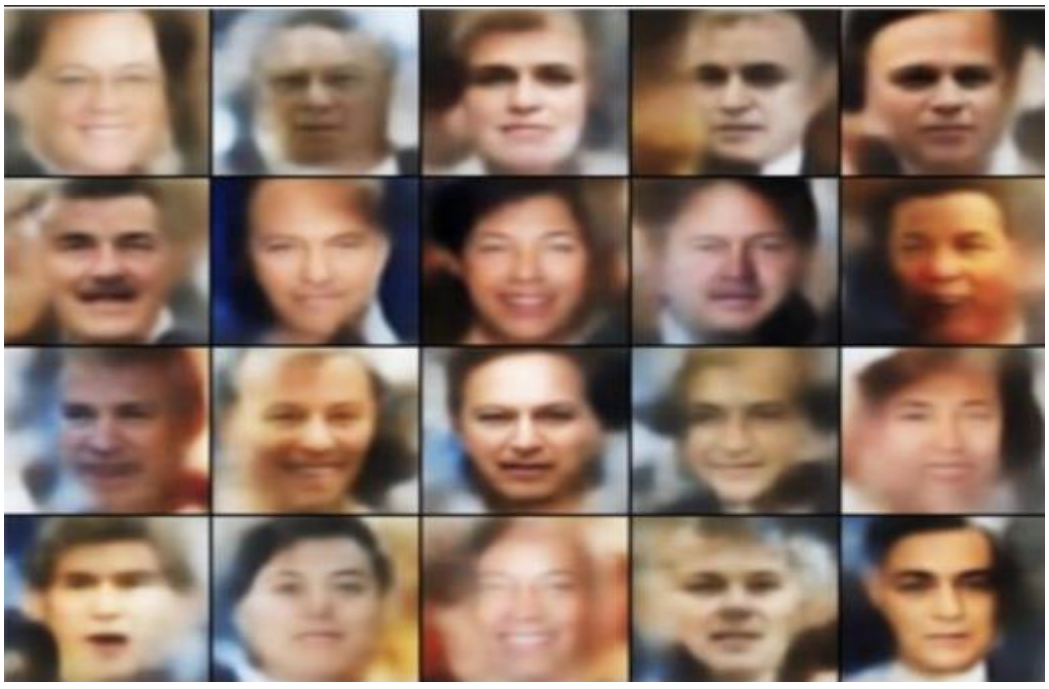


Figure 4 VAE generated samples

1.2.2. Markov chain

In general, stochastic approximation [8] that can be seen even in the minibatch selection where samples are randomly selected from training data. This stochastic approximation work well as far as the variance between selected images for training is not so high but with Markov chain we made an expensive generation as the Markov chain concept can be summarized in generating samples by repeatedly drawing a sample $x_f \sim q(x_f \mid x)$ which guarantee that x will eventually converge to a sample from $p_{model}(x)$. Unfortunately, this convergence can be very slow without being able to verify the method convergence. The efficiencies of Markov chain is so low when we deal with high-dimensional spaces. Actually, the Markov chain approximation techniques have not scaled to problems like ImageNet[1]generation and even if Markov chain methods scaled well enough to be used for training, they won't be used to generate with the presence of single-step generation methods because the multi-step Markov chain approach has higher computational cost.

In our work, we are interested in generative adversarial networks that belongs to the second type of generative model. This in next section, we will explain the generative model concept and illustrate its different application

[1] http://www.image-net.org/

III. GANs

1. GAN process

As we have said, GANs belong to the second type of neural networks which implicitly define the data probability distribution by generating samples similar to them. Actually, main idea of GAN [9] is to set up a game between two players: the generator and the discriminator.

First let's define the generator. The latter is the player that creates samples. Those samples are intended to come from the same distribution as the training data. The discriminator is the second player that examines samples to decide whether they are real or fake. The discriminator usually learns using traditional supervised learning techniques to divide inputs into two classes (real or fake). The generator is trained to fool the discriminator. The GAN's purpose is to create samples, which are able to deceive humans and even computers. Hence, they are often compared to money counterfeiting: generator aims to create fake money whilst discriminator tries to differentiate between real and fake one. The two networks improve themselves by competing against each other. Each of our players has its own differentiable function with respect to its parameters and its inputs.

The discriminator has a function D that takes x as input and uses θ_D as parameters. The generator is defined by a function G that takes z as input and uses θ_G as parameters. Both players have cost functions that are defined in terms of both players' parameters. This cost function is defined by the following equation:

$$\min \theta_G \max \theta_D(E_x \log(D_{\theta_D}(x)+E_z \log(1- D_{\theta_D}(G_{\theta_G}(z)) \qquad (2)$$

So, let us explain this relation, we have a real image x that will be examined by the discriminator D. For this image, it will give a value near to zero. Hence, for a fake image, it will give a higher value near to one. For the generator, he will take a randomly generated vector from a very simple and well-known distribution and produce an image that will be used also to train the discriminator. The latter will be alternatively shown a real and fake image. Therefore, the generator's role is to minimise the output by providing more realistic images while the other player tries to maximise the same thing. Evidently, each player cost depends on the other player's parameter but they can only control their own parameter. This scenario is most straightforward to describe as a 'minimax' game where the solution is a Nash equilibrium. The following figure (5) resume the process

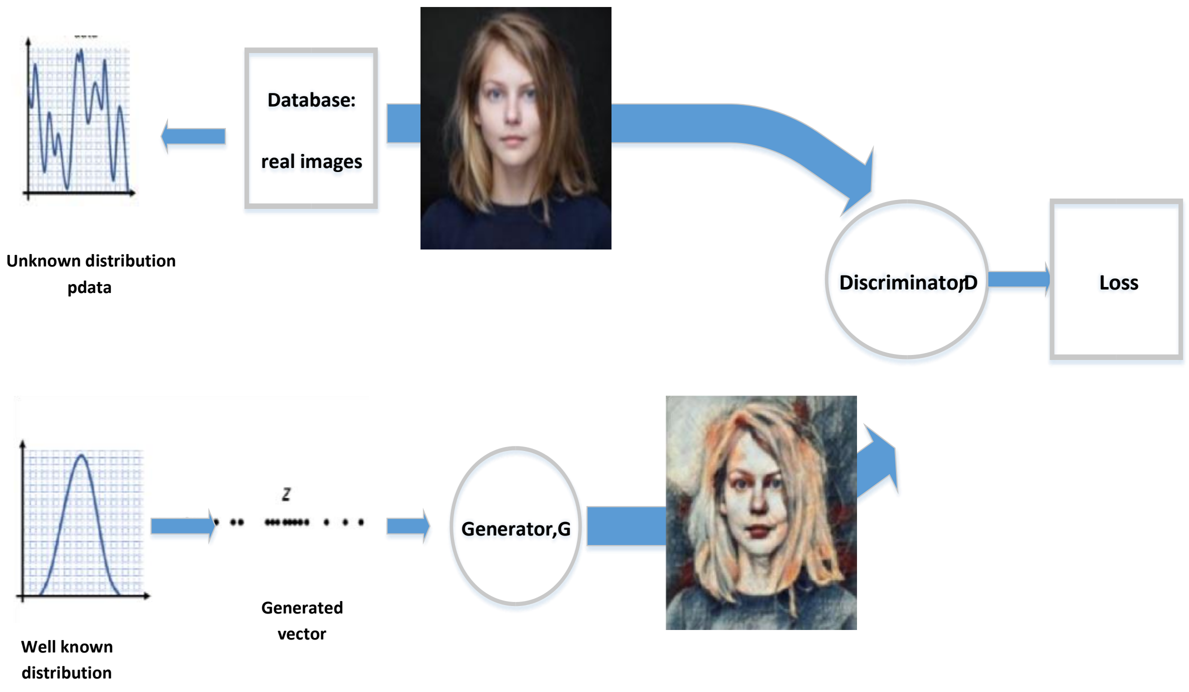


Figure 5 GANs process

2. Advantage of GANs

As we have said the Generative adversarial network (GAN) are networks, which are able to provide samples similar to a training distribution. There are several reasons that GANs models are important [9]:

- ✓ **Represent and manipulate high-dimensional probability distributions:**

In applied math and the domains of engineering, the high-dimensional probability distributions have a big importance. Hence, in order to test our ability to represent and manipulate high-dimensional probability distributions, the generative models are the suitable tools

- ✓ **Reinforcement learning:**

The generative models of time-series data can be used to simulate possible futures and such models could be used for planning in reinforcement learning in a variety of ways.

- ✓ **Missing data and semi-supervised learning:**

Missing data is not a problem for generative models, which are able to provide predictions on inputs with missing data. An interesting case of missing data is the semi-supervised learning, in which the labels for many or even most training examples are missing.

- ✓ **Working with multi-modal outputs:**

GANs enable machine learning to work with multi-modal outputs. For many tasks, a single input may correspond to many different correct answers, each of which is acceptable.

- ✓ **Data generation:**

Many tasks intrinsically require realistic generation of samples from some distribution. In figure (6) we illustrate samples generated by GAN

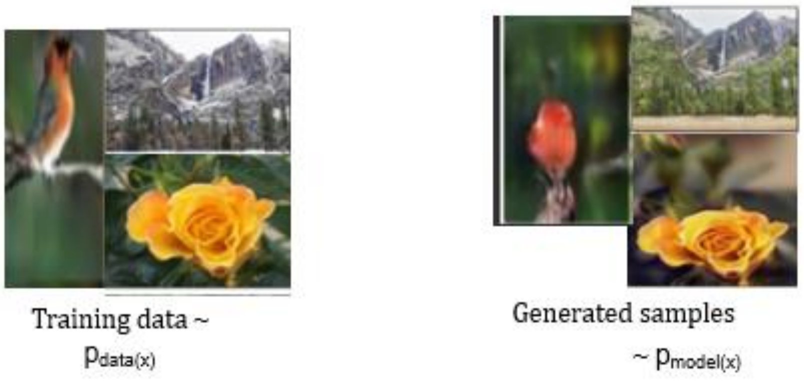


Figure 6 GAN's output

3. GAN’s Application

GANs open the door to many applications such:

- ✓ **Generation of images with super-resolution:** take a low- resolution image and synthesize a high-resolution equivalent.
- ✓ **Create art:** GANs can be used to create interactive programs that assist the user in creating realistic images that correspond to rough scenes in the user’s imagination [10].
- ✓ **Image-to-image translation**: convert one type of photos into another type, or convert sketches to images.
- ✓ **Text to image translation:** generate realistic image from a text description and vice versa
- ✓ **Image inpaiting:** completing an image with missing patches.

In the following figure (2), we illustrate some of GAN applications:

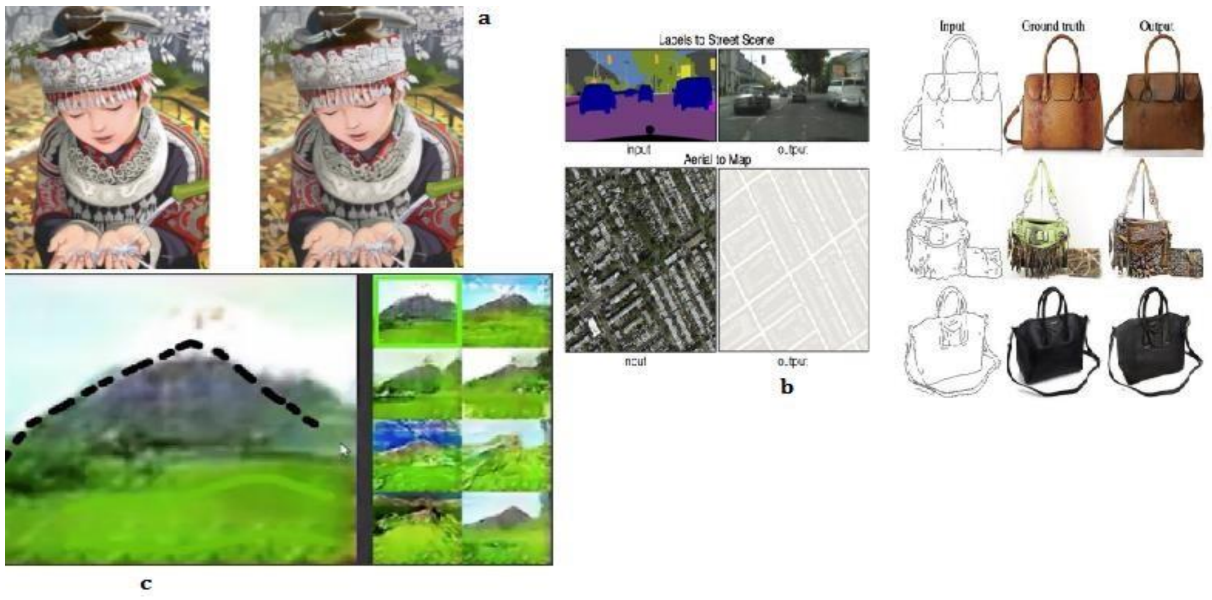


Figure 7 GAN applications example a) The high resolution image generation using GAN a b)image to image translation c) an interactive application developed by Zhu and al. [10]

Now, after giving an overview about generative model and explain the GANs. We are going to illustrate in the next section our project

IV. Developed Task

As we have mentioned, the principle of GAN is managing a game between two networks. Implementing this in Python is an old-hat since there are many pre-build solutions. For our work, we have mostly followed the base-code from [11]

1. Dataset:

As a first step, we have chosen the dataset from which we are aiming to generate similar samples. This dataset contains tongue-ultrasound images. We have collected those images from a file generated by the ultrasound software -The dataset contain the recording of a Hungarian female with normal speaking abilities while reading sentences aloud (altogether 209 sentences) [12]. The tongue movement was recorded in midsagittal orientation using the “Micro” ultrasound system of Articulate Instruments Ltd. at 82 fps- The latter file contains raw ultrasound images stored after each other. Each pixel is stored as a 1 byte unsigned integer, which has actually a grayscale pixel intensity. Using the extracted raw ultrasound images, we can build ultrasound frames. Those frames can be used to produce a video illustrating the tongue’s movement.

In our case, we are interested in raw ultrasound images. Therefore, after successfully extracting those images , we have built a data set of 27925 raw images which dimension is 64*842 pixels that we have split it into two groups:

The first group is made of 2/3 of images, which is used for training and the second one made of 1/3 of the dataset used for testing.

The figure below (3) illustrates a sample of raw ultrasound images and ultrasound frames as well :

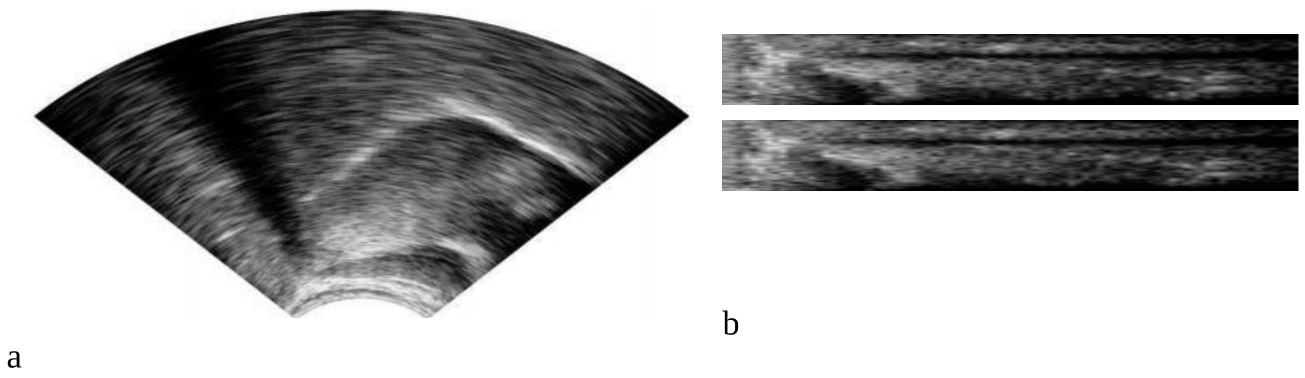


Figure 3 : a) ultrasound image – b) raw ultrasound image

2. Deep learning Framework

As a framework, we have used the TensorFlow 1due to its features. Actually, the TensorFlow which has been made by Google enables us to define static graphs i.e. define the graph then run it. In addition, those graphs are compiled at compile time, which ensure a faster compilation. In addition, Tensorflow contains high-level wrappers such Keras and Sonnet. After choosing the framework, we have to build the model that is able to process our images

3. Model:

For our project, we have chosen to use the deep convolutional neural network for both the generator and the discriminator. Hence as a first step, we are going to explain the architecture of each network.

a) Discriminator:

It is a downsampling network using strided convolutional layers. The discriminator main idea is to check real images and save variables in order to use them in fake image checking. Hence, we have to define the number of hidden layer, the kernel for each layer and its dimension. For our image with size of 64*842. The number of hidden layers N_{hl} is the same for both the generator and discriminator and it can be defined by the following equation :

$$N_{hl}=\text{int(math.log(image\_size) / math.log(the stride number))} \quad (3)$$

In addition, for each layer we have to apply the batch normalization on our data before going through the non-linear function. Finally, the last layer is a linear layer providing the unbounded result from the networks. As a classification task, we apply the sigmoid function on the result of the final layer in order to get a probability between 0 and 1.

b) Generator:

An upsampling neural network takes as input a z vector randomly generated from a known distribution. After linearly transforming z, it will be fed through the layers in order to get as a result an image with the same size as our original image. This is well done by fixing the number of hidden layers, number of convolutional filters in each layer and their size.

After defining the two networks, we have to define our tensorflow graph where we have to put together the data, the discriminator and the generator. Then, we have to choose the loss function. In our case, we have used the cross entropy loss. Using this cost function, we can calculate the discriminator

[1] https://www.tensorflow.org/

loss by comparing its result to one for both a fake and a real image but for the generator loss, we compare its result to zero. Finally, in order to update our network parameter, we have to define an optimization method; in our case, we have used the Adam optimizer.

Before starting the training process, we have to define our network hyperparamaters such as the size and the number of kernels for the discriminator, the batch size and the number of epochs.

We can summarize this process in the following pseudo code:

```
                                                          Maximise G
-   for number of iterations:
-   for k steps:
            ○  Sample minibatch of noise samples and generate images  ○
               Sample minibatch from the image dataset
            ○  Update the discriminator using gradient ascent
                                                       Minimise D
-   Sample minibatch of noise samples and generate images
-   Update the generator
```

V. Experimentation and result

In our model, we have fixed the number of hidden layer with respect to equation (2) Therefore, we have eight hidden layers for both discriminator and generator. As hyperparamaters, we have fixed 64 as batch size and 25 for the number of epoch. Equally, we have chosen as convolutional kernels filters with size 5*5 and stride equal to two. In the following table (1), we will summarize the process of the two neural networks with respect to their input and output image size and the number of filters for each layer.

Table 1 : discriminator and generator process

	Discriminator			Generator		
layer	Number of filter	Input image	Output image	Number of filter	Input image	Output image
l	64	842*642	32*421*64	512	1*4*512	1*7*256
2	64	32*421*64	16*211*64	256	1*7*256	1*14*128
3	64	16*211*64	8*106*64	128	1*14*128	2*27*64
4	64	8*106*64	4*53*64	64	2*27*64	4*53*64
5	64	4*53*64	2*27*64	64	4*53*64	8*106*64
6	128	2*27*64	1*14*128	64	8*106*64	16*211*64
7	256	1*14*128	1*7*256	64	16*211*64	32*421*64
8	512	1*7*256	1*4*512	64	32*421*64	842*642

The following figure will explain better our model where for the generator we have as input the noise with a normal distribution and as output an image with a size equal to the real one and for the discriminator he takes as input an image with size of 64*842 and produce an output with a size of 1*4*512:

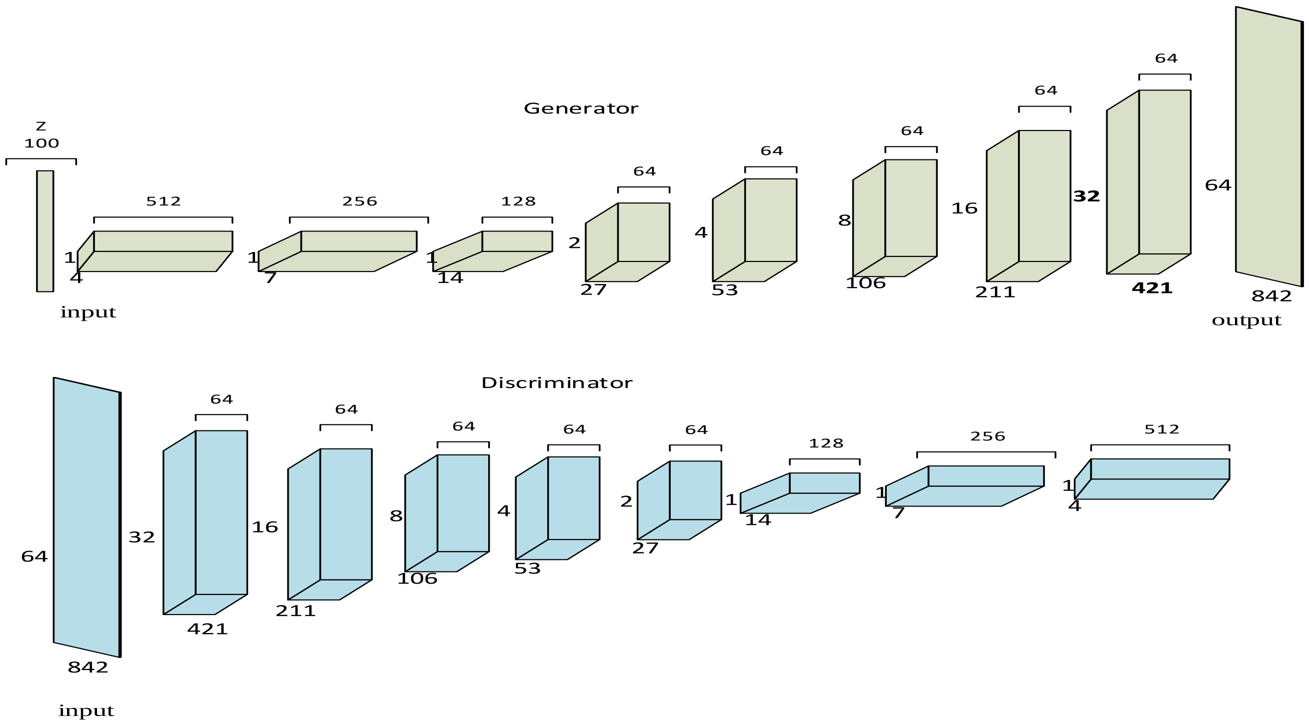


Figure 8 the model architecture

1. Result

After every 100 iterations, we generate 64 ultrasound images. In figure (9); we illustrate some generated images and in figure 10 the corresponding images with real physical orientation

The final loss of our model is 0.67 for the discriminator and 1.673 for the generator

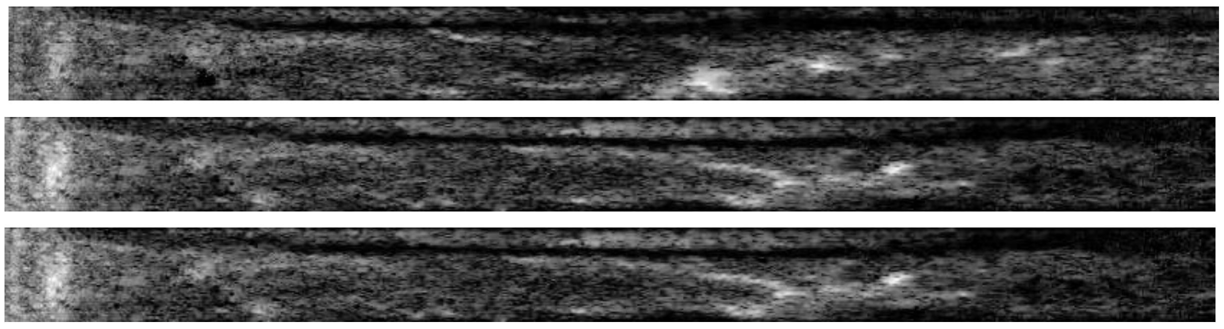


Figure 9 generated images in the last epoch

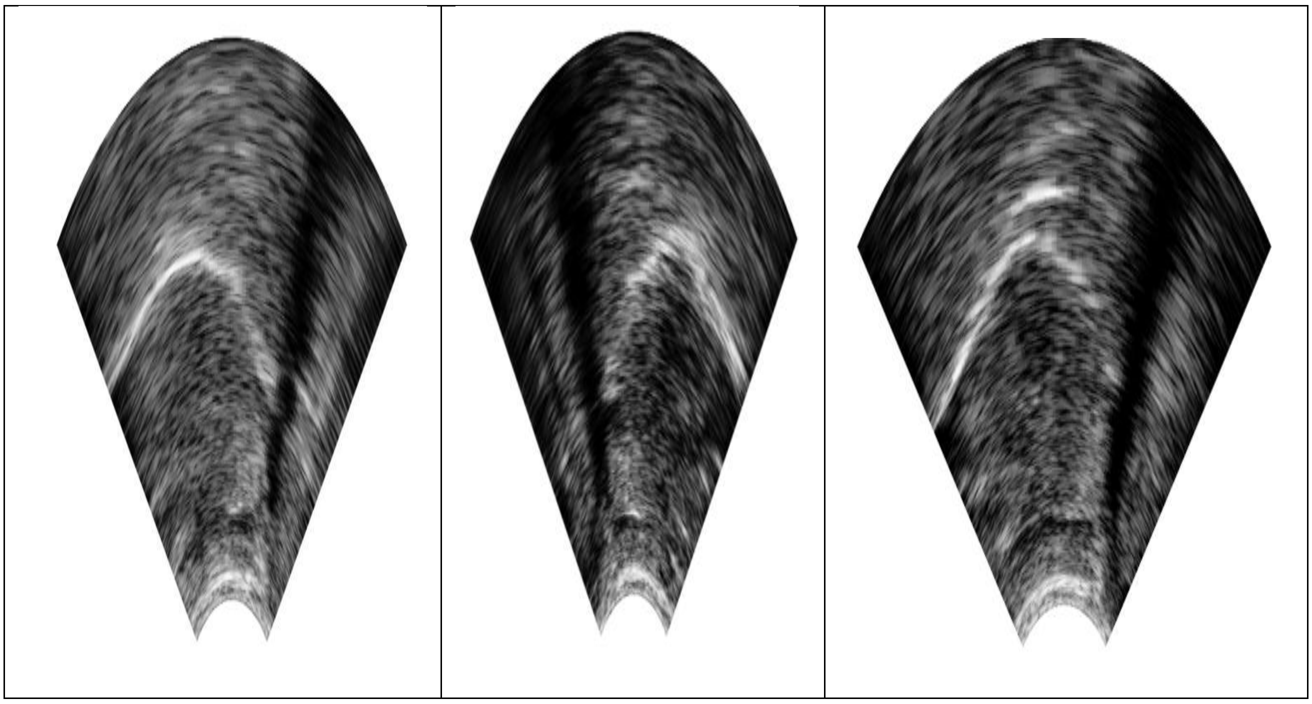


Figure 10 real physical orientation of tongue ultrasound images

Obviously, the performance of generative adversarial networks did not start as good as in final epochs because as we have said during the training game the generator will improve its performance and this can be noticed by the quality of the generated images. In figure 11, we will illustrate images generated from first epochs and others from last epochs which explain the progress of the generator

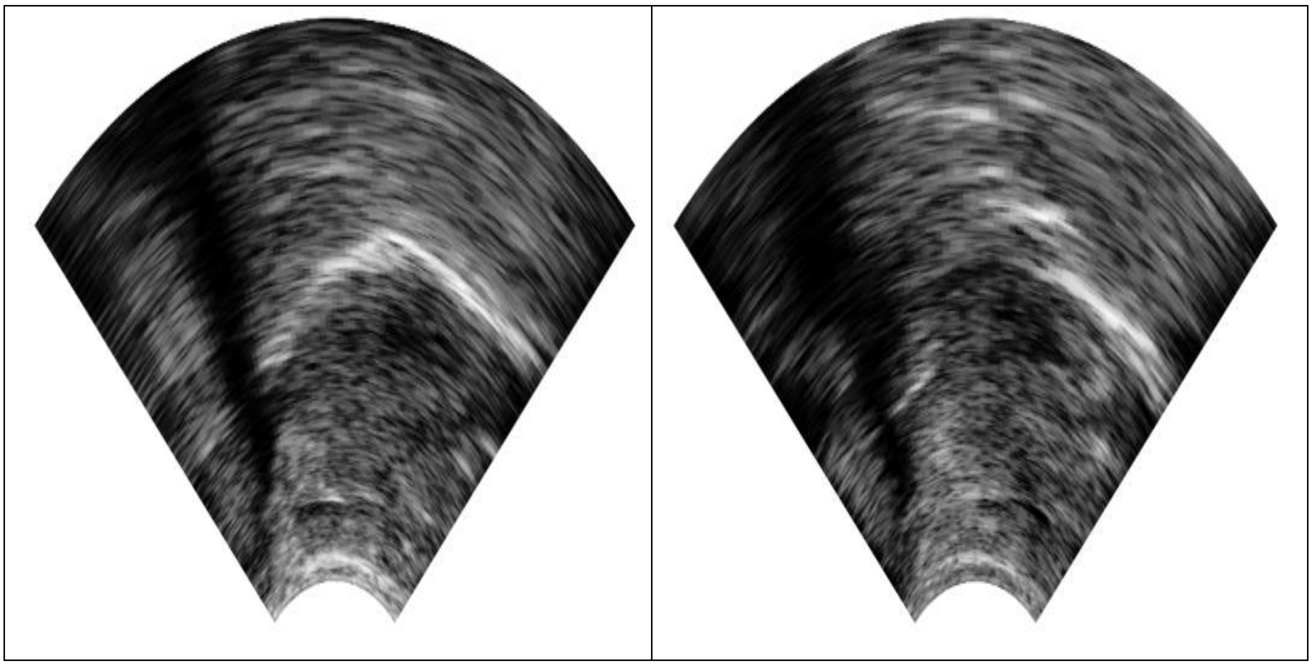
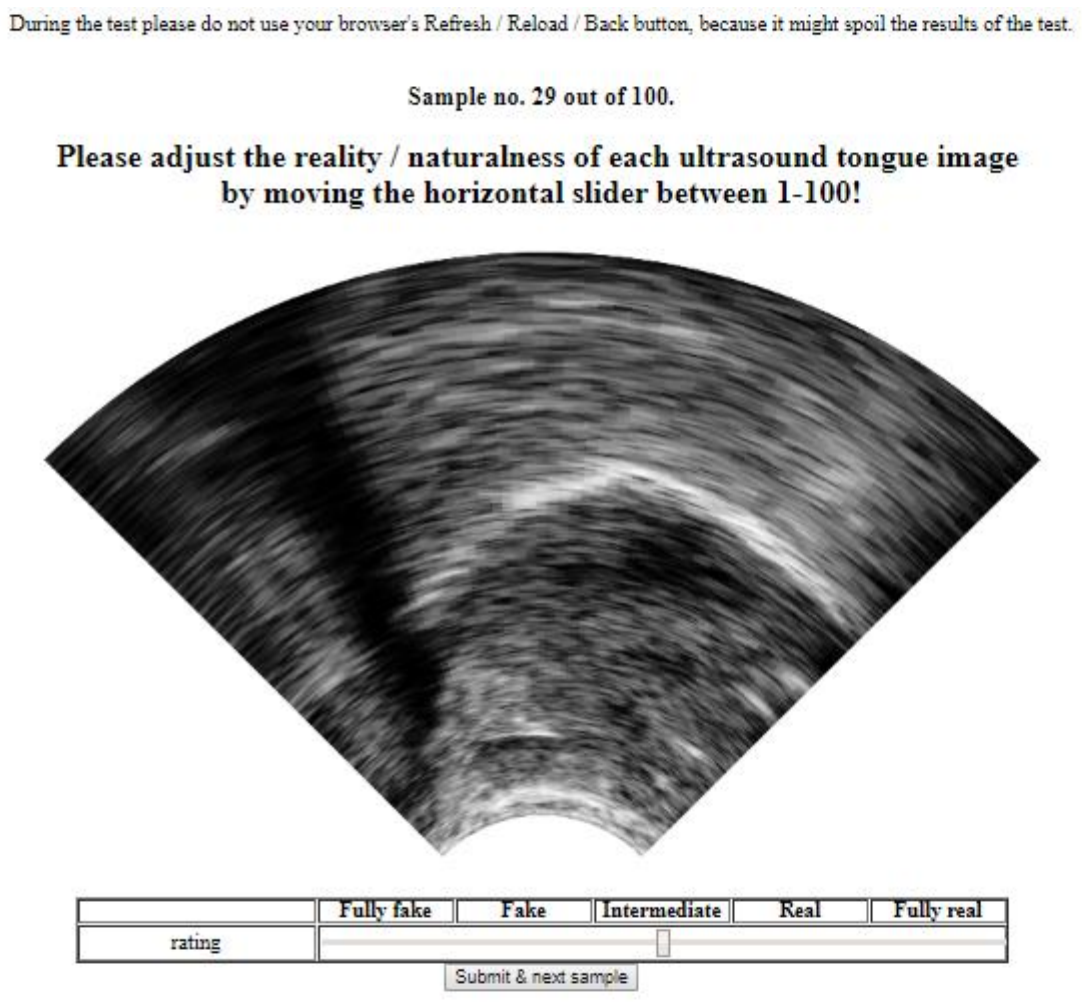


Figure 11 generated images a - from last epoch b - from earlier epoch

2. Evaluation

In order to investigate the quality of the generated raw images, we have created an internet test based1. In this test, we have 100 samples made of real and generated images. The generated images used for training are made of bad samples generated from the first iterations and good ones generated from the last iterations. The participant's task is to assess the reality of images without any prior knowledge about their origin either they are real or generated ones. The following figure (5) explains more the process and we can easily notice that we have used for the test the tongue ultrasound images with real physical orientation



During the test please do not use your browser's Refresh / Reload / Back button, because it might spoil the results of the test.

Sample no. 29 out of 100.

Please adjust the reality / naturalness of each ultrasound tongue image by moving the horizontal slider between 1-100!

	Fully fake	Fake	Intermediate	Real	Fully real
rating					

Figure 12 the internet-based test

[1] http://leszped.tmit.bme.hu/gan2018/

In this test 9 persons have participated. The test result can summarize in the table (2):

Table 2: result of images evaluation

	Good Images	Bad images	Real images
Number used in test	60	20	20
Average result	70.73	36.1	90.1

Analysing the result, we can say that the GANs are very efficient in image generation and deceive humans. Actually, the average result for real images in 90.1% which mean that the rest of real images where classified as good images as the participants were confused either they are treating a real or a fake one and this is the common feedback given by the participant

3. Future Work

The performance shown by the GANs in generating realistic tongue ultrasound images, encourages us to improve the used model by taking into consideration the time dimension to be able to predict the next input for the generator which may enhance the performance and get better and accurate result.

Conclusion

Our project consists in creating a tongue ultrasound dataset which can be used as benchmark for speech research methods evaluation. In order to achieve this goal, we have used the generative adversarial networks known as performant networks for samples synthesis

In this report, we have given an overview of generative models. Afterwards, a detailed explanation of generative adversarial networks and their interest is presented. Then, we have detailed our work starting with the creation of the dataset, to the conception of the network model and finally, the investigation of the obtained result.

According to the experiments, it is possible to generate realistic tongue ultrasound images and we can take advantage of the neural network to build a huge dataset which can be used as benchmark to assess the several methods in speech research

References and list of related documents

References:

[1] C. Wu, S. Chen, G. Sheng, P. Roussel, and B. Denby, “Predicting tongue motion in unlabeled ultrasound video using 3D convolutional neural networks,” in Proc. ICASSP, 2018

[2] Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014b). Generative adversarial networks. In NIPS’ 2014 .

[3] S.Mohamed, B. Lakshminarayanan “Learning in Implicit Generative Models”, arXiV: 1610.03483v3,2017

[4] K.Gregor, I.Danihelka, A.Graves, D.J.Rezende, D.Wierstra “DRAW : a Recurrent Neural Networks for Image Generation”, arXiv : arXiv:1502.04623v2 ,2016

[5] A.v.d.Oord, S.Dieleman, H.Zen, K.Simonyan, O.Vinyals, A.Graves, N.Kalchbrenner, A.Senior, K. Kavukcuoglu “WavNet: A Generative Model For Raw Audio” arXiv:1609.03499v2,2016

[6] K.Gregor, I.Danihelka, A.Graves,D.J.Rezende,D.Wierstra, “DRAW: A Recurrent Neural Network For Image Generation”,arXiv 1502.04623v2, 2015

[7] D.P.Kingma,M.Welling, ”Auto-Encoding variational Bayes”, arXiv :13126114,2013

[8] D.J. Rezende, S. Mohamed,, and D.Wierstra . “Stochastic backpropagation and approximate inference in deep generative models.” Technical report,2014, arXiv:1401.4082.

[9] I.Goodfellow; “Generative adversarial networks”. ar Xiv: 1701.00160v4;2016

[10] P.Isola, J.-Y.Zhu,T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks”, CVPR, 2016, pages: 5967-5976.

[11] A.Radford, M.Luke, and C.Soumith. “Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.”, arXiv: 1511.06434v2, 2016

[12] T.G.Csapo,A.Dence,T.E.Gráczi,A.Markó,G.Varjasi. “Synchronized speech, tongue ultrasound and lip movement video recordings with the “Micro” system ”

Workshop on Challenges in Analysis and Processing of Spontaneous Speech (CAPSS2017), 2017

Related Documents:

Code and results :

https://drive.google.com/drive/folders/1C3osmWeb70uwpIRuSdMuyilqBCVuOUlW?usp=sharing