**Report**

Assignment 4, Deep Learning

by Carmel Shablin 305812661

and Gregory Koushnir 321889479

## Task

Implement two generative architectures: standard GAN, and a modified GAN architecture.

## Dataset

The datasets include 2 files

a. German\_credit.arff – credit card samples with tabular and numeric features, class is good/bad customer

b. diabetes.arff – for blood measurements, class is has diabetes or not.

## Environment

Since none of the students has or has access to GPU physically, the assignment was done in the Google Colab environment in order to speed up the training process. All shown results are taken from the Jupiter notebook.

## Method

The general method is to build a GAN for each dataset, GAN is combined from 2 NN one call Generator and other Discriminator, and the main goal is make Generator to generate samples that are similar to real ones.

### Preprocessing:

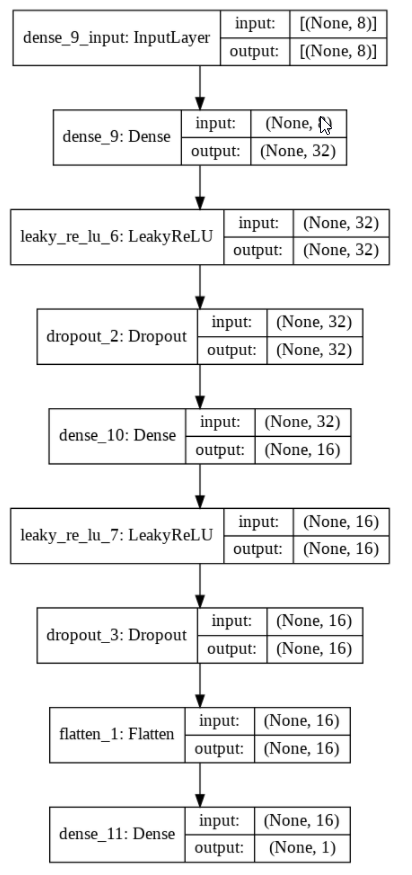
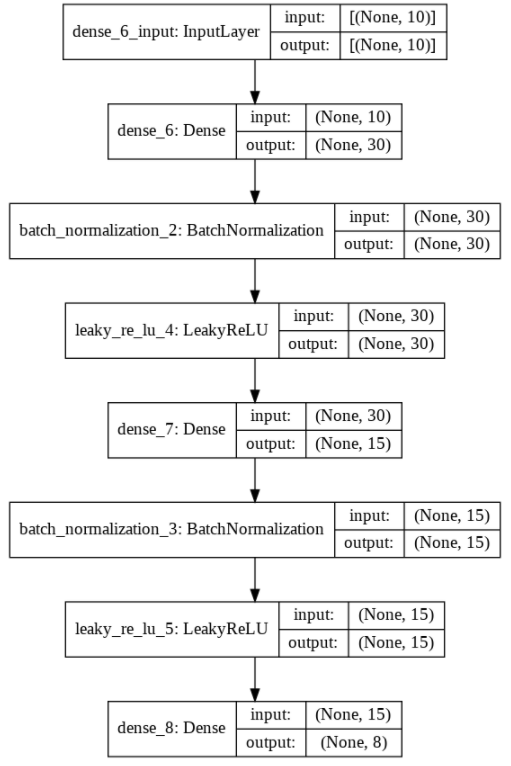
diabetes.arff – all columns were numeric so we just transformed all columns to [-1, 1]

German\_credit.arff – some of columns where categorial so for those we performed one-hot-encoding than transformed all columns to [-1, 1]

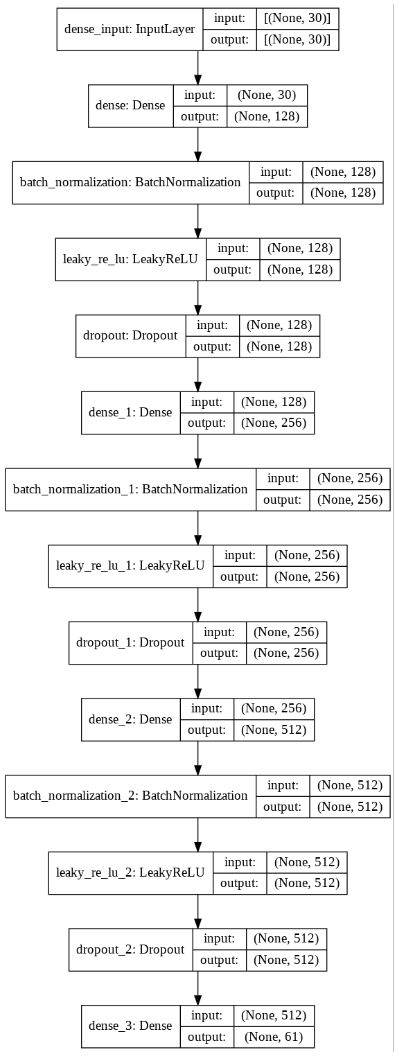
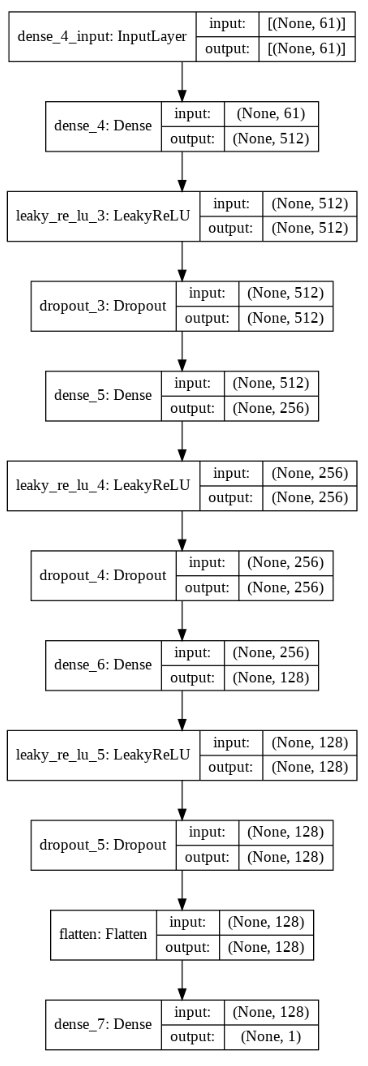
### Architecture

Similar architecture was implemented (per dataset) for both parts of Assignment

Diabetic GAN, Generator + Discriminator respectively:



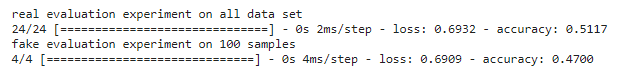
Credit GAN, Generator + Discriminator respectively:

### Part 1

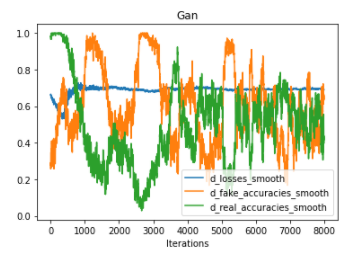
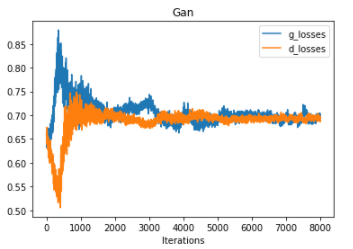
Diabetic dataset

1. NA
2. GAN convergence
   1. GAN losses reached convergence as can be seen in plot of ‘3.c’
3. Analysis
   1. TODO – check within each feature whether the sample is in the middle of range of real distribution of dataset. For fooled samples we expect them to be in range of majority of features, and for those who didn’t we expect them to be not in range of majority of features.
   2. About 47% passed as real samples:



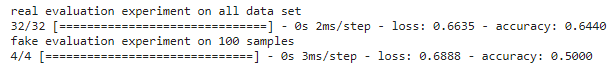
~~we can calculate the Gaussian distribution of each column and check how close it is to real sample distribution~~

* 1. As can be seen the models did go back and forth there was no consistent leader:

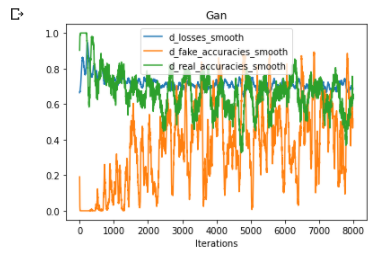
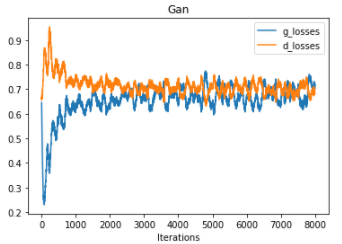


Credit dataset

1. NA
2. GAN convergence
   1. GAN losses reached convergence as can be seen in plot of ‘3.c’
3. Analysis
   1. TODO- check within each feature whether the sample is in the middle of range of real distribution of dataset. For fooled samples we expect them to be in range of majority of features, and for those who didn’t we expect them to be not in range of majority of features.
   2. About 50% passed as real samples:



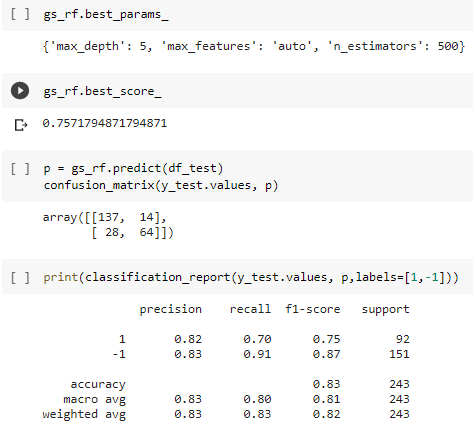
* 1. As can be seen the models did go back and forth there was no consistent leader:



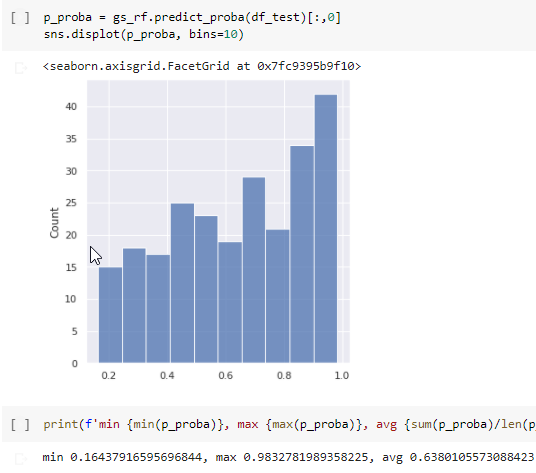
### Part 2

Diabetic dataset

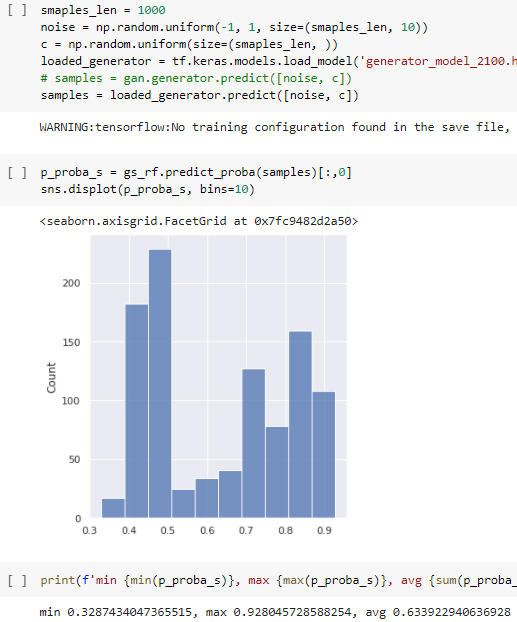
1. – Random forest trained (70%) using 5-fold CV and Grid search and the following is the best:



1. NA
2. NA
3. Following information provided:
   1. Random Forest test (30%) performance:



* 1. NA
  2. Statistics on score distribution of BB using generated samples:



Looks like the resulting distribution is Uniform-alike, boundaries are wide almost as in real distribution (much better than in the other dataset GAN). The perfect expectation would have been close to ~ U(0,1) same as the distribution of sampled “Confidences”.

TODO answer 2 q:

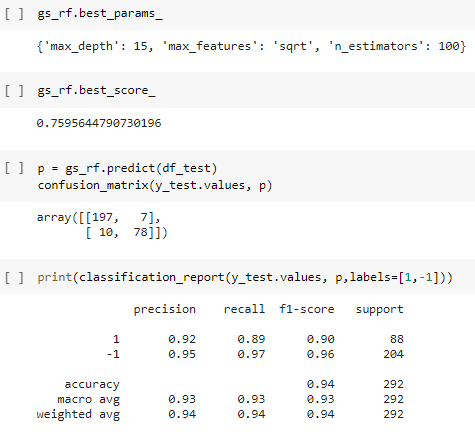
Were you more successful for a specific class of samples? 🡺 means low confidence is class\_1 and high is class\_2, so we can test/plot distribution for both classes

Were you more successful for a specific range of confidence scores? 🡺 to sample from deferent ranges “Confidences” and check the output range, desired expectation would be c = Y i.e. the resulting range will be same

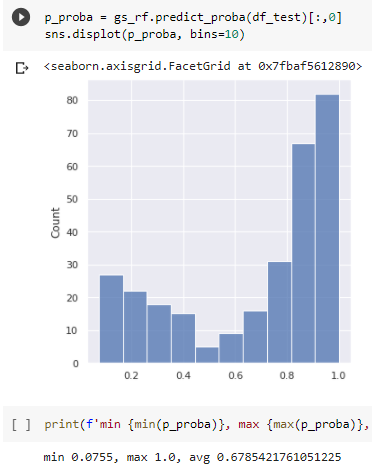
* 1. problematic aspect we had encountered influenced the resulting ‘Confidence’ distribution - Firstly our ‘noise’ for generation was sampled from ~ N(0, 1) and the BB output distribution was worse from now (final state) it was always Normal-alike narrow distribution, we expected it to be more Uniform alike with boundaries close to [0, 1]. So what we did is to sample ‘noise’ from ~ U[-1, 1] this improved greatly the BB output distribution that looked unformal-alike~ U[a, b] as can be seen in ‘4.c’ plot.

Credit dataset

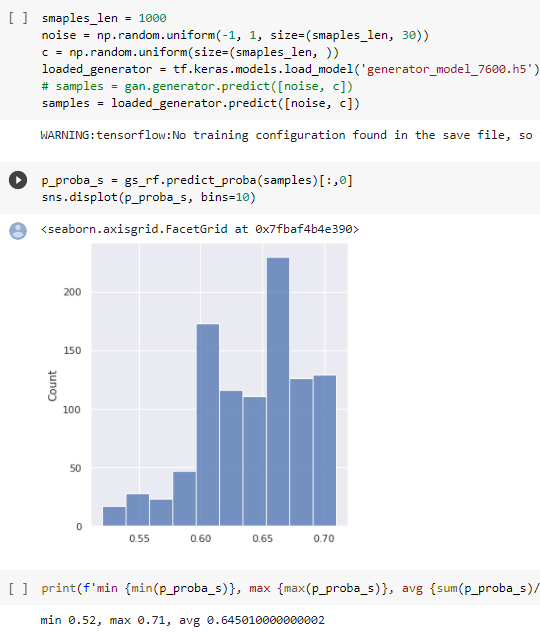
1. – Random forest trained (70%) using 5-fold CV and Grid search and the following is the best:



1. NA
2. NA
3. Following information provided:
   1. Random Forest test (30%) performance:



* 1. NA
  2. Statistics on score distribution of BB using generated samples:



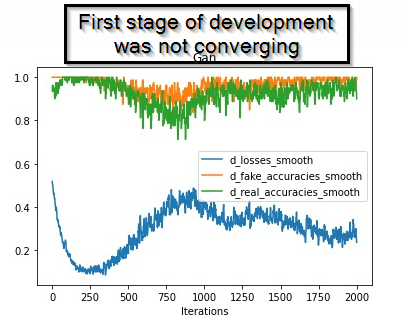
Looks like the resulting distribution is lacking samples near the boundaries (0 and 1) the perfect expectation would have been ~ U(0,1) same as the distribution of sampled “Confidences”.

TODO answer 2 q:

Were you more successful for a specific class of samples? 🡺 means low confidence is class\_1 and high is class\_2, so we can test/plot distribution for both classes

Were you more successful for a specific range of confidence scores? 🡺 to sample from deferent ranges “Confidences” and check the output range, desired expectation would be c = Y i.e. the resulting range will be same

* 1. At Early stages of development, the model suffered from not converging – the Discriminator was much better than Generator, as showed below in plot the discriminator can tell what is real and fake and the desire is to fool it by 50% of samples:



So we had to make them converge / learn in equal rate, we slowed down ‘D’ by changing learning rate by x10 to 0.0002 and later fine-tuned ‘D’ beta\_1🡺0.8, ‘G’ beta\_1🡺0.6 .

Another problematic aspect we had encountered influenced the resulting ‘Confidence’ distribution - Firstly our ‘noise’ for generation was sampled from ~ N(0, 1) and the BB output distribution was worse from now (final state) it was always Normal-alike narrow distribution, we expected it to be more Uniform alike with boundaries close to [0, 1]. So what we did is to sample ‘noise’ from ~ U[-1, 1] this improved greatly the BB output distribution that looked unformal-alike~ U[a, b] as can be seen in ‘4.c’ plot.