Faculty of Computer Science

Data Science and Business Analytics

Moscow 2023

Deep Learning Neural Networks to Identify Phlebology Disease (Joint Project with Antireflux Hospital)

Bachelor Thesis

Author: Rutsinskiy Arsenty
Academic Supervisor: Gromov V. A., Full Professor, Deputy Head of Department

Introduction

- Nearly the third of all death in the World are caused by cardiovascular diseases.
- Worsening of the situation due to covid and sedentary lifestyle.
- Rapid adoption of computer technologies in the medical sector.
- Attempts to exclude the possibility of medical errors and increase the capacity of diagnostic centers.

Tasks and Objectives

Key objective:

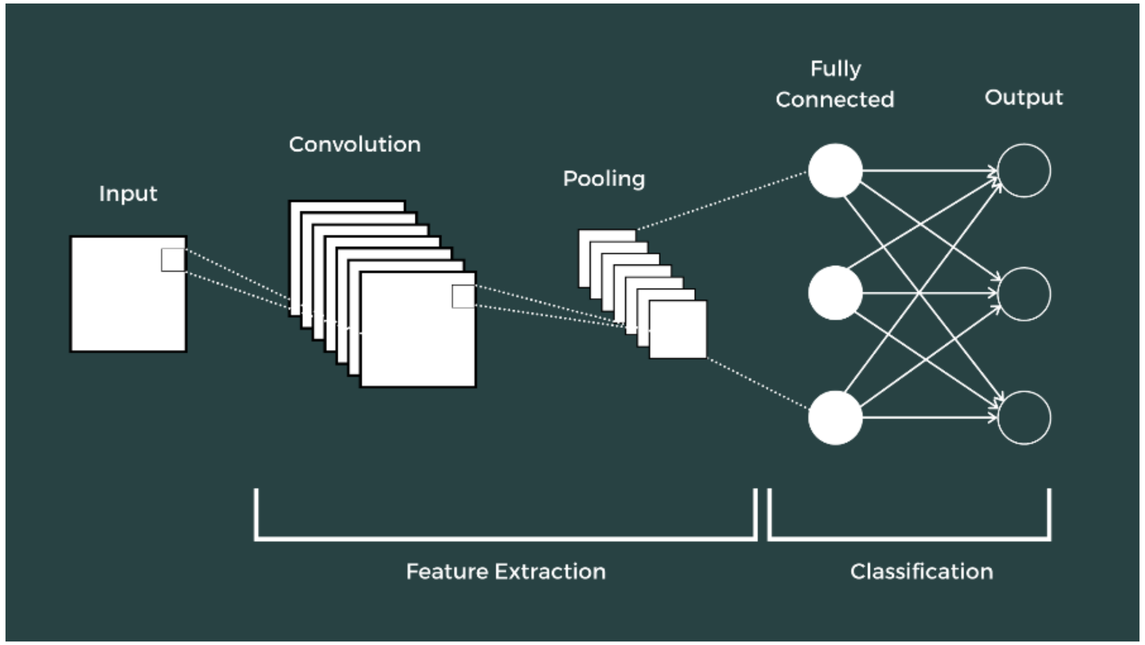
Develop deep learning neural network to identify phlebology disease.

Tasks:

- Review of works on phlebology disease identification.
- Explore and preprocess provided data.
- Study the cardiovascular system and human hemodynamics.
- Develop deep learning neural network for cardiovascular disease identification based on MRI images and Physics Informed Neural Network (PINN) paradigm.
- Carry out experimental studies of the effectiveness of the proposed methods.

Convolutional Neural Networks

- Commonly used for image recognition and classification
- Widely applied in problems concerning medicine
- Reduces spatial dimension, subsequently lowering computational costs



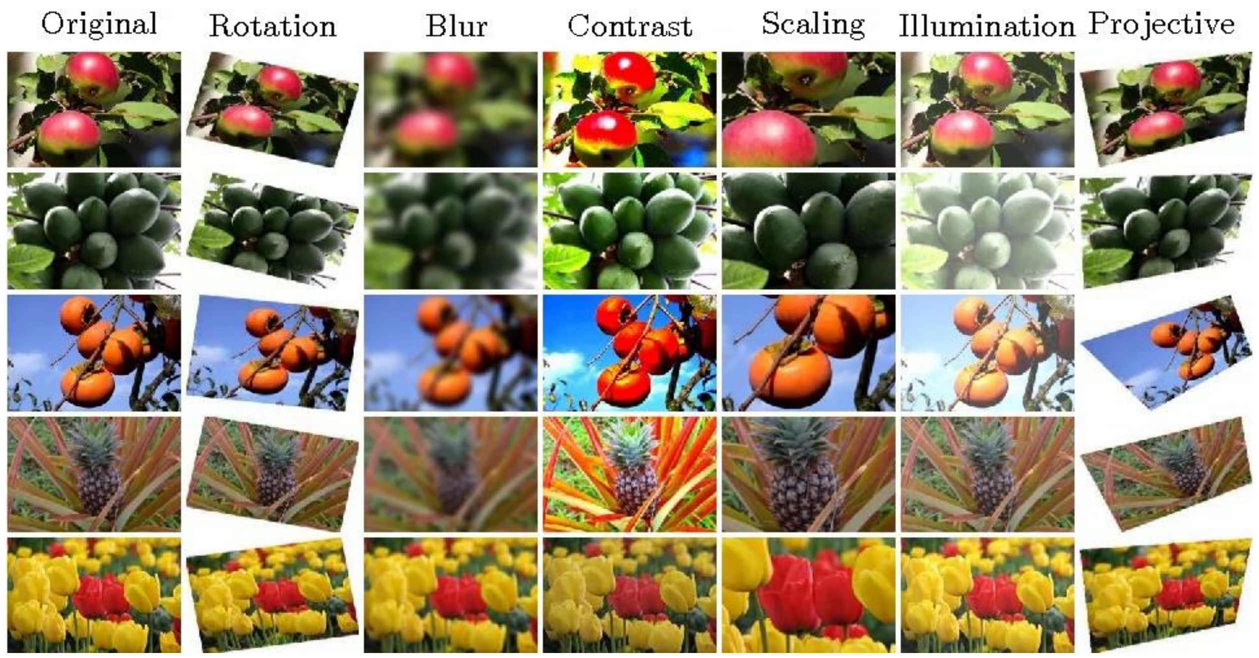
Data Augmentation

Basic augmentation methods:

- horizontal or vertical display, rotation, shift
- changing brightness and contrast
- randomly shuffling RGB channels distortion, blurring

Problems:

- No fundamentally new images appear
- MRI images are strictly standardized: centered, vertically and horizontally aligned, brightness and contrast normalized

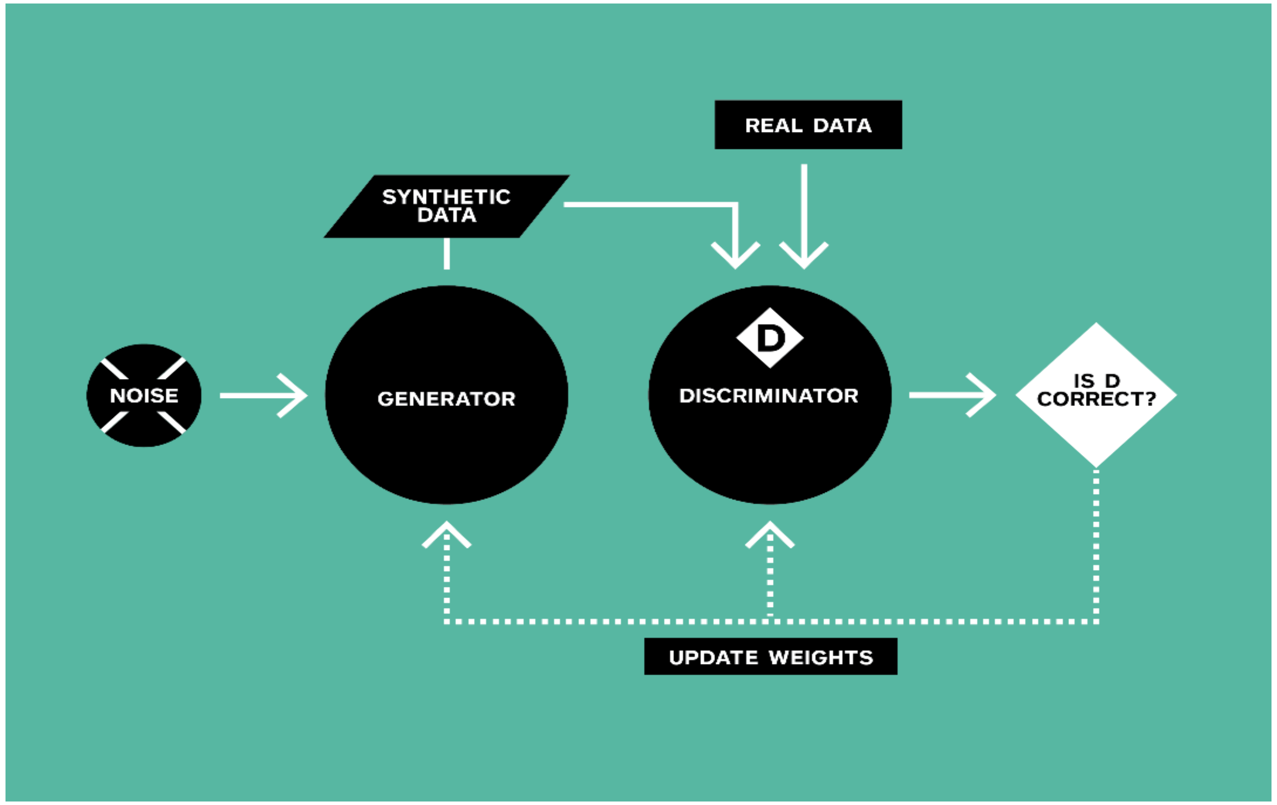


Data Augmentation: Generative Adversarial Network (GAN)

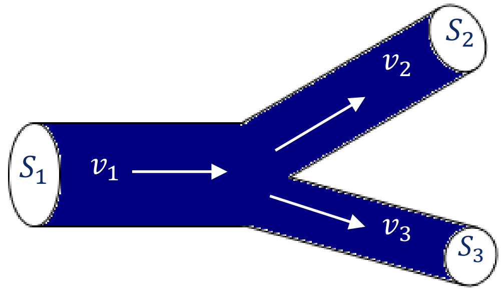
Two competing networks:

- A generator creates images from random noise
- A discriminator tries to distinguish the real images from the generated ones

After a while the generated images become indistinguishable from the real ones.



Human hemodynamics



Two main equations:

- Law of conservation of mass:

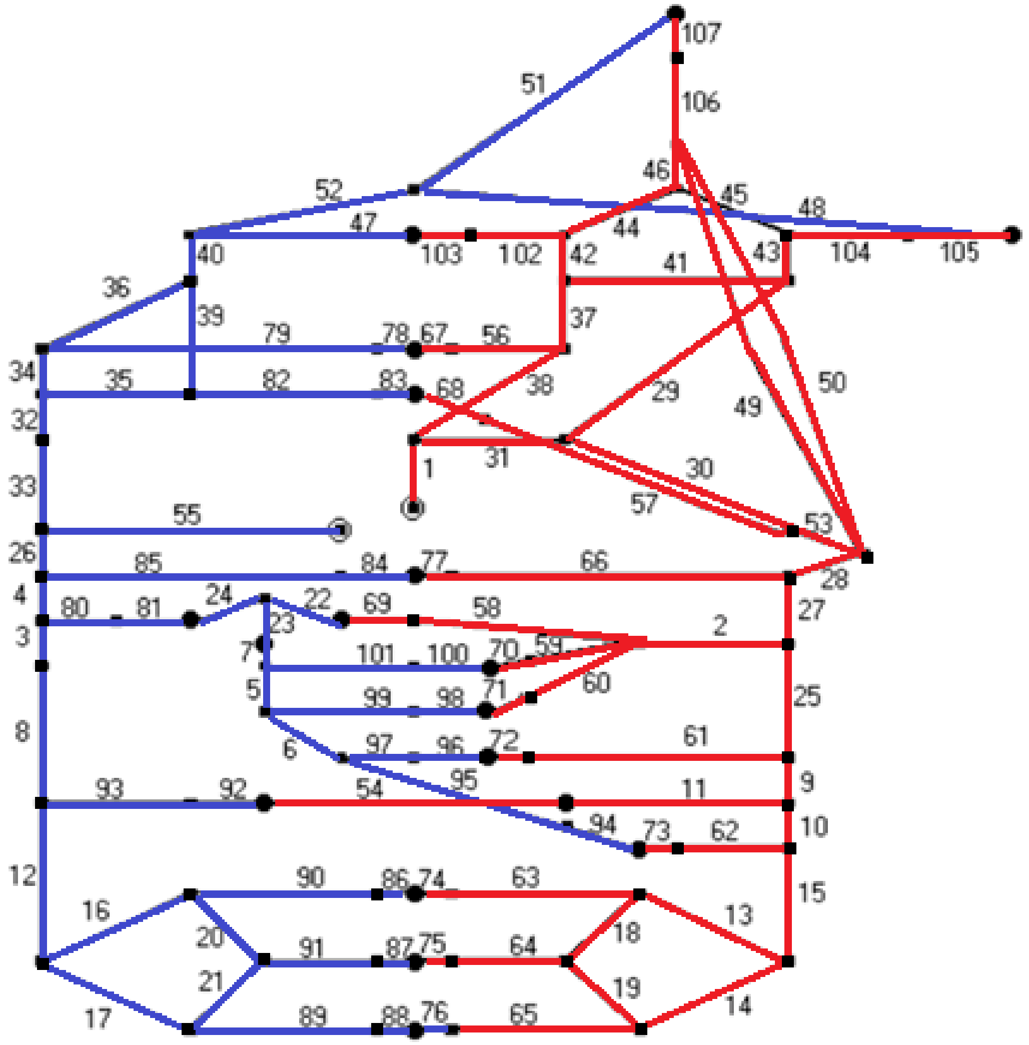
$$S_1 v_1 = S_2 v_2 + S_3 v_3,$$

where S – cross-sectional area, v – blood flow rate.

- Bernoulli’s equation:

$$\rho \frac{v^2}{2} + \rho * g * h + p = const,$$

where ρ – density (blood), v – blood flow rate, h - height, p – vessel pressure, g – gravitational acceleration.



Data preprocessing

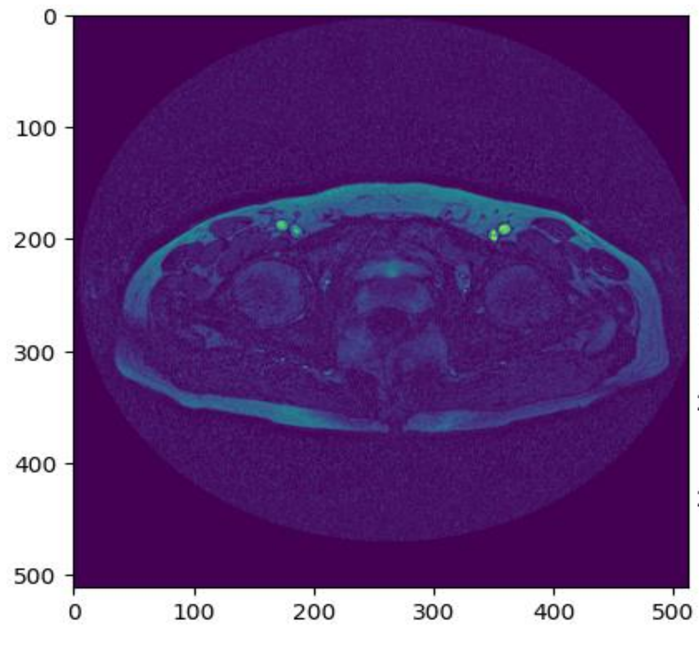
81 patients: 10 MRI scans each

CEAP diagnosis:

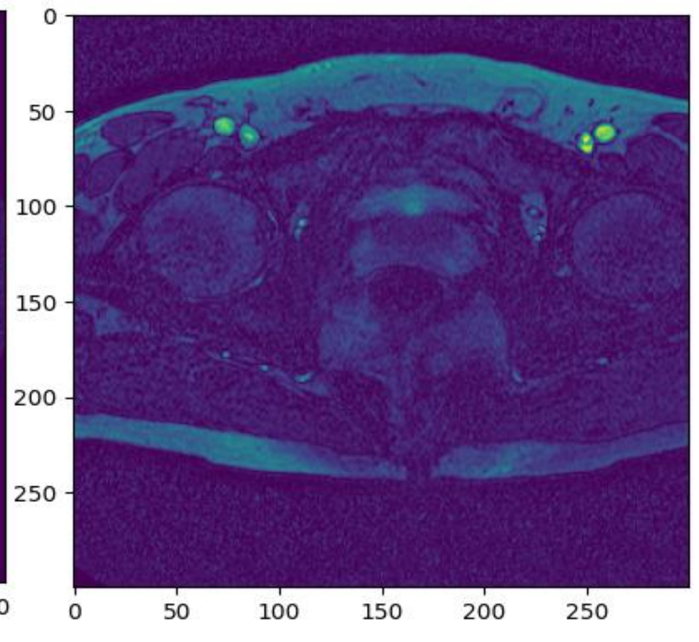
- 59 healthy (C0-C2)
- 22 sick (C3-C6)

Preprocessing:

- cutting uninformative edges,
- merging into a single image,
- organizing into the batches



Before

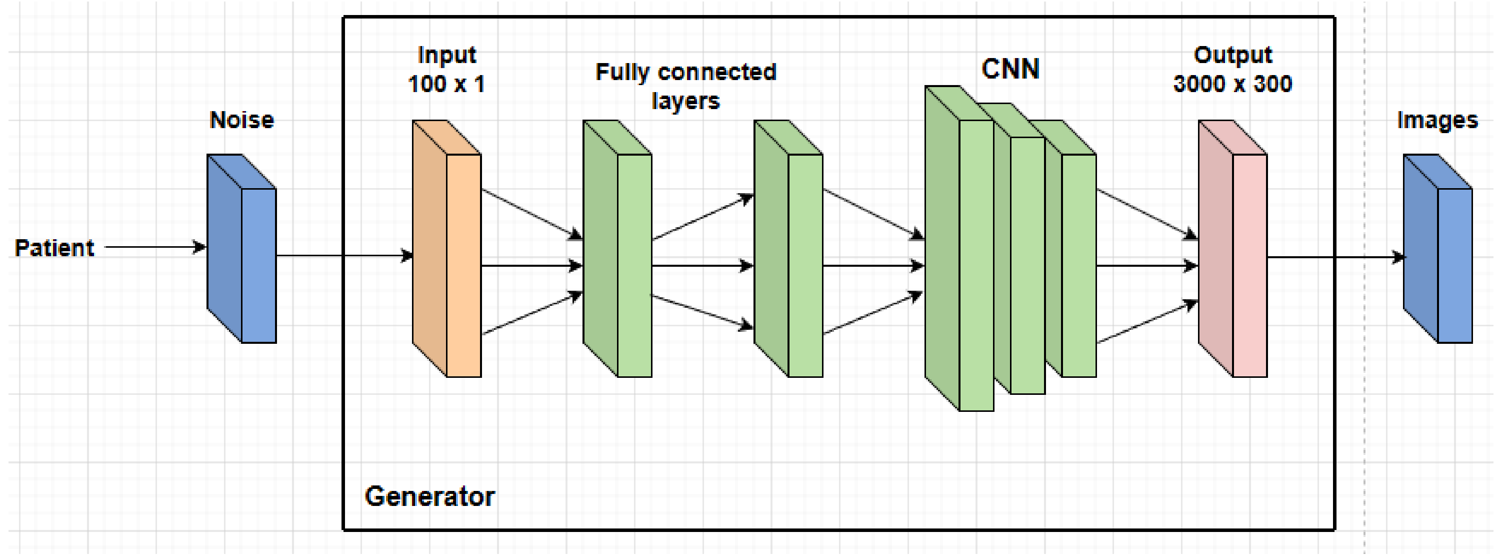


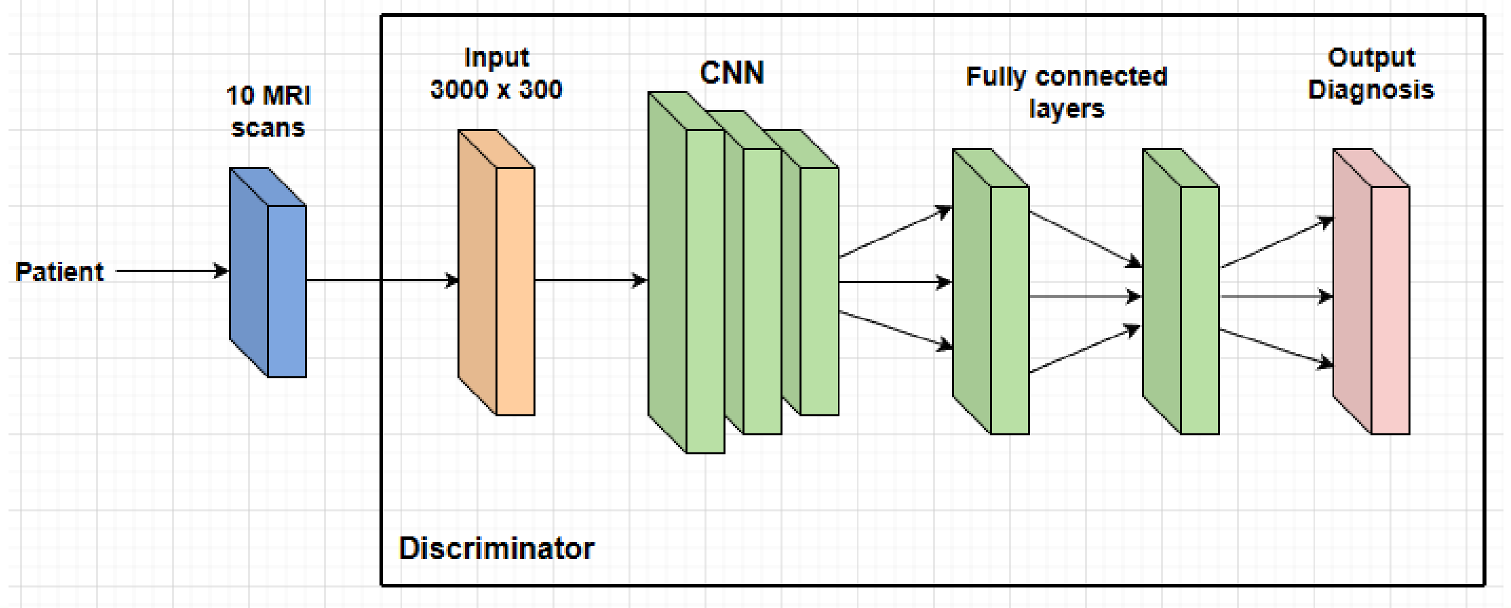
After

Standard GAN model

- Standard GAN architecture
- Loss function:

  Binary cross-entropy
- Batch normalization layers with Leaky RELU activation function
- Dropout rate equal to 0.3





GAN with PINN

- Hidden layer with 50 neurons added to the model
- Improved loss function

$$Loss_{residuals} = \frac{1}{N}\sum_{i=0}^{N}\left(S_0^i v_0^i - \sum_{j=1}^{n_i} {S^i}_j v_j^i\right)^2 + \frac{1}{N}\sum_{i=0}^{N}\left(\sum_{j=1}^{n_i}\left(p_0^i + \frac{1}{2}\rho(v_0^i)^2 - p_j^i - \frac{1}{2}\rho(v_j^i)^2\right)\right)^2$$

$$+\frac{1}{N_c}\sum_{i=N}^{N+N_h}\left(\sum_{j=1}^{n_i}(p_0^i + \frac{1}{2}\rho(v_0^i)^2 - p_j^i - \frac{1}{2}\rho(v_j^i)^2 - \Delta p_{heart})\right)^2$$

$$+\frac{1}{N_к}\sum_{i=N+N_h}^{N+N_h+N_c}\left(\sum_{j=1}^{n_i}(p_0^i + \frac{1}{2}\rho(v_0^i)^2 - p_j^i - \frac{1}{2}\rho(v_j^i)^2 - \Delta p_{capillaries})\right)^2$$

where N_h - number of collocation points with Δp_{heart}, N_c - number of collocation points with $\Delta p_{capillaries}$, N – number of remaining collocation points, n_i – number of edges, ${S^i}_j, v_j^i, p_j^i$ – parameters for i-th point.

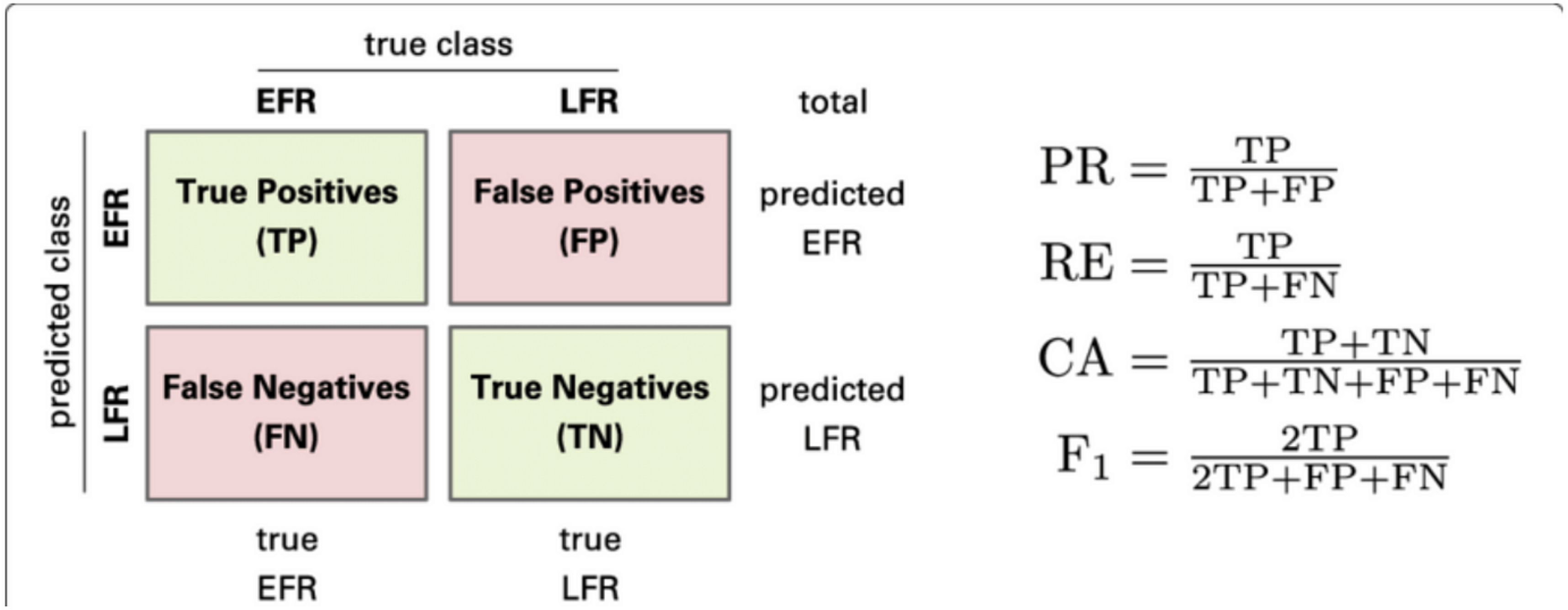
Model evaluation

Model efficiency:

- Loss functions

Prediction accuracy:

- F1-score
- Accuracy
- Precision
- Recall



$$\mathrm{PR} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$$

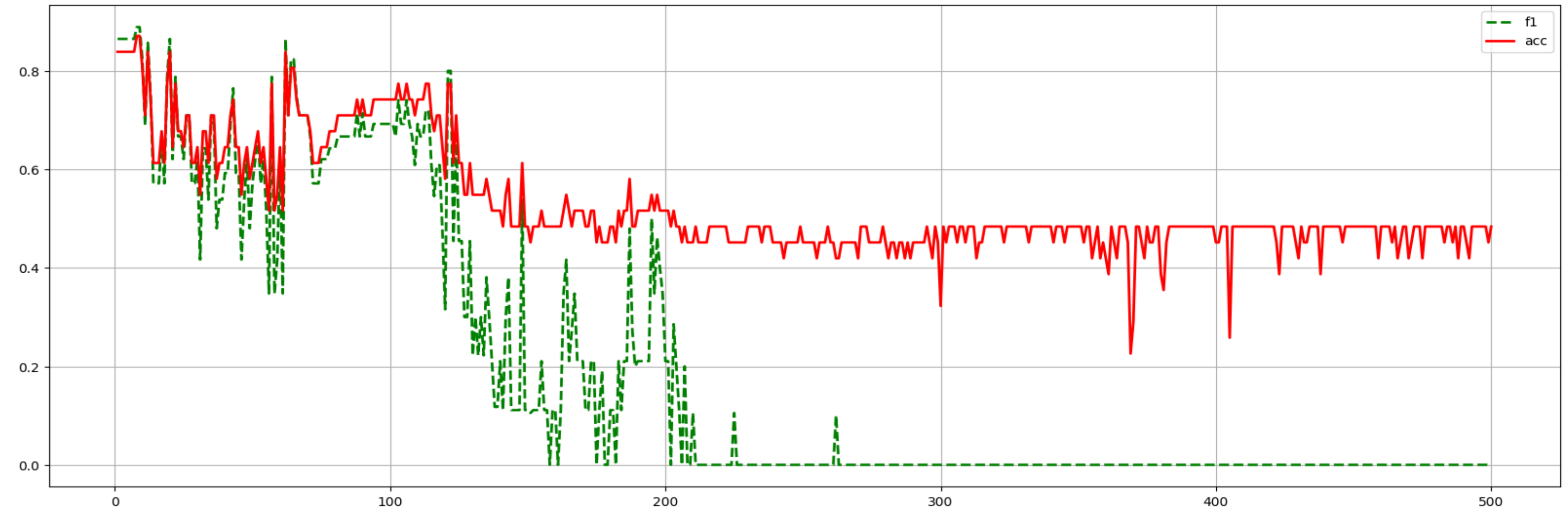
$$\mathrm{RE} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$$

$$\mathrm{CA} = \frac{\mathrm{TP+TN}}{\mathrm{TP+TN+FP+FN}}$$

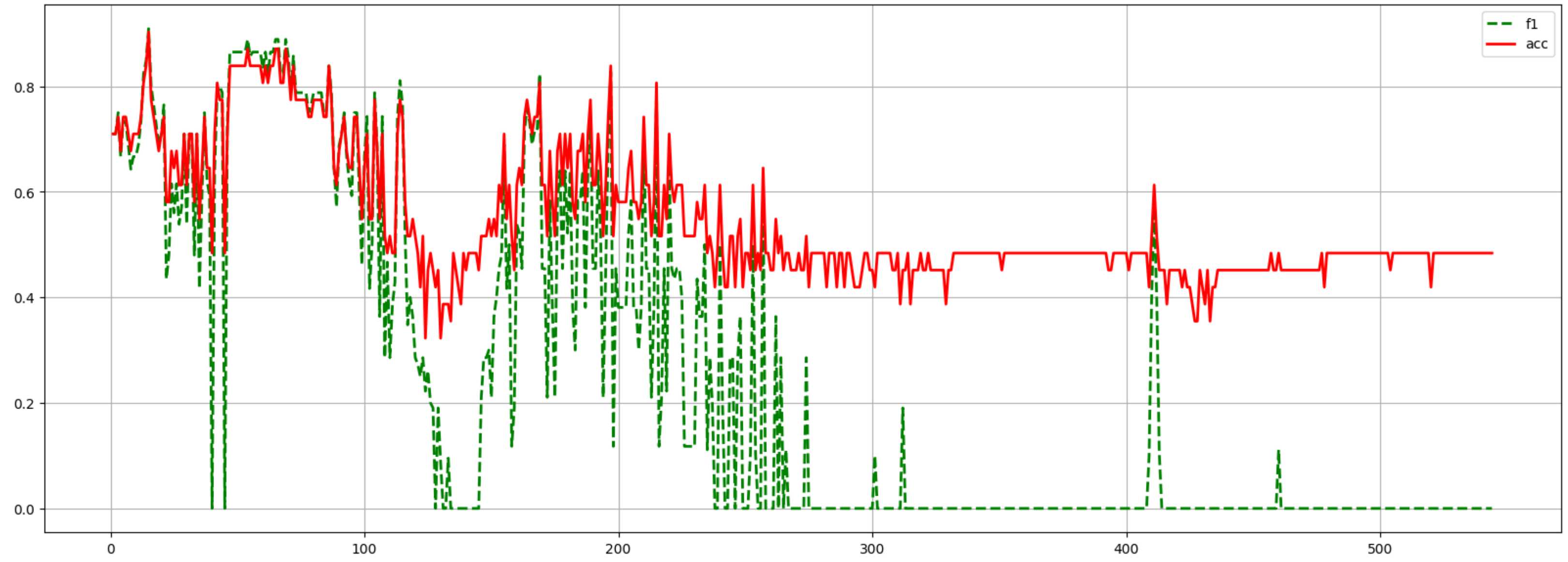
$$\mathrm{F}_1 = \frac{\mathrm{2TP}}{\mathrm{2TP+FP+FN}}$$

Confusion matrix. Exemplified CM with the formulas of precision (PR), recall (RE), accuracy (CA), and F 1-measure

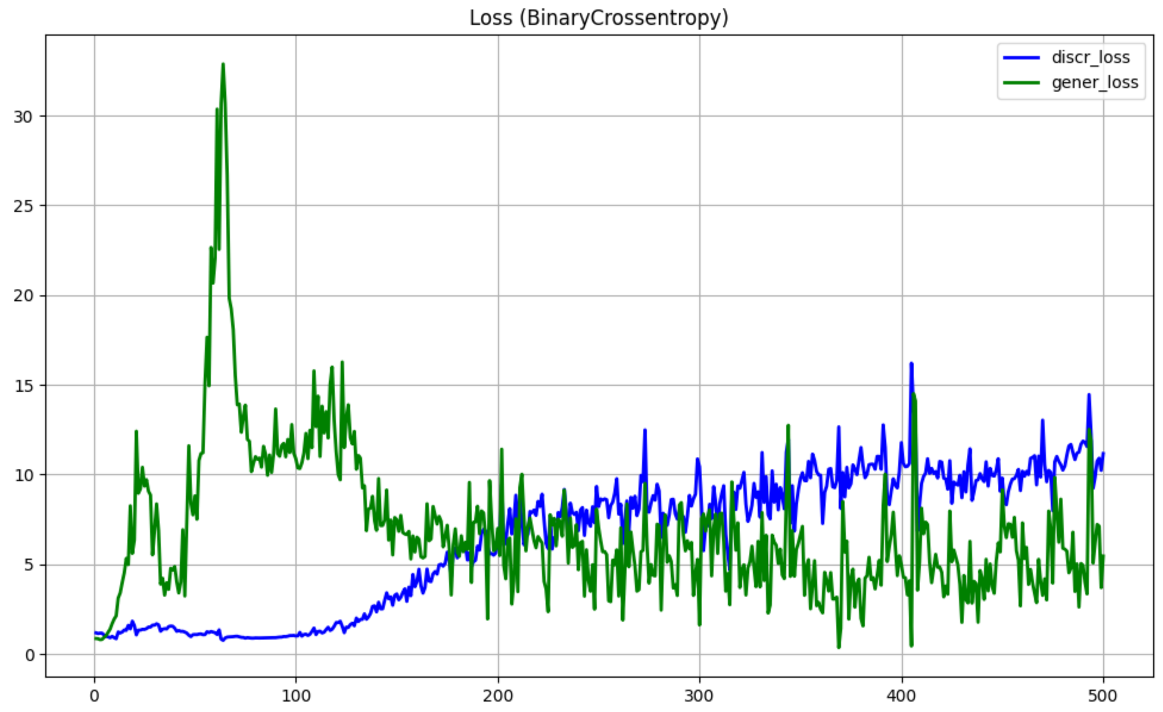
Generated GAN images quality could be estimated by sight.



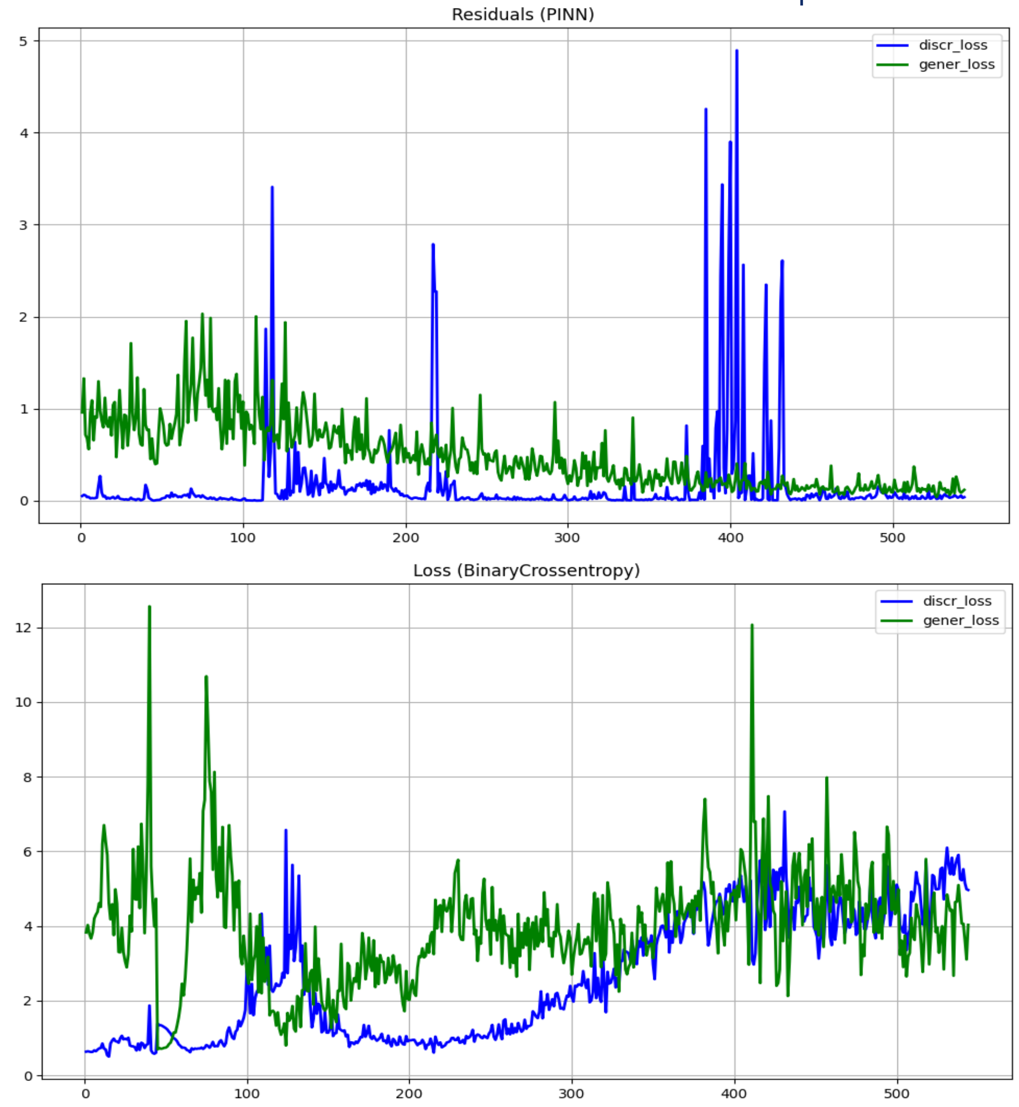
Standard GAN



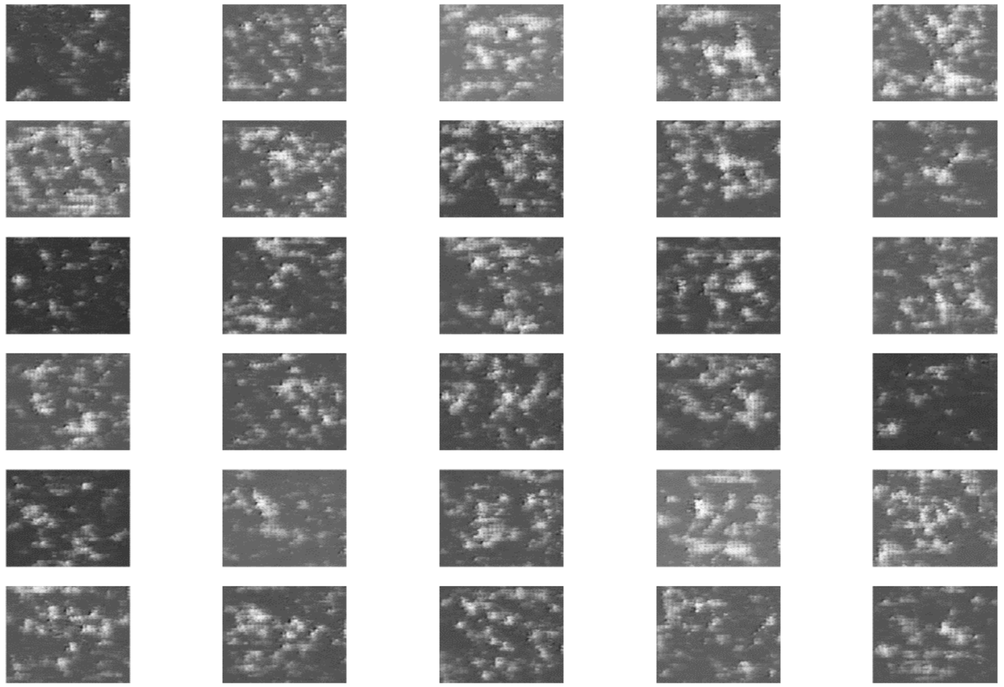
GAN with PINN

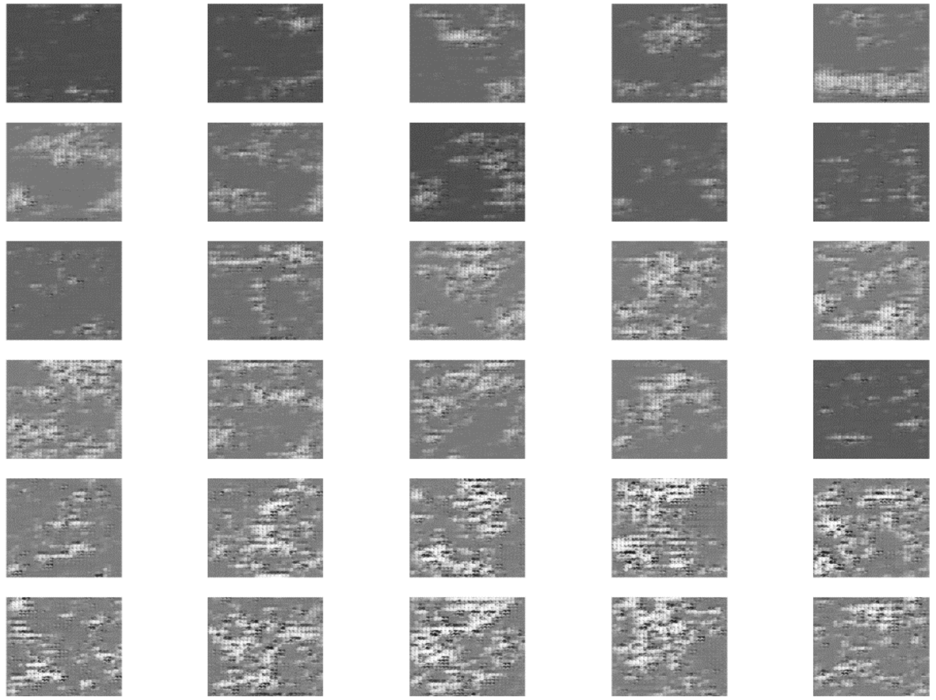


Standard GAN



GAN with PINN


Standard GAN


GAN with PINN

Conclusion

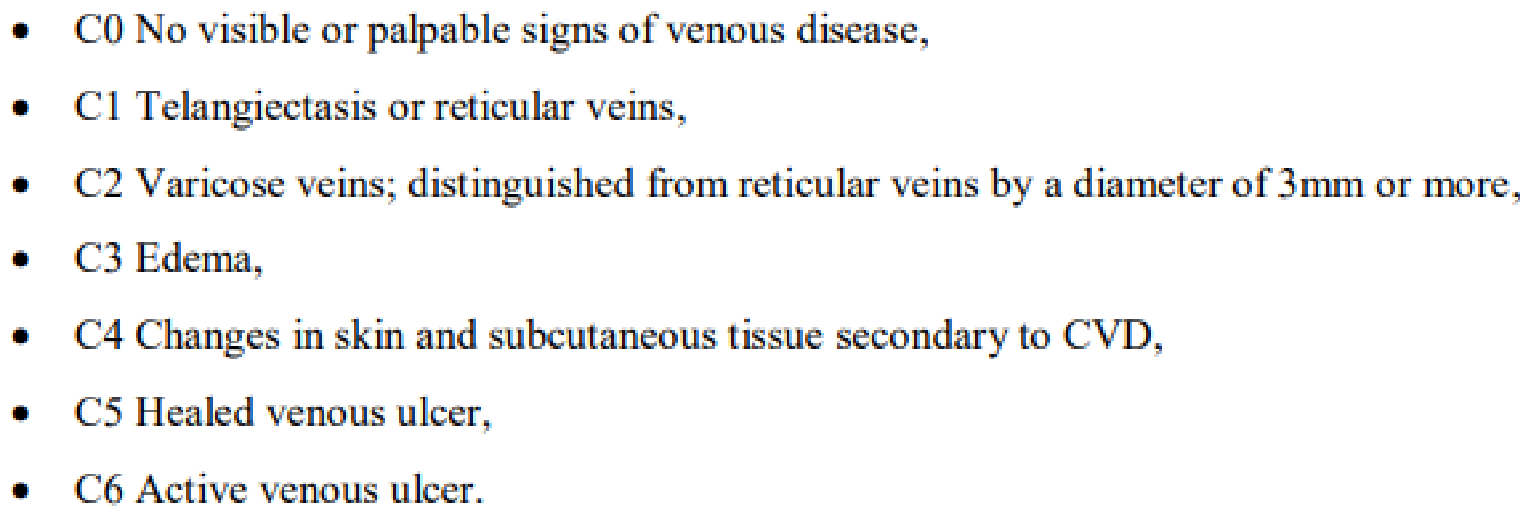
- Including physics information about human cardiovascular system and hemodynamics laws improves the model's efficiency
- The results are still too low for implementing the model into real-world application
- Disadvantages of the models have been detected, determining the ideas for further work

Further work

- Limiting blood flow parameters in specified intervals
- Implementing full CV graph into the model
- Transfer learning

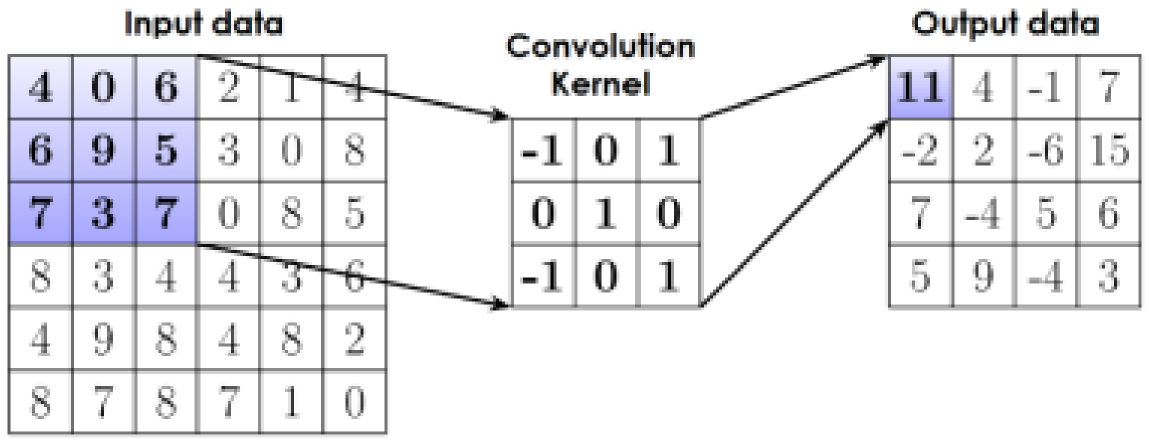
Thank you for attention!

CEAP classification

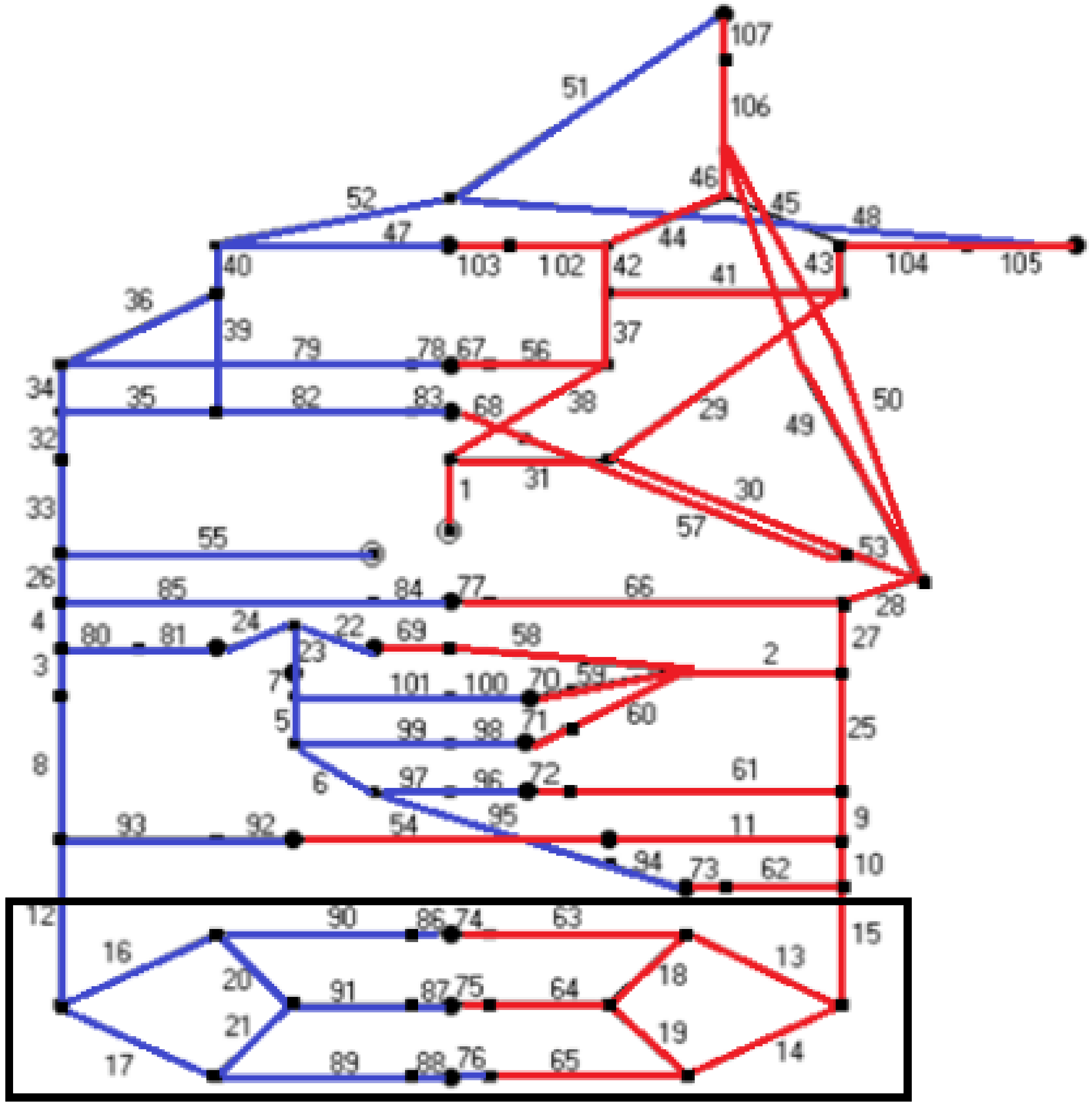


- C0 No visible or palpable signs of venous disease,
- C1 Telangiectasis or reticular veins,
- C2 Varicose veins; distinguished from reticular veins by a diameter of 3mm or more,
- C3 Edema,
- C4 Changes in skin and subcutaneous tissue secondary to CVD,
- C5 Healed venous ulcer,
- C6 Active venous ulcer.

Convolution process



Simplified cardiovascular graph



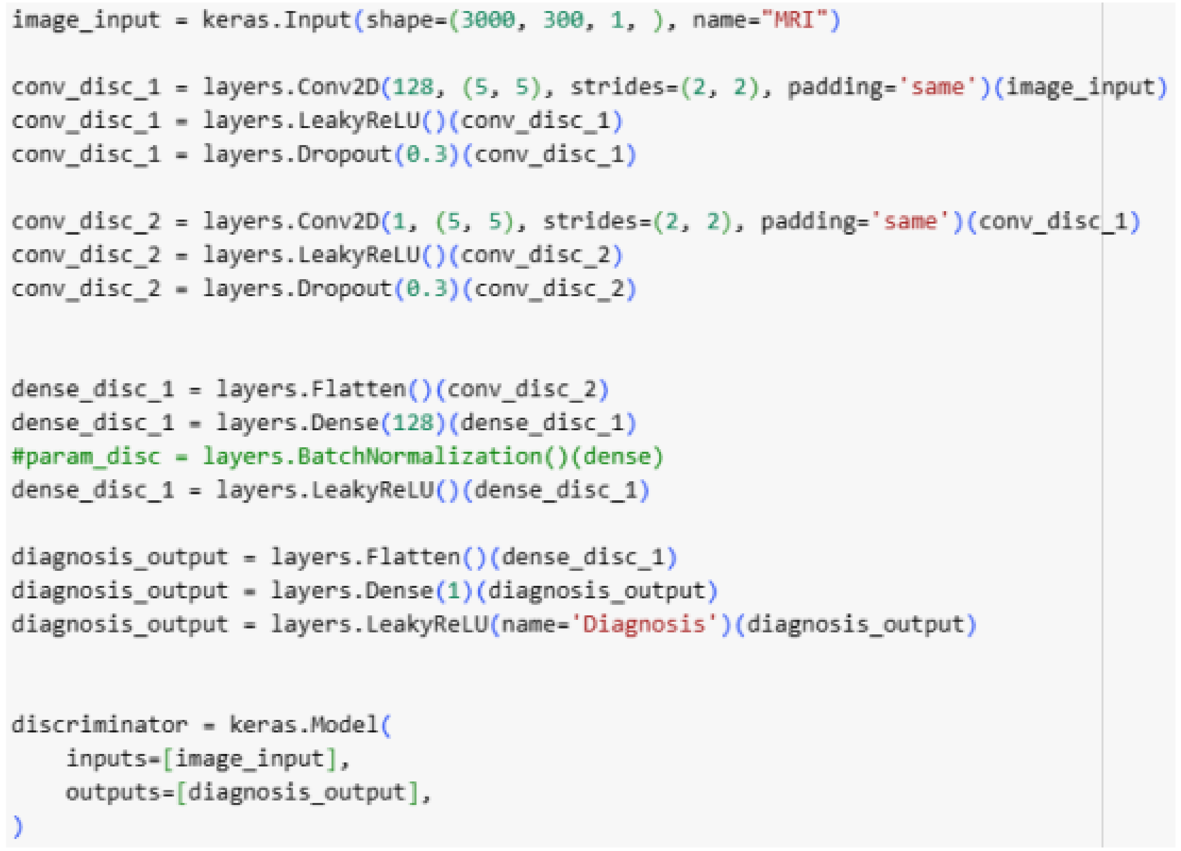
Binary cross-entropy

Model training takes place by the standard method of error backpropagation using the Adam optimizer with the learning step $\alpha = 1e - 04$, and the binary cross-entropy is used as the loss function:

$$BinaryCrossentropy = -(y \cdot log(p) + (1 - y) \cdot \log(1 - p)),$$

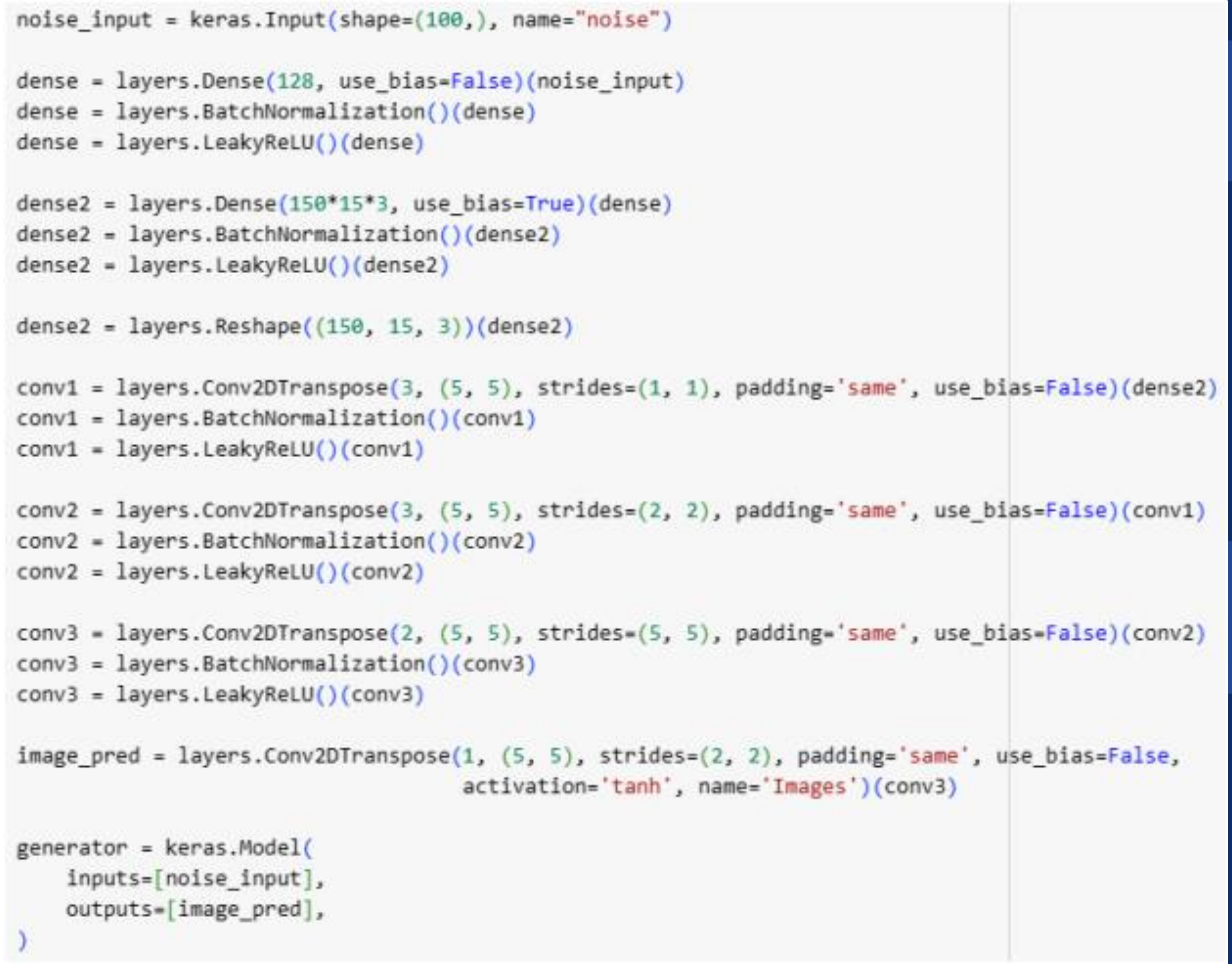
where y is a binary indicator (0 or 1), and p is the predicted probability of belonging to the class. The loss is calculated using TensorFlow functions.

GAN architecture



```
image_input = keras.Input(shape=(3000, 300, 1, ), name="MRI")

conv_disc_1 = layers.Conv2D(128, (5, 5), strides=(2, 2), padding='same')(image_input)
conv_disc_1 = layers.LeakyReLU()(conv_disc_1)
conv_disc_1 = layers.Dropout(0.3)(conv_disc_1)

conv_disc_2 = layers.Conv2D(1, (5, 5), strides=(2, 2), padding='same')(conv_disc_1)
conv_disc_2 = layers.LeakyReLU()(conv_disc_2)
conv_disc_2 = layers.Dropout(0.3)(conv_disc_2)


dense_disc_1 = layers.Flatten()(conv_disc_2)
dense_disc_1 = layers.Dense(128)(dense_disc_1)
#param_disc = layers.BatchNormalization()(dense)
dense_disc_1 = layers.LeakyReLU()(dense_disc_1)

diagnosis_output = layers.Flatten()(dense_disc_1)
diagnosis_output = layers.Dense(1)(diagnosis_output)
diagnosis_output = layers.LeakyReLU(name='Diagnosis')(diagnosis_output)


discriminator = keras.Model(
    inputs=[image_input],
    outputs=[diagnosis_output],
)
```

Figure 15. Developed discriminator code



```
noise_input = keras.Input(shape=(100,), name="noise")

dense = layers.Dense(128, use_bias=False)(noise_input)
dense = layers.BatchNormalization()(dense)
dense = layers.LeakyReLU()(dense)

dense2 = layers.Dense(150*15*3, use_bias=True)(dense)
dense2 = layers.BatchNormalization()(dense2)
dense2 = layers.LeakyReLU()(dense2)

dense2 = layers.Reshape((150, 15, 3))(dense2)

conv1 = layers.Conv2DTranspose(3, (5, 5), strides=(1, 1), padding='same', use_bias=False)(dense2)
conv1 = layers.BatchNormalization()(conv1)
conv1 = layers.LeakyReLU()(conv1)

conv2 = layers.Conv2DTranspose(3, (5, 5), strides=(2, 2), padding='same', use_bias=False)(conv1)
conv2 = layers.BatchNormalization()(conv2)
conv2 = layers.LeakyReLU()(conv2)

conv3 = layers.Conv2DTranspose(2, (5, 5), strides=(5, 5), padding='same', use_bias=False)(conv2)
conv3 = layers.BatchNormalization()(conv3)
conv3 = layers.LeakyReLU()(conv3)

image_pred = layers.Conv2DTranspose(1, (5, 5), strides=(2, 2), padding='same', use_bias=False,
                                    activation='tanh', name='Images')(conv3)

generator = keras.Model(
    inputs=[noise_input],
    outputs=[image_pred],
)
```

Figure 13. Developed generator code