Deep Learning Assignment
2AMM10
Technische Universiteit Eindhoven

Panagiotis Banos (1622773)

1 Task 2-Anomaly detection in an article database

1.1 Problem Description

We used a Wiki-CS dataset with no anomalous articles in the previous task, but we have a new scenario for this task. Wiki-CS is growing as users are adding new articles, some can be anomalous, and these articles should not be part of Wiki-CS because they are misplaced or not meeting the quality standards. We define anomaly as something we have not seen yet and is out of the ordinary. The goal of this task is to implement an anomaly detection model to identify anomalous articles.

1.1.1 Dataset Properties

In order to evaluate the performance of the model, we used an extended dataset with an additional 2000 articles, half of which are anomalous. We used this extended dataset just as an evaluation dataset. The evaluation dataset contains 12701 nodes and 302103 edges, from which 2000 nodes and 50176 edges are newly added.

1.2 Problem formulation and proposed approach

For this task we will be using a Variational Autoencoder to detect anomalous articles in the new part of the dataset.

We will be building a metric that will be used at testing articles in order to identify anomalous data. In order to do that we will feed 2000 test articles to the encoder part, which will output its μ and σ in the latent space. Then the decoder will sample z from a multivariate normal distribution whose mean and variance are those output by the encoder, feed z and propagate it into the decoder network. Since this is an unsupervized task, at its output the VAE will compare the distribution of the reconstructed data points to those he has learned during training and try to identify the anomalous ones.

We have built a model that consists of two parts, the encoder and the decoder. The encoder will encode each input variable x^d of d dimensions into a compressed lower dimensional latent representation space $z = (z_1, z_2 ... z_k)^T$ of according to the amount of compression k we decide. Since this is a VAE and not a vanilla autoencoder, the encoder will output parameters to $q_\theta(z|x_i) \ \forall i \in X$ with a latent variable z that is computed from a mean μ, a $\log(\sigma)$ and ε such that $z = \mu + \sigma \odot \varepsilon$ where $\varepsilon \sim Normal(0,1)$. The decoder will use z as its input, propagate it through its network and output the parameters to a probability distribution $p_\phi(\hat{x}|z)$.

To calculate how well the autoencoder has learned to compress and reconstruct each input x we will be using ELBO(Evidence Lower Bound) that contains the following:

- The reconstruction loss which is calculated by the $MeanSquaredError(\hat{x}_i, x_i)$
- The Kullback-Leibler divergence between the encoder's distribution $q_\theta(z|x_i)$ and $p(z)$
- A constant term

$$MSE = \sum_{i=1}^{n} (x_i - \hat{x}_i)^2 \tag{1}$$

$$D_{KL} = q(z|x_i)||p(z) \tag{2}$$

$$c_{term} = output_{channels} * 0.5 * \log \pi \tag{3}$$

At this point lets move a bit backwards again. We know that the decoder samples from the z of the encoder. But where does the encoder sample from? The answer is, that it does so from the input x_i. This can be expressed using Baye's Theorem for a single input datapoint like this:

$$P(z|x_i) = \frac{P(x_i|z)P(z)}{\int_z P(x_i|z)P(z)dz} \tag{4}$$

The main problem here is that computing the integral of z is computationally really difficult. In order to avoid doing so we are introducing a new function $q(z|x_i)$ which approximates the real value of $p(z|x_i)$.

The Kullback-Leibler divergence can also be written as:

$$D_{KL} = (\mathcal{N}(\mu, \sigma) \parallel \mathcal{N}(0,1)) = \sum_{x \in X} \left(\sigma^2 + \mu^2 - \log \sigma - \frac{1}{2} \right) \tag{5}$$

By Jensen's inequality, the $D_{KL} \geq 0$. That means, that minimizing the D_{KL} results in maximizing the ELBO. As maximizing is not possible for a loss function in PyTorch, we will instead be minimizing the negative ELBO.

To tune the autoencoder, ideally we would try and minimize the loss for both the reconstruction loss and the KL loss. But as we mentioned, minimizing the reconstruction loss is incredibly difficult due to some intractable terms, we will only be focusing on minimizing the KL loss.

1.2.1 Network architecture

While initially, an attention based neural network was constructed using the model from task_1, we chose to instead build a graphSAGE network because we were getting some mixed results. Nevertheless, the architecture is as follows:

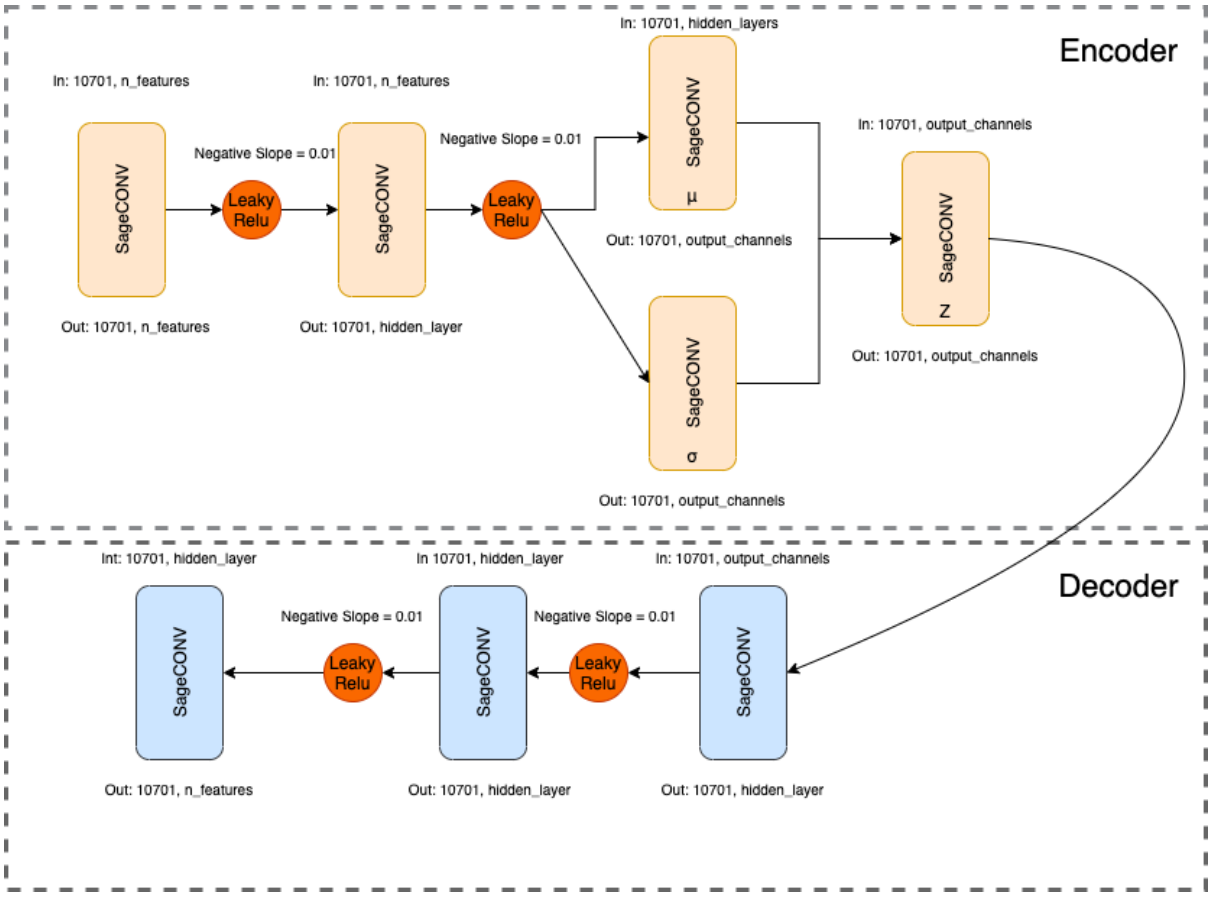

Figure 1: VAE for task 2

For the autoencoder we try to keep the representation of each datapoint x as close as it is to the original that is why we have not included anything other than a LeakyReLU activation function inside the network. As we know, we are starting with 300-D embeddings. As these embeddings move through the network, we keep their original size and compress them to 64-D for the latent space. Concerning the choice of layers, we assume that keeping the topological ordering and the neighbourhood relationships to be very important when constructing the autoencoder, thats why every layer in the network is also aware of the adjacency matrix. We try to keep the information contained by the embedding as true to its original form as possible.

1.3 Results

First lets see how the loss decreases while training the autoencoder. See in Figure 5, Figure 6 and Figure 7 how the each component of ELBO is diminishing.

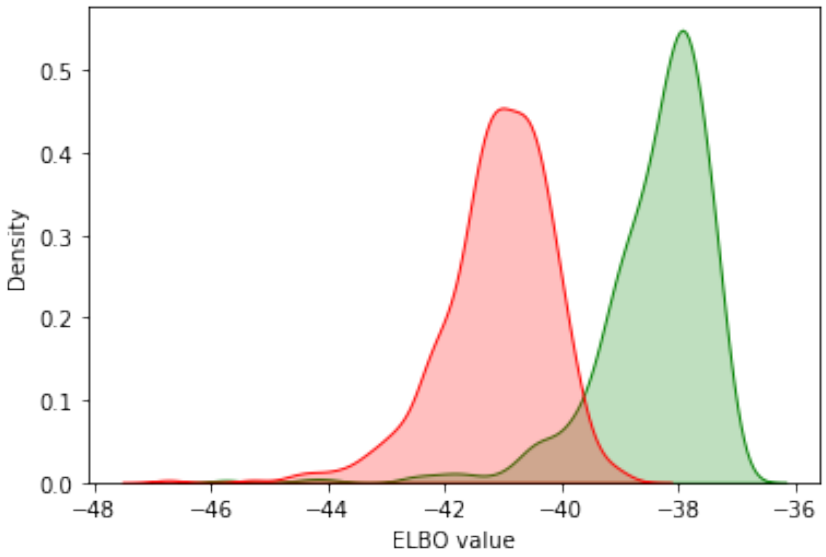

Figure 2: ELBO densities of normal(green) vs anomalous(red) data

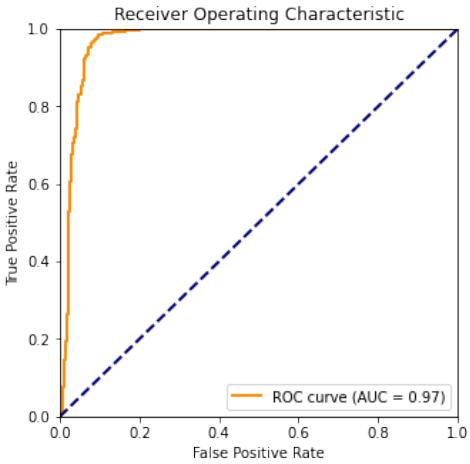

Figure 3: Area under curve plot.

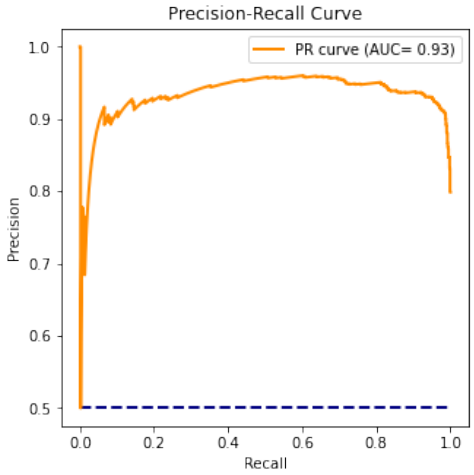

Figure 4: Precision-Recall curve.

Looking at the results in Figure 2 we can see that there is a clear seperation between the densities of anomalous and normal results with some minor overlapping.The distributions are very seperable which intuitively means that the autoencoder is able to discern between real and anomalous data. We can also verify this quantitatively by looking at Figure 4 and Figure 3 which show a AUC of 0.97 for the former and 0.93 for the later. Them being so close to 1 are good indicators of a network capable to recognize anomalies.

1.4 Implementation

The implementation can be found in gnn-and-vae.ipynb

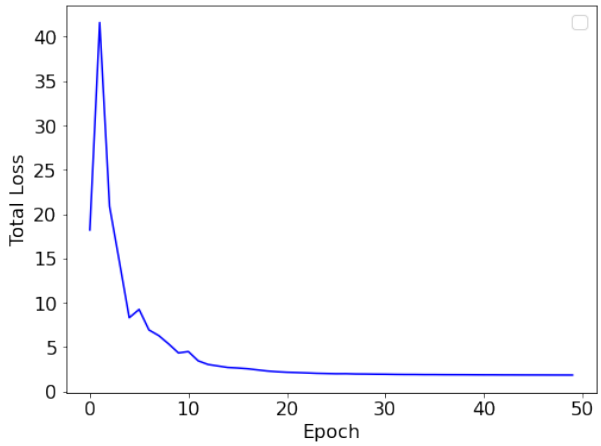

Figure 5: Total Loss with 50 epochs

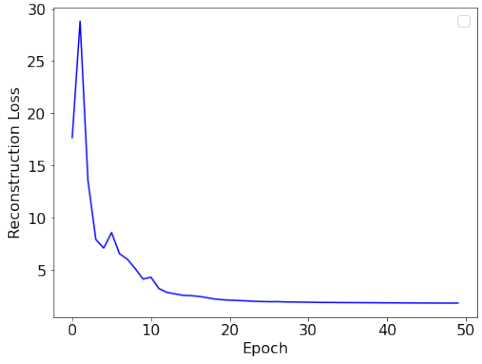

Figure 6: Reconstruction Loss with 50 epochs

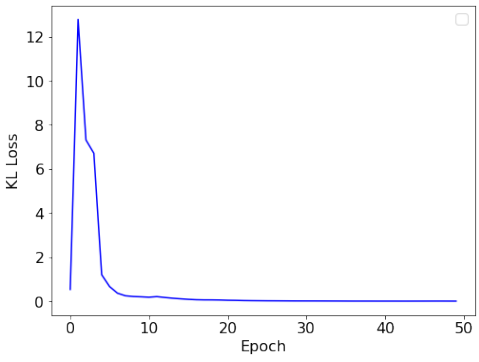

Figure 7: KL Loss with 50 epochs