# Drug Discovery via Graph Neural Networks

Anastasia Nichiporchuk

1 февраля 2023 г.

# Содержание

## Abstract

The paper considers a method for predicting solubility in water for various molecules using graph neural networks. The architecture of three graph neural networks is given, neural networks training on the AqSol dataset is considered, a test is conducted on the ESOL benchmark.

// TODO: дописать про генеративную часть


## 1 Introduction

The need to treat illness and maintain health has accompanied humans throughout their existence as a species. The development of the pharmaceutical industry has improved the quality of life, increased its duration, and made it possible to cope with diseases that were previously considered incurable. The development of vaccines for preventive protection against disease, both for the individual and for society as a whole, as demonstrated by the recent COVID-19 pandemic, is also crucial.

Modern drug development is inextricably linked to the use of various technologies. Pharmaceutical companies compete for primacy in the development of drugs for diseases that place the greatest burden on the health care system. Before the discovery of penicillin, these were mostly diseases of an infectious nature. Nowadays, cardiovascular diseases, cancer and neurodegenerative diseases account for the largest percentage.

Because time is of the essence in drug development, pharmaceutical companies use a variety of cutting-edge research methods. In addition to laboratory research, computer simulations of different processes, mathematical modeling, and other methods that can reduce the amount of laboratory research are actively used.

This is due to the fact that conducting any laboratory research requires time and financial expenses. Also, you need highly qualified specialists who are able to conduct this research. For example, a substance can be tested in the laboratory for a fairly small number of properties, so you need to try empirically to exclude those tests that are known to give a bad result.

With the development of computing power, as well as biotechnology and cheminformatics, it has become possible to use neural networks as such empirical

methods. A neural network trained for a specific task can predict certain data and minimize unsuccessful laboratory tests.

## 2 Solubility

Solubility is the maximum amount of solute that can be dissolved in a known amount of solvent at a particular temperature.

Factors affecting solubility:

- temperature. Solubility depends significantly on temperature and can be changed by increasing or lowering the temperature. As a rule, the range from 20°C to 100 °C is considered suitable for dissolution in water.
- pressure. Pressure has practically no effect on the solubility of solids, but for gaseous substances, solubility is directly proportional to the pressure of the gas over the liquid.
- the nature of the solvent and the solute. Water belongs to polar solvents and, accordingly, dissolves polar substances better.

Aqueous solubility, denoted by S, or its logarithm value log Is very important molecular property. Identification of molecules with undesirable solubility in water at early stages is of great importance in drug development and in other related fields of pharmacy, since solubility affects the processes of absorption, distribution, metabolism and elimination (ADME).

A number of existing QSPR models developed to predict solubility show that solubility is related to experimental indicators such as melting point and separation coefficient. However, the data obtained experimentally often contain measurement errors. This increases the difficulty of predicting solubility.

The most using formula of solubility is:

$$logS = 0.5 - 0.01(MP - 25) - logP, \tag{1}$$

where MP is melting point and logP is the log of the octanol-water partition coefficient.

Both partition coefficient and aqueous solubility reveal how absolute dissolves in a solvent.

Solubility of molecules in water is an important property in terms of drug development. Substances with good solubility can be incorporated into drugs "as is,"while substances with low solubility require the addition of various impurities to increase the final solubility.

The most popular forms of drugs are tablets and solutions for injection. It is not uncommon for laboratory tests to find a substance that has the desired effect (capable of acting as a medicine for a specific disease). Such a substance needs to be submitted for clinical trials already in the dosage form in which it is supposed to be taken by patients. At this stage there may be a hitch, precisely because the active ingredient is poorly soluble in water, and it is not technologically easy to make a tablet from it.

Solubility can also affect the absorption of the active ingredient and its stability while in the dosage form. Thus, it can be determined that the better the solubility of the substance in water, the easier and cheaper it will be to use to make the drug.

# 3 Graph Neural Networks

Graph neural networks are a class of neural networks that allow you to apply deep learning techniques to a variety of data that can be represented as a graph. Graph data has no regular structure, like pictures (it is known exactly how many neighbors a pixel has) or texts (there is a sequence of words in a sentence). Therefore, classical neural networks are not well suited to this kind of data. At the same time, the variety of such data in the real world is very large - as an example, we can consider social networks, routes, and molecular structures.

There are several different types of graph problems:

1. node-level prediction
2. edge-level prediction
3. graph-level prediction
4. community detection (the problem of searching for clusters in a graph).

The first type can include various tasks of prediction and classification of

graph vertices, for example, prediction of the topic of an article on the basis of citations.

The second type includes tasks related to a pair of vertices or edges, the so-called "link prediction"tasks. As an example of such a problem, we can consider recommendation systems that predict whether a user is suitable for a service offered.

The third type, graph prediction, involves problems where a graph is treated as an object and the output is an estimate of its property. These can be both classification and regression problems. Such problems are often used when considering molecules to predict their properties.

Clustering tasks belong to unsupervised learning and can be used, for example, to distinguish groups of familiar users in social networks.

Below we will consider in more detail in which areas graph neural networks are used.

## 3.1 Graph neural networks in different tasks

**Computer vision**

In computer vision classical neural network models are actively used, in particular CNN. However, if we consider an image as a graph, in which each pixel is a vertex of the graph, and the neighborhood with other pixels determines the presence of edges between the corresponding vertices, it becomes possible to apply GNN for appropriate tasks related to images.

In addition, graphs are actively used for more complex tasks involving relations between image objects. One such task is scene graph generation, in which the goal of the model is to parse an image into a semantic graph consisting of objects and their semantic relations. From the image data, the model detects and recognizes objects and predicts semantic relationships between pairs of objects.

Another important task is to match objects in scenes. It makes it possible to compile three-dimensional models from the available video footage, which are used for the modeling and mapping (SLAM). If certain hardware conditions are met, such models can also work in real time.

### Natural language processing

In natural language processing it is possible to represent a sequence of words as a simple graph. Such a format seems quite intuitive, but not much in demand. The representation of natural language in the form of graphs, such as dependency graphs, constituency graphs, AMR graphs and knowledge graphs, as well as text graphs containing multiple hierarchies of elements, i.e. document, sentence and word.

In some tasks, when graph contains text information. associated with each node, it is useful to combine text and network embeddings to increase total accuracy of the model [3].

### Recommendation systems

Recommendation systems are designed to analyze user interests and predict what will be interesting to a particular user at the moment.

Recommendation systems are divided into two fundamentally different types - off-line models (detection of global patterns, personalized model for a specific object) and online models (fast response, detection of current actual trends).

Graphs are actively used for offline models. are used by large companies in their products (Alibaba, Uber, Pinterest) to prepare personalized offers for users based on their actions and reactions. In this case, the user and the product can be represented as vertices of the graph, and their interaction as an edge of the graph.

### Natural science

Chemistry, biology, and other natural sciences have been and remain one of the main driving forces in the development of GNN. The reason for this is that these sciences pose problems closely related to data in the form of graphs - molecules and materials. Also, these problems often require significant computational power when solved by classical methods, so it is appropriate to use machine learning methods for them. In general, when comparing classical methods of machine learning and GNN, the latter show better results.

# 4 Recent work

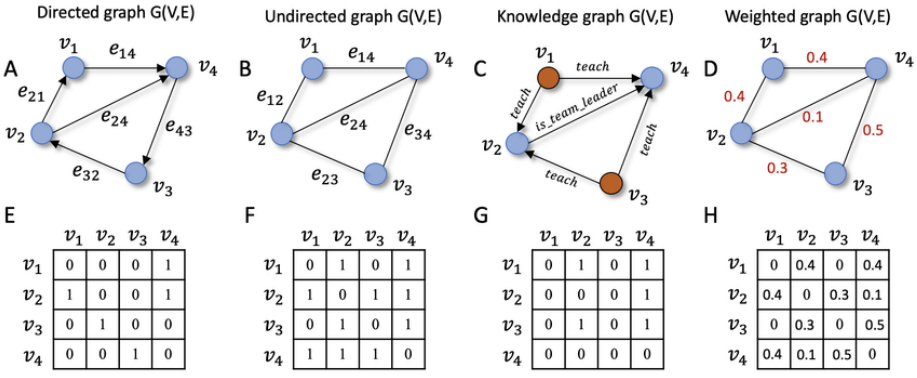
In this paper, the problems of predicting the solubility of a substance in water based on the structure of a molecule are considered. As described above, solubility is an important property for the production of drugs, which can be calculated using experimental (and therefore not very accurate) data. Therefore, it is advisable to use a neural network to predict this property.

Since the molecule is essentially a graph, it is possible to use a graph neural network in the problem, preserving the original structure of the input data.

// TODO: дописать тут про генеративную часть задачи

# 5 GNN approach

How could working with a graph using traditional machine learning methods look like? One could take the adjacency matrix of a graph, which is a matrix of zeros and ones (for graphs without weights) or a matrix of various numbers for graphs with weights. The vertices of the graph are located vertically and horizontally of such a matrix. If there is an edge between the vertices, units (or the weight of the edge) are set at the intersection of these vertices. If the vertices are not connected by an edge, zero remains in the matrix cell. Example of such matrix is below:



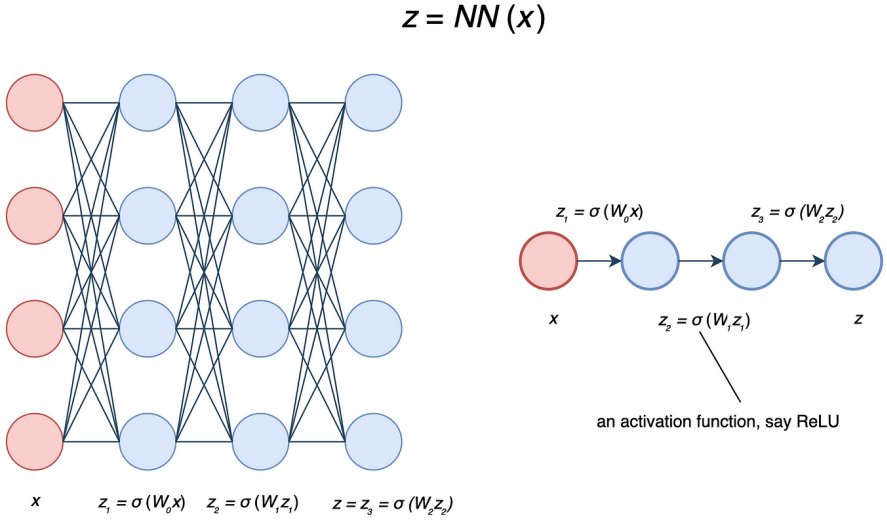

Intuitively, this seems to be a way out of the situation, but in fact, such a representation of the graph has a serious drawback. The vertices of the graph are not ordered, that is, it is impossible to uniquely arrange them by numbers 1, 2, 3 ... so that this order is always preserved. The vertices can be swapped (keeping the edges), while the graph will remain the same, and the adjacency

matrix will change, that is, we will get one object with many different sets of features.

For this reason, classical machine learning methods are poorly suited for working with graph structures.

## 5.1 Graph convolution neural networks (GCN)

GCNs are neural networks designed to perform convolutions over undirected graph data. Let's consider the main difference from CNN.



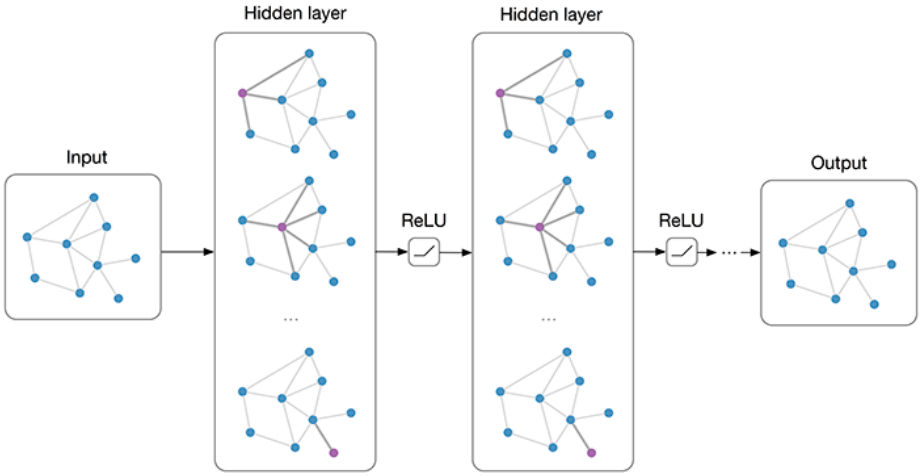

Here, x is a well-structured data - maybe, image or feature vector. And in GCN, the input will be a graph. Also, instead of inferring a single $z$, it infers the value $z_i$ for each node $i$ in the graph. And to make predictions for $Z_i$, GCN utilizes both $X_i$ and its neighboring nodes in the calculation.

In GNN, single layer can be described with the following equation:

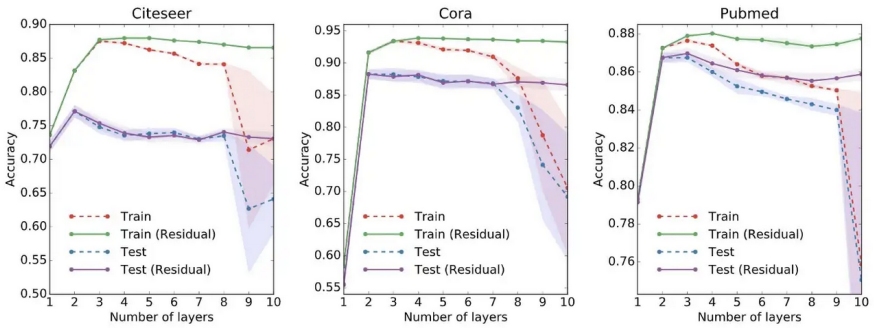
$$H^{l+1} = \sigma(D - \frac{1}{2}AD - \frac{1}{2}H^{(l)}W^{(l)}) \tag{2}$$

Here H represents the node embeddings within a graph, W represents a projection matrix, A represents the adjacency matrix of the graph with added self-connections, D represents a diagonal matrix that stores the degree of each node, and sigma represents an element-wise nonlinearity (like ReLU). The equation simply projects the node embeddings, computes each new embedding as the average of all of a node's neighbors, and applies an element-wise nonlinearity

to the new embeddings. These operations comprise a single graph convolutional layer, and these layers can be stacked to build a GCN:




During some tests, in [KIPF-WELLING] was obtained, that the best results are obtained with a 2- or 3-layer model. Besides, with a deep GCN (more than 7 layers), it tends to get bad performances (dashed blue line). One solution is to use the residual connections between hidden layers (purple line)




## 5.2 GAT

Graph Attention Network (GAT) [2] is based on the idea of computing the hidden representations of each node in the graph by attending over its neighbors using a self-attention strategy.

By staking layers, nodes can attend to their neighborhood node features by specifying different weights to different nodes and not needing costly matrix operation or knowing the graph structure upfront.

To compute one single GAT layer, we initialized weight matrix $W$ and define

the equation to compute attention coefficient:

$$e_{ij} = a(Wh_i, Wh_j), \tag{3}$$

which means the inportance of node j's features to node i. In this equation $a$ called shared attentional mechanism. Then we computes $e_{ij}$ only for neighbors and use them to compute $\alpha_{ij}$:

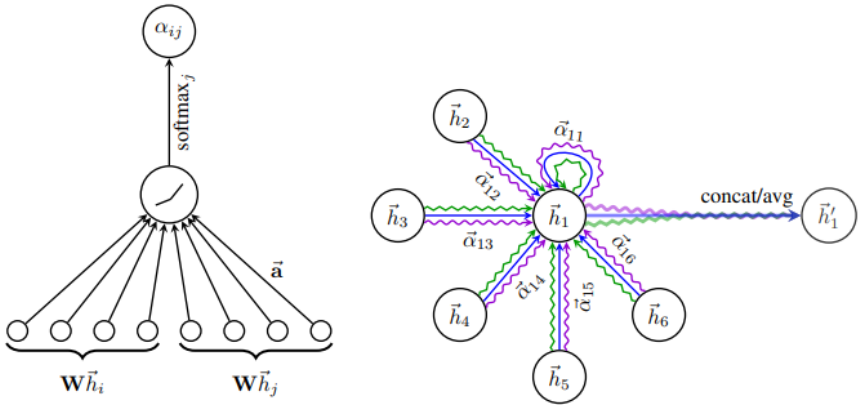
$$\alpha_{ij} = softmax(e_{ij}) \tag{4}$$

In the original work [GAT] authors use LeakyReLu function as attention mechanism.

The aggregated features of each node can be compute by equation:

$$h_i = \sigma(\sum_{j \in neibs(i)} \alpha_{ij} W h_j), \tag{5}$$

where sigma is some nonlinear activation function. This process is illustrated by the image:



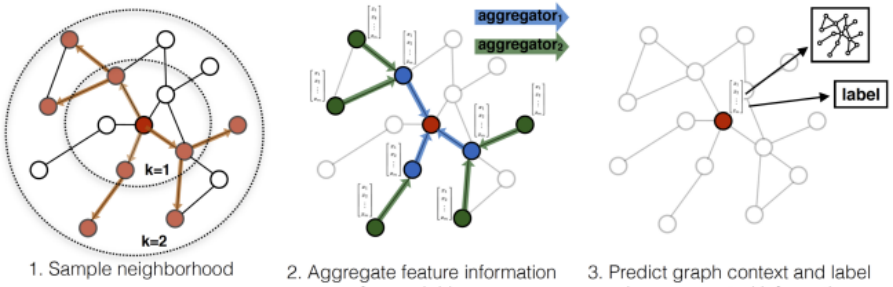

## 5.3 GraphSAGE

GraphSAGE [1] is based on SAmpling and aggreGatE, where it samples only a subset of neighboring nodes at different depth layers. It then aggregates the neighbors of the previous layers using an aggregator.

At each iteration, nodes aggregate information from their local neighbors based on sampling. Each aggregator function aggregates information from a different number of search depth, away from a given node. As a result, nodes incrementally gain more and more information from further reaches of the graph.

The goal of GraphSAGE is to learn a representation for every node based on some combination of its neighbouring nodes.

GraphSAGE is that it was the first work to create inductive node embeddings in an unsupervised way. At the first stage, forward propagation occurs, which results in embeddings for the nodes of the graph. Next, aggregator functions aggregate feature information from nodes, which were embedded earlier. Then we use aggregated information to predict graph context.




This architecture using for single SAGEConv layer, and GraphSAGE neural network is usually consists of some SAGEConv layers.

## 5.4 Selected models

My work uses three models with different layers and similar architecture: GCN, GAT and GraphSAGE with three layers and a linear output. As a function of the transition between layers, the $leaky_relu$ function showed the best results in the tests. Below is the GCN model code:

```
class GCN(torch.nn.Module):
    def __init__(self, num_features, hidden_channels):
        super(GCN, self).init()
        torch.manual_seed(42)

        self.conv1 = GCNConv(num_features, hidden_channels)
        self.conv2 = GCNConv(hidden_channels, hidden_channels)
        self.conv3 = GCNConv(hidden_channels, hidden_channels)
        self.out = Linear(hidden_channels*2, 1)
```

```
def forward(self, x, edge_index, batch_index):
    hidden = F.leaky_relu(self.conv1(x, edge_index))
    hidden = F.leaky_relu(self.conv2(hidden, edge_index))
    hidden = F.leaky_relu(self.conv3(hidden, edge_index))

    hidden = torch.cat([gmp(hidden, batch_index),
                        gap(hidden, batch_index)], dim=1)

    out = self.out(hidden)
    return out, hidden
```

GAT and GraphSAGE are the similar, but using GATConv and SAGEConv layers. There are two input parameters in this model. The first,

`num_features`

is the number of features, also this is the size of tensor with features. This parameters is obtains from dataset. Also, the second parameter,

`hidden_channels`

is the model's parameter and may vary. During the tests, a parameter with a value of 64 showed good results.

This model has a fairly simple architecture compared to, for example, MPNN (Message Passing Neural Network) or GAT (Graph Attention Network). She learns quite quickly, although the training time depends significantly on the size of the dataset. Metrics show a steady improvement in the loss function.

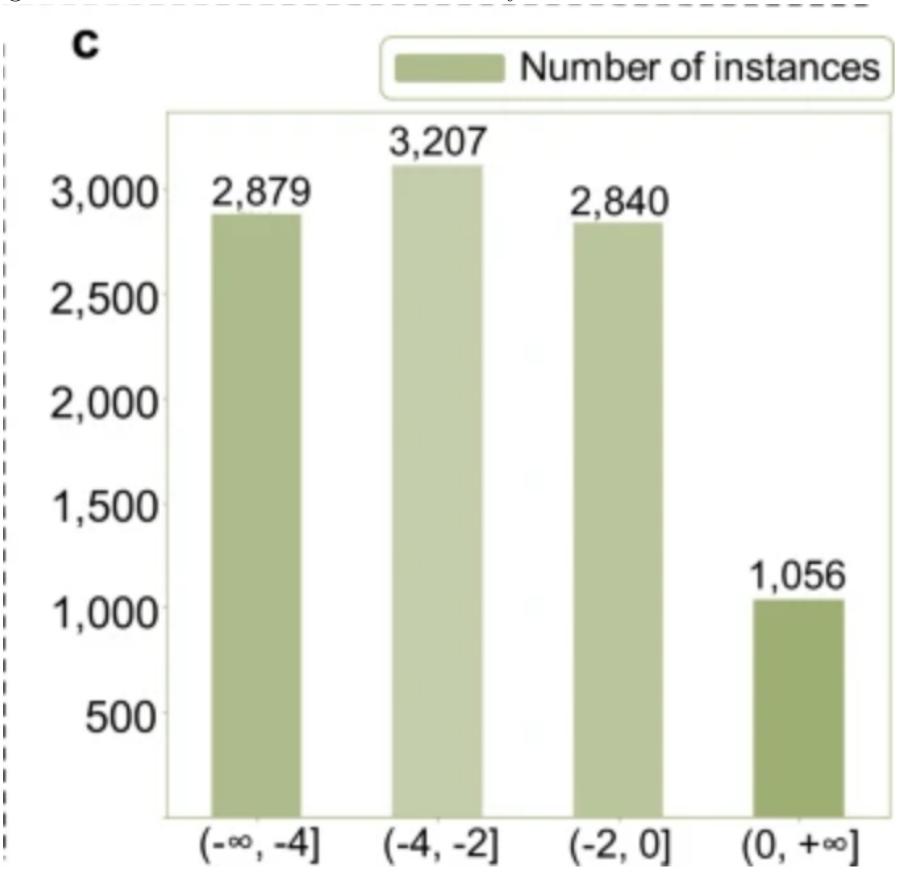
# 6 Datasets

A large number of datasets have been collected for tasks related to the study of molecules. Since this work is aimed at predicting a specific property of the molecule, accordingly, datasets related to solubility were selected.

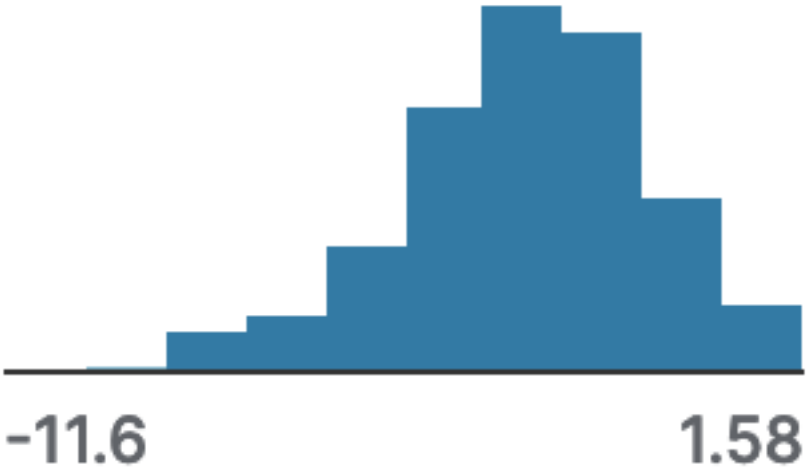
The model was trained on the AqSol dataset [https://www.nature.com/articles/s41597-019-0151-1 ], consists of aqueous solubility values of 9,982 unique compounds, along with some relevant topological and physico-chemical 2D descriptors.

To work with the model, it is important to have a SMILES string - one of the generally accepted formats for recording molecular structures in the form of a string, and AqSol contains such data.

There are several solubility groups in this dataset: compounds with 0 and higher solubility value (highly soluble), those in the range of 0 to -2 (soluble), those in the range of -2 to -4 (slightly soluble) and insoluble if less than -4. Image below shows the distribution of solubility values.




The ESOL (Delaney) dataset is used to test the performance of the model on data other than the training dataset. It consists of 1128 molecules presented in SMILES format and data on their solubility. The solubility distribution can be seen on the graph:




## 7 Obtained results

GCN models were chosen to predict solubility. The AqSol dataset was used to train the model, and the ESOL dataset was used to test the trained model. Parameters with which the model was trained:

$$hidden\_channels = 64$$

$$batch\_size = 64$$

$$optimizer = Adam$$

$$lr = 0.0007$$

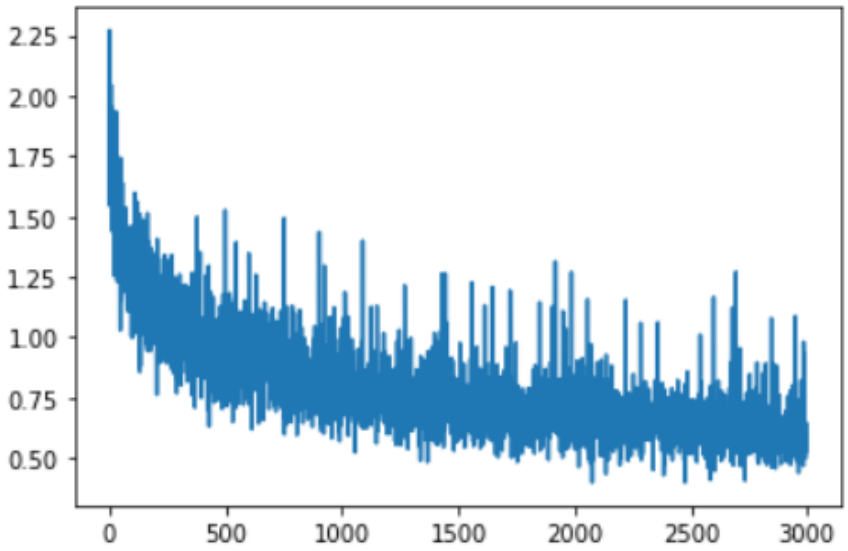
$$metric = MSE$$

$$epochs = 3000$$

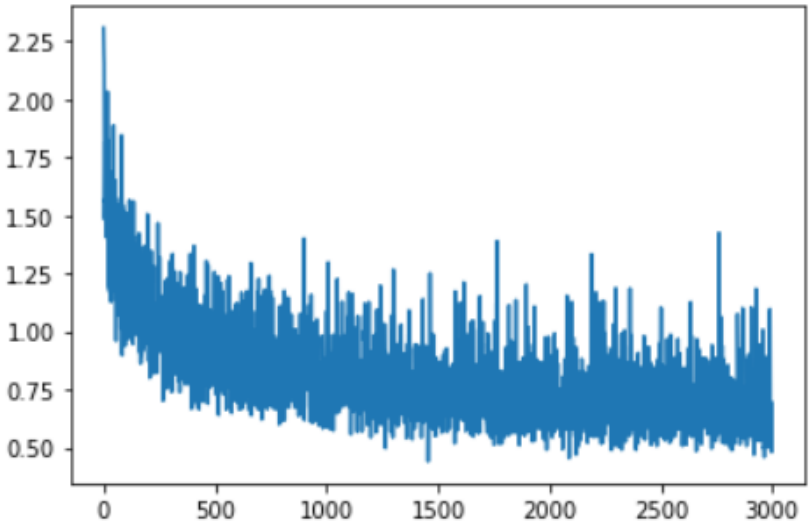
On the training dataset, the minimum value of the MSE metric corresponded to 0.254 with GraphSAGE.

| | GCN | GAT | GraphSAGE |
|---|---|---|---|
| Min MSE | 0.398 | 0.440 | 0.254 |
| Mean MSE | 0.791 | 0.804 | 0.536 |

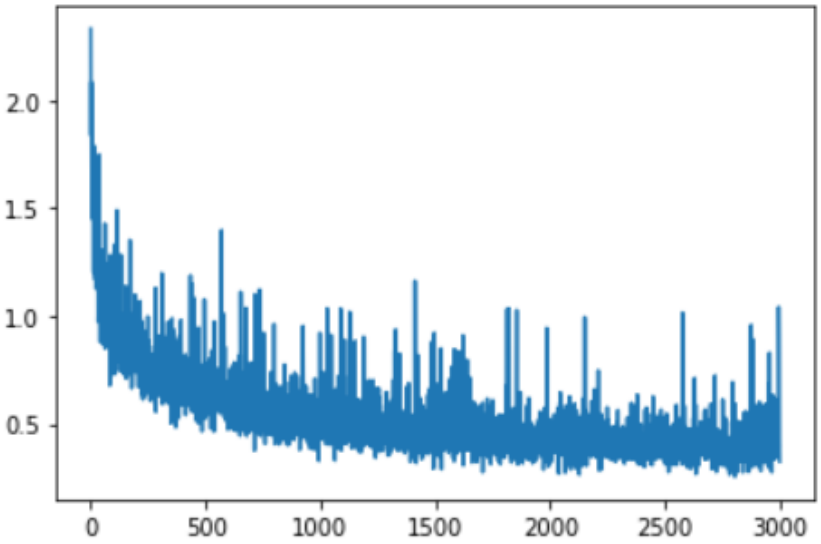
The change in values by epoch on GCN is reflected in the graph:




The change in values by epoch on GAT is reflected in the graph:



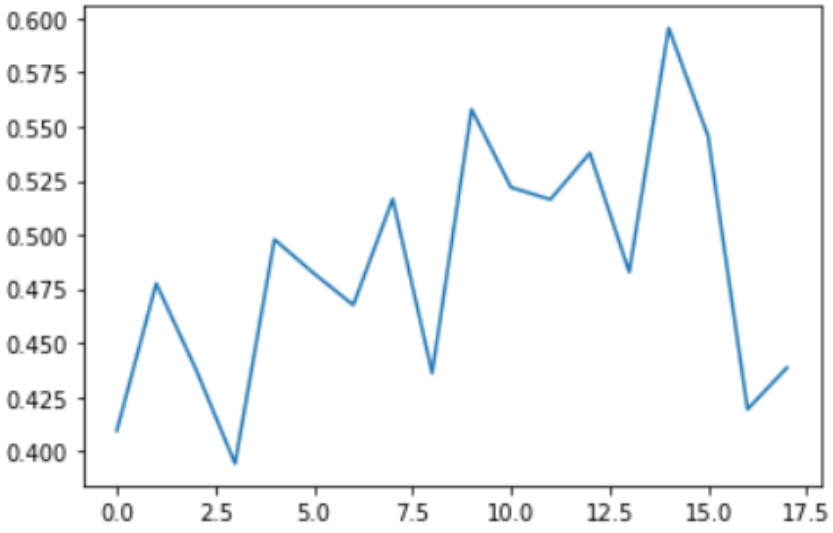

The change in values by epoch on GraphSAGE is reflected in the graph:



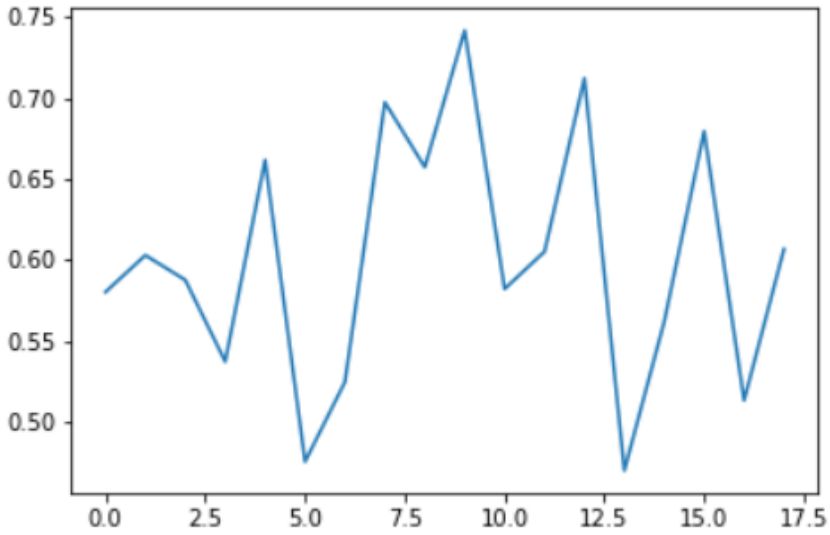

On the benchmark dataset, the minimum value of the MSE metric corresponded to 0.281 with GraphSAGE.

| | GCN | GAT | GraphSAGE |
|---|---|---|---|
| Min MSE | 0.394 | 0.469 | 0.281 |
| Mean MSE | 0.485 | 0.599 | 0.406 |

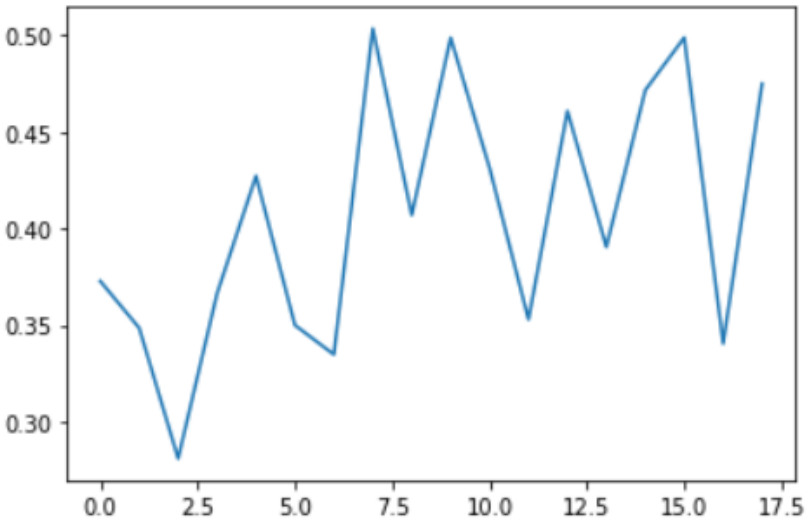
The change in values by epoch on GCN is reflected in the graph:




The change in values by epoch on GAT is reflected in the graph:




The change in values by epoch on GraphSAGE is reflected in the graph:




If we look for the solubility value by the formula using experimental data, the error can range from 0.5 to 1.6 logarithmic units. Therefore, such indicators for the metric are acceptable for the model and in some cases exceed the accuracy of the calculation using experimental data.

# 8 Conclusion

The paper considered the problem of predicting solubility in water by molecular structure. The substantiation of the relevance of the task is carried out. The GCN, GAT and GraphSAGE models were used for the solution, those training was performed on the AqSol dataset. The ESOL benchmark was used to test the model.

The trained models is able to predict solubility for molecules in the SMILES format.

Metrics show that the quality of the models is comparable to the error that occurs when calculating solubility based on experimental data, and in some cases the models shows better results.

If we compare the results shown by the models, then GraphSAGE showed the best minimum values for the training and test datasets and the best average values for all epochs.

# References

# Список литературы

[1] Hamilton, W., Ying, Z., Leskovec, J. (2017), *Inductive Representation Learning on Large Graphs*, Advances in Neural Information Processing Systems , 1024-1034.

[2] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y. (2017), *Graph Attention Networks*, 6th International Conference on Learning Representations.

[3] Makarov I, Makarov M, Kiselev D. 2021. *Fusion of text and graph information for machine learning problems on networks.* PeerJ Computer Science 7:e526.