Skoltech
ADASE NEURO

The Advanced Data Analytics in Science and Engineering Group

# Benchmarking of Machine Learning and Deep Learning approaches for Neuroimaging Data

**Team CogniNet**: Antipushina E., Knyshenko M., Kalimullin R., Morozova M.

**Mentors**: Petr Mokrov, Maria Zubrikhina
**Course instructors**: Evgeny Burnaev, Alexey Zaytsev

***Machine Learning course. Term 3. 2023***

# Outline

**Theory**

- Relevance of the task: problem statement and project goals
- Data description (fMRI, Connectivity matrices and ROIs)

**Practical part**

- Classical Machine Learning algorithms
- Deep Learning approaches

# Relevance of the task

**Problem statement:**

- At the moment, there is no clear understanding which methods will work best for fMRI data.
- Problem with publication results
- Globally: problem of identifying biomarkers isn't solved.

**Project goals:**

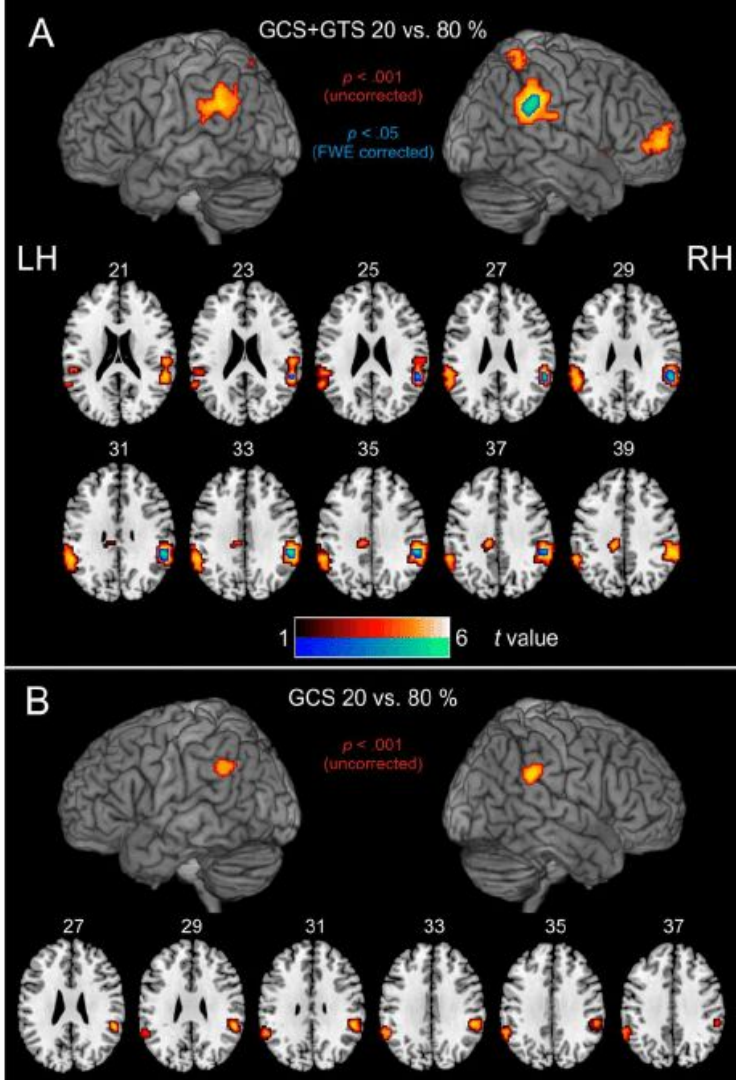
- Implement classic ML algorithms and fix the metrics.
- Compare with at least one graph approach.

**Relevant papers** 

# About fMRI

Functional magnetic resonance imaging (fMRI) measures the small changes in blood flow that occur with brain activity.

It may be used to examine which parts of the brain are handling critical functions, evaluate the effects of stroke or other disease, or to guide brain treatment.



# Datasets description

**COBRE Dataset+AAL labeling:** 163 connectivity matrices (116x116), 79 patients with schizophrenia and 73 healthy controls
**ABIDE Dataset+Craddock labeling:** 1035 connectivity matrices (200x200), 530 with Autism spectrum disorder (ASD) and 505 healthy controls.

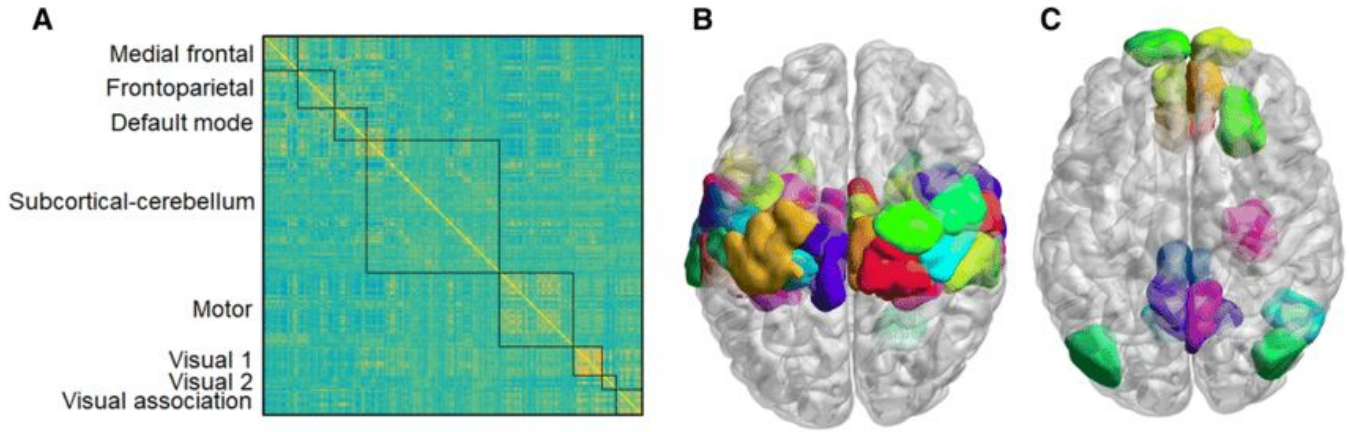



*Figure 1 - Brain network-related regions of interest (ROIs)*



*Lee YB, Yoo K, Roh JH, Moon WJ, Jeong Y. Brain-State Extraction Algorithm Based on the State Transition (BEST): A Dynamic Functional Brain Network Analysis in fMRI Study. Brain Topogr. 2019 Sep;32(5):897-913. doi: 10.1007/s10548-019-00719-7. Epub 2019 Jun 3. PMID: 31161473.*

# Classical Machine learning approach

Considered models: Logistic Regression; Random Forest; SVM; SVM + PCA; XGBoost.

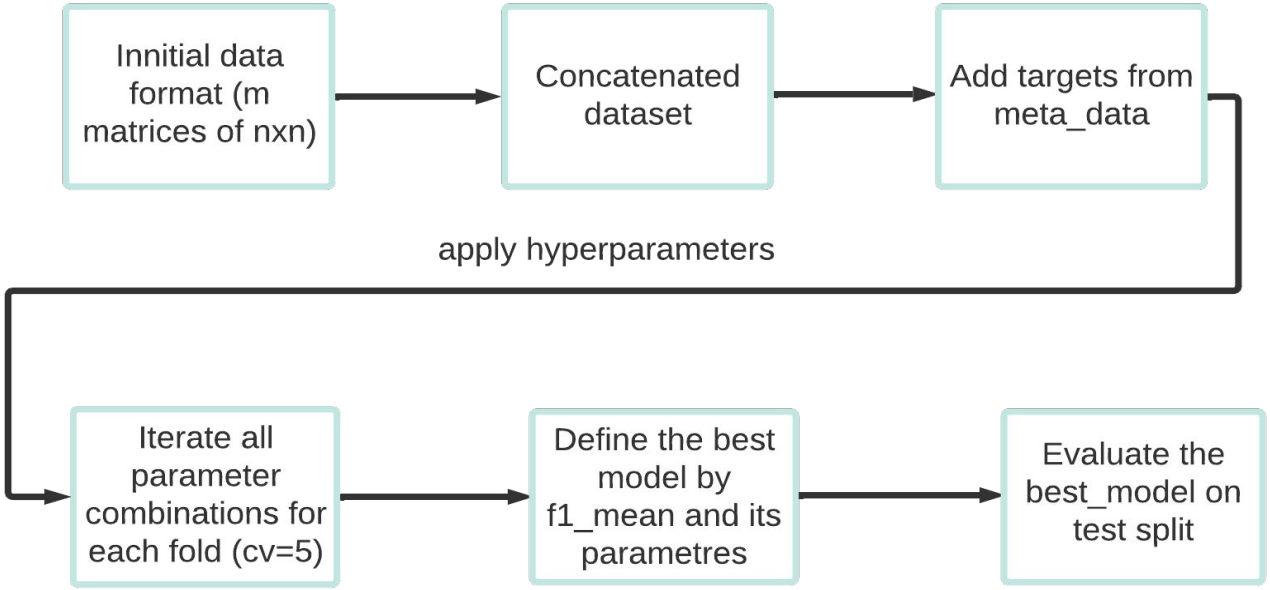



*Figure 2 - Code logic block diagram*

# Classical ML results for COBRE dataset

| | F1 score | Accuracy | ROC-AUC | Best hyperparameters |
|---|---|---|---|---|
| Logistic Regression | 0.73 ± 0.056 | 0.76 ± 0.045 | 0.76 ± 0.049 | 'class_weight': None, 'penalty': 'l2', 'C': 0.1, 'solver': 'liblinear' |
| Random Forest | 0.65 ± 0.073 | 0.69 ± 0.064 | 0.69 ± 0.065 | n_estimators': 50, 'max_depth': 3 |
| **SVM** | **0.76 ± 0.058** | **0.78 ± 0.039** | **0.79 ± 0.047** | 'C': 0.01, 'kernel': 'linear','probability': True |
| SVM + PCA | 0.71 ± 0.099 | 0.76 ± 0.062 | 0.75 ± 0.072 | 'C': 10, 'kernel': 'linear','class_weight': 'balanced', 'probability': True, n_components=70 |
| XGBoost | 0.67 ± 0.044 | 0.72 ± 0.029 | 0.71 ± 0.033 | 'objective': 'binary:logistic', 'eta': 0.1, 'max_depth': 10, 'scale_pos_weight': 1.0, 'alpha': 0 |

# Classical ML results for ABIDE dataset

| | F1 score | Accuracy | ROC-AUC | Best hyperparameters |
|---|---|---|---|---|
| Logistic Regression | 0.66 ± 0.013 | 0.65 ± 0.012 | 0.66 ± 0.013 | 'class_weight': None, 'penalty': 'l2', 'C': 10 |
| Random Forest | 0.65 ± 0.037 | 0.67 ± 0.030 | 0.67 ± 0.031 | 'n_estimators': 100, 'max_depth': 3 |
| SVM | 0.67 ± 0.012 | 0.67 ± 0.013 | 0.67 ± 0.013 | 'C': 10, 'kernel': 'rbf' |
| **SVM + PCA** | **0.69 ± 0.017** | **0.68 ± 0.015** | **0.68 ± 0.016** | 'C': 0.01, 'kernel': 'linear','class_weight': 'balanced', 'probability': True, n_components = 500 |
| XGBoost | 0.65 ± 0.007 | 0.63 ± 0.010 | 0.63 ± 0.011 | 'max_depth': 10, 'scale_pos_weight': 1.0, 'alpha': 0 |

# Graph-Attention Neural Network

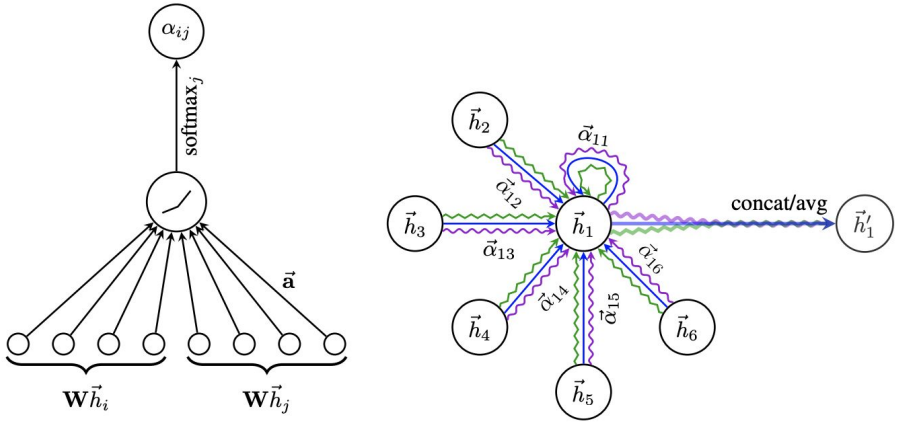



Figure 1: **Left:** The attention mechanism $a(\mathbf{W}\vec{h}_i, \mathbf{W}\vec{h}_j)$ employed by our model, parametrized by a weight vector $\vec{\mathbf{a}} \in \mathbb{R}^{2F'}$, applying a LeakyReLU activation. **Right:** An illustration of multi-head attention (with $K = 3$ heads) by node 1 on its neighborhood. Different arrow styles and colors denote independent attention computations. The aggregated features from each head are concatenated or averaged to obtain $\vec{h}'_1$.

*Figure 3 - GAT architecture*



https://arxiv.org/abs/1710.10903v3

# Deep Learning approach. Graph Neural Networks

```
BrainGAT(
  (convs): ModuleList(
    (0): Sequential(
      (0): GATConv(200, 32, heads=2)
      (1): Linear(in_features=64, out_features=32, bias=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (1): Sequential(
      (0): GATConv(32, 32, heads=2)
      (1): Linear(in_features=64, out_features=32, bias=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
    (2): Sequential(
      (0): GATConv(32, 32, heads=2)
      (1): Linear(in_features=64, out_features=64, bias=True)
      (2): LeakyReLU(negative_slope=0.2)
    )
  )
  (prepool): Sequential(
    (0): Linear(in_features=64, out_features=8, bias=True)
    (1): LeakyReLU(negative_slope=0.2)
    (2): BatchNorm1d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  )
  (fcn): BasicMLP(
    (net): Sequential(
      (0): Sequential(
        (0): Linear(in_features=1600, out_features=32, bias=True)
        (1): LeakyReLU(negative_slope=0.2)
        (2): Dropout(p=0.6, inplace=True)
      )
      (1): Sequential(
        (0): Linear(in_features=32, out_features=32, bias=True)
        (1): LeakyReLU(negative_slope=0.2)
        (2): Dropout(p=0.6, inplace=True)
      )
      (2): Sequential(
        (0): Linear(in_features=32, out_features=2, bias=True)
      )
    )
  )
)
```

**torch_geometric.nn.conv.GATConv**
graph attentional operator

**fully connected layers** (fcn)

*number of GATConv layers*
*number of neurons in GATConv*
*GATConv parameters (heads)*
*fcn parameters (number of neurons, layers)*
*regularizations (BatchNorm, Dropout)*

# Deep Learning approach. Baseline: multilayer perceptron

```
MLP(
  (layers): Sequential(
    (0): Flatten(start_dim=1, end_dim=-1)
    (1): Linear(in_features=13456, out_features=1024, bias=True)
    (2): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): ReLU()
    (4): Dropout(p=0.1, inplace=False)
    (5): Linear(in_features=1024, out_features=512, bias=True)
    (6): ReLU()
    (7): Dropout(p=0.1, inplace=False)
    (8): Linear(in_features=512, out_features=2, bias=True)
  )
)
```

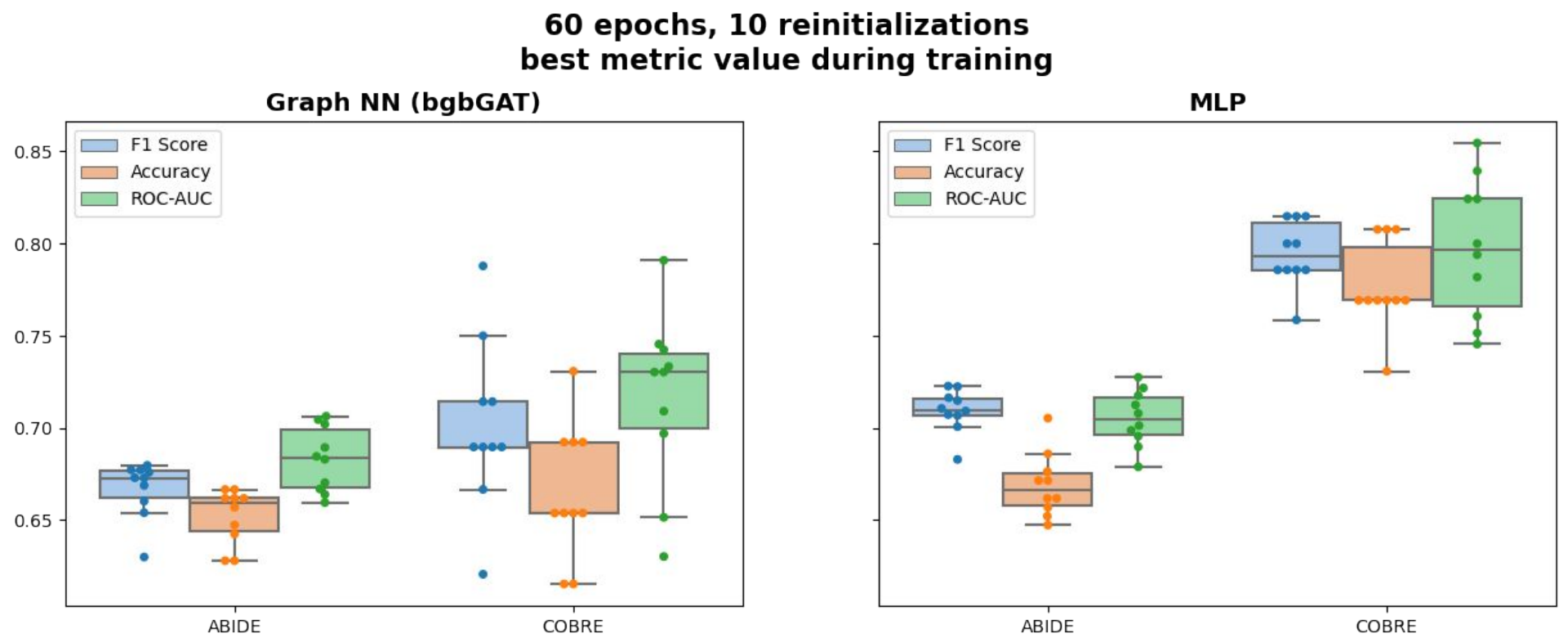
*number of neurons*
*number of layers*
*regularizations*

# Training

1. Classic **AdamW, lr=0.001, 60 epochs** for training
2. After each epoch, **F1 Score, Accuracy, ROC-AUC** are estimated on the test set
   NB: train test split is the same as in classical ML
3. Best F1 Score, Accuracy, ROC-AUC combination is saved

Repeat NN initialization and training as above **10 times** to estimate variation of metrics

# Results


60 epochs, 10 reinitializations
best metric value during training
Graph NN (bgbGAT)
MLP
F1 Score
Accuracy
ROC-AUC
0.85
0.80
0.75
0.70
0.65
ABIDE
COBRE

# Comparison of Classic ML and DL

## COBRE results

| | F1 Score | Accuracy | ROC-AUC |
|---|---|---|---|
| Logistic Regression | 0.73 ± 0.056 | 0.76 ± 0.045 | 0.76 ± 0.049 |
| Random Forest | 0.65 ± 0.073 | 0.69 ± 0.064 | 0.69 ± 0.065 |
| SVM | **0.76 ± 0.058** | 0.78 ± 0.039 | 0.79 ± 0.047 |
| SVM + PCA | 0.71 ± 0.087 | 0.76 ± 0.062 | 0.75 ± 0.072 |
| XGBoost | 0.67 ± 0.044 | 0.72 ± 0.029 | 0.71 ± 0.033 |
| bgbGAT | 0.70 ± 0.045 | 0.67 ± 0.036 | 0.72 ± 0.042 |
| MLP | **0.79 ± 0.018** | 0.78 ± 0.024 | 0.79 ± 0.038 |

## ABIDE results

| | F1 Score | Accuracy | ROC-AUC |
|---|---|---|---|
| Logistic Regression | 0.66 ± 0.013 | 0.66 ± 0.013 | 0.66 ± 0.013 |
| Random Forest | 0.65 ± 0.037 | 0.67 ± 0.030 | 0.67 ± 0.031 |
| SVM | 0.67 ± 0.012 | 0.67 ± 0.013 | 0.67 ± 0.013 |
| SVM + PCA | **0.69 ± 0.017** | 0.68 ± 0.015 | 0.68 ± 0.016 |
| XGBoost | 0.65 ± 0.007 | 0.63 ± 0.010 | 0.63 ± 0.011 |
| bgbGAT | 0.67 ± 0.015 | 0.65 ± 0.015 | 0.68 ± 0.017 |
| MLP | **0.71 ± 0.012** | 0.67 ± 0.017 | 0.71 ± 0.015 |

# Conclusions

1. Usually, we do not have “Big Data” in biomedical research
2. Classic ML approaches are very applicable to biomedical data
3. Well advertised Graph-Attention Neural Networks may not give the best result in biomed
4. For our comparatively small medical datasets, old classic Multilayer Perceptron and SVM beat all the other models

# thx!


**Link to our github:**
https://github.com/utoprey/CogniNet

# GATConv

The graph attentional operator from the "Graph Attention Networks" paper

$$\mathbf{x}'_i = \alpha_{i,i}\mathbf{\Theta}\mathbf{x}_i + \sum_{j \in \mathcal{N}(i)} \alpha_{i,j}\mathbf{\Theta}\mathbf{x}_j,$$

where the attention coefficients $\alpha_{i,j}$ are computed as

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{\Theta}\mathbf{x}_i \,\|\, \mathbf{\Theta}\mathbf{x}_j]\right)\right)}{\sum_{k \in \mathcal{N}(i)\cup\{i\}} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{\Theta}\mathbf{x}_i \,\|\, \mathbf{\Theta}\mathbf{x}_k]\right)\right)}.$$

If the graph has multi-dimensional edge features $\mathbf{e}_{i,j}$, the attention coefficients $\alpha_{i,j}$ are computed as

$$\alpha_{i,j} = \frac{\exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{\Theta}\mathbf{x}_i \,\|\, \mathbf{\Theta}\mathbf{x}_j \,\|\, \mathbf{\Theta}_e\mathbf{e}_{i,j}]\right)\right)}{\sum_{k \in \mathcal{N}(i)\cup\{i\}} \exp\left(\text{LeakyReLU}\left(\mathbf{a}^\top [\mathbf{\Theta}\mathbf{x}_i \,\|\, \mathbf{\Theta}\mathbf{x}_k \,\|\, \mathbf{\Theta}_e\mathbf{e}_{i,k}]\right)\right)}.$$