KLCE loss and Graph Information Bottleneck (GIB)on Network Datasets For Class Imbalance

Final Project Presentation

Student: Pranjal Umesh Kalekar

CS 7840: Foundations and Applications of Information Theory (fa24)

https://northeastern-datalab.github.io/cs7840/fa24/

Lecturers: Wolfgang Gatterbauer, Javeed Aslam

DEC 9, 2024

Understanding of KL Divergence

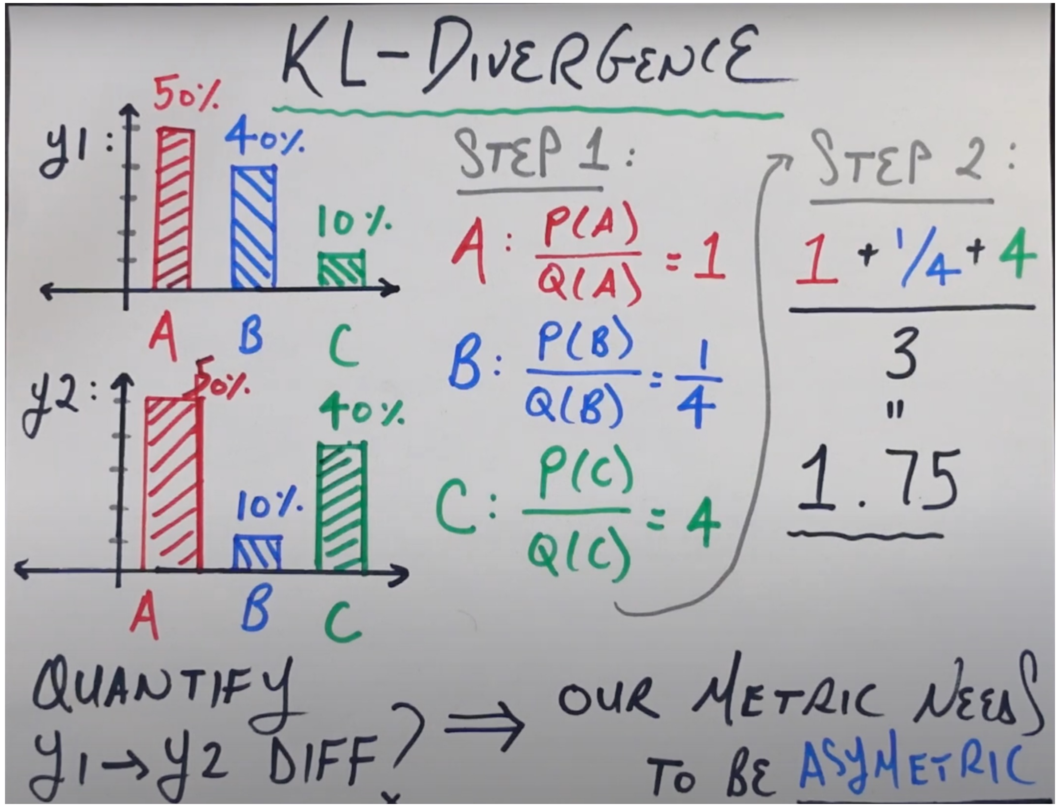
KL divergence is a statistical measure used to compare probability density functions (PDFs). In simpler terms, it quantifies how much one probability distribution differs from another.

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \, \log\left(\frac{P(x)}{Q(x)}\right).$$

$D_{\mathrm{KL}}(P \parallel Q)$

$D_{\mathrm{KL}}(P \parallel Q)$ is a type of statistical distance: a measure of how much a model probability distribution Q is different from a true probability distribution P

Understanding KL Divergence continued



KL-DIVERGENCE

STEP 1:

A: $\frac{P(A)}{Q(A)} = 1$

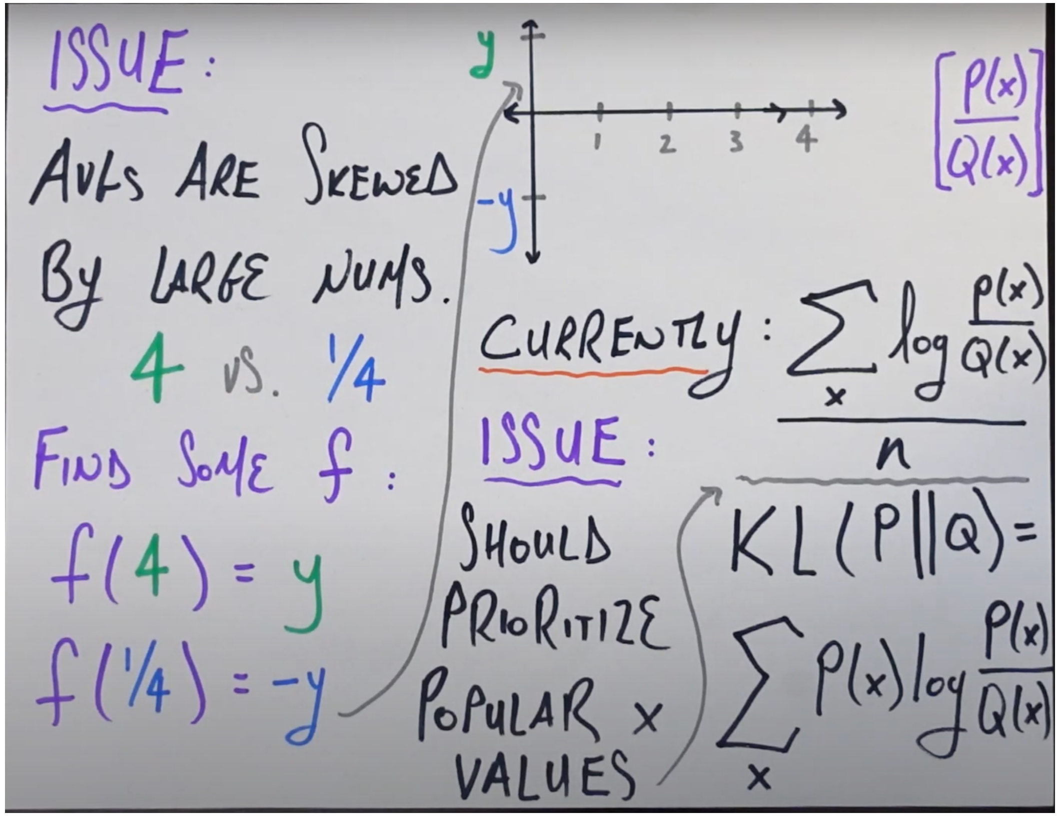
B: $\frac{P(B)}{Q(B)} = \frac{1}{4}$

C: $\frac{P(C)}{Q(C)} = 4$

STEP 2:

$$\frac{1 + 1/4 + 4}{3} = 1.75$$

QUANTIFY $y_1 \rightarrow y_2$ DIFF? ⇒ OUR METRIC NEEDS TO BE ASYMETRIC



ISSUE:

AVGS ARE SKEWED BY LARGE NUMS.

4 VS. 1/4

FIND SOME f:

$f(4) = y$

$f(1/4) = -y$

CURRENTLY: $\frac{\sum_x \log \frac{P(x)}{Q(x)}}{n}$

ISSUE:

SHOULD PRIORITIZE POPULAR x VALUES

$$KL(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

Kindergarten student performances example to understand why KL divergence needs to be asymmetric to understand the difference between the two distributions. It also depicts how adding the bias term P(X) improves the prioritization of highly occurring events and avoid skewness that just the average(1/n as a equal bias to the changes)

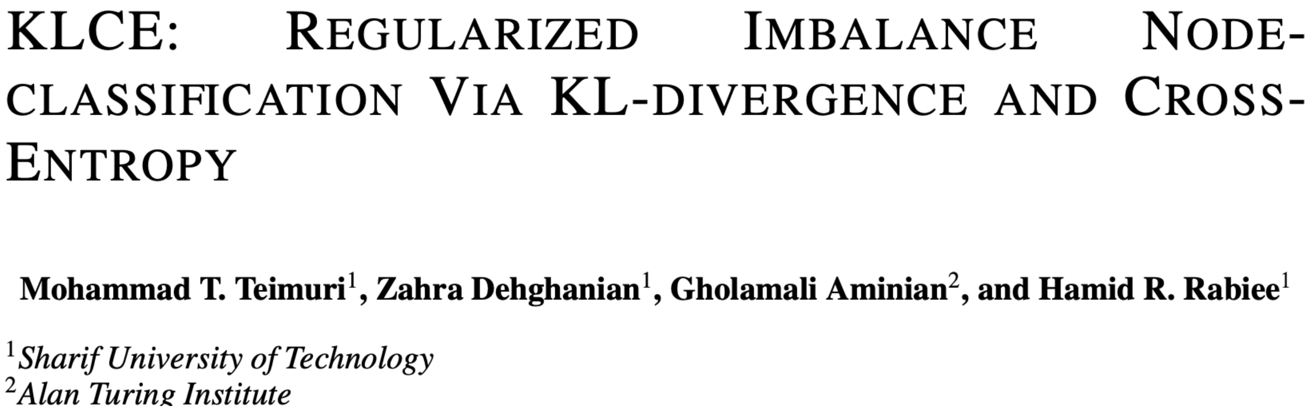
This work also resembles Prof. Wolfgangs hammer and nail ideology while leaning new concept

Source: YT Ritwikmath

Project Idea and Concept of using KL divergence in combination of cross entropy as regularizing term

Aim of the project is deriving the KLCE regularization loss introduced at ICLR 2024 in a tiny paper and implement on following

Datasets: **Cora**, **PubMed**, and **CiteSeer** are well-known benchmark datasets used for node classification tasks in graph neural networks (GNNs)



KLCE: Regularized Imbalance Node-classification Via KL-divergence and Cross-Entropy

Mohammad T. Teimuri[1], Zahra Dehghanian[1], Gholamali Aminian[2], and Hamid R. Rabiee[1]

[1] *Sharif University of Technology*
[2] *Alan Turing Institute*

KLCE regularization quantifies the difference between the GNN's predicted class distribution (P(Ŷ|X)) and a target distribution (Pk) that is designed to give more weight to minority classes using KLD term and CE term with hyperparameters λ and λKLCE

More about the datasets

- The Cora dataset is a citation network of 2,708 machine-learning papers, organized into seven distinct classes. These papers are interlinked by 5,429 citations, forming a directed graph that maps out how papers cite each other. Each paper is represented by a binary word vector, derived from a dictionary of 1,433 unique words, indicating the presence or absence of specific words in the paper.
- The CiteSeer dataset is similar to the Cora dataset but focuses on a different set of scientific papers. It contains a citation network where each paper belongs to one of six categories. There are 6 classes: Case-Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, and Robotics
  - 3327 nodes
  - 4732 edges
  - 3703 features
  - 6 classes
- The PubMed dataset is another citation network of scientific papers, but it is focused on biomedical research papers. Each paper is also assigned to one of three categories based on its content.
  - Size:19717 nodes
  - 44338 edges
  - 500 features (dimensionality of the paper feature vectors)
  - 3 classes (topics)

Formulation of the concept in mathematical form

KLCE regularization, which combines KL divergence with cross-entropy in the GNN's loss function eventually added to the baseline

$$\mathrm{KLCE} := \lambda_{\mathrm{KLCE}} \mathrm{H}(P(\hat{\mathbf{Y}}|\mathbf{X}), \mathbf{P}_k) + \mathrm{KL}(P(\hat{\mathbf{Y}}|\mathbf{X}) \| \mathbf{P}_k)$$

Target Class Distribution (Pk): KLCE uses a target class distribution (Pk) that can be adjusted to give more weight to minority classes

KL Divergence Term: measures the difference between the learned distribution of class predictions by the GNN, denoted as P(Ŷ|X), and the target distribution **(Pk)**

$$\mathrm{KL}(P(\hat{\mathbf{Y}}|\mathbf{X}) \| \mathbf{P}_k) = \sum_{j=1}^{k} P(\hat{Y} = j|\mathbf{X}) \log \left(\frac{P(\hat{Y} = j|\mathbf{X})}{P_j} \right)$$

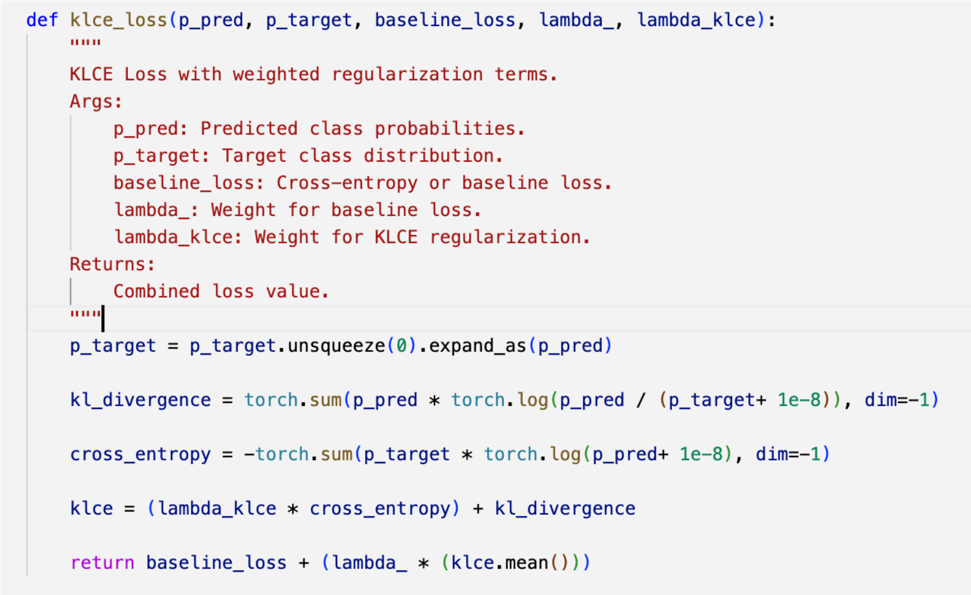
Cross-Entropy Term: It focuses on the expected coding length of the data, providing an additional measure of how well the model's predictions align with the desired distribution.

$$\mathrm{H}(P(\hat{\mathbf{Y}}|\mathbf{X}), \mathbf{P}_k) = -\sum_{j=1}^{k} P(\hat{Y} = j|\mathbf{X}) \log(P_j)$$

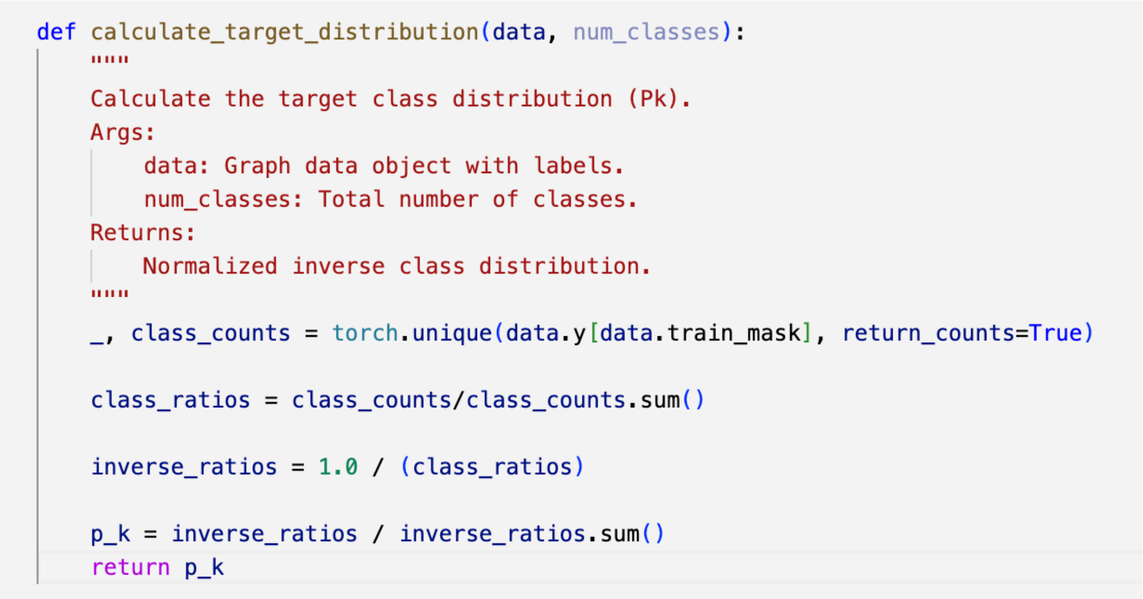
Tools used to build the project around this concept

The project has been implemented in python utilizing pytorch
And trained using VS studio Code

Construction of the custom KLCE regularizing loss function and addressing class imbalance while constructing the target distribution



```python
def klce_loss(p_pred, p_target, baseline_loss, lambda_, lambda_klce):
    """
    KLCE Loss with weighted regularization terms.
    Args:
        p_pred: Predicted class probabilities.
        p_target: Target class distribution.
        baseline_loss: Cross-entropy or baseline loss.
        lambda_: Weight for baseline loss.
        lambda_klce: Weight for KLCE regularization.
    Returns:
        Combined loss value.
    """
    p_target = p_target.unsqueeze(0).expand_as(p_pred)

    kl_divergence = torch.sum(p_pred * torch.log(p_pred / (p_target+ 1e-8)), dim=-1)

    cross_entropy = -torch.sum(p_target * torch.log(p_pred+ 1e-8), dim=-1)

    klce = (lambda_klce * cross_entropy) + kl_divergence

    return baseline_loss + (lambda_ * (klce.mean()))
```



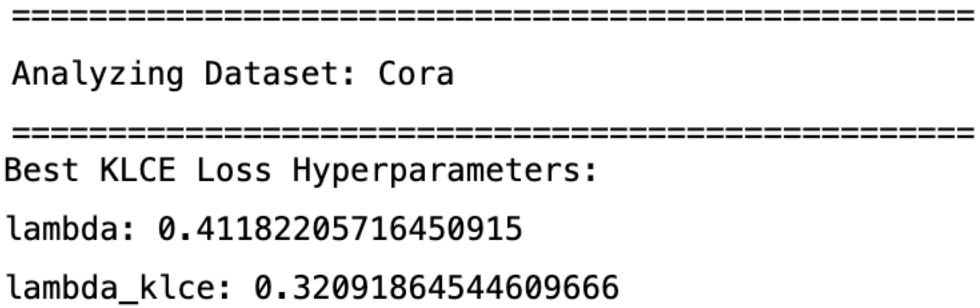
```python
def calculate_target_distribution(data, num_classes):
    """
    Calculate the target class distribution (Pk).
    Args:
        data: Graph data object with labels.
        num_classes: Total number of classes.
    Returns:
        Normalized inverse class distribution.
    """
    _, class_counts = torch.unique(data.y[data.train_mask], return_counts=True)

    class_ratios = class_counts/class_counts.sum()

    inverse_ratios = 1.0 / (class_ratios)

    p_k = inverse_ratios / inverse_ratios.sum()
    return p_k
```

Pk for the regularization term involves two steps. the ratio of each class and a normalized inverse ratio each class. The inversion is intended to emphasize the significance of the minority class.

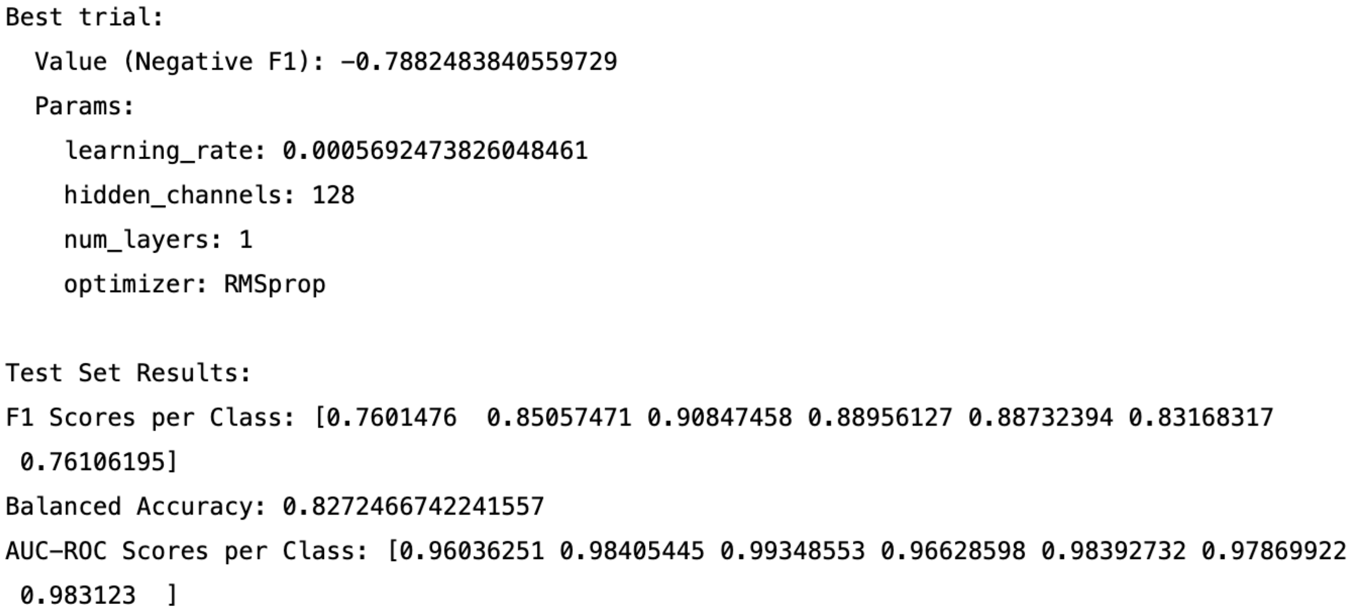
Hyper Parameter Tuning

- 4 models were trained where
  - A baseline model was trained utilizing log loss. This model was tuned using `Optuna`

- 3 other models were tuned for Lambda and LambdaKLCE values for each dataset building on the tuning results on the baseline model

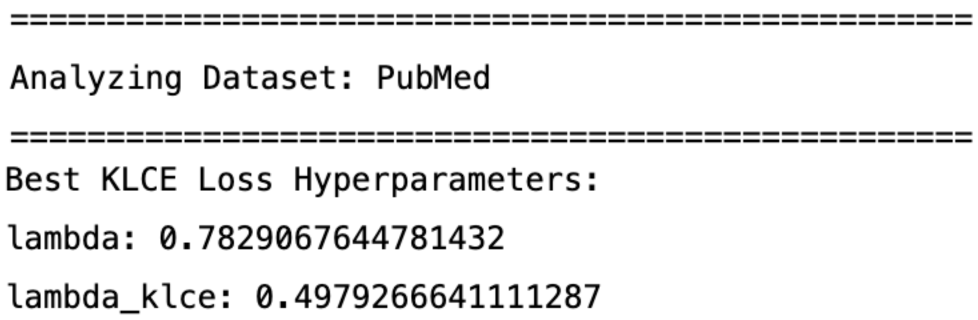


```
==================================================
Analyzing Dataset: Cora
==================================================
Best KLCE Loss Hyperparameters:
lambda: 0.41182205716450915
lambda_klce: 0.32091864544609666
```

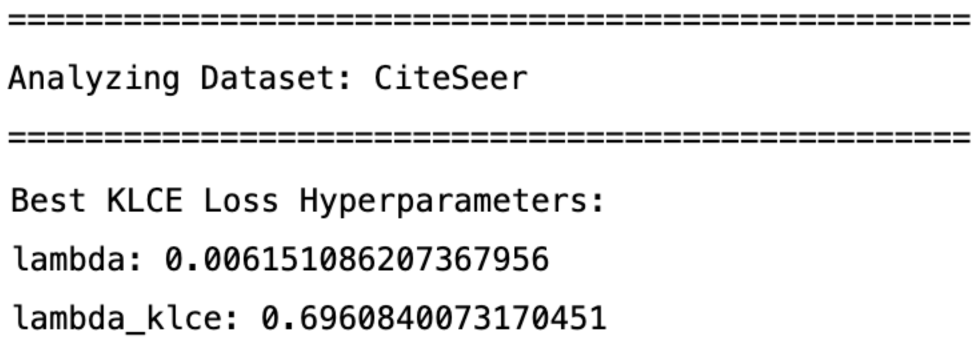


```
Best trial:
  Value (Negative F1): -0.7882483840559729
  Params:
    learning_rate: 0.0005692473826048461
    hidden_channels: 128
    num_layers: 1
    optimizer: RMSprop

Test Set Results:
F1 Scores per Class: [0.7601476  0.85057471 0.90847458 0.88956127 0.88732394 0.83168317
 0.76106195]
Balanced Accuracy: 0.8272466742241557
AUC-ROC Scores per Class: [0.96036251 0.98405445 0.99348553 0.96628598 0.98392732 0.97869922
 0.983123  ]
```

Hyper Parameter Tuning Continued



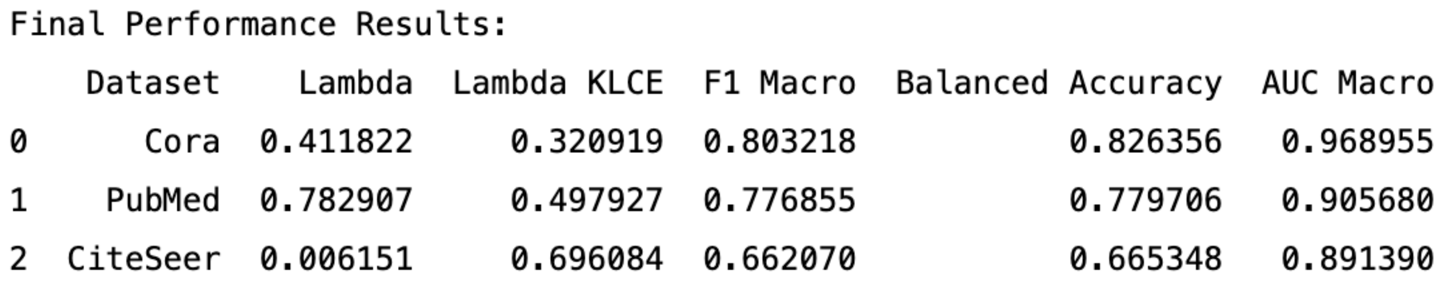
```
==================================================
Analyzing Dataset: PubMed
==================================================
Best KLCE Loss Hyperparameters:
lambda: 0.7829067644781432
lambda_klce: 0.4979266641111287
```



```
==================================================
Analyzing Dataset: CiteSeer
==================================================

Best KLCE Loss Hyperparameters:
lambda: 0.006151086207367956
lambda_klce: 0.6960840073170451
```

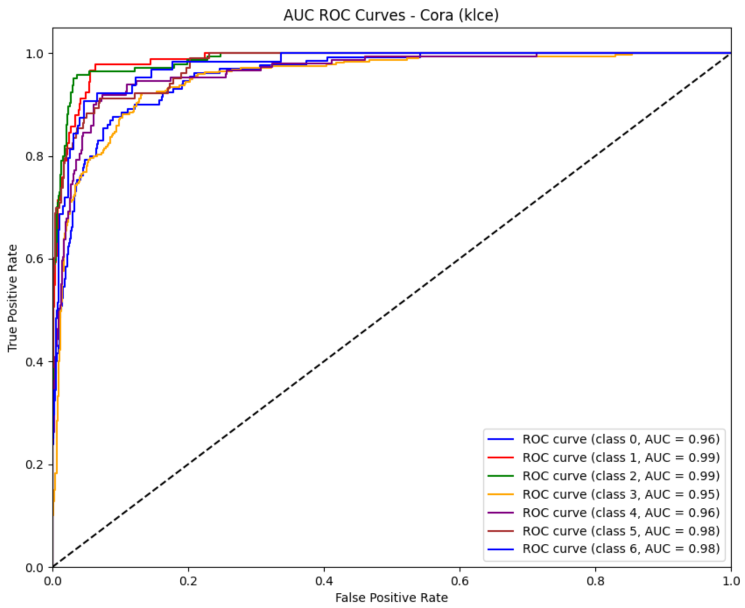
Performance evaluation Metrics and results

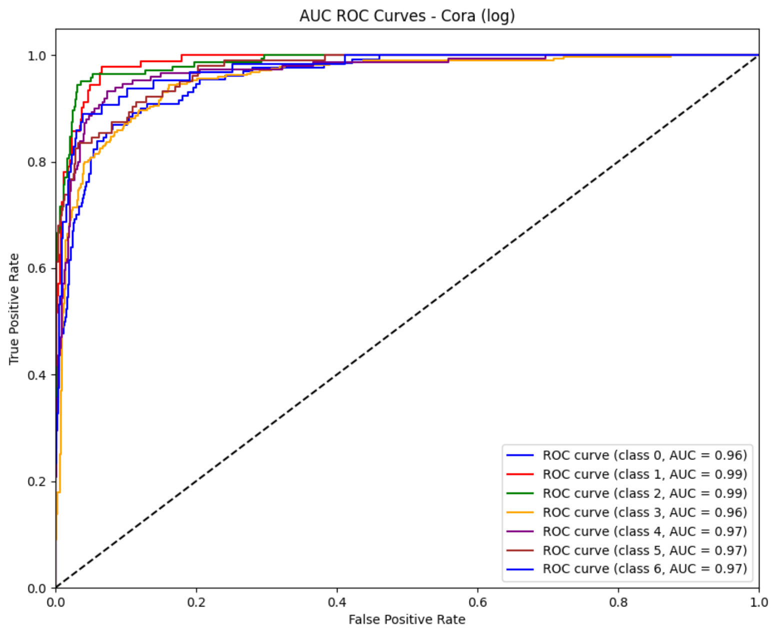
Performance was measured using F1- score, Balanced Accuracy (bAcc), AUC score and ROC plots



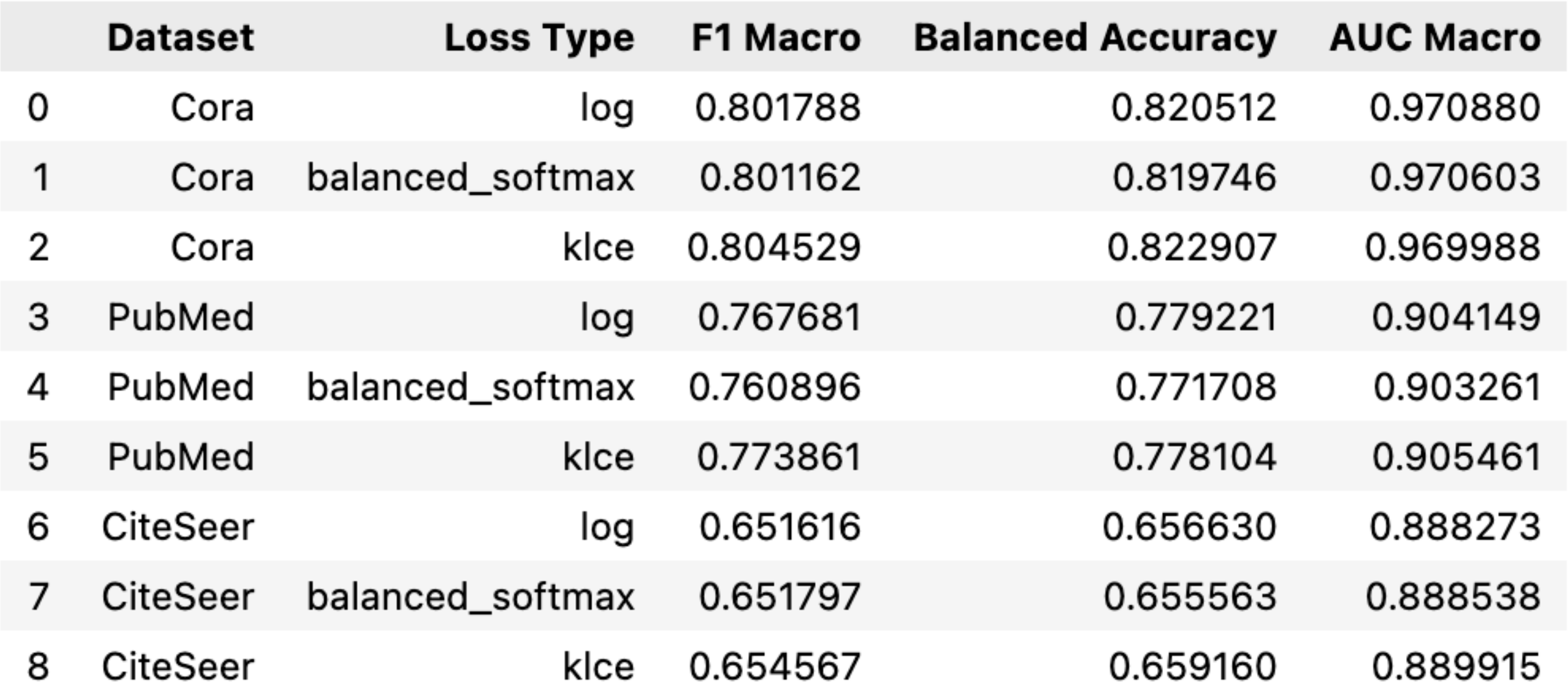
```
Final Performance Results:
    Dataset    Lambda  Lambda KLCE  F1 Macro  Balanced Accuracy  AUC Macro
0      Cora  0.411822     0.320919  0.803218           0.826356   0.968955
1    PubMed  0.782907     0.497927  0.776855           0.779706   0.905680
2  CiteSeer  0.006151     0.696084  0.662070           0.665348   0.891390
```

AUC Plots:





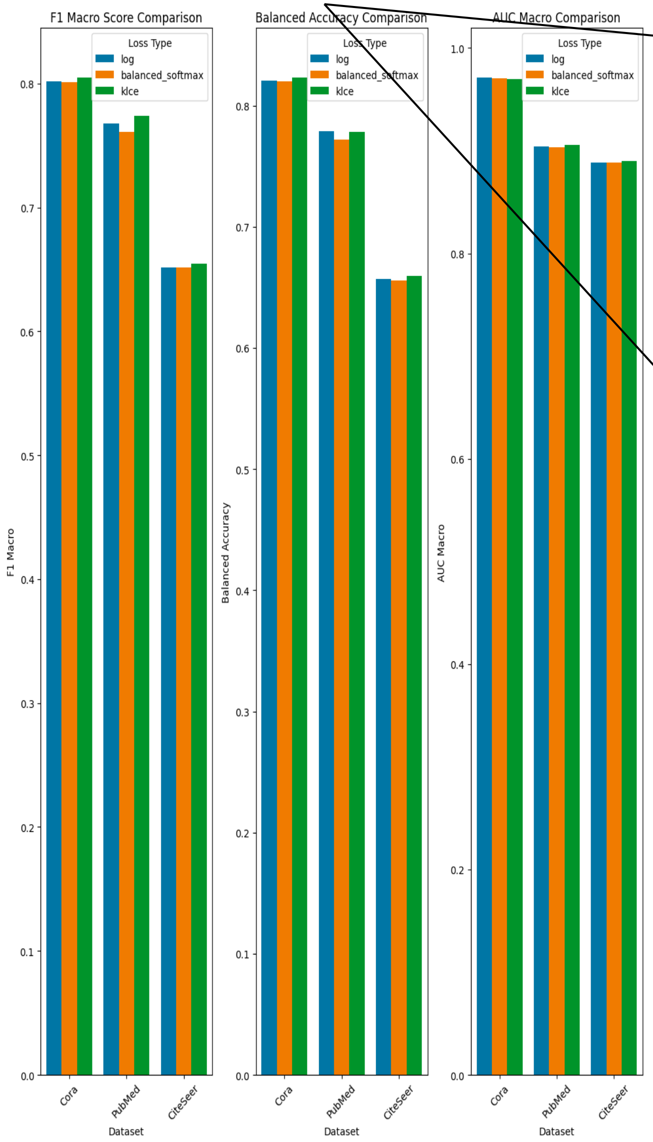
Results continued

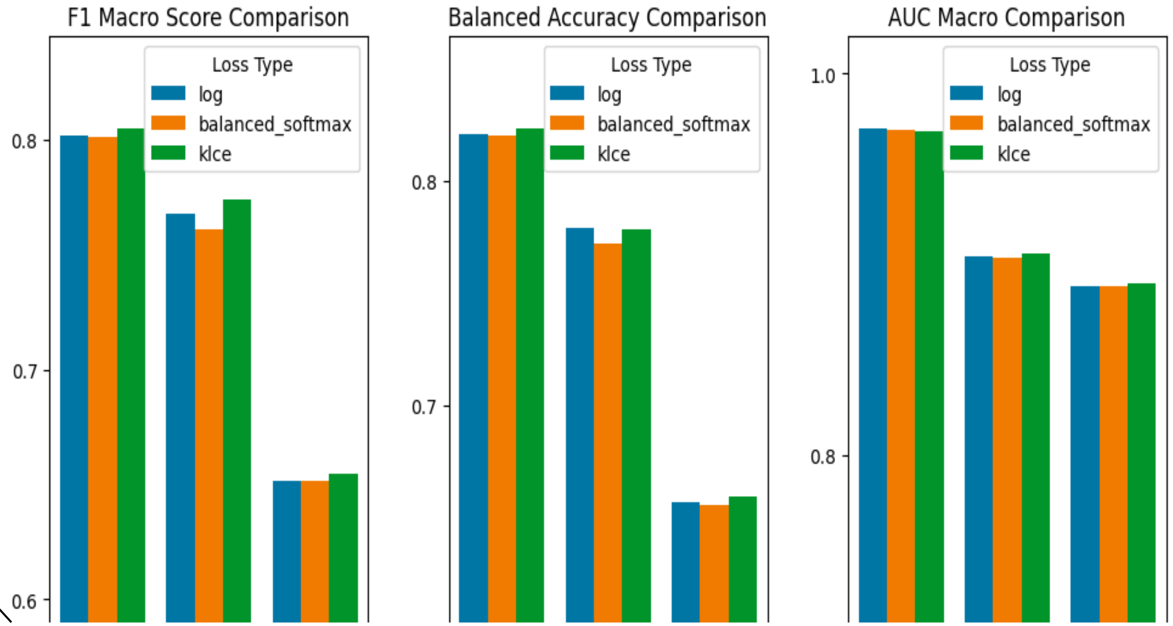


	Dataset	Loss Type	F1 Macro	Balanced Accuracy	AUC Macro
0	Cora	log	0.801788	0.820512	0.970880
1	Cora	balanced_softmax	0.801162	0.819746	0.970603
2	Cora	klce	0.804529	0.822907	0.969988
3	PubMed	log	0.767681	0.779221	0.904149
4	PubMed	balanced_softmax	0.760896	0.771708	0.903261
5	PubMed	klce	0.773861	0.778104	0.905461
6	CiteSeer	log	0.651616	0.656630	0.888273
7	CiteSeer	balanced_softmax	0.651797	0.655563	0.888538
8	CiteSeer	klce	0.654567	0.659160	0.889915

Model performances on all datasets with different loss functions

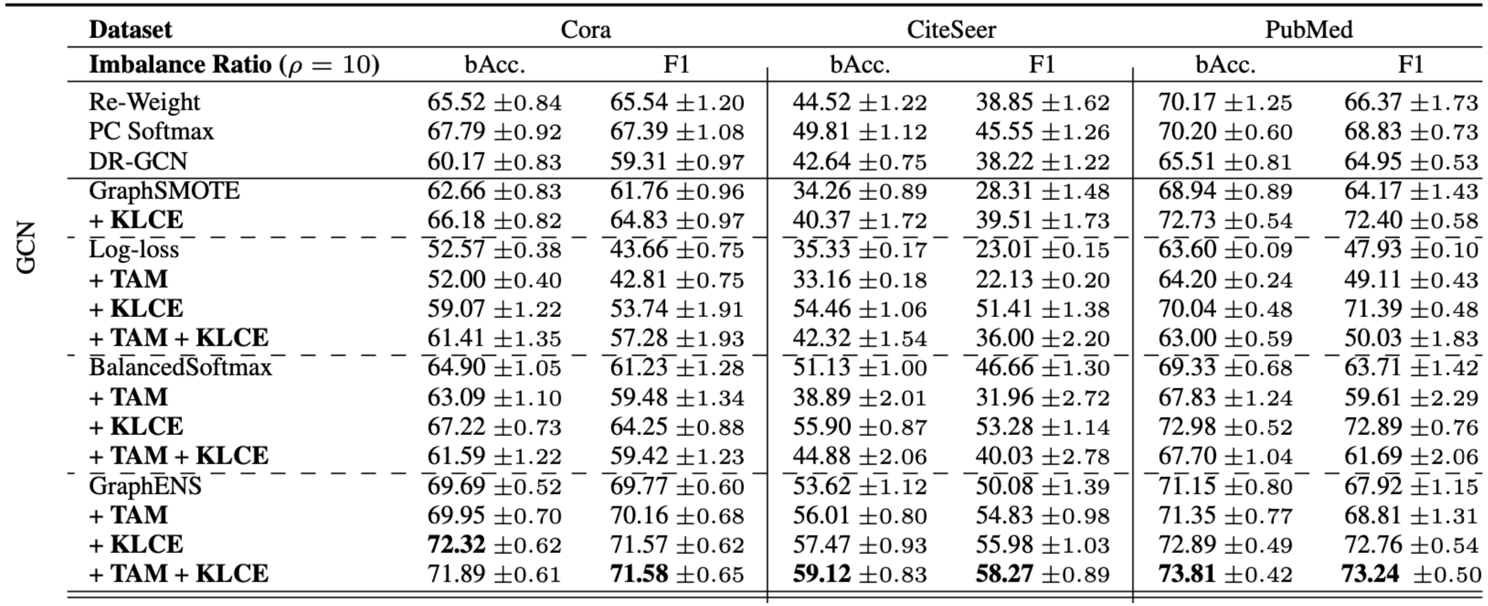
KLCE performs better than pre-existing Loss functions on all datasets





Results are visible!
Ideally the hyper parameters for lr, model architecture, weight decay and type of optimizer should be tuned per dataset as well – I believe that will show significant improvement

Comparing results with the published tiny paper



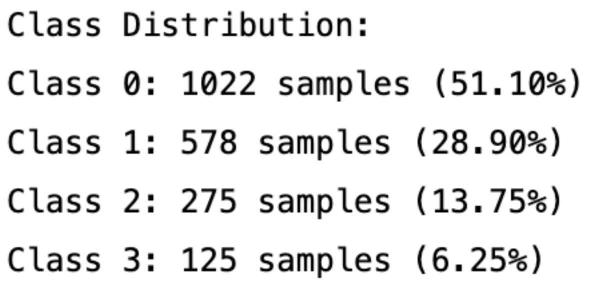
	Dataset	Cora		CiteSeer		PubMed	
	Imbalance Ratio ($\rho = 10$)	bAcc.	F1	bAcc.	F1	bAcc.	F1
GCN	Re-Weight	65.52 $\pm$0.84	65.54 $\pm$1.20	44.52 $\pm$1.22	38.85 $\pm$1.62	70.17 $\pm$1.25	66.37 $\pm$1.73
	PC Softmax	67.79 $\pm$0.92	67.39 $\pm$1.08	49.81 $\pm$1.12	45.55 $\pm$1.26	70.20 $\pm$0.60	68.83 $\pm$0.73
	DR-GCN	60.17 $\pm$0.83	59.31 $\pm$0.97	42.64 $\pm$0.75	38.22 $\pm$1.22	65.51 $\pm$0.81	64.95 $\pm$0.53
	GraphSMOTE	62.66 $\pm$0.83	61.76 $\pm$0.96	34.26 $\pm$0.89	28.31 $\pm$1.48	68.94 $\pm$0.89	64.17 $\pm$1.43
	+ **KLCE**	66.18 $\pm$0.82	64.83 $\pm$0.97	40.37 $\pm$1.72	39.51 $\pm$1.73	72.73 $\pm$0.54	72.40 $\pm$0.58
	Log-loss	52.57 $\pm$0.38	43.66 $\pm$0.75	35.33 $\pm$0.17	23.01 $\pm$0.15	63.60 $\pm$0.09	47.93 $\pm$0.10
	+ **TAM**	52.00 $\pm$0.40	42.81 $\pm$0.75	33.16 $\pm$0.18	22.13 $\pm$0.20	64.20 $\pm$0.24	49.11 $\pm$0.43
	+ **KLCE**	59.07 $\pm$1.22	53.74 $\pm$1.91	54.46 $\pm$1.06	51.41 $\pm$1.38	70.04 $\pm$0.48	71.39 $\pm$0.48
	+ **TAM** + **KLCE**	61.41 $\pm$1.35	57.28 $\pm$1.93	42.32 $\pm$1.54	36.00 $\pm$2.20	63.00 $\pm$0.59	50.03 $\pm$1.83
	BalancedSoftmax	64.90 $\pm$1.05	61.23 $\pm$1.28	51.13 $\pm$1.00	46.66 $\pm$1.30	69.33 $\pm$0.68	63.71 $\pm$1.42
	+ **TAM**	63.09 $\pm$1.10	59.48 $\pm$1.34	38.89 $\pm$2.01	31.96 $\pm$2.72	67.83 $\pm$1.24	59.61 $\pm$2.29
	+ **KLCE**	67.22 $\pm$0.73	64.25 $\pm$0.88	55.90 $\pm$0.87	53.28 $\pm$1.14	72.98 $\pm$0.52	72.89 $\pm$0.76
	+ **TAM** + **KLCE**	61.59 $\pm$1.22	59.42 $\pm$1.23	44.88 $\pm$2.06	40.03 $\pm$2.78	67.70 $\pm$1.04	61.69 $\pm$2.06
	GraphENS	69.69 $\pm$0.52	69.77 $\pm$0.60	53.62 $\pm$1.12	50.08 $\pm$1.39	71.15 $\pm$0.80	67.92 $\pm$1.15
	+ **TAM**	69.95 $\pm$0.70	70.16 $\pm$0.68	56.01 $\pm$0.80	54.83 $\pm$0.98	71.35 $\pm$0.77	68.81 $\pm$1.31
	+ **KLCE**	**72.32** $\pm$0.62	71.57 $\pm$0.62	57.47 $\pm$0.93	55.98 $\pm$1.03	72.89 $\pm$0.49	72.76 $\pm$0.54
	+ **TAM** + **KLCE**	71.89 $\pm$0.61	**71.58** $\pm$0.65	**59.12** $\pm$0.83	**58.27** $\pm$0.89	**73.81** $\pm$0.42	**73.24** $\pm$0.50

This paper have not mentioned the information about the architecture used to train the GCN or the other hyperparameters. so hard to point out why my model gives better results - but I am assuming the hyper parameter tuning for baseline might be the deciding factor

Performing the similar analysis on synthetically highly imbalanced data to understand behavior of hyperparameters

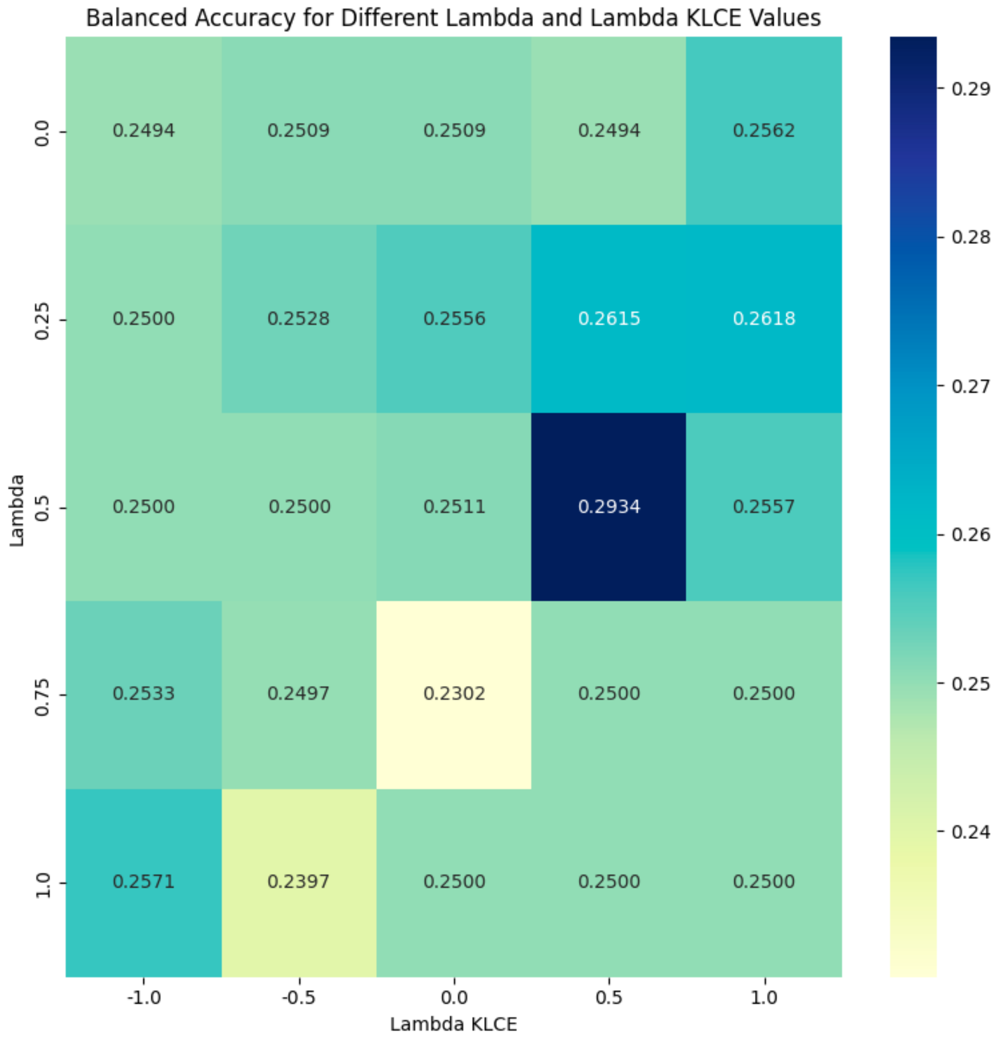
I wanted to observe if the dataset is highly imbalanced will the effect of KLCE be significantly more hence I generated Synthetic data with variable class imbalance parameter

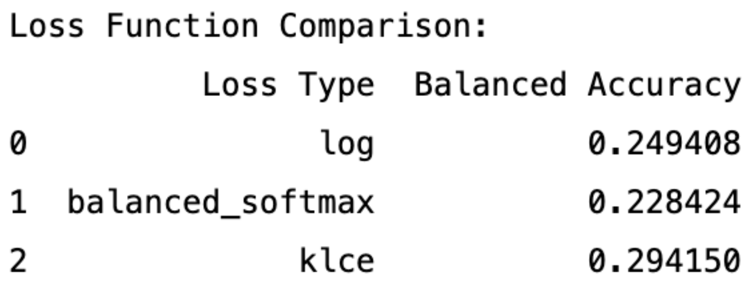
Synthetically generated data was trained on 3-layer linear network and softmax activation and performance was evaluated using balanced accuracy and F1 score with variable range of Lamda and LambdaKLCE



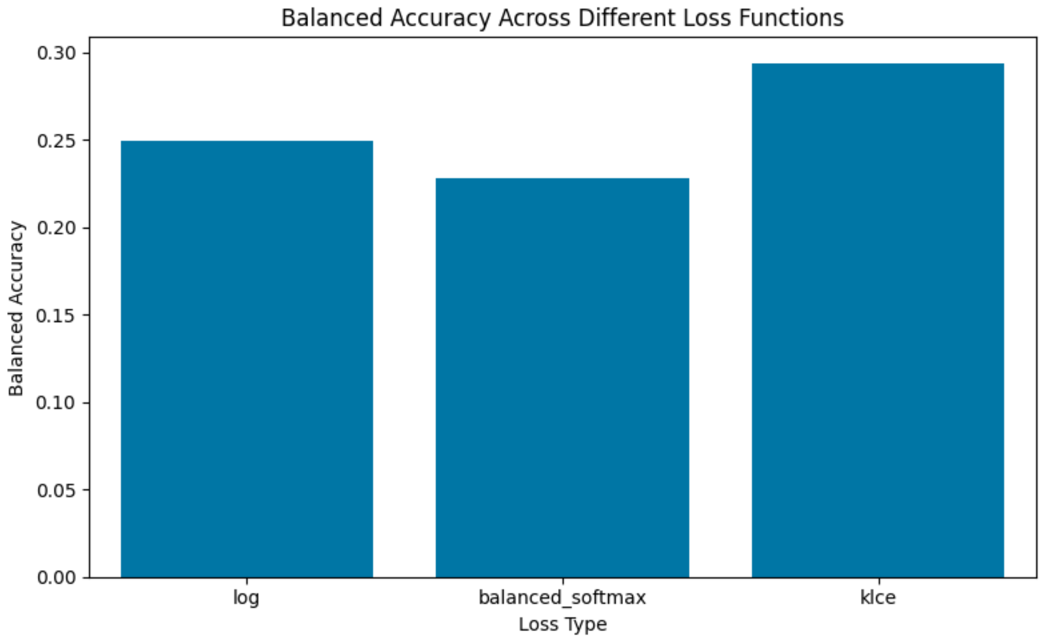
```
Class Distribution:
Class 0: 1022 samples (51.10%)
Class 1: 578 samples (28.90%)
Class 2: 275 samples (13.75%)
Class 3: 125 samples (6.25%)
```

Results on Synthetic data

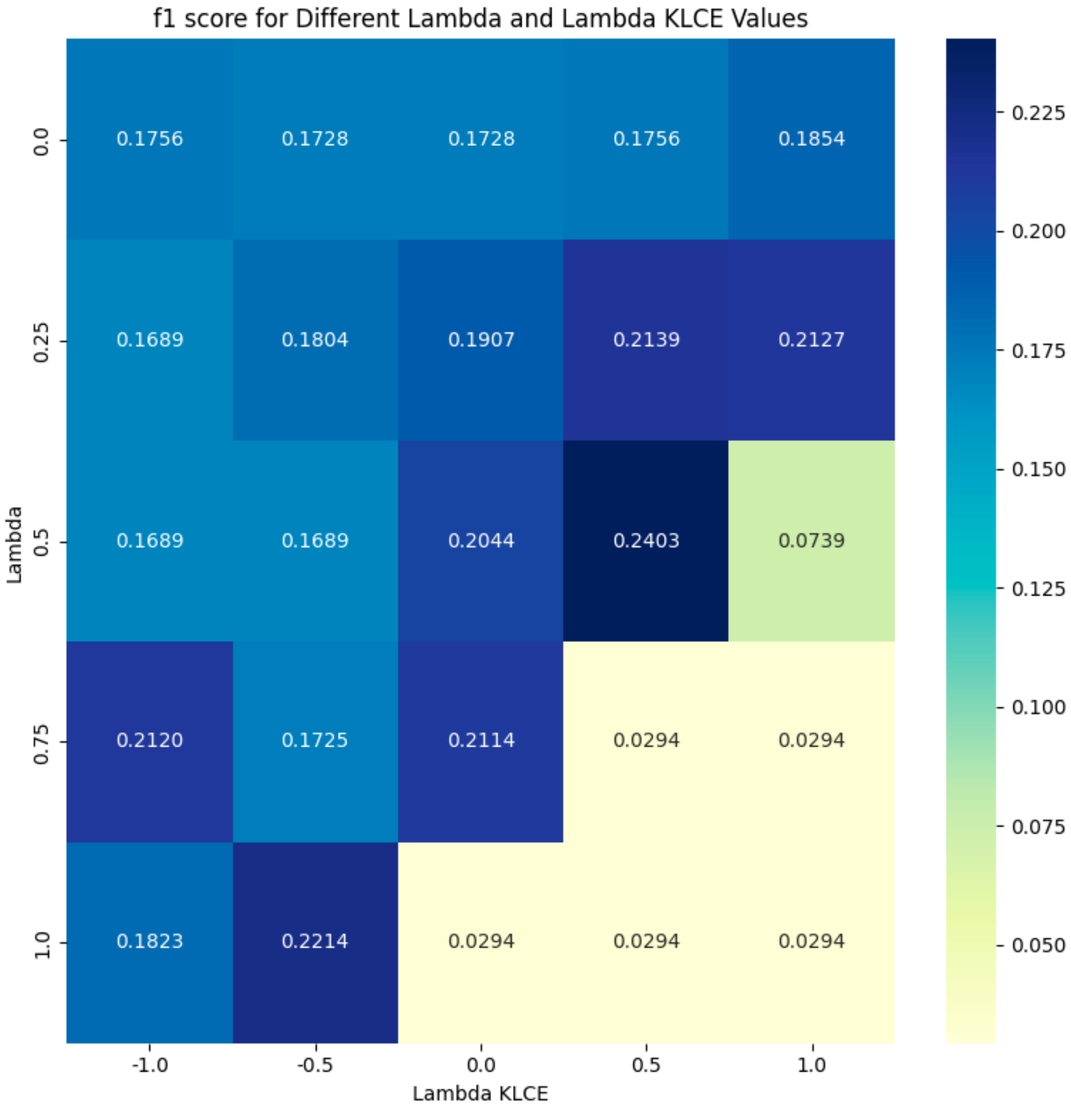


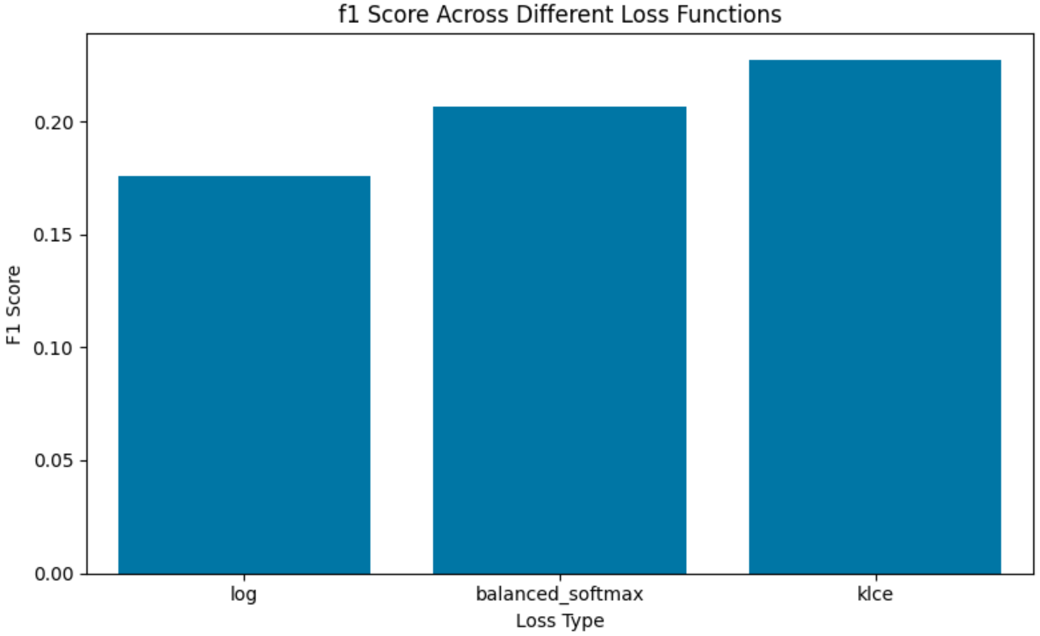


```
Loss Function Comparison:
          Loss Type  Balanced Accuracy
0               log           0.249408
1  balanced_softmax           0.228424
2              klce           0.294150
```



Results on Synthetic data





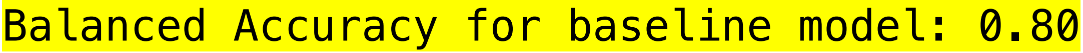
Trying similar analysis using Information bottleneck – as it’s a measure of similarity

So the construction of loss term using the tradeoff of relevance and compression

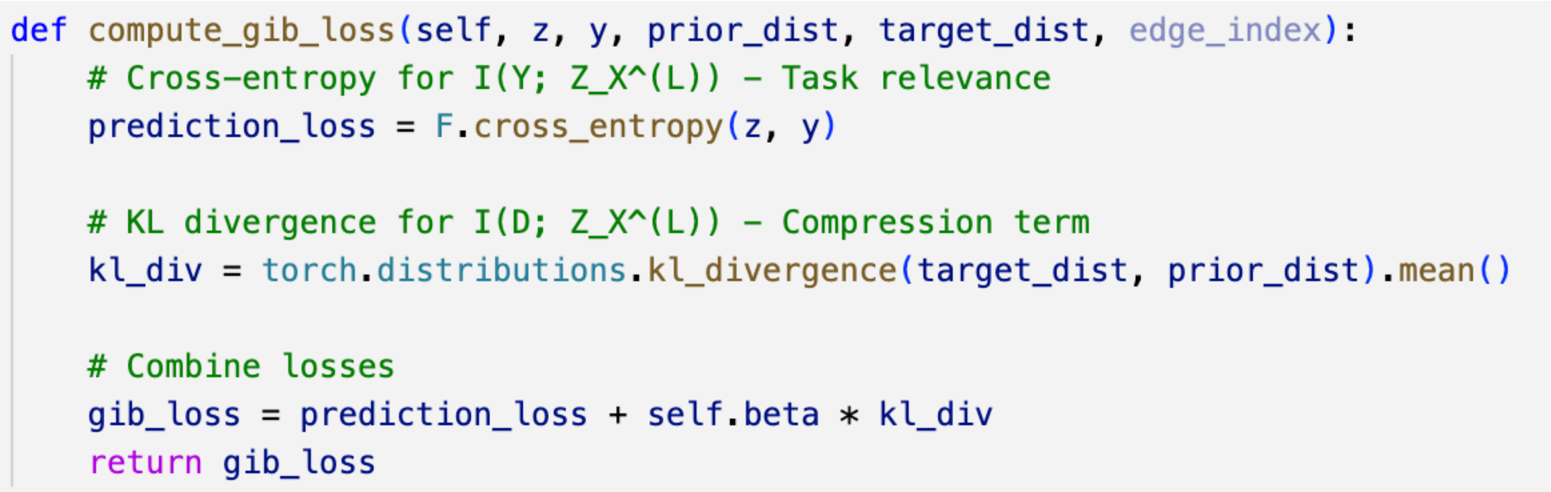
Graph Information Bottleneck:

$$\min_{\mathbb{P}(Z|\mathcal{D})\in\Omega} \mathrm{GIB}_{\beta}(\mathcal{D}, Y; Z) \triangleq [-I(Y; Z) + \beta I(\mathcal{D}; Z)]$$

Training the models on PubMed:



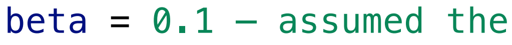
```
Balanced Accuracy for baseline model: 0.80
```



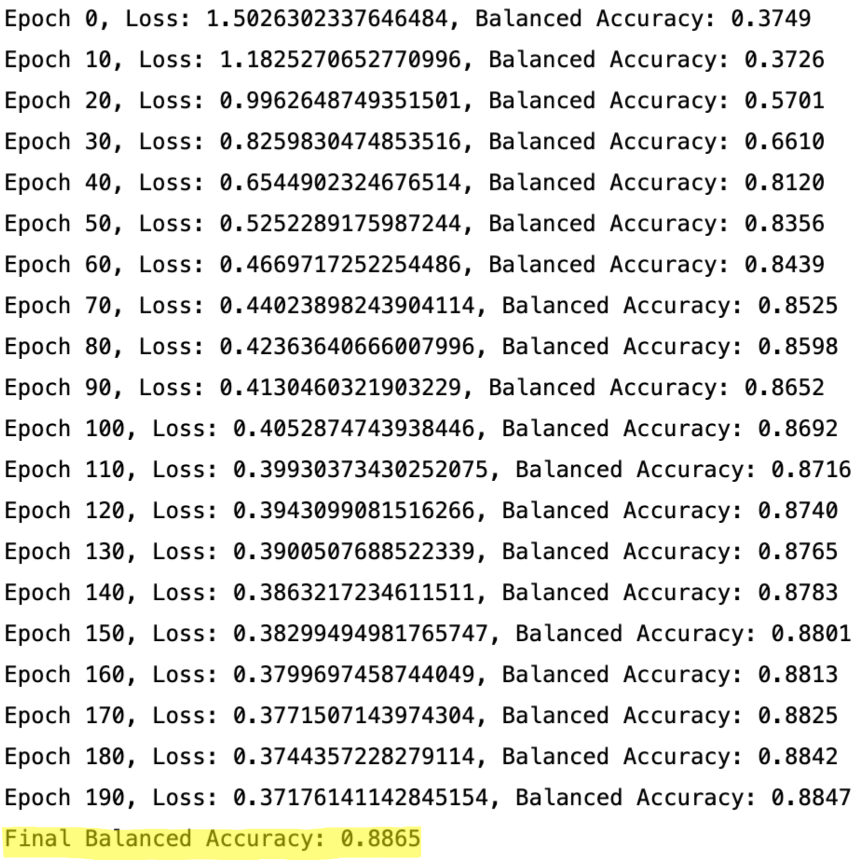
```
def compute_gib_loss(self, z, y, prior_dist, target_dist, edge_index):
    # Cross-entropy for I(Y; Z_X^(L)) - Task relevance
    prediction_loss = F.cross_entropy(z, y)

    # KL divergence for I(D; Z_X^(L)) - Compression term
    kl_div = torch.distributions.kl_divergence(target_dist, prior_dist).mean()

    # Combine losses
    gib_loss = prediction_loss + self.beta * kl_div
    return gib_loss
```

The term β * I(D; Z) acts as a regularization term in the loss function. It penalizes representations that contain too much information ---> avoiding overfitting

Training with GIB loss term



```
beta = 0.1 – assumed the
```



```
Epoch 0, Loss: 1.5026302337646484, Balanced Accuracy: 0.3749
Epoch 10, Loss: 1.1825270652770996, Balanced Accuracy: 0.3726
Epoch 20, Loss: 0.9962648749351501, Balanced Accuracy: 0.5701
Epoch 30, Loss: 0.8259830474853516, Balanced Accuracy: 0.6610
Epoch 40, Loss: 0.6544902324676514, Balanced Accuracy: 0.8120
Epoch 50, Loss: 0.5252289175987244, Balanced Accuracy: 0.8356
Epoch 60, Loss: 0.4669717252254486, Balanced Accuracy: 0.8439
Epoch 70, Loss: 0.44023898243904114, Balanced Accuracy: 0.8525
Epoch 80, Loss: 0.42363640666007996, Balanced Accuracy: 0.8598
Epoch 90, Loss: 0.4130460321903229, Balanced Accuracy: 0.8652
Epoch 100, Loss: 0.4052874743938446, Balanced Accuracy: 0.8692
Epoch 110, Loss: 0.39930373430252075, Balanced Accuracy: 0.8716
Epoch 120, Loss: 0.3943099081516266, Balanced Accuracy: 0.8740
Epoch 130, Loss: 0.3900507688522339, Balanced Accuracy: 0.8765
Epoch 140, Loss: 0.3863217234611511, Balanced Accuracy: 0.8783
Epoch 150, Loss: 0.38299494981765747, Balanced Accuracy: 0.8801
Epoch 160, Loss: 0.3799697458744049, Balanced Accuracy: 0.8813
Epoch 170, Loss: 0.3771507143974304, Balanced Accuracy: 0.8825
Epoch 180, Loss: 0.3744357228279114, Balanced Accuracy: 0.8842
Epoch 190, Loss: 0.37176141142845154, Balanced Accuracy: 0.8847
Final Balanced Accuracy: 0.8865
```

Mass General Brigham
Mass General Cancer Center

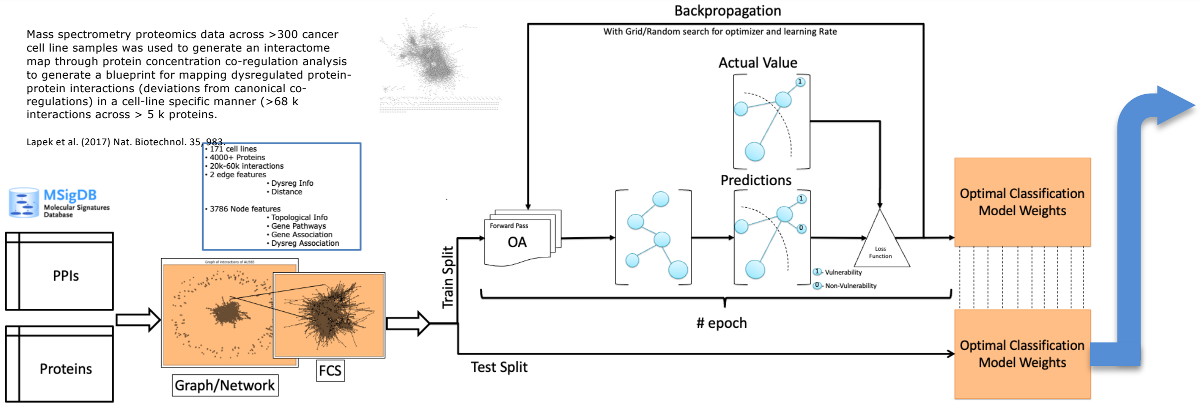
HARVARD MEDICAL SCHOOL
TEACHING HOSPITAL

Krantz Family Center for Cancer Research

Graph Neural Networks for Identifying Cancer Vulnerabilities

Pranjal U. Kalekar[1,2], Robert T Morris[1], Soroush Hajizadeh[1,3], Zohair Shafi[2], Johannes Kreuzer[1], Wilhelm Haas[1]

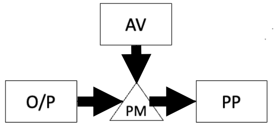
[1]Krantz Family Center for Cancer Research, MGH, and Harvard Medical School, [2]Northeastern University, [3]University of Graz, Austria


Mass spectrometry proteomics data across >300 cancer cell line samples was used to generate an interactome map through protein concentration co-regulation analysis to generate a blueprint for mapping dysregulated protein-protein interactions (deviations from canonical co-regulations) in a cell-line specific manner (>68 k interactions across > 5 k proteins.

Lapek et al. (2017) Nat. Biotechnol. 35, 983.

- 171 cell lines
- 4000+ Proteins
- 20k-60k interactions
- 2 edge features
  - Dysreg Info
  - Distance
- 3786 Node features
  - Topological Info
  - Gene Pathways
  - Gene Association
  - Dysreg Association

MSigDB
Molecular Signatures Database
PPIs
Proteins
Graph of interactions of AU565
Graph/Network
FCS
Train Split
Test Split
Backpropagation
With Grid/Random search for optimizer and learning Rate
Forward Pass
OA
Actual Value
Predictions
1
0
Loss Function
1 - Vulnerability
0 - Non-Vulnerability
# epoch
Optimal Classification Model Weights
Optimal Classification Model Weights



Abbreviations	Meaning
PPIs	Protein-Protein Interactions
FCS	Fully Connected Subgraph
OA	Optimal Architecture
AV	Actual Values
PM	Performance Measure
PP	Performance Parameter

Layer Type	Dysreg Info	Hidden channels	Heads	Targeted	Average AUC	Average F1 score	Average Precision	Average Recall
GCNConv	-	32	-	Y	0.49810	0.0	0.0	0.0
GCNConv	-	32	-	N	0.493438	0.0	0.0	0.0
GATConv	W	32	1	Y	0.545851	0.0002334	0.000117	0.0147
GATConv	W	32	1	N	0.649732	0.0019122	0.001826	0.002
GATConv	WO	32	3	N	0.557120	0.0040291	0.00205	0.1134
TransformerConv	W	16	3	Y	0.594267	0.2197760	0.1649	0.3753
TRansformerConv	W	16	3	N	0.9375	0.538962003	0.8713	0.4367
TRansformerConv	WO	16	3	N	0.8375	0.5137072	0.7760	0.3871

Workflow from mass spec proteomics data to a list of cancer-specific vulnerabilities through using graph neural network models

Classification Model Performances

Background

Mass spectrometry-based proteomics as a tool to map **interactome-wide protein-protein interaction dysregulations** at high throughput

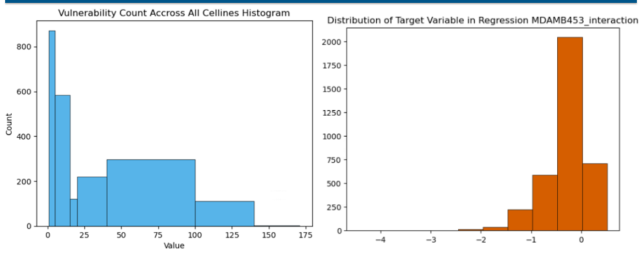
Cell line-specific **fitness genes** are **enriched in dysregulated protein interaction networks**

Network data structures as a base to classify proteins (nodes) as cell line-specific fitness genes

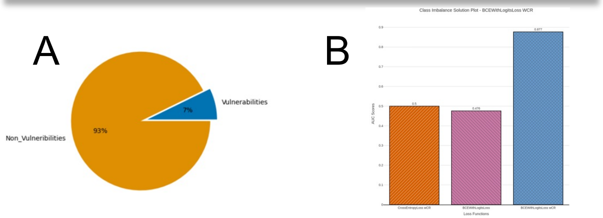
Graph Convolution Network (GCN) models to classify node properties based on node (protein and its properties), egde (protein-protein interaction), and topology information (protein interaction network structure)

Training dataset based on **proteomics and genetic screening data of 171 cancer cell lines**

Data Insights and Need of Targeted Mask

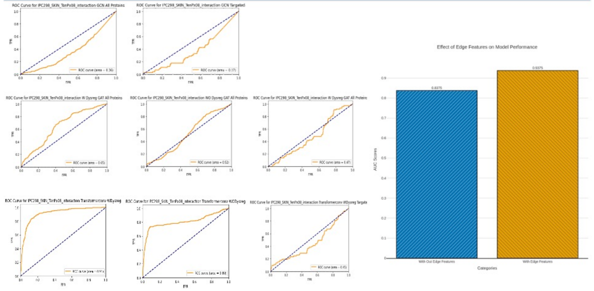


Class Imbalance and Solutions



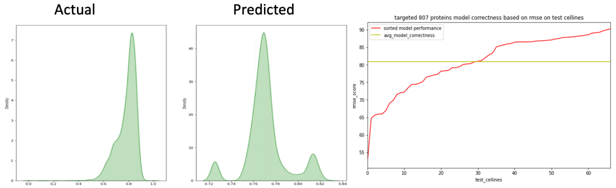
Only a small number of proteins in cancer cell lines are cell line-specific vulnerabilities (5-10 % of total proteins). (A) shows the distribution of vulnerabilities and non-fitness proteins in MDAMB453. The class imbalance is a challenge for prediction models. (B) Shows performance improvements upon addressing the imbalance using the Binary Classification Entropy Method.

Effect of DysReg Interaction and Case Study



Regression For Remedy

$$\text{Model correctness} = 1 - \frac{rmse}{Avg\ gene\_effect_{celline}} \times 100$$



Summary

- Effective representation of PPI as a network
- Optimal Model training for node classification
- Handling class imbalance
- Performance evaluation on targeted and non -Targeted proteins-eventually regression Model Training
- Custom Regression Model Performance Measure

References:

1. Jervakani, M. T. T., Dehghanian, Z., Aminian, G., & Rabiee, H. R. (2024, August). *KLCE: Regularized Imbalance Node-classification via KL-divergence and Cross-Entropy*. Retrieved from OpenReview.
2. Moreno, P., Ho, P., & Vasconcelos, N. (2004, March). *A Kullback-Leibler Divergence Based Kernel for SVM Classification in Multimedia Applications*. Retrieved from ResearchGate.
3. Wu, T., Ren, H., Li, P., & Leskovec, J. (2020). *Graph Information Bottleneck*. In *Proceedings of the Springer Lecture Notes in Computer Science* (pp. xx-xx). Springer. Retrieved from Springer Link.
4. *Dynamic Graph Information Bottleneck*. (2024, February). Retrieved from ACM Digital Library.
5. Ritvikmath. (2023, January). *The KL Divergence: Data Science Basics* [Video]. Retrieved from YouTube.