Assignment 3
Report

Name: Jonas Schrade
Student Number: 01/1080887
Course (Instructor): Deep Learning for Social Science (Giordano Di Marzo)

1 Introduction

The objective of Assignment 3 is to prepare and assess large-scale graph data and to construct, train, and evaluate Graph Convolutional Networks (GCNs) for two tasks: classifying Twitch channels based on popularity and language homophily.

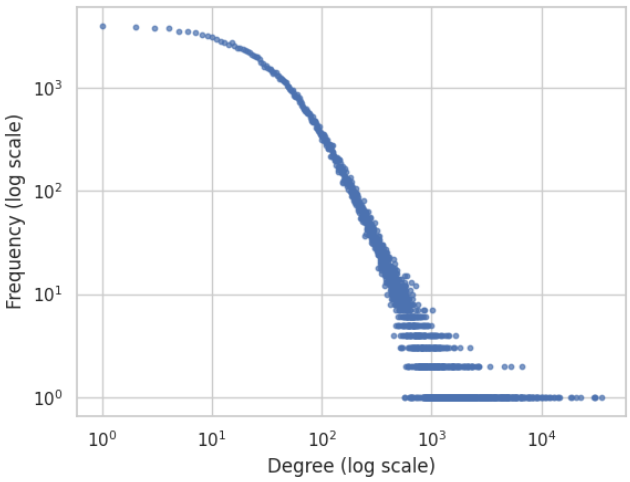


Figure 1: Degree distribution (log-log scaled).

2 Data Analysis

The dataset comprises 168,114 nodes and 6,797,557 undirected edges, representing Twitch channels and their mutual following relationship. Despite an average degree of 80.87 suggesting density, the heavy-tailed degree distribution (Figure 1) indicates a few high-degree nodes skew the average. Nodes include categorical and continuous features, such as maturity, affiliate, and account status; timestamps for last update and creation; and language. Datetime features are converted to UNIX time, continuous variables are log-scaled, and languages are one-hot encoded. Due to perfect multicollinearity (Figure 2), account lifetime is dropped in favor of creation date. The two classification targets are: (1) a binary popularity label (views > 10,000), and (2) a multi-class label for the top five languages. Both targets are highly imbalanced (Figure 3), so class-weighing of losses is applied later. To mitigate overemphasis on dominant classes, weights are log-smoothed, improving balance and training stability. The data is split into training, validation, and testing sets in a 60%/20%/20% ratio, with non-dummy features standardized across all splits using a scaler fit exclusively on the training set; for GNNs, data separation is done by masking nodes — preserving the graph structure — while maintaining overall class proportions.

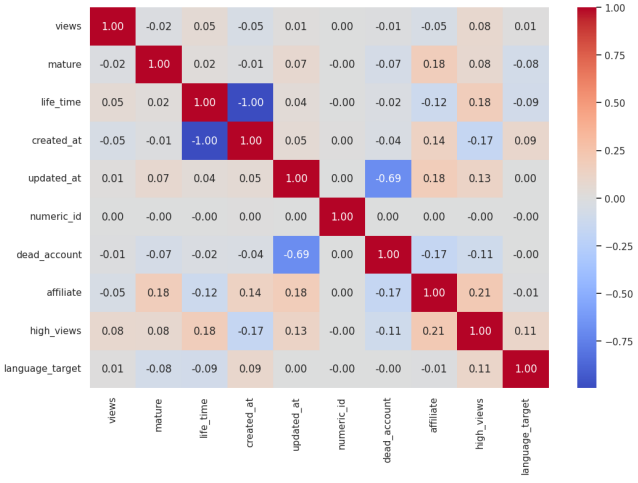


Figure 2: Correlation matrix.

3 Methods

For benchmarking, MLPs use two hidden layers (128–64–32) with ReLU activation, dropout (0.5, 0.3, 0.2), and a linear output layer (6 nodes for multi-class, 2 for binary). GCNs apply two graph convolution

layers with ReLU, 0.5 dropout, a hidden size of 32, and a linear classifier. All models are trained for 500 epochs using Adam optimization algorithm, class-weighted cross-entropy, early stopping (patience 25, min delta 0.0001), and L2 regularization. MLPs use a batch size of 64, learning rate 0.001, and weight decay 0.001; GCNs use a learning rate of 0.005 and weight decay of 0.0005.

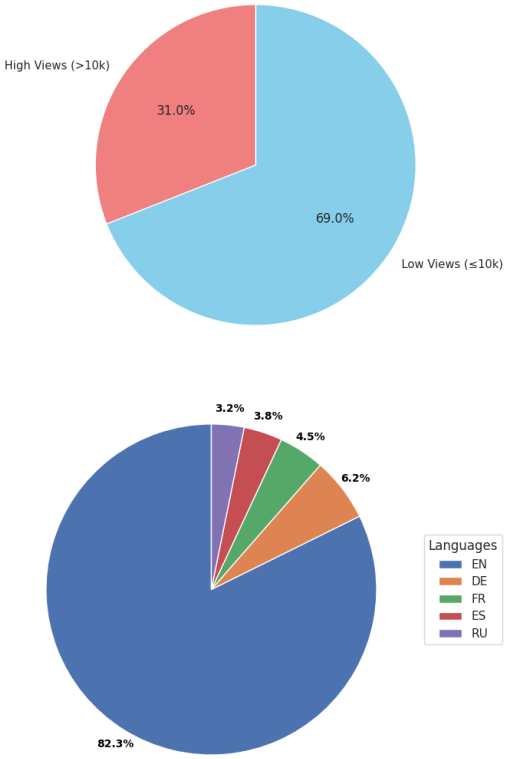


Figure 3: Target variables.

Learning curves for Task A and B are shown in Figures 4 and 5. GCNs show smooth convergence, with a sharp loss drop in the first 100 epochs and gradual improvement thereafter, reaching validation losses of 0.420 (A) and 0.969 (B), and training losses of 0.447 and 1.096. MLPs decline quickly at first but plateau irregularly, stopping early at epoch 44 (A) and 36 (B). Final losses are 0.564/0.558 (train/val) for Task A and 1.635/1.621 for Task B.

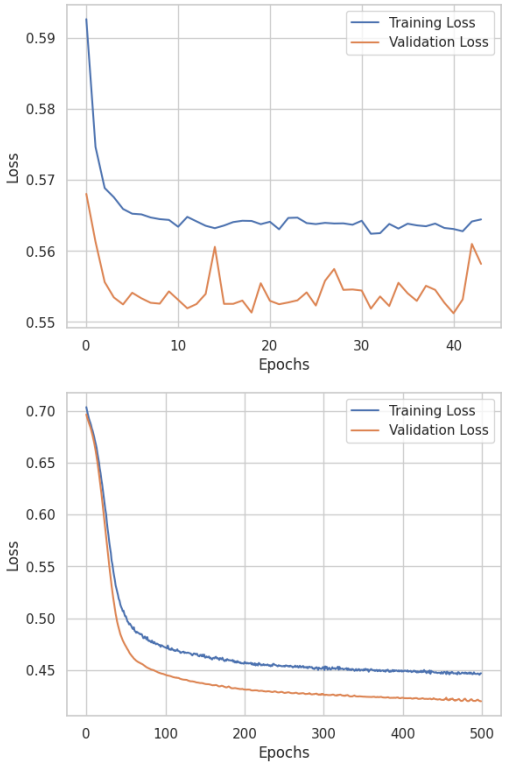


Figure 4: Learning curves for MLP (top) and GCN (bottom) of Task A.

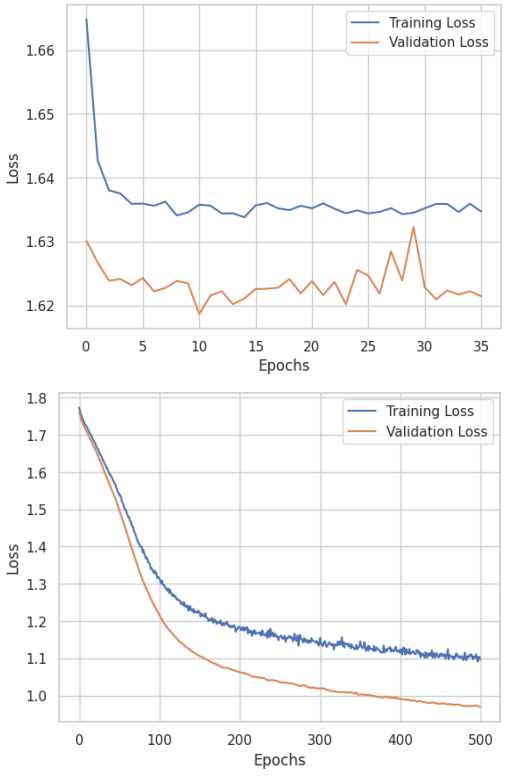


Figure 5: Learning curves for MLP (top) and GCN (bottom) of Task B.

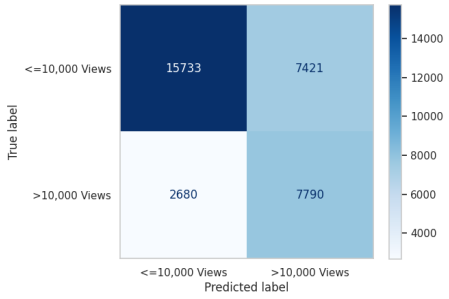
4 Results

Table 1: **Task A**: Test: MLP vs GCN

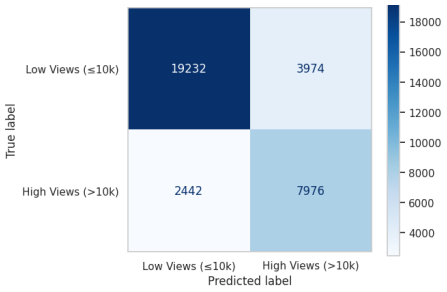
Metric	Class	MLP	GCN
Accuracy	-	0.700	0.809
F1-score	≤10k Views	0.757	0.857
	>10k Views	0.607	0.713
Macro Avg	Precision	0.683	0.777
	Recall	0.712	0.797
	F1-score	0.682	0.785

Table 2: **Task B**: Test: MLP vs GCN

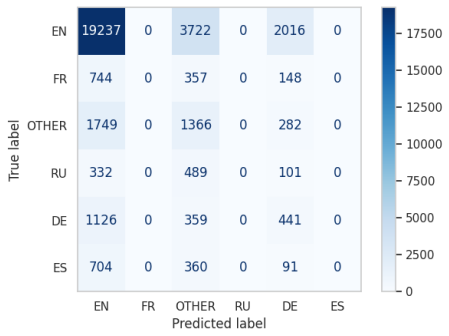
Metric	Class	MLP	GCN
Accuracy	-	0.626	0.834
F1-score	EN	0.787	0.928
	FR	-	0.397
	OTHER	0.272	0.535
	RU	-	0.522
	DE	0.176	0.772
	ES	-	0.470
Macro Avg	Precision	0.192	0.612
	Recall	0.234	0.600
	F1-score	0.206	0.604



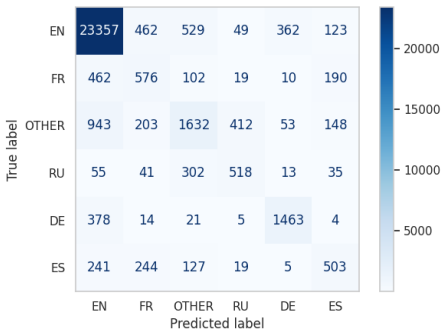
(a) Task A: MLP



(b) Task A: GCN



(c) Task B: MLP



(d) Task B: GCN

Figure 6: Confusion matrices for MLP and GCN models on Task A and Task B.

5 Discussion

Both GCN models clearly outperform the MLP benchmarks across their respective classification tasks in terms of accuracy, overall F1-score, precision, recall, and class-specific F1-scores. In Task B, the GCN demonstrates a notable advantage by effectively retrieving the majority class and achieving reasonable performance on minority classes—an area where the MLP benchmark struggles heavily. Leveraging the graph structure and the clustering of language groups (i.e. language homophily), it achieves decent to strong classification performance. Furthermore, the GCN in Task A benefits from a smaller, more balanced class distribution, yielding good accuracy, precision, recall, and F1-scores.

6 GraphSAGE

For Task B, a GraphSAGE model with two SAGEConv layers, ReLU, dropout, and global mean pooling is trained on the same data masks as the GCN. It slightly outperforms the GCN (see Table 3).

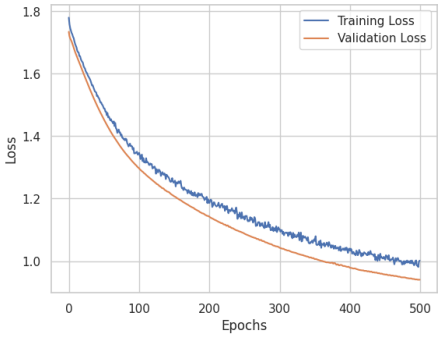


Figure 7: Learning curve for GraphSAGE model of Task B.

Table 3: **Task B**: Test Performance: GraphSAGE

Metric	Class	GraphSAGE
Accuracy	-	0.841
F1-score	EN	0.932
	FR	0.485
	OTHER	0.595
	RU	0.611
	DE	0.742
	ES	0.397
Macro Avg	Precision	0.623
	Recall	0.632
	F1-score	0.627

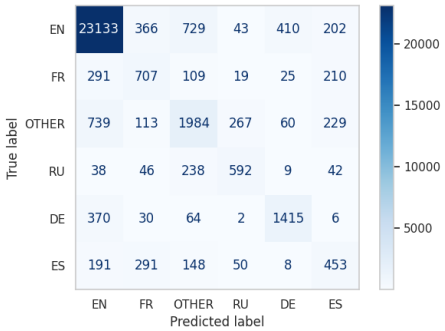


Figure 8: Learning curves for MLP (top) and GCN (bottom) of Task A.

7 Regression

The regression task predicts continuous views using a two-layer GCN with ReLU and dropout and a four-layer MLP with LeakyReLU and dropout. Features are split into dummy and numerical types, with numerical features standardized on training data. Both models are trained with early stopping based on validation MSE, using the Adam optimizer with weight decay and learning rates of 0.005 (GCN) and 0.001 (MLP). The GCN has hidden dimensions of 32 and 16, while the MLP uses layer sizes 256, 128, and 64. In contrast to the classification tasks, the GCN does not appear to outperform the MLP.

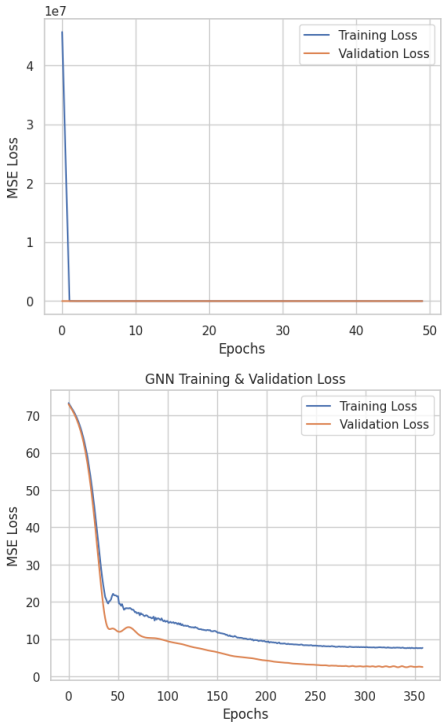


Figure 9: Learning curves for MLP (top) and GCN (bottom) of Task A.

Table 4: **Regression Performance** on Original Scale: MLP vs GNN

Metric	MLP	GNN
MSE	8.48×10^{12}	1.08×10^{13}
MAE	1.62×10^{5}	1.91×10^{5}
R^2	-0.0030	-0.0028