Universiteit Leiden

Master Computer Science

[Salary Prediction Using Graph Attention Networks and Mixture Density Network]

Name:	[Zhipei Qin]
Student ID:	[3977226]
Date:	[August 2025]
Specialisation:	[Computer Science: Data Science]
1st supervisor:	[Niels van Weeren]
2nd supervisor:	[Frank Takes]

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Accurate salary prediction is a significant challenge due to the complex and heterogeneous nature of job market data. Traditional predicting models typically yield a single point estimate for salaries. This approach often fails to capture the inherent uncertainty and multi-modal distributions of compensation. Additionally, these models are generally incapable of leveraging the complex relational and hierarchical structures between key predictive attributes. This paper proposes a framework that combines graph attention networks (GATs) [38] with a mixture density network (MDN) [5] to model the full conditional probability distribution of salaries. We construct distinct graphs for three key attribute domains: location, occupation, and industry. Each node in our graphs is represented by a unique, low-dimensional embedding vector to capture its core semantic content. The edges that connect these nodes are defined by two principles: Hierarchical edges, which link parent-child attributes, and (Weighted) Similarity Edges, which connect peer-level nodes based on semantic relatedness. Our multigraph GAT architecture learns rich, contextual representations for all nodes by leveraging information propagated along explicitly defined hierarchical and similarity-based edges. The resulting node representations from each attribute domain are then integrated to form a single, comprehensive feature vector, which is subsequently mapped by the MDN head to the parameters of a Gaussian Mixture Model (GMM). A priority-based hierarchical selection method is employed to select the most granular available feature representation for any given input sample. Experimental results show that our GAT-based model achieves significantly higher prediction accuracy compared to the simple MLP-MDN baseline model that does not exploit graph information. This approach allows us to quantify prediction uncertainty and identify complex salary structures, offering a more transparent and insightful tool for analyzing the labor market.

Keywords: Salary Prediction, Graph Attention Networks (GATs), Mixture Density Network (MDN), Gaussian Mixture Model (GMM) Hierarchical Edges, Weighted Similarity Edges, Multi-Graph Model.

Contents

1 Introduction

The Human Resources and recruitment industry takes on an intermediary role in the labor market, ideally matching the right talent to the appropriate organizational role. In this dynamic, salary operates as the most significant mechanism through which skills and experience are valued and directly related to an organization's ability to attract talent and manage operational costs. Thus, determining the right and fair market value for any specific position stands as one of the most precarious tasks in this field.

A primary manifestation of this challenge is the salary information gap between job seekers and employers, a persistent problem where applicants often lack clarity on compensation until a final offer is made, while employers risk either losing candidates with uncompetitive offers or inflating labor costs with overly high ones. Bridging this gap, where the objective market value of a position is determined, ensures minimization of job mismatches and enhanced hiring outcomes for both sides. [34] In addition, the effective integration of heterogeneous salary data improves the accuracy of salary prediction and the transparency of the labor market. Beyond the improvement in salary prediction accuracy that could be achieved, integrating heterogeneous salary data sources gives a holistic view of salary structures in various industries and regions. [29] Addressing these challenges will improve salary estimation models, improve labor market efficiency, and support better career decision-making. Salary prediction models have been extensively studied using various methodologies. Traditional salary prediction approaches, such as regression models and even advanced machine learning techniques, often fall short in two critical aspects. Firstly, by predicting a single value (point estimate), they are insufficient to represent the full conditional distribution of salaries, which may be multi-modal or exhibit significant variance even for individuals with similar job profiles. Secondly, these models typically treat predictive attributes as independent features, thus failing to exploit the valuable information embedded in the complex hierarchical and relational structures within the data.

These limitations directly motivate the central research questions of this thesis. First, to what extent can we improve salary prediction accuracy by explicitly modeling the hierarchical and similarity-based relationships between attributes using a graph-based framework? Second, how can a probabilistic model be designed to capture the full, often multi-modal, distribution of salaries, thereby providing a more realistic and insightful output than a single point estimate?

This limitation is particularly pronounced when dealing with real-world recruitment data, which is the focus of our work. Our study uses a dataset of real-world job postings where the hierarchical and similarity structure of a domain is evident. We focus mainly on three domains in job and contract information: Location (the geographical location of the job position), Industry (the industry to which the job position belongs), and Occupation (the professional title corresponding to the job position). A key characteristic of these data is its inherent hierarchical nature; for instance, the Location domain contains multiple levels of granularity, such as provinces and the cities nested within them. Furthermore, there are also similarity relationships between nodes at the same level, e.g., all cities, and different attribute values may exhibit varying degrees of similarity. For instance, the similarity between Amsterdam and Rotterdam is higher than that between Rotterdam and London. These two types of structured information, which are often overlooked by traditional models, provide a unique opportunity to build a more context-sensitive predictive framework.

To seize this opportunity and address the above limitations, this study proposes a novel framework for predicting salary distribution via the relational and hierarchical information encoded in the different salary-influencing attributes like geographic location, occupation, and industry sector. Moving beyond point estimates, the framework combines domain-specific graph representation learning through GAT with an MDN head. It first establishes a multi-graph structure in which edges record information about explicit hierarchical containment and functional similarity. Next, the GAT architecture dynamically learns the importance of these diverse relations, focusing solely on those that matter most. A masked self-attention mechanism achieves this by letting the model compute attention coefficients for each node in its local, first-order neighborhood. Based on features from nodes and edge types, these coefficients determine the weight a given neighbor's contribution bears during the subsequent feature aggregation step. This multi-head attention mechanism allows the model to learn different relational patterns in parallel, creating a powerful yet highly cohesive representation per node. The model integrates a priority-based, feature selection logic in terms of hierarchical over-writing. In the process of selecting features for each sample, it prioritizes the most specific information, using a hierarchical overwrite logic that enables the model to use the most granular attribute values while being robust to broader attribute ranges or even missing inputs through general representations. Finally, the MDN head uses such richly context-aware

features to model the conditional probability of salaries as a flexible Gaussian Mixture Model. The number of components in this mixture is a tunable hyperparameter. Therefore, our framework can better fit the actual, often complex, salary distribution, captured prediction uncertainty, and potential multimodality than a single distribution.

In the following sections, our study firstly introduces the related work in Sect. 2. Then, we formally introduce the preliminaries and define the problem of salary prediction with the GAT-MDN framework in Sect. 3. Sect.4. provides a detailed description of our research methodology and specific model architecture. Following this, Sect. 5. presents an exploratory data analysis in our experiments. The complete experimental setup and the results of our salary predictions are then reported in Section 6. Finally, our study presents our discussions in Sect.7. and discuss our future work in Section 8.

2 Related Work

This section reviews the literature in three key areas that form the foundation of our proposed model. Our study begins by surveying the field of salary prediction to contextualize our work and highlight existing challenges. Subsequently, our study discusses the literature on multiple forms of graph models, which are central to our relational feature extraction methodology. Finally, our study discusses prior research on mixture density networks, the Gaussian mixture model, and the mechanism that enables our model's probabilistic output.

2.1 Data-Driven Salary Prediction

Predicting salary changes involves applying various analytical techniques to estimate how salaries will evolve over time, considering factors like economic conditions, industry trends, and job market dynamics. These methods are used to forecast salary growth, fluctuations, and the impacts of specific events on compensation levels. Recent studies have increasingly employed data-driven approaches to examine the factors that influence salary levels [43]. A linear regression approach could be used for the problem. However, statistical models often struggle with non-linearity, high-dimensional features, and data sparsity, whereas machine learning, deep learning and time series models excel at capturing intricate patterns in salary data. These models integrate data sources including job postings, employee surveys, company attributes, and regional economic indicators, allowing for more accurate and dynamic salary estimations. [25] predicts workforce salaries in the Saudi Arabian economy evaluated by five machine learning algorithms, finding that non-linear models significantly improve the goodness-of-fit for the regression task across all positions, economic activities, and organization sizes. [1] applied the Support Vector Machine technique to demographic data of talent in order to forecast income levels. [44] predicts the salary of data science professionals using decision trees, random forests, and gradient boosting, and the Decision Tree Regression model is most efficient when it comes to salary prediction. For forecasting the univariate time series salary data, [32] discusses the effectiveness of five different machine learning and time series methods (Autoregression, Moving Average ARIMA, MLP, and CNN) for salary prediction, where autoregression achieved the best results with the fewest parameters. [42] introduces a salary dynamic fluctuation trend prediction model based on multivariate time series, the BP neural network was constructed, factors affecting salary fluctuations were analyzed and future salary trends were predicted. Meng et al.(2018)[29] proposes a data-driven approach for intelligent salary benchmarking using large-scale, fine-grained online recruitment data. It addresses the challenges of traditional methods that struggle with dynamic scenarios and timely benchmarking by framing the salary benchmarking problem as a matrix completion task and developing the Holistic Salary Benchmarking Matrix Factorization (HSBMF) model. Meng et al.(2022)[28] further aim at the problems of data sparsity, cold start and poor model interpretability in traditional salary benchmarking (JSB), a data-driven model based on nonparametric Dirichlet process (NDP-JSB) was proposed. Sun, Y.et al. (2021)[35] propose a novel data-driven model called Salary-Skill Combination Network (SSCN), which transforms the salary prediction task into a collaborative process to separate and evaluate the value of individual skills without direct value labels. [18] introduces LGDESetNet, a set-based neural prototyping method that, by combining a skill graph-enhanced disentangled subset selection layer and set-oriented prototype learning, is able to identify influential skill subsets and provide clear insights into the composition effects of these skill sets. A significant body of research has also been dedicated to predicting salaries from the unstructured text within job descriptions. In a novel application for salary prediction from text, Wang et

al. (2019)[40] developed a Bi-directional GRU-CNN model, a deep hybrid architecture that surpasses the performance of established models such as TextCNN and RCNN. While these data-driven salary predicting studies primarily focus on predicting a single point estimate or leveraging specific feature types, our work introduces a GAT-MDN framework that enables our model to not only learn from complex hierarchical and relational structures within job attributes but also to predict the full conditional probability distribution of salaries.

2.2 Graph-Based Model

Graph-based models offer a powerful paradigm for salary prediction because they have the unique ability to go beyond the limitation of treating attributes as independent inputs. By representing job market data as networks, these models can explicitly learn from complex hierarchical and relational structures. [45] presents HetGNN, a model designed for representation learning on heterogeneous graphs, which contain diverse types of nodes and content. It addresses the limitations of traditional methods by using a random walk-based strategy to sample different types of neighbors, followed by a neural network to aggregate their features. In the context of salary prediction, it can effectively model career progression by treating job positions as nodes and transitions as edges, helping to handle sparse data by inferring salary trends from related career movements. Furthermore, its ability to aggregate diverse information makes it suitable for integrating various data sources like job descriptions and company details to improve prediction accuracy.

There are techniques for identifying and extracting meaningful patterns (subgraphs) within large graph structures — subgraph mining. In the context of salary modeling, subgraph mining can be applied as a powerful technique for discovering complex, structural patterns within the labor market that are not apparent from analyzing individual attributes alone. This technique can identify frequent and significant substructures in graph data, providing a rich source of features for graph-based models. By representing the entire dataset as a large, interconnected graph of jobs, skills, industries, and locations, subgraph mining can identify "career archetypes" or "economic clusters" as frequently recurring subgraphs. [16] proposes a novel frequent subgraph mining algorithm, FFSM, which uses a vertical search strategy within an algebraic graph framework to reduce redundant candidate subgraphs. [41] proposes a new graph-based pattern mining algorithm called gSpan, which discovers frequent subgraphs without candidate generation by building a new lexicographic order among graphs and using depth-first search. The method first constructs a lexicographically ordered collection of graphs, then assigns a canonical label to each one by transforming it into its unique minimal DFS code. This code is a string representation of the graph, generated by listing the edges in the order they are visited during a depth-first search. By finding the lexicographically smallest of all possible strings for a given graph, this "minimal" code provides a unique and canonical identifier, ensuring that structurally identical graphs can be easily and efficiently compared. Additionally, many methods exist for searching for information-rich subgraph patterns, such as AGM [17], Gaston [31], and MoFa[6].

Chen et al. (2020) [8] propose a novel semi-supervised approach to job salary prediction by constructing a graph database from job postings and leveraging both metadata, e.g., job type and skills, and relational features, i.e., connections between similar positions. They demonstrate that these two feature sets are conditionally independent yet each sufficient for prediction, and integrate them as dual "views" within a Graph Convolutional Network (GCN) framework. By exploiting large amounts of unlabelled data alongside a smaller labelled set, their GCN model significantly outperforms baseline methods that simply concatenate all features. This work is among the first to formally cast salary estimation as a machine learning task and to employ semi-supervised deep learning on graph-structured job data for improved accuracy. [24] further proposes an enhanced Gate Graph Neural Network (GGNN) that incorporates gated recurrent units (GRUs) and modern optimization techniques to effectively process graph structures and generate sequence outputs. [30] introduces a CNN framework, PATCHY-SAN for arbitrary graphs, where the key technique is a universal approach to extracting locally connected regions, thus generalizing the convolution operation to graph data.

Our model employs the Graph Attention Networks (GATs) architecture proposed by Veličković et al. [38]. Unlike the aforementioned methods, which typically aggregate neighborhood information uniformly, the core innovation of the GAT architecture is its ability to dynamically learn and assign a different "attention" weight to each neighboring node, thereby enabling it to selectively focus on the most influential attributes and relationships for a more nuanced prediction. There are also some non-spectral graph convolution methods that are in the same category as GATs and provide good context for other ways of defining convolutions directly on the graph, see [10] and [3].

2.3 Mixture Density Network

The mixture density network (MDN) approach, originally proposed by Bishop [5], enables the network to predict the parameters of a probability distribution for the target variable, conditioned on the input features. This makes it exceptionally well-suited for problems where the target variable has a complex, non-Gaussian relationship with the inputs, such as multi-modal or having input-dependent variance data. [21] proposed a hybrid speech enhancement framework, in which the core role of the MDN is to receive noisy speech and predict the complete probability distribution of all possible clean speeches, thus providing critical prior knowledge containing uncertainty information for the subsequent step of VTS-based enhancement. In the field of handwriting synthesis, [12] combined MDN with LSTM neural network to generate realistic cursive handwriting by predicting a single data point, where MDN used a mixture of bivariate Gaussian distribution and Bernoulli distribution to model the complex probability distribution of pen coordinates at the end of a stroke. More recent developments on the MDN model can be found in [7].

The Gaussian Mixture Model, which stands out as one of the most powerful and widely used models within the MDN framework, serves two basic purposes: a semi-parametric approach of unknown data distribution and probabilistic data clustering. However, determining the appropriate number of components for a GMM is a core research problem in itself, as it directly impacts model performance. This topic has been extensively reviewed in [26]. In many applications, the GMM proved to be very flexible; for example, in computer vision, in [33], a classical real-time background subtraction method was proposed in which the color distribution for each pixel is modeled using a GMM, allowing a good separation of foreground objects from dynamic background scenes. In bioinformatics, the GMM covariance structure has been extended to better cluster gene expression data while acquiring intergene correlation in small samples [27].

The integration of GMMs with deep learning is a prominent and active research direction. Rather than applying GMMs as standalone models, modern approaches leverage deep neural networks to first learn meaningful, low-dimensional feature representations from complex data, which then serve as a more effective input for GMM-based modeling and clustering. For example, the Variational Deep Embedding (VaDE) framework embeds a GMM as a prior distribution within the latent space of a Variational Autoencoder (VAE), jointly optimizing for both deep representation learning and clustering parameters to achieve state-of-the-art results and generate realistic samples for each cluster [19]. GMMs are also used as output heads for Recurrent Neural Networks (RNNs), where an RNN compresses past observations into a state representation which an MDN then uses to predict the probability distribution of the next state [13]. In other applications, a framework by Belciug et al. uses GMM clustering as part of a collaborative voting system with multiple deep learning algorithms to classify fetal morphological scan view planes [4]. Our model is based on a similar concept, which innovatively uses multi-graph GATs to extract deep features from the inherent hierarchical and relational structures of salary data. These features are then fed into the head of an MDN to directly predict the parameters of a GMM that can describe complex salary possibilities.

3 Preliminaries

This section formally defines the salary prediction task and details the architecture of our proposed GAT-MDN model. The methodology is presented in a structured manner to clearly distinguish between the foundational techniques we adopt from prior work and the novel adaptations we introduce to solve the specific challenges of our task. We will begin by defining our graph-based data representation, then formalize the probabilistic prediction goal, describe the model's architecture, and finally, specify the training objective.

3.1 Graph-Based Data Representation

To address the limitation that predictive attributes are treated as independent features, we adopt a graph-based representation here and formally model the job market data for the attribute domains of Location, Occupation, and Industry. In this framework, the graph's nodes (V_m) represent the unique attribute values themselves, e.g., 'Amsterdam', 'Software Engineer', 'Retail', thereby allowing the model to learn about the context representations and not from the data points' isolation.

The expressiveness of this approach will be to define the edges E_m. In previous studies, edges were often constructed as a single type. As our primary contribution in data representation, we construct the edge set as a union of two complementary types to capture the full complexity of our specific dataset: $E_m = E_{\text{hier},m} \cup E_{\text{sim},m}$, which is the union of Hierarchical Edges ($E_{\text{hier},m}$) and weighted Similarity Edges ($E_{\text{sim},m}$). First, there are Hierarchical Edges, which encode all explicit structural parent-child relationships that the data express. Then, to express the next-level implicit relationships among nodes at the same level, our study adds Similarity Edges based on the semantic similarity quantified by pre-trained language models. This dual-edge strategy provides a rich structural foundation for our model, the formal implementation of which is detailed in the following definitions.

Let $A = \{a_1, a_2, \ldots, a_N\}$ represent the set of N specific attribute values defining an instance of interest, e.g., $a_1 =$ `'Rotterdam'`, $a_2 =$ `'Dairy Farmer'`, $a_3 =$ `'Administrative And Support Service Activities'`. These N attributes belong to $M = (m_1, m_2, ..., m_n)$ distinct domains. In our research, we set $n = 3$, presenting three domains: Location m_{loc}, Occupation m_{occ}, and Industry m_{ind}.

For each attribute domain $m \in M$, our study constructs a corresponding graph $G_m = (V_m, E_m)$. Here, V_m is the set of all possible unique values (nodes) for the attribute in domain m. Each node $v \in V_m$ is associated with an initial feature vector x_v, forming the feature matrix X_m for graph G_m. The specific attribute value a_m for the instance A corresponds to a node in V_m. The edge set $E_m = E_{\text{hier},m} \cup E_{\text{sim},m}$ is the union of Hierarchical Edges ($E_{\text{hier},m}$) and weighted Similarity Edges ($E_{\text{sim},m}$).

Hierarchical Edges: Let D_{all} be the dataset of all instances. Let $L_{m,p}$ and $L_{m,c}$ be two adjacent hierarchical levels within domain m, where p denotes the parent level and c denotes the child level, e.g., $L_{\text{loc, province}}$ and $L_{\text{loc, city}}$. For any instance $d_i \in D_{all}$, let $d_i(L_{m,p})$ and $d_i(L_{m,c})$ be the attribute values for these levels. The set of hierarchical edges for domain m, $E_{\text{hier},m}$, is defined as:

$$E_{\text{hier},m} = \bigcup_{d_i \in D} \{\{u, v\} \mid u = d_i(L_{m,p}), v = d_i(L_{m,c}), u, v \neq \texttt{null}\} \tag{1}$$

Weighted Similarity Edges: The set of weighted similarity edges for all levels in domain m, $E_{\text{sim},m}$, is defined as the union of edges for each level l:

$$E_{\text{sim},m} = \bigcup_{l} \{(u, v, w_{uv}) \mid u, v \in V_{m,l}, u \neq v, w_{uv} > \tau\} \tag{2}$$

where the weight w_{uv} is the cosine similarity between the embedding vectors produced by a large pre-trained model for the node names u and v:

$$w_{uv} = \text{sim}(u, v) = \frac{\Phi(u) \cdot \Phi(v)}{\|\Phi(u)\| \|\Phi(v)\|} \tag{3}$$

Here, τ is a predefined similarity threshold to filter out weak or noisy connections.

Given a user-defined query consisting of sets of desired attribute values for each domain, $\mathcal{Q}_{\text{loc}}, \mathcal{Q}_{\text{occ}}, \mathcal{Q}_{\text{ind}}$, our study first defines the complete set of attribute combinations $\mathcal{A}_Q$ as their Cartesian product:

$$\mathcal{A}_Q = \mathcal{Q}_{\text{loc}} \times \mathcal{Q}_{\text{occ}} \times \mathcal{Q}_{\text{ind}} \tag{4}$$

Our goal is twofold: first, to predict the salary distribution and the point estimate for each individual combination $A_j \in \mathcal{A}_Q$; and second, to derive an aggregate distribution and point estimate for the entire queried group.

3.2 Probabilistic Prediction Goal

For each individual instance subset $A \in \mathcal{A}_Q$, our study aims to learn a parametric model f_θ that maps the input attributes A (leveraging the context provided by the graphs $\{G_m\}$ and features $\{X_m\}$) to a predicted probability distribution $\hat{p}_\theta(y|A)$, which approximates the true underlying distribution $p(y|A)$. Here, $y \in \mathbb{R}$ denotes the target variable, a continuous value representing the salary associated with the instance A.

Our study models this predicted conditional distribution $\hat{p}_\theta(y|A)$ using a **Gaussian Mixture Model (GMM)** with K components:

$$\hat{p}_\theta(y|A) = \sum_{k=1}^{K} \pi_k(A;\theta)\, \mathcal{N}(y|\mu_k(A;\theta), \sigma_k(A;\theta)^2) \tag{5}$$

where, for each component k:

- $\pi_k(A;\theta)$ is the mixture weight (prior probability) of the k-th Gaussian component, conditioned on A and model parameters θ. $\sum_{k=1}^{K} \pi_k = 1$ and $\pi_k \geq 0$.
- $\mu_k(A;\theta)$ is the mean of the k-th Gaussian component.
- $\sigma_k(A;\theta)$ is the standard deviation of the k-th Gaussian component ($\sigma_k > 0$).
- $\mathcal{N}(y|\mu, \sigma^2)$ denotes the probability density function of a Gaussian distribution.

While the primary output of our model is the full conditional probability distribution $\hat{p}_\theta(y|A)$, a single-value **point estimate** $E[y|A]$ can also be derived. This point estimate is the predicted average salary for the combination A.

$$E[y|A] = \sum_{k=1}^{K} \pi_k(A;\theta) \cdot \mu_k(A;\theta) \tag{6}$$

To analyze the queried group as a whole, our study also computes an aggregate salary distribution and an aggregate point estimate. This is achieved by calculating a weighted average over all individual combinations $A_j \in \mathcal{A}_Q$.

The weight for each combination, $W(A_j)$, is derived from its prevalence in the source dataset, e.g., based on sample counts. The **aggregate salary distribution** for the entire query set $\mathcal{A}_Q$ is then defined as the weighted mixture of the individual predicted distributions:

$$\hat{p}_\theta(y|\mathcal{A}_Q) = \sum_{A_j \in \mathcal{A}_Q} W(A_j) \cdot \hat{p}_\theta(y|A_j) \tag{7}$$

Correspondingly, the **aggregate point estimate** is the weighted average of the expected values of all individual combinations:

$$E[y|\mathcal{A}_Q] = \sum_{A_j \in \mathcal{A}_Q} W(A_j) \cdot E[y|A_j] \tag{8}$$

or

$$\hat{y}_{\text{mode}}(A_Q) = \sum_{A_j \in \mathcal{A}_Q} W(A_j) \cdot \hat{y}_{\text{mode}}(A) \tag{9}$$

3.3 Model Architecture

The architecture of our model represents a novel synthesis of graph representation learning and probabilistic prediction, specifically tailored to overcome the limitations of prior approaches. While graph-based models have been applied to relational data, our framework introduces a key innovation by constructing a multi-relational graph with both hierarchical and similarity-based edges. This provides the Graph Attention Network (GAT) feature extractor with a far richer and more realistic structural context than models that rely on a single type of relationship. Furthermore, whereas most salary prediction models yield a single point estimate, our approach distinguishes itself by employing a Mixture Density Network (MDN) as the predictive head.

The model $f_\theta(g_{\theta_g}, d_{\theta_d})$ consists of two main parts:

1. **A multi-graph GAT feature extractor,** g_{θ_g}. This extractor itself operates in two main stages:

   (a) Parallel Representation Learning: For each attribute domain m, a dedicated GAT sub-module, $g_{\theta_{g,m}}$, is employed. This process begins by creating an augmented feature matrix, $X'_m = \text{concat}(X_m, e'_m)$, by concatenating the initial node features X_m with a dedicated matrix of learnable embeddings e'_m, in order to provide a richer signal. The GAT sub-module then operates on these augmented features and the edge set $E_m = E_{\text{hier},m} \cup E_{\text{sim},m}$ to produce a final matrix of learned, context-aware node representations:

   $$H_m = g_{\theta_{g,m}}(X'_m, E_m) \tag{10}$$

   Here, H_m contains the learned high-dimensional embeddings for all unique nodes within that attribute domain, enriched by both structural and semantic relationships among nodes.

   (b) **Feature Integration:** Based on the specific attribute values in the input instance $A = \{a_1, \ldots, a_M\}$, a selection function retrieves the corresponding node representation vector $\mathbf{h}_m$ from each matrix H_m. These vectors are then concatenated to form the final integrated feature representation, $\mathbf{h}_{\text{comb}}$:

   $$\mathbf{h}_{\text{comb}} = \text{concat}(\mathbf{h}_{\texttt{loc}}, \mathbf{h}_{\texttt{occ}}, \mathbf{h}_{\texttt{ind}}) \tag{11}$$

2. **An MDN head,** d_{θ_d}, which maps the integrated representation $\mathbf{h}_{\text{comb}}$ to the GMM parameters: $\{\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\sigma}\} = d_{\theta_d}(\mathbf{h}_{\text{comb}})$.

The overall model parameters are $\theta = \{\theta_g, \theta_d\}$.

3.4 Training Objective

Given a training dataset $D = \{(A^{(i)}, y^{(i)})\}_{i=1}^{N_{\text{samples}}}$ consisting of attribute sets and corresponding single observed salaries, the model parameters θ are learned by minimizing the combination of **negative log-likelihood (NLL)** loss and **mean squared error (MSE)** loss over the dataset. The foundation of our training objective is the standard Negative Log-Likelihood (NLL) loss, which is defined as:

$$\begin{aligned}\mathcal{L}_{\text{NLL}}(\theta) &= -\sum_{i=1}^{N_{\text{samples}}} \log \hat{p}_\theta(y^{(i)}|A^{(i)}) \\ &= -\sum_{i=1}^{N_{\text{samples}}} \log \left(\sum_{k=1}^{K} \pi_k(A^{(i)};\theta)\, \mathcal{N}(y^{(i)}|\mu_k(A^{(i)};\theta), \sigma_k(A^{(i)};\theta)^2) \right)\end{aligned} \tag{12}$$

However, relying solely on NLL can sometimes lead to slower convergence, particularly when anchoring the mean of the distribution. To address this for our specific task of salary prediction, we introduce a supplementary Mean Squared Error (MSE) loss component as our novel adaptation. This term explicitly penalizes the difference between the true salary and the expected value of the predicted distribution, which helps to stabilize the training process and guide the model towards a more accurate point estimate.

$$\mathcal{L}_{\text{MSE}}(\theta) = \sum_{i=1}^{N_{\text{samples}}} \left(y^{(i)} - E[y|A^{(i)}] \right)^2 \tag{13}$$

where the expected value $E[y|A^{(i)}]$ is $\sum_{k=1}^{K} \pi_k(A^{(i)};\theta) \cdot \mu_k(A^{(i)};\theta)$.

The final combined loss, $\mathcal{L}_{\text{Combined}}(\theta)$, is therefore:

$$\mathcal{L}_{\text{Combined}}(\theta) = \underbrace{-\sum_{i=1}^{N_{\text{samples}}} \log\left(\sum_{k=1}^{K} \pi_k(A^{(i)})\mathcal{N}(y^{(i)}|\dots)\right)}_{\text{NLL Component}} + \alpha \cdot \underbrace{\sum_{i=1}^{N_{\text{samples}}} \left(y^{(i)} - \sum_{k=1}^{K} \pi_k(A^{(i)})\mu_k(A^{(i)})\right)^2}_{\text{MSE Component}} \tag{14}$$

where α is a hyperparameter that balances the two objectives. This objective function trains the model f_θ to output GMM parameters that assign high probability density to the observed salaries given their corresponding attribute combinations.

For a comprehensive list of the major notations used in this paper, please refer to major notations 3 in the appendix.

4 Methodology

This section is an in-depth technical description of our proposed Graph Attention Networks - Mixture Density Network (GAT-MDN), an end-to-end methodology from data ingestion to probabilistic prediction. The selection of this specific GAT-MDN architecture is a direct response to two fundamental challenges inherent in salary prediction. First, the predictive attributes are not independent; they possess a rich relational structure (e.g., hierarchical and similarity-based connections) that traditional models fail to exploit. We chose a Graph Attention Network (GAT) specifically to address this problem, as it is designed to learn context-aware representations by modeling these explicit relationships. Second, a salary is rarely a single, deterministic value but rather a distribution, often with multiple modes (e.g., for junior vs. senior roles). To solve this, we chose a Mixture Density Network (MDN) as the model's head, enabling it to predict the full, flexible probability distribution of a salary instead of an insufficient single point estimate. This combined GAT-MDN setup is therefore a principled choice, tailored to solve these core issues presented by our research question and the nature of our data.

Our study first discusses the pipeline for cleaning, transforming, and standardizing raw job market data. This is followed by a description of one of the essential components of our work: graph construction, in which we derive a rich multi-graph structure for each domain from relational attribute data. Then, we discuss the model's architecture when the data is represented as graphs. First, our study introduces the key engine, GAT, responsible for learning contextually rich representations from the graphs, and we explain how the priority-based hierarchical selection is used to deal robustly with inputs at different levels of granularity. The output of this graph-based feature extractor will be fed into an MDN head, which will produce the final probabilistic salary predictions. For optimization of the entire system, our study creates a hybrid loss function to account for the different objectives underlying the training process. Then, we provide an overall framework that illustrates how these pieces fit together, and we finally discuss the model's inherent capability for generalization over unseen combinations of attributes.

4.1 Data Preprocessing

To prepare the raw job posting data for our graph-based model, our study applies a systematic preprocessing pipeline. This section details the critical steps taken to ensure data quality and suitability for training, including outlier removal, the calculation of a representative target salary, and a two-step transformation to standardize the salary data.

Outlier Removal and Data Cleaning: The raw dataset's $salary\_from$ (salary lower bound) and $salary\_to$ (salary upper bound) fields were both first converted to numeric types, with any unconvertible values being set to missing (NaN). Subsequently, a series of filtering criteria is applied to identify and remove anomalous or invalid salary records. The specific filtering logic is as follows:

$$\begin{cases} 10,000 < S_f \leq 1,000,000 \\ 10,000 < S_t \leq 1,000,000 \\ S_f \leq S_t \end{cases}$$

- S_f represents salary lower bound.
- S_t represents salary upper bound.

Average Expected Salary: The original data does not include the salary value (average) of each data item. To convert the salary range [$salary\_from$, $salary\_to$] into a single, representative numerical value, our study computed the average of the salary range for each job posting:

$$S = \frac{S_f + S_t}{2}$$

- S represents average expected salary.

Salary Data Log-transformation and Standardization: To create a more normally distributed target variable, which is better suited for model training, and to prevent the large scale of salary values from adversely affecting the model training process, e.g., leading to large gradients, slow convergence, our study applies a two-step

transformation process: a logarithmic transform followed by Z-score standardization. The entire transformation for the i-th data point is defined by:

$$S'_i = \frac{\log(S_i) - \mu_{\log}}{\sigma_{\log}} \tag{15}$$

where:

- S_i: represents the raw average salary value for the i-th data point.
- $\log(S_i)$: is the log-transformed salary.
- $\mu_{\log}$: is the mean of all log-transformed salary data, defined as $\mu_{\log} = \frac{1}{N}\sum_{i=1}^{N}\log(S_i)$.
- $\sigma_{\log}$: is the standard deviation of all log-transformed salary data.
- S'_i: represents the final log-transformed and standardized salary value used for training the model.

4.2 Graph Construction for Each Domain

This section describes how our study builds the graphs upon which our model will be based. We decided to take a graph-based approach because of one key limitation in traditional models: processing predictive attributes as a flat, independent feature vector. This does not allow for capturing the rich, contextual interplay of factors that determine a salary. For example, the market value of a 'Data Scientist' position is very different across industries such as 'FinTech' and the 'Public Sector' and then again in different geographical locations like 'Amsterdam' or smaller towns.

To explicitly model these critical interactions, our study represents each attribute domain as a distinct graph: $m_{\texttt{loc}} \to G_{\texttt{loc}}$, $m_{\texttt{occ}} \to G_{\texttt{occ}}$, and $m_{\texttt{ind}} \to G_{\texttt{ind}}$. This process, illustrated in fig. 1, involves defining a comprehensive set of nodes and establishing a rich edge structure composed of two distinct types of relationships: hierarchical and similarity-based.

Node definition: For each attribute domain m our study previously constructed a distinct graph, denoted $G_m = (V_m, E_m)$. The graph construction process begins by initializing an empty graph object. We then iterate through each row of the source dataset. For a given domain m, unique attribute values across all of its defined hierarchical levels are identified. Each unique value is added as a node $v \in V_m$ to the graph if it does not already exist, and its level is stored as a node attribute. This set of nodes (V_m) comprises all unique categorical values present within a given domain. For example, in our location-based graph, the vertices represent both provinces, such as `'Zuid Holland'`, and cities, such as `'Amsterdam'`.

Edge construction: The edge set (E_m) encodes the structured relationships between these vertices, allowing the GAT model to effectively learn from the attribute's inherent structure. The relationships between job attributes are twofold. There are explicit, formal structures, but there are also implicit, functional similarities. A robust model must understand both types of relationships to grasp the nuances of the job market. Therefore, in our study, the nature of these relationships depends on the domain and includes two types:

- **Hierarchical relationships ($E_{\text{hier},m}$):** A hierarchical edge $(i, j) \in E_{\text{hier},m}$ is created between the corresponding nodes i and j if a parent-child relationship is observed within a single data record. This is determined by the co-occurrence of non-null values in adjacent hierarchical fields (e.g., `industry_1_name` and `industry_2_name`, for the Industry domain), the full schema for which is provided in the appendix. For example, in a job posting data row, an edge may connect a general job category to a more specific role, e.g., $\{\texttt{Engineer}, \texttt{Software Engineer}\} \in E_{\text{hier,occ}}$, or a province may connect a city within it, e.g., $\{\texttt{Zuid Holland}, \texttt{Amsterdam}\} \in E_{\text{hier,loc}}$.
- **Similarity relationships ($E_{\text{sim},m}$):** Edges can link nodes that often share a similar context in domain m to capture peer-level relationships. An example in the occupation domain is to connect job titles that require a similar skill set, e.g., $\{\texttt{Data Scientist}, \texttt{Data Analyst}\} \in E_{\text{sim,occ}}$. The construction of the $E_{\text{sim},m}$ is achieved through the following computational steps: (1) Peer-Level Grouping: For each hierarchical level within a domain, e.g., all nodes with type=`'city'`, our study groups the nodes together. (2) Semantic Embedding: our study uses the pre-trained Sentence-Transformers model as our embedding function ϕ. This model generates a dense, 384-dimensional vector representation for the name of each node in the

group. The base model of sentence-transformers is introduced in [39]. (3) Similarity Calculation: The pairwise cosine similarity $sim(v_{\phi_i}, v_{\phi_j})$ is then computed for all vector embeddings within the group. (4) Edge Creation: A similarity edge $E_{\text{sim},m}(i,j)$ is added to the graph between any two nodes if their similarity score $sim(v_{\phi_i}, v_{\phi_j})$ exceeds a predefined hyperparameter threshold τ.

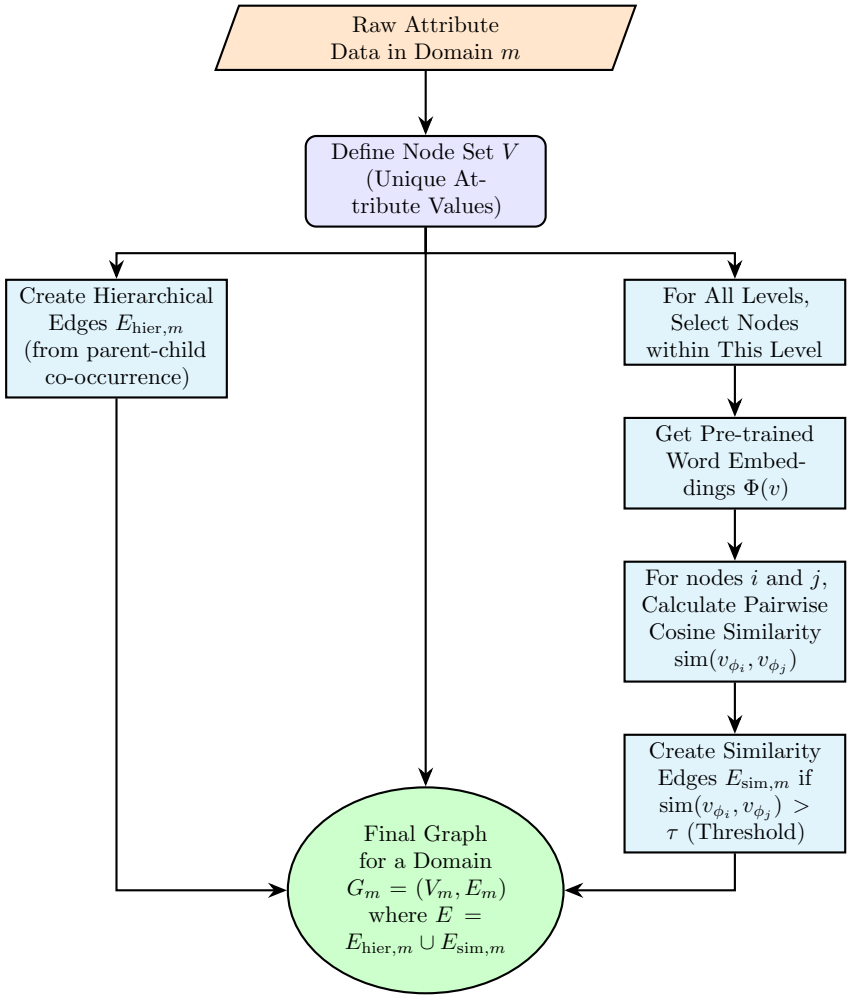


Figure 1: The construction process for a single domain graph G_m.

4.3 Graph Attention Networks

In this section, our study will introduce the process of learning node representations from each attribute graph $G_m = (V_m, E_m)$ using GAT architecture, which involves two main stages: GAT Feature Preparation, followed by using Graph Attention-based Representation Learning. The decision to employ an attention mechanism is crucial given the heterogeneous nature of our graph's edge structure. Our graphs contain both hierarchical and similarity-based connections, and not all neighbors provide equally valuable information for a given prediction.

A standard graph convolution would aggregate information from these diverse neighbors uniformly, failing to distinguish between the influence of a parent node and a semantically similar peer. In contrast, the attention mechanism addresses this by allowing our model to dynamically learn the relative importance of each neighbor, assigning higher weights to more influential nodes and lower weights to less relevant ones. The overall process involves two main stages: GAT feature preparation, followed by Graph Attention-based representation learning. The complete workflow of the GAT module is depicted in fig. 2.

4.3.1 GAT Feature Preparation

With the graph structure defined, our study prepares the initial feature representation for each node v_i. A distinctive feature of our model is the creation of an augmented feature vector, x'_i, which serves as the input to the GAT layers. For each node $v_i \in V_m$, we first assign an initial feature vector $x_i \in \mathbb{R}^{F_{\text{in}}}$, typically a one-hot encoding corresponding to its unique ID, which forms the initial feature matrix X_m. If a given domain graph contains N_m unique nodes, the initial node feature matrix X_m is constructed as an $N_m \times N_m$ identity matrix. Additionally, a dedicated embedding layer learns a unique, dense vector $e_i \in \mathbb{R}^{F_{\text{embed}}}$ for each node, capturing latent characteristics related to the node's identity directly from the salary prediction task. The final augmented feature vector x'_i is then created by concatenating the initial feature with the learned embedding, as expressed by the equation $x'_i = \text{concat}(X_i, e_i)$. This combined representation, with dimension $F_{\text{in}} + F_{\text{embed}}$, provides a rich and expressive input for the subsequent representation learning stage. In this paper, we prepare the augmented feature vector for our three domains.

4.3.2 Graph Attention-based Representation Learning

Graph Attention Networks (GATs) are a class of graph-based neural networks, which were first proposed by Petar et al [38]. Unlike standard GCNs which assign fixed, uniform weights to all neighbors, GAT uses a masked self-attention mechanism. This allows the model to dynamically compute and assign different importance weights to each node in a neighborhood, enabling a more expressive and powerful feature aggregation.

The core mechanism of our GAT is a **masked self-attention** strategy, which adapts the powerful attention concept for graph-structured data. While a general self-attention mechanism could theoretically compute interactions between every pair of nodes in a graph, this would be computationally prohibitive and would ignore the relational structure we have defined. Therefore, GAT constrains this process. For any given node i, the model only computes attention weights, α'_{ij}, for nodes j that belong to its local neighborhood $\mathcal{N}_i$. In our model, this neighborhood $\mathcal{N}$ is defined as the set of first-order neighbors of node i, i.e., all nodes directly connected to i by an edge. Consequently, the edge set E_m that we construct serves as the definitive set of permissible pathways for information flow. The GAT layer does not attend to nodes where a pre-defined edge does not exist.

The operation of a single graph attentional layer can be broken down into the following steps. The input to our layer is a set of node features in domain m, $\mathbf{x}' = \{\vec{x}'_1, \vec{x}'_2, ..., \vec{x}'_{N_m}\}$, $\vec{x}'_i \in \mathbb{R}^{F_{\text{in}}+F_{\text{embed}}}$, where N_m denotes the number of nodes in G_m, and $F_{\text{in}} + F_{\text{embed}}$ denotes the number of the node features. First, a shared, learnable linear transformation, parameterized by a weight matrix $\mathbf{W}$, is applied to every node's input feature vector $\vec{x}'_i \in \mathbb{R}^{F_{\text{in}}+F_{\text{embed}}}$ to enhance its expressive power. Following this, for a given node i, the model computes the raw attention weights α_{ij}, for each of its neighbors j in $\mathcal{N}_i$. To make the attention mechanism aware of the different types of relationships (hierarchical vs. similarity), this weight is calculated by an attention mechanism a that considers not only the features of the two nodes but also the features of the edge $\vec{x'_{ij}}$ connecting them:

$$\alpha_{ij} = a(\mathbf{W}\vec{x'}_i, \mathbf{W}\vec{x'}_j, \mathbf{W_E}\vec{x'_{ij}}) \tag{16}$$

where $\vec{x'}_{ij}$ is a feature vector for the edge and $\mathbf{W_E}$ is a dedicated learnable transformation for edge features. The attention mechanism a can be implemented as:

$$\text{LeakyReLU}(\vec{\mathbf{a}}^T[\mathbf{W}\vec{x'}_i|\mathbf{W}\vec{x'}_j|\mathbf{W_E}\vec{x'}_{ij}]) \tag{17}$$

where $\vec{\mathbf{a}}$ is a learnable weight vector and $\|$ represents the concatenation operation.

These raw scores are then normalized across all of node i's neighbors using the softmax function to produce the final attention weights, α'_{ij}:

$$\alpha'_{ij} = \text{softmax}_j(\alpha_{ij}) = \frac{\exp(\text{LeakyReLU}(\vec{\mathbf{a}}^T[\mathbf{W}\vec{x'}_i|\mathbf{W}\vec{x'}_j|\mathbf{W_E}\vec{e}_{ij}]))}{\sum_{k\in\mathcal{N}_i}\exp(\text{LeakyReLU}(\vec{\mathbf{a}}^T[\mathbf{W}\vec{x'}_i|\mathbf{W}\vec{x'}_k|\mathbf{W_E}\vec{e}_{ik}]))} \tag{18}$$

The new feature vector for node i, $\vec{h}'_i$, is computed as an attention-weighted sum of its neighbors' transformed

features, followed by a non-linear activation σ:

$$\vec{h}_i^{'} = \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha'_{ij}\mathbf{W}\vec{h}_j\right) \tag{19}$$

To stabilize the learning process and capture diverse relational patterns, each GAT layer employs a multi-head attention mechanism. This involves executing K_a independent attention mechanisms in parallel. For intermediate layers, the feature vectors resulting from these K_a heads are concatenated to form the final node representation. This preserves the information learned from each attention head independently:

$$\vec{H}_i^{'} = \Big\|_{k=1}^{K_a} \vec{h}_i^{'k} = \Big\|_{k=1}^{K_a} \sigma\left(\sum_{j\in\mathcal{N}_i} \alpha'^{k}_{ij}\mathbf{W}^k\vec{h}_j\right) \tag{20}$$

$\vec{H}_i' \in \mathbb{R}^{K\times F_{\text{hidden}}}$ represents the concatenated output for node i from the K attention heads, where F_{hidden} is the feature dimension of each head. $||$ represents concatenation and α'^{k}_{ij} are the attention weights computed by the k-th attention head.

Finally, to ensure the output conforms to the precise dimensionality required for subsequent stages of our model, for the final layer of the model, the outputs from the attention heads are passed through a linear layer. This provides a final, aggregated node representation, forming the rows of the ultimate node representation matrix $H_i \in \mathbb{R}^{F_{\text{out}}}$ for node i in graph G_m:

$$\vec{H}_i = \mathbf{P}\vec{H}_i^{'} \tag{21}$$

where $\mathbf{P} \in \mathbb{R}^{F_{\text{out}}\times(K\cdot F_{\text{hidden}})}$ is the learnable weight matrix of the final linear layer, which projects the concatenated vector to the final output dimension, F_{out}.

For a given attribute domain m, the set of representations for all nodes is formed by stacking the individual node representation vectors $\vec{H}_i$ for all nodes $v_i \in V_m$. This forms the **node representation matrix**, H_m. If the domain m contains a total of $N_m = |V_m|$ nodes, and the dimension of each representation vector is F_{out}, then the matrix H_m has the dimensions $N_m \times F_{\text{out}}$. This matrix is formally defined as:

$$H_m = \begin{pmatrix} \vec{H}_1 \\ \vec{H}_2 \\ \vdots \\ \vec{H}_{N_m} \end{pmatrix} \in \mathbb{R}^{N_m\times F_{\text{out}}} \tag{22}$$

4.4 Priority-based Hierarchical Selection

A key component of our model's forward flow is the priority-based hierarchical selection, a mechanism designed to robustly handle input samples with varying levels of attribute specificity. This logic ensures that the model always leverages the finest-grained (most specific) information available for each attribute category (region, occupation, and industry) when building the final integrated feature vector. The process for each attribute in h_comb first initializes the feature representation of each sample in the batch with the corresponding learnable "unknown" embedding. The model then systematically checks the attributes for the presence of lower-level data, allowing embeddings from more specific levels to override embeddings from more general levels.

For example, when selecting occupation features, the model first populates the representation with Level 1 (major-level category) embeddings, if available. It then checks Level 2 (sub-category) embeddings, and if present for a given sample, it overrides the Level 1 embedding for that sample. Next, it checks the most granular Level 3 (minor-level category) embedding, and if present, it replaces any Level 1 or Level 2 information. and finally, resorts to a general "Unknown" representation. This strategy applies to all three attribute domains, enabling the model to gracefully handle incomplete data while maximizing the use of the most precise descriptive features available for each prediction.

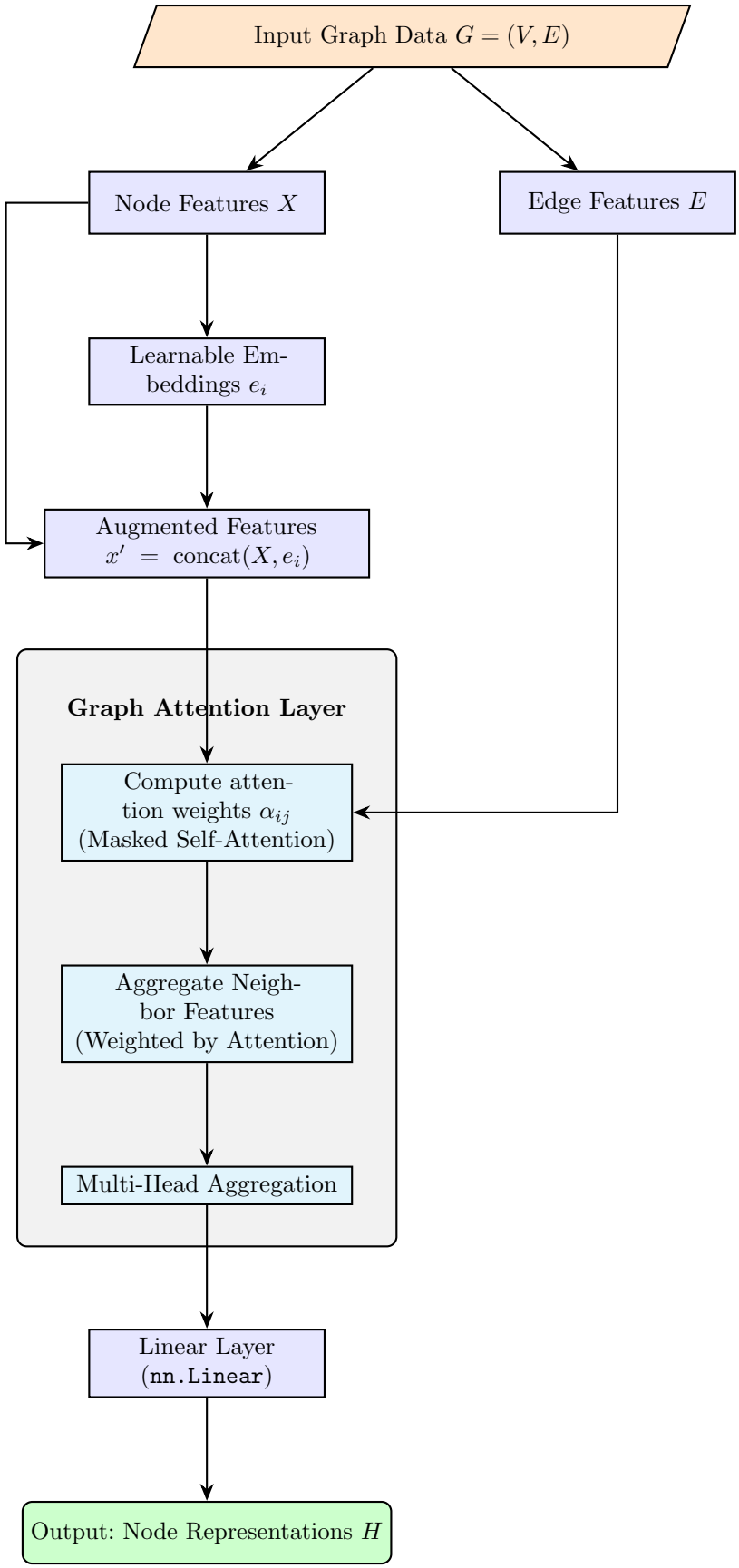


Figure 2: Detailed architecture of a GAT module.

4.5 Mixture Density Network

To move beyond single point estimates and capture the inherent uncertainty and potential multimodality in salary prediction, we extend the core Graph Attention Network (GAT) architecture, described in fig. 2, by incorporating a Mixture Density Network head that outputs the parameters of a Gaussian Mixture Model (GMM). The complete workflow of the model is depicted in fig. 3.

For each attribute domain, the corresponding learned representation vector h_m is selected from the GAT output matrix H_m. These individual vectors, each encapsulating the relevant information learned from its graph's context and node characteristics, are then concatenated to form the final integrated feature representation, h_{comb}. Subsequently, the head of the MDN receives h_{comb} as input and is responsible for mapping it into the parameters of the GMM. This process involves two stages: first, h_{comb} is passed through a hidden layer, which is a standard Multi-Layer Perceptron (MLP) with a ReLU activation function. This hidden layer transforms the integrated features into a hidden vector with a dimension of F_{hidden}. Second, this hidden representation is then fed into a final linear output layer that computes the parameters defining the GMM. Assuming a mixture of K Gaussian components, the MDN head outputs $3K$ values for each input h. These values parameterize: 1. The means (μ_k) of the K Gaussian components. 2. The standard deviations (σ_k) of the K Gaussian components. 3. The mixture weights (π_k) for combining the K components.

To ensure the parameters are valid, specific activation functions are applied to the raw outputs (a) of the final linear layer:

- The outputs for standard deviations (σ_k) are passed through an exponential function ($\sigma_k = \exp(a_k^\sigma)$) to guarantee positivity.
- The outputs for mixture weights (π_k) are passed through a Softmax function ($\pi_k = \frac{\exp(a_k^\pi)}{\sum_{j=1}^{K} \exp(a_j^\pi)}$) to ensure they are positive and sum to unity ($\sum_{k=1}^{K} \pi_k = 1$).
- The outputs for the means (μ_k) typically use a linear activation, i.e., $\mu_k = a_k^\mu$, allowing them to take any real value.

These parameters ($\boldsymbol{\mu}(h), \boldsymbol{\sigma}(h), \boldsymbol{\pi}(h)$) together define the conditional probability density function $p(y|h)$ for the target salary y given the representation h:

$$p(y|h) = \sum_{k=1}^{K} \pi_k(h) \mathcal{N}(y|\mu_k(h), \sigma_k(h)^2) \tag{23}$$

where $\mathcal{N}(y|\mu, \sigma^2)$ denotes the Gaussian probability density function with mean μ and variance σ^2.

4.6 Loss Function

We employ a hybrid loss function that combines two distinct objectives: the primary Negative Log-Likelihood (NLL) loss to accurately model the distribution's shape, and a supplementary Mean Squared Error (MSE) loss to explicitly anchor the distribution's expected value.

The primary objective is to minimize the NLL of the true target salaries y_{true} under the predicted conditional mixture distribution:

$$\mathcal{L}_{\text{NLL}} = -\log p(y_{\text{true}}|h) = -\log \left(\sum_{k=1}^{K} \pi_k(h) \mathcal{N}(y_{\text{true}}|\mu_k(h), \sigma_k(h)^2) \right) \tag{24}$$

This loss function encourages the model to assign high probability density to the true target values.

To further guide the training process and ensure that the mean of the predicted distribution is a precise point estimate, we introduces a secondary MSE loss. This loss measures the squared difference between the true salary and the expected value of the predicted GMM, $E[y|h]$, which is calculated as:

$$E[y|h] = \sum_{k=1}^{K} \pi_k(h) \cdot \mu_k(h) \tag{25}$$

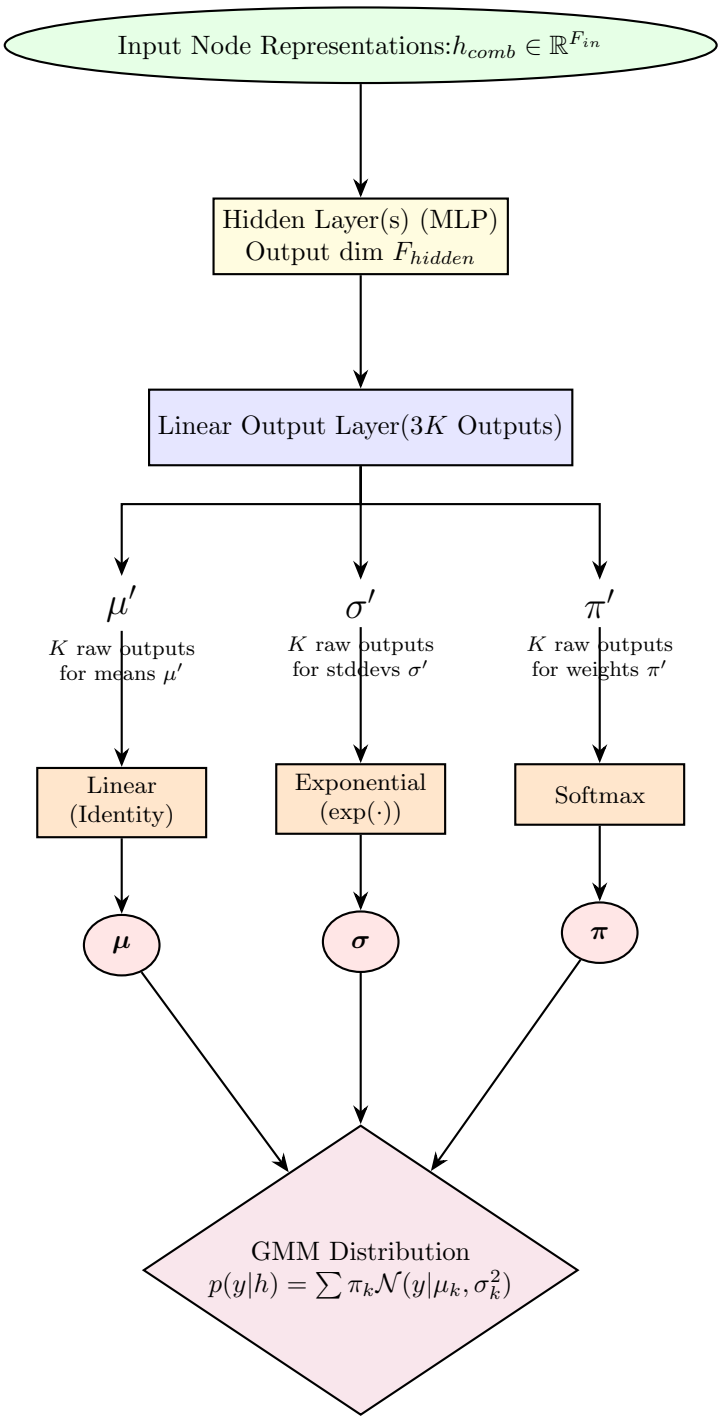


Figure 3: Simplified architecture of the GMM with K Gaussian Components.

The MSE loss is then defined as:

$$\mathcal{L}_{\text{MSE}} = (y_{\text{true}} - E[y|h])^2 \tag{26}$$

The entire GAT-MDN model is trained end-to-end by minimizing the combined loss of the true target salaries y_{true} under the predicted conditional mixture distribution:

$$\mathcal{L}_{\text{Combined}} = \mathcal{L}_{\text{NLL}} + \alpha \cdot \mathcal{L}_{\text{MSE}} \tag{27}$$

where α is the MSE weight that manages the contribution of the MSE loss components. This hybrid approach trains the model to learn not only the shape and uncertainty of the salary distribution but also to produce an accurate expected value.

4.7 Framework Outline

This section outlines the overall architecture of our proposed model for salary distribution prediction, visually summarized in fig. 4. The framework adopts a multi-graph approach to capture diverse relational information from various attribute domains ($m_1, m_2, \ldots, m_M$) known to influence salary levels. The core pipeline proceeds as follows: First, for each relevant attribute domain $m_i, i \in (1, 2, ..M)$, a specific graph (G_i) is constructed, defining the nodes (representing attribute values, denoted X_i in the diagram's node definition) and the edges (E_i) that capture relationships within that domain. Following graph construction, learnable embeddings (e_i) are associated with each node, initialized via an embedding layer. Subsequently, each domain's graph structure (E_i), the embedding lookup e_i, and associated initial node features (X_i) are processed by a dedicated Graph Attention Network module (GAT Module i). As detailed in Section 4.3, each GAT module uses graph convolutions, feature concatenation (combining X_m and e_m), and activation functions to generate contextualized node representations (H_m). To generate a prediction for a specific instance characterized by a set of attribute values across the domains, the corresponding learned node representations (h_m) are retrieved from the outputs (H_m) of each relevant GAT module. These domain-specific representations are then integrated via concatenation into a unified feature vector (h_{comb}) in the Feature Integration step. Finally, this integrated representation h_{comb} is fed into a Mixture Density Network (MDN) head. As detailed in Section 4.4, the MDN head receives and maps the integrated features to the parameters ($\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\pi}$) of a Gaussian Mixture Model. These parameters collectively define the model's final output: the conditional probability distribution (and point estimate) of salaries ($p(y|h_{\text{comb}})$) for the given input attribute combinations.

4.8 Model Generalization

A crucial capability of the proposed multi-graph GAT-MDN framework is its inherent ability to generalize and provide predictions for combinations of attributes that were not explicitly present in the training dataset. In real-world scenarios, the number of possible attribute combinations, e.g., specific job role within a specific industry in a specific city can be vast, making it infeasible to observe examples for every single combination during training. Methods relying solely on memorization or direct lookups would fail in such cases. Our framework addresses this challenge through a combination of component-based representation learning and functional mapping. The framework's generalization power stems from its two-stage process:

Component Representation Learning: Instead of learning a representation for each complete attribute combination, the model first learns representations for individual attribute values within their respective domains using dedicated GAT modules. For instance, the location GAT learns vector representations (h_{loc}) for cities and regions, capturing similarities based on geographical relationships and potentially shared characteristics learned from salary patterns associated with them during training (even when they appear in different combinations). Similarly, the occupation GAT learns representations (h_{occ}) for job roles, and the industry GAT learns representations (h_{ind}) for industries. Crucially, a representation is learned for every attribute value, e.g., every city, every job title present somewhere in the training data.

Composition and Functional Mapping: When predicting for a potentially unseen combination, A_{new},the model first retrieves the learned representations for each constituent attribute value. These component representations are then concatenated, to form a composite feature vector h_{comb}. The MDN head, d_{θ_d}, subsequently acts as a learned function that maps this (potentially novel) h_{comb} vector to the parameters of the GMM describing the

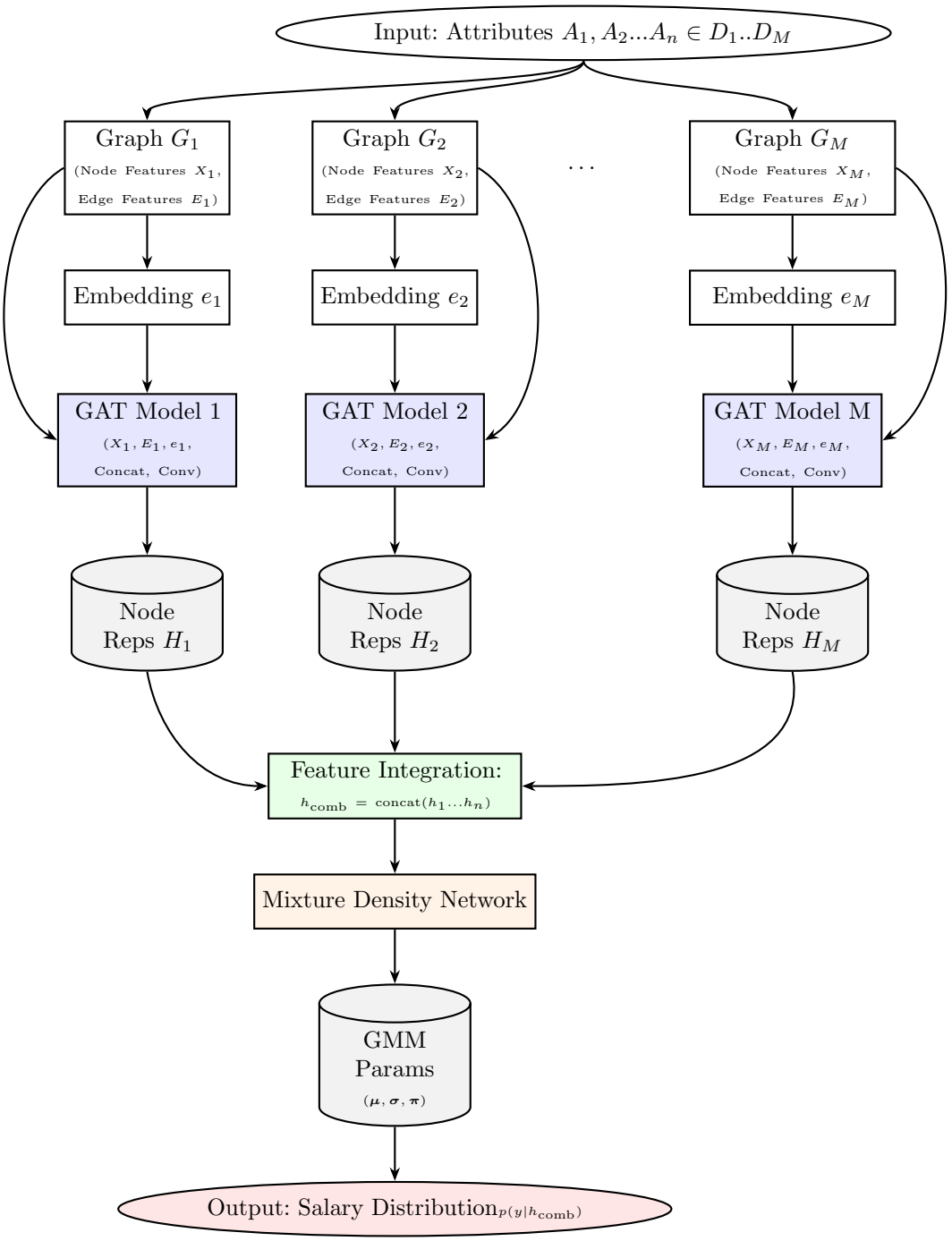


Figure 4: The whole framework of the salary prediction model. In our study, we set M=3.

salary distribution. Because the MDN head learns a continuous mapping function over the space defined by h_{comb} based on diverse examples seen during training, it can effectively interpolate or reasonably extrapolate to generate parameter estimates for new h_{comb} vectors corresponding to unseen attribute combinations.

Illustrative Example

Consider predicting the salary distribution for a `'Data Scientist'` in `'Amsterdam'` within the `'Consulting'` industry. Suppose this specific combination did not appear in the training dataset. However, assume the training data did include instances such as:

– (`'Amsterdam'`, `'Data Scientist'`, `'ICT'`)

– (`'Utrecht'`, `'Data Scientist'`, `'Consulting'`)

– (`'Amsterdam'`, `'Software Engineer'`, `'Consulting'`)

The framework would proceed by first retrieving the learned representations for each attribute from their respective GATs: $h_{\text{Amsterdam}}$ (having learned about Amsterdam from other job/industries), h_{DS} (having learned about this role from other locations/industries), and $h_{\text{Consulting}}$ (having learned about this industry from other locations/occupations). These vectors are then concatenated to form a composite vector: $h_{\text{comb}} = \text{concat}(h_{\text{Amsterdam}}, h_{\text{DS}}, h_{\text{Consulting}})$. Although this h_{comb} vector may be unique, its components capture characteristics from related instances, placing it within the learned feature space. Finally, the trained MDN head takes this vector and is still able to output the GMM parameters (μ, σ, π) based on the continuous function it learned from all training data.

4.9 Ablation Study

Our study designed and implemented an ablation study baseline model, called **"Embedding-only MLP-MDN"**. This baseline model aims to answer a core question: "How will the model perform if it only learns independent embedding representations of each attribute value without leveraging their connections in the graph?"

In contrast to our full GAT-MDN model, this baseline model completely removes the graph-convolutional modules. For a given prediction instance $A = \{a_1, \ldots, a_M\}$, instead of computing the context-aware node representation matrix H_m through message passing, we directly retrieve the 'raw' embedding vector e_m for each attribute value a_m from its corresponding learnable embedding matrix E'_m. These embedding vectors, retrieved independently from each attribute domain, are then concatenated to form a composite vector:

$$h_{\text{comb\_mlp}} = \text{concat}(e_1, e_2, \ldots, e_M) \tag{28}$$

This concatenated vector is subsequently fed into a standard Multi-Layer Perceptron (MLP). The role of the MLP is to learn the interactions between these independent features within a non-structured vector space. Finally, the output of the MLP is passed to the same MDN head, d_{θ_d}, to predict the salary's probability distribution. By directly comparing the performance of our GAT-MDN model against this Embedding-only MLP-MDN baseline, we can isolate and quantify the performance gain attributable to graph convolution.

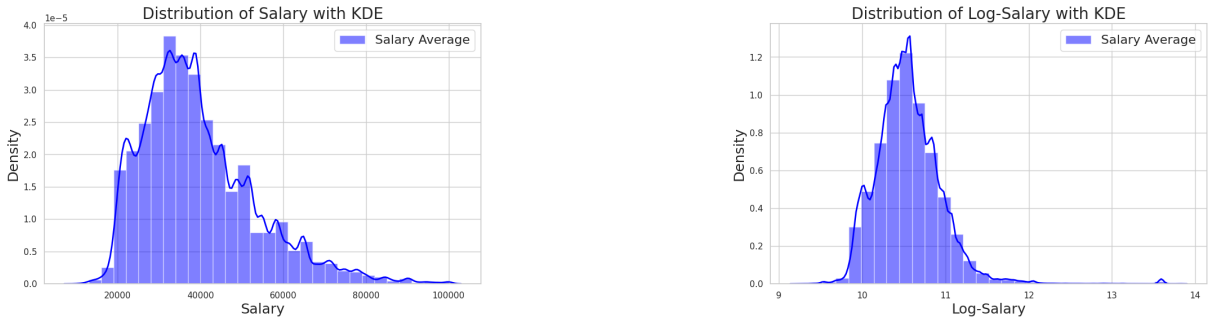
5 Descriptive Data Analysis

In this section, our study conducted a series of data introductions and exploratory data analyses to gain a deeper understanding of the inherent characteristics and distribution of our dataset. The data is called the NL salaries source dataset, which consists of real-world job postings from the Netherlands, provided by the labor market data platform Jobdigger [20]. We first provide a high-level overview of the salary data, including its origin, basic statistics, and overall distribution. We then provide a more detailed analysis of the three core attribute domains: Location, Occupation, and Industry. In detail, within each domain, the internal hierarchical structure is also explored—for instance, the level of province versus city within the Location domain. Furthermore, counts of unique attribute values at each level within the hierarchy are reported, sizes of resultant graphs in terms of nodes and edges, and top followed by bottom average salaries across respective criterion values for each domain

hierarchy are discussed and depicted in graphs, thus presenting a very initial view into the drivers of market value.

The structured dataset is derived from real-world job postings. Each job posting provides detailed salary information, including the upper and lower salary bands, payment category, and currency. Additionally, the postings contain descriptive contextual data, such as job and contract information, e.g., city, province, occupation, language, alongside specific skill requirements, such as the minimum education level and a list of explicitly required skills. We use salary data where the values are not empty, from January 1, 2022, to January 1, 2025. This selection gives us a total of 1040918 unique records, with each record having a unique salary value and corresponding attribute indices and values.

The average salary for the whole dataset is €42604.05 per year. fig. 5a shows the distribution of the salary. The salary distribution is right-skewed, indicating that most of the salary data is concentrated in the lower range, while there are a few high-paid jobs. In addition, the distribution is multimodal, which may mean that there is more than one core salary group in the data. After logarithmic transformation, the distribution is close to a normal distribution, shown in fig. 5b.



(a) Distribution of Salary with KDE.

(b) Log Distribution of Salary with KDE.

Figure 5: Comparison of salary distributions before and after log transformation.

As mentioned in Section 3, our model performs the prediction task using a set of three engineered job-related features, which include Location, Occupation, and Industry. To capture the varying levels of granularity within these domains, we define each with a specific hierarchical structure. The Location domain m_L pertains to the geographical area of a job posting, modeled with two levels: a broader provincial scope, e.g., `'Zuid Holland'` and a more specific city scope, e.g., `'Rotterdam'`. The Occupation domain m_O represents the job title through a three-tiered hierarchy, ranging from a major-level functional category, e.g., `'Transportation and Logistics'`, to a sub-category, e.g., `'Transportation'`), down to a granular role, e.g., `'Bus Driver'`. Finally, the Industry domain m_I specifies the economic sector to which the job belongs, structured with two levels following the international standard for the classification of economic activities (NACE Rev. 2) [11]: a major industry group, e.g., `'Administrative And Support Service Activities'` and a specific sub-sector, e.g., `'Payrolling'`. The statistics of each domain graph are shown in the Table 1.

Table 1: Statistics of the three constructed graphs

Graph Name	Number of Nodes	Number of Edges
Region Graph	478	10177
Occupation Graph	3,128	605767
Industry Graph	98	1159

Locations: We consider 12 Dutch provinces and 408 cities in the dataset. The provinces with the highest averaged salaries are "Zuid-Holland" "Utrecht" and "Noord-Holland", which are all well above €45000. The lowest salaries are in "Drenthe" and "Limburg", which are both below €37000. Approximately 3.9% of cities (16 in total) feature an average salary below €30,000, with the lowest recorded on the northern island of Terschelling at €21,672. In contrast, 4.9% of cities (20 in total) have an average salary exceeding €50,000.

The highest average salary was observed in the southern city of Haaren at a significant €139,828.

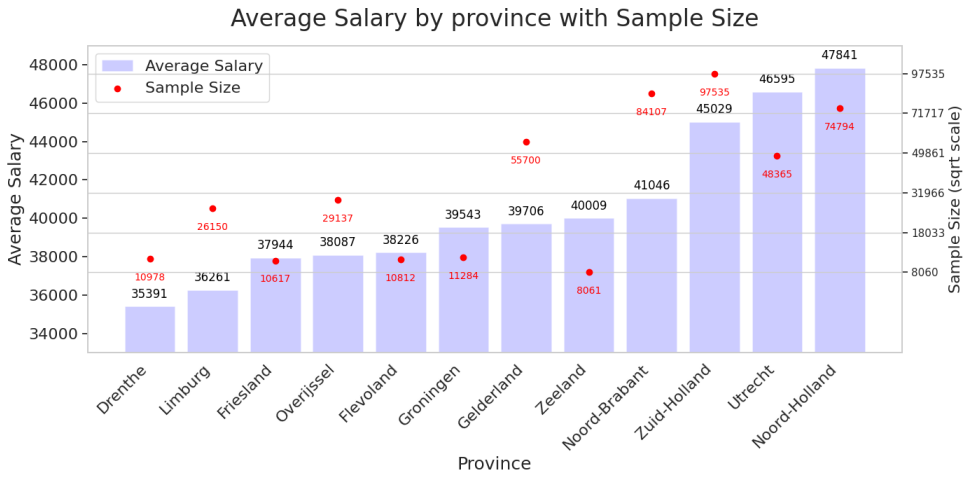


Figure 6: Average Salary by Province with Sample Size.

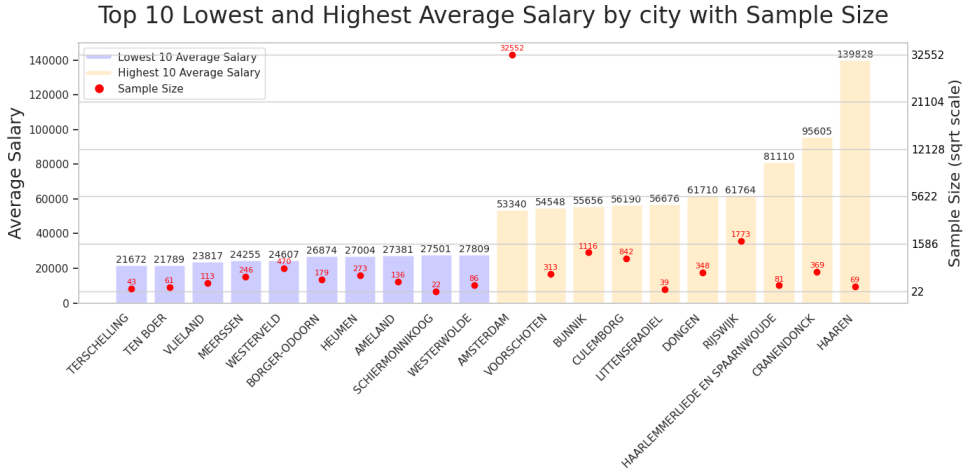


Figure 7: Top 10 Lowest and Highest Average Salary by City with Sample Size.

Industries: The dataset contains a total of 20 major and 83 minor industry categories, and among the major categories, the top three highest-paying industries are Government and Military (€51501), Environmental Services (€50844) and Information and Communication (€48620). The least three paying industries are Home-based employment (€28287), Hospitality (€31336) and Agriculture (€33728).

Occupations: The dataset contains a total of 17 major, 42 mid-level, and 3,389 minor occupation categories, and among the major categories, the top three highest-paying occupations are Finance (€63,105), ICT (€52,819), and Social Care (€52,138). The least three paying occupations are Cleaning (€24461), Agriculture and Gardening (€27271), and Transportation and Logistics (€31245).

Our exploratory analysis of the dataset evidences large and predictable variance in average salaries when disaggregated along the three main attribute domains, as would be expected from the very complicated Dutch labor market. The disparities within each domain are substantial. For instance, the average salary in the highest-paying province is 37% greater than in the lowest, and this gap widens dramatically at the city level, where the highest-paying city commands a salary more than six times that of the lowest. Clearly, geographical context emerges as a crucial variable with respect to empirical expectations with regard to higher remuneration levels in large metropolitan areas and economically strong regions.

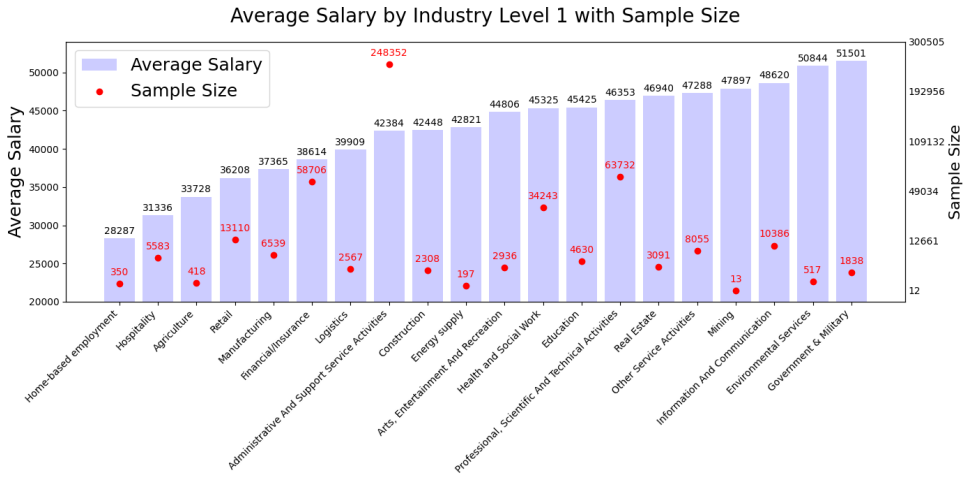


Figure 8: Average Salary by Industry Level 1 (major category) with Sample Size.

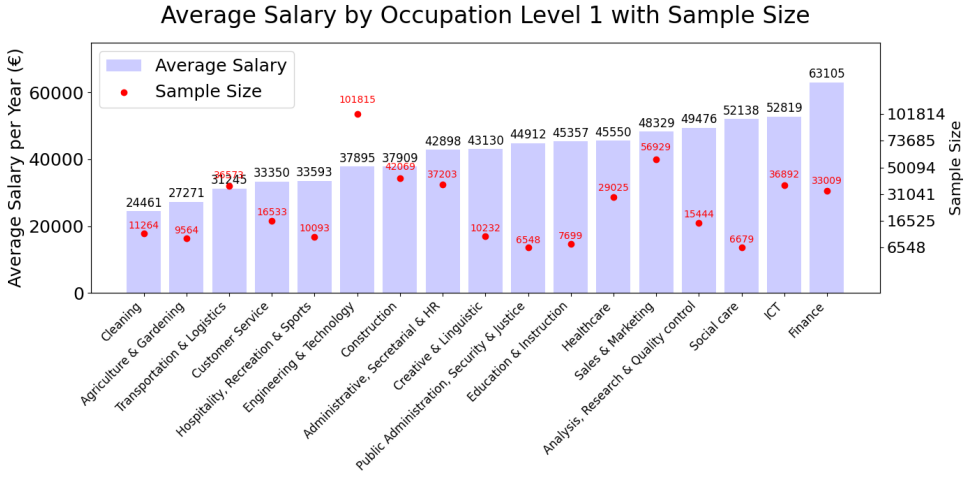


Figure 9: Average Salary by Occupation Level 1 (major category) with Sample Size.

Similar trends are observed in the industry and occupation domains. The highest-paying industry offers an average salary 82% higher than the lowest, with roles in sectors like 'Government Military' being far more lucrative than home-based work. The gap between professional fields is equally pronounced; occupations traditionally considered high-paying, such as Finance and Information Technology, show average salaries more than double those in fields like Cleaning or Gardening. These strong disparities underscore that industry and occupation are primary drivers of market value, and any predictive model must be capable of capturing these powerful signals.

Finally, an analysis of the target variable itself reveals crucial characteristics. The log-transformed salary distribution, represented in fig. 5b, indicates that the distribution was not entirely normal and there was a prominent spike at about 13.6, reflecting a small cluster of extremely high salaries. This pointed to a possible case of extreme outliers or a distinct subpopulation of high earners, underscoring the need for robust pre-processing and a modeling approach such as our Mixture Density Network that can directly accommodate non-standard, multi-modal distributions.

6 Experiment Results

This section presents the comprehensive experimental evaluation of our proposed GAT-GMM framework, which is about establishing its performance and robustness for practical purposes. Our study begins by by a detailed overview of the experimental environment, including the hardware and software configurations. To ensure our work is reproducible, we first present all fixed parameters, which provide the architecture underpinning our core model. We then discuss hyperparameter optimization, in which we follow a systematic approach to optimizing the most sensitive parameters—learning rate and number of Gaussian components. To rigorously assess the contribution of the graph-based components, our study introduces an ablation study centered on an MLP-MDN baseline and details its experimental setup. The model results section provides a quantitative comparison of the learning dynamics and ultimate performances of our GAT-GMM model against the baseline, based on NLL and MSE loss metrics. Finally, we conduct several case studies to demonstrate the real applicability of the model and qualitatively assess its predictive power under conditions of single-attribute queries, combinations of multi-attributes, and instances of missing data.

6.1 Experimental Environment

The experiments were conducted on a single server running Windows Server 2022. This server was equipped with 1 x NVIDIA RTX 4000 Ada GPU, 16 vCPUs, and 62 GB of RAM. The primary software environment was Python 3.10.

Implementation Details: The entire deep learning architecture was built using the `PyTorch` framework, with a specific reliance on the `PyTorch Geometric (PyG)` library for all layers and operations of the graph-based neural network. The initial construction of graph structures was managed using the `NetworkX` [14] libraries. For creating similarity-based edges, semantic embeddings were generated using the `sentence-transformers` library. Standard machine learning tasks, such as data standardization and splitting, were handled by `scikit-learn`. The hyperparameter optimization process was systematically conducted using `Optuna`, and all visualizations were created with `Matplotlib`.

6.2 Fixed Parameters Settings

To ensure a fair and reproducible hyperparameter search and reduce the model's computational cost, architectural parameters of the GAT-MDN model below were held constant across all trials. These fixed parameters, which define the core capacity and structure of the network, are summarized in Table 2.

6.3 Hyperparameter Optimization

To systematically determine the optimal hyperparameters for our GAT-MDN model, we want to set an objective function, which receives a set of experimental hyperparameters and returns the performance of the model evaluated on the validation set. Therefore, our study employed Optuna [2], a specialized framework for automated hyperparameter optimization. The primary objective was to find the hyperparameter combination that minimizes the model's combined loss on a held-out validation set. In our model, the number of gaussian

Table 2: Model Architectural and Fixed Hyperparameters

Parameter	Value	Description
`embedding_dim`	32	The dimension of the learnable embedding vector (e_i) for each node in GAT.
`GAT_hidden_channels`	64	The number of output feature channels from the first GAT convolution layer.
`GAT_out_channels`	32	The output dimension of the final node representation (H_m) from each GAT module.
`head`	3	The number of independent attention mechanisms in multi-head attention.
`mdn_hidden_dim`	128	The dimension of the hidden layer within the MDN head.
`GAT Layers`	2	The number of sequential convolution layers in each GAT module.
`Batch Size`	512	The number of samples processed in each training iteration (batch).
`MSE Weight`	0.5	The contribution of the MSE loss components.
`Optimizer`	Adam	The optimizer used during training.

components `num_gaussians` and learning rate (`lr`) of our Adam optimizer are the two hyperparameters that have the greatest impact on the training results. Therefore, our study only performs optimal search for these two hyperparameters: the learning rate was selected from the discrete set $5e-4, 1e-4, 5e-5, 1e-5, 5e-6$, and the `num_gaussians` was sampled from the integer range of 1 to 10. Other architectural parameters described in the previous section are kept constant during this process. For each trial, Optuna selected a new set of hyperparameters, and a corresponding model was instantiated and trained for up to 100 epochs. To improve efficiency, an early-stopping mechanism known as pruning was used, allowing the study to terminate unpromising trials that showed poor performance on the validation loss compared to others. A total of 60 trials were conducted to thoroughly explore the search space and identify the configuration that yielded the best model performance. Best parameters found after 50 trials are: `lr` $= 1e-4$, `num_gaussians` $= 9$.

6.4 Ablation Study: MLP Baseline

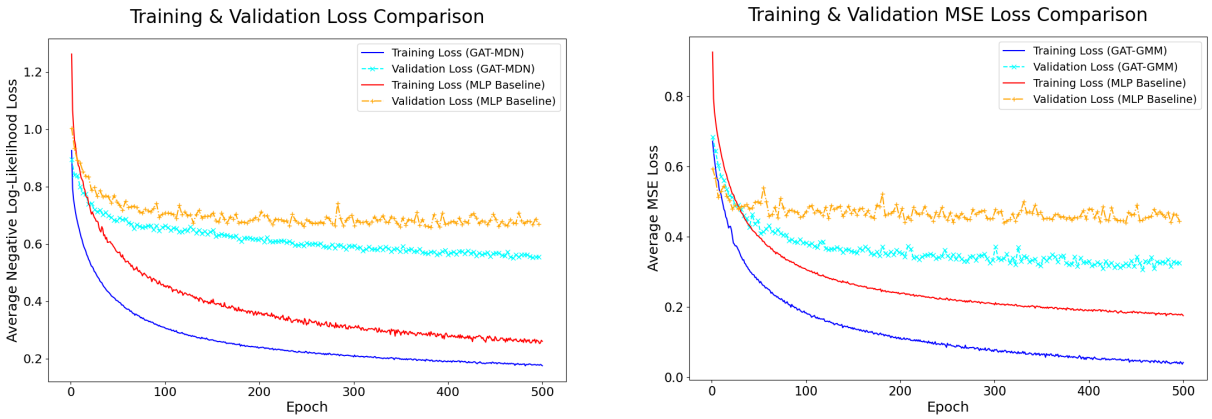
Experimental Setup. To ensure a fair comparison, the Embedding-only MLP-MDN baseline was trained under the same experimental conditions as the primary GAT-MDN model. The training and validation data sets were used identically and the training was performed in the same computational environment. Key architectural hyperparameters were kept consistent where applicable to isolate the impact of the graph convolution. Specifically, the dimension of the learnable node embeddings (`embedding_dim`) was set to 32, and the hidden dimension of the MDN head (`mdn_hidden_dim`) was set to 128. For the new Multi-Layer Perceptron component, its hidden layer dimension (`mlp_hidden_dim`) was also set to 128. For hyperparameter optimization, our study uses the same method as GAT-MDN to explore the hyperparameter space. Best parameters found after 50 trials are: `lr` $= 0.0001$, `num_gaussians` $= 7$.

6.5 Model Results

Our study performed a comparative analysis of the learning process between our proposed GAT-MDN model and the Embedding-only MLP baseline. Both models were trained for 500 epochs on the dataset, which was partitioned into an 80% training set and a 20% validation set. We record the negative log-likelihood loss (NLL) and mean squared error loss (MSE) of the two models respectively. During the training process, the three types of average training loss ($\mathcal{L}_{\text{NLL}}, \mathcal{L}_{\text{MSE}}$ and $\mathcal{L}_{\text{combined}}$) were recorded respectively at the end of each epoch for

both models, and the validation losses were recorded for each of the three epochs. The resulting learning curves illustrating the training and validation losses for both the GAT-MDN and the MLP baseline throughout the training process are presented in fig. 10.

As shown in fig. 10a, from the initial epochs onward, the training NLL losses for the GAT-MDN are substantially lower than those of the MLP-MDN baseline. In terms of validation loss, the gap between the two models is initially small. However, after the 200-epoch mark, the MLP baseline's performance improvement stagnates, whereas the GAT-MDN model's loss continues to register a marginal yet consistent decline. This significant and persistent performance gap strongly validates our central hypothesis: that explicitly modeling the relational and hierarchical structures within the attribute data is crucial for this task. Furthermore, according to fig. 10b, our model demonstrates similar superiority regarding the MSE loss component. By the end of the 500-epoch training period, both the final training and validation MSE losses for the GAT-GMM are markedly lower than those achieved by the baseline model.



(a) NLL Loss Comparison Between GAT-MDN and MLP Baseline.

(b) MSE Loss Comparison Between GAT-MDN and MLP Baseline.

Figure 10: NLL and MSE Loss Comparison Between GAT-MDN and MLP Baseline.

6.6 Case Study

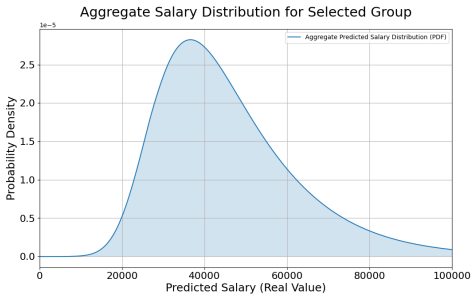
Our study presents a series of case studies designed to evaluate our GAT-MDN's predictive performance under various input scenarios. The first set of cases (Case 1 and Case 2) examines the core capability of the model: generating salary distributions for specific, fully defined instances where only one attribute value is provided per attribute domain. Building on this, Cases 3 and 4 demonstrate the model's advanced capabilities for handling queries involving multiple attribute values, computing overall salary distributions for a wider range of population groups. Finally, Cases 5 and 6 analyze the model's resilience to incomplete information by evaluating its predictions when key attribute domains are intentionally unspecified, testing the effectiveness of our priority-based hierarchical selection. For each case study, we will present the full predicted salary distribution. To detail its composition, we will explicitly report the mixture weights (π) of each Gaussian component.

Case-Insensitive Matching: When a user provides an attribute value for prediction , e.g., 'amsterdam' or 'AMSTERDAM', the case-insensitive function first converts the input to a consistent lowercase format. It then iterates through the keys of the corresponding node mapping dictionary, similarly converting each key to lowercase for comparison. If a match is found, the function retrieves the original, correctly-cased key from the dictionary, e.g., 'Amsterdam' to ensure that the proper node ID is selected from the graph, thereby accommodating variations in user input capitalization without sacrificing lookup accuracy.

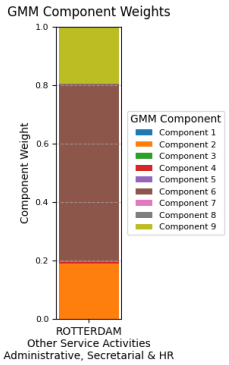
6.6.1 Single Combination

Case 1 (fig. 11): '`region`': 'ROTTERDAM' (city level, all uppercase letters), '`industry`': 'Other Service Activities' (major level), '`occupation`': 'Administrative, Secretarial and HR' (major level). The salary point estimate in this case is €54091. fig. 11a shows the predicted distribution. fig. 11b shows that the salary is

mainly composed of three Gaussian distributions.



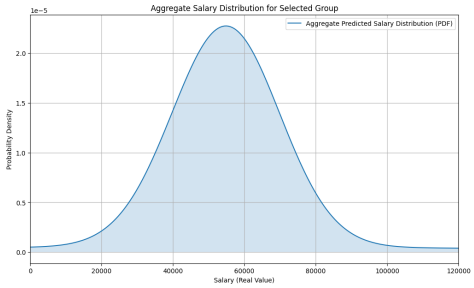
(a) Salary Prediction Distribution for Case 1.



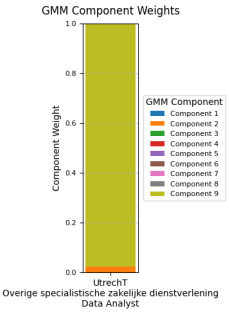
(b) Component Weights.

Figure 11: Salary Prediction Results for Case 1.

Case 2 (fig. 12): '`region`': 'UtrechT' (province level, mixed-case), '`industry`': 'Overige specialistische zakelijke dienstverlening (Other specialist business services)' (minor level), '`occupation`': 'Data Analyst' (minor level). The salary point estimate in this case is €52623. fig. 12a shows the predicted distribution. fig. 12b shows that the salary is mainly made up of a Gaussian distribution.



(a) Salary Prediction Distribution for Case 2.



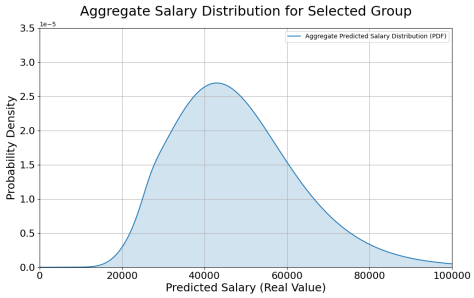
(b) Component Weights.

Figure 12: Salary Prediction Results for Case 2.

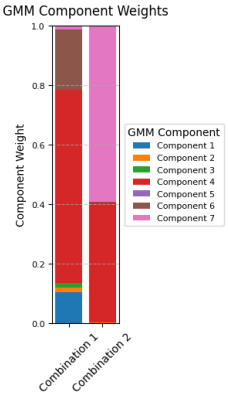
6.6.2 Multiple Combinations

Case 3 (fig. 13): '`region`': 'ROTTERDAM' (city level, all uppercase letters), '`industry`': 'Other Service Activities' (major level), '`occupation`': 'Administrative, Secretarial and HR','Security Inspector' (two values, minor level). The salary point estimate in this case is €46148. fig. 13a shows the predicted distribution for two combinations: ('ROTTERDAM','Other Service Activities' , 'Administrative, Secretarial and HR'), and ('ROTTERDAM','Other Service Activities' , 'Security Inspector'). fig. 13b shows that combination 1 is mainly composed of three Gaussian distributions, while combination 2 is mainly composed of two Gaussian distributions.

Case 4 (fig. 14): '`region`': 'Noord-Brabant' (province level, mixed-case), '`industry`': 'Administrative And Support Service Activities','Professional, Scientific And Technical Activities' (two values, major level), '`occupation`': 'ICT' (minor level). The salary point estimate in this case is €57892. fig. 14a shows the predicted distribution for two combinations: ('Noord-Brabant','Administrative And Support Service Activities' , 'ICT'), and ('Noord-Brabant','Professional, Scientific And Technical Activities' , 'ICT'). fig. 14b shows that combination 1 is mainly composed of three Gaussian distributions, while combination 2 is mainly composed of one Gaussian distribution.

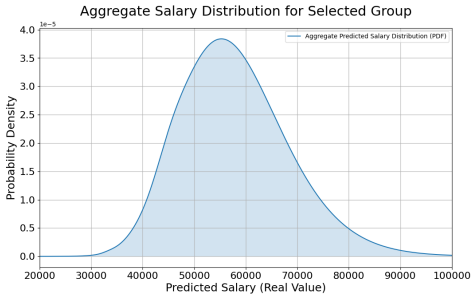


(a) Salary Prediction Distribution for Case 3.

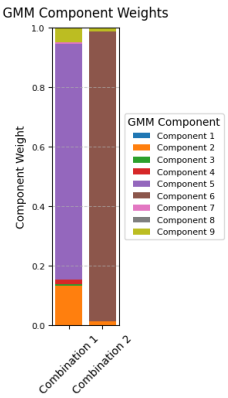


(b) Component Weights.

Figure 13: Salary Prediction Results for Case 3.



(a) Salary Prediction Distribution for Case 4.

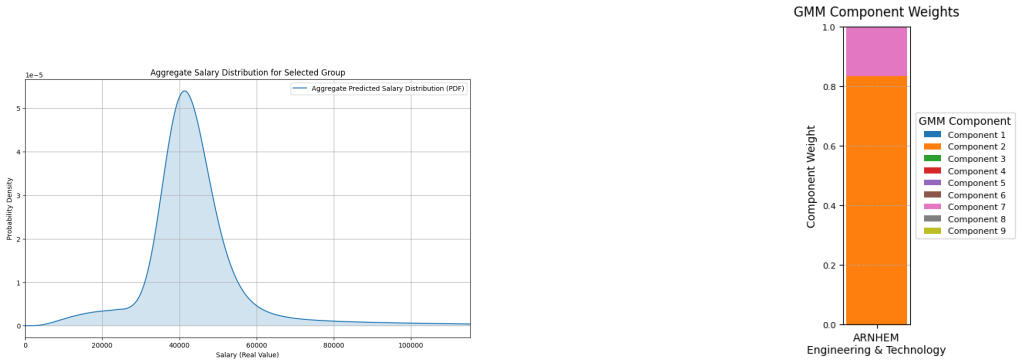


(b) Component Weights.

Figure 14: Salary Prediction Results for Case 4.

6.6.3 Combinations with Empty Values

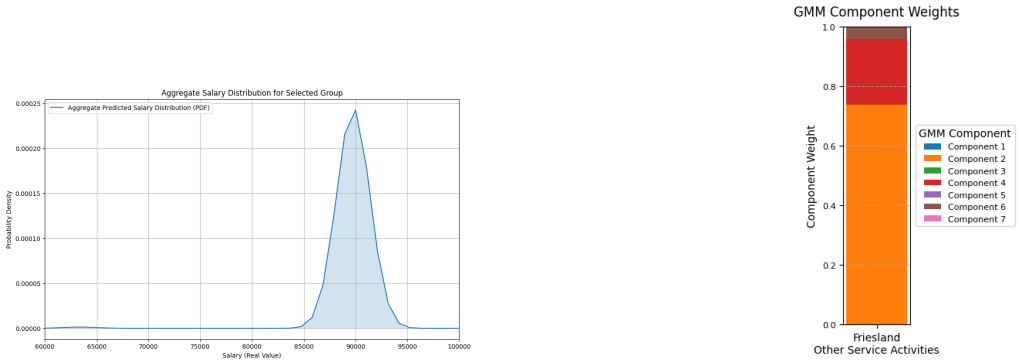
Case 5 (fig. 15): '`region`': 'ARNHEM' (city level, all uppercase letters), '`industry`': empty, '`occupation`': 'Engineering and Technology' (major level). The salary point estimate in this case is €48333. fig. 15a shows the predicted distribution for this case. fig. 15b shows that the salary is mainly made up of two Gaussian distributions.



(a) Salary Prediction Distribution for Case 5. **(b)** Component Weights.

Figure 15: Salary Prediction Results for Case 5.

Case 6 (fig. 16): '`region`': 'Friesland' (Province level, mixed-case), '`industry`': 'Other Service Activities (major level), '`occupation`': empty. The salary point estimate in this case is €115195. fig. 16a shows the predicted distribution for this case. fig. 16b shows that the salary is mainly made up of three Gaussian distributions.



(a) Salary Prediction Distribution for Case 6. **(b)** Component Weights.

Figure 16: Salary Prediction Results for Case 6.

7 Conclusion

In this study, our study proposed and evaluated a new salary prediction framework, the GAT-MDN model, which overcomes the limitations of traditional methods through an explicit representation of the rich relation and hierarchy inherent in job market data. We have conducted a comprehensive experimental campaign to demonstrate the strong advantages that this graph-based, probabilistic approach possesses.

The experimental results testify that our GAT-MDN model greatly outperforms the non-graph MLP-MDN baseline. As can be seen from the learning curves, the GAT-MDN achieves a much lower NLL loss on both training and validation sets. This is a powerful finding, as it indicates that our model is not only more accurate at predicting the expected salary but is also superior at capturing the true underlying shape, spread, and potential multimodality of the salary distribution. The enriched hierarchical and similarity-based edge graph structure gave the model the right kind of context to learn a better and more appropriate representation of salary uncertainty. The attention mechanism proves crucial in navigating this complex information landscape, learning to weigh the influence of hierarchical parents and semantically similar peers to produce a more robust and accurate final prediction.

Furthermore, the superiority of the GAT-MDN is also clearly reflected in the Mean Squared Error (MSE) loss. This model showcases its strength in consistently achieving lower MSE loss due to its nature, which allows for better prediction of a point estimate, or "center of mass," of the distribution of salaries as compared to the baseline. That is, the graph attention mechanism in aggregating contextual information allows the model, from simple attribute-level correlation, to learn a complicated function for the market value of a position. Although there exists a generalization gap between training and validation performances for the GAT-MDN, its validation losses have consistently and significantly been placed under those recorded for the simpler MLP model.

In addition to quantitative metrics, multiple case studies demonstrate the applicability and robustness of our framework. The model generates reasonable point estimates and true distributions for a wide range of use cases, including for single attributes, multiple combinations, and cases with missing information. Overall, the predicted distributions are non-unimodal, demonstrating that the model is able to capture the complex structure of real-world salaries, which can be challenging for simpler models. Notably, while our model is configured with nine Gaussian components, the final predictions are typically dominated by only two or three main components. This demonstrates that the model learns to adapt its complexity to specific queries, using only the necessary components to fit the underlying salary structure, rather than overfitting to all available capacity. This flexibility is further demonstrated by its case-insensitive input and ability to work with different hierarchical attributes, strongly suggesting its scalability and potential for real-world applications. However, as shown in the fitting results for Cases 5 and 6, the model's prediction accuracy for the finest-grained attributes and queries with null attributes in a domain remains limited due to data sparsity.

In conclusion, this work validates our central hypothesis: that using the explicit structure of attribute data through Graph Attention Networks provides a significant performance boost for the task of salary prediction. Combining such a powerful representation learning technique with a Mixture Density Network, the framework proposed here is more accurate, transparent, and insightful in analyzing labor markets and is able to provide not just one value, but a complete view of salary ranges.

8 Future Work

While powerful, our model's generalization capability also has limitations. The model cannot predict for attribute values that were entirely absent during training, e.g., a never-before-seen city or specific skills, due to the lack of learned representations for such attributes. Furthermore, predictions for combinations whose combined vector is very far from the distribution of vectors seen during training (far extrapolation) may be less reliable. Dealing with novel entities requires a different strategy, such as a Graph Representation Learning with GraphSAGE [15]. This architecture addresses the challenge of unseen input by learning universal aggregator functions that sample and combine feature information from a node's local neighborhood. This means that instead of memorizing a fixed embedding for each node, the model learns a process for generating embeddings, which can then be applied on-the-fly to new nodes as they appear in the graph. Alternatively, we can leverage node

features from pre-trained language models (LMs), which creates initial node features directly from the attribute names themselves rather than relying on one-hot encodings, making the model inherently open-world. When a previously unseen attribute appears, its string is simply fed into a pre-trained model like BERT to generate an initial feature vector. This vector is semantically meaningful and located in a similar feature space to related attributes, allowing the GAT to process it effectively. gives a great example of a large-scale GNN that successfully initializes node features from raw text using language models. [9] investigating how large language models (LLMs) can act as powerful feature enhancers to enrich node text attributes before they are processed by a GNN.

Augmenting with Numerical Features: Our current study is limited to categorical attributes. A key direction for future research is to integrate numerical attributes, such as years of experience and candidate age, which are absent from our existing data. Enhancing the model to process these quantitative features alongside the structural graph embeddings would likely yield a significant improvement in predictive performance. A direct approach to incorporating numerical attributes is to integrate them at the node feature level, prior to the graph convolution process. This strategy enables the adaptation of GNNs to rich tabular data, which inherently involves a mixture of categorical and numerical characteristics [23]. Our future work would also include the use of an advanced GNN for heterogeneous data. [36] introduces Collaborative Graph Neural Networks (CONN), an advanced architecture designed to create a richer and more comprehensive node embedded in the learning process.

Real-world salary distributions can have complex shapes that may be difficult for even the GMM to perfectly capture. A more advanced research direction is to design a model capable of learning and adapting to arbitrarily complex distributional forms. One of the most relevant state-of-the-art techniques for this is Normalizing Flows. A Normalizing Flow learns a complex distribution by starting with a simple base distribution and applying a series of invertible, learnable neural network transformations to progressively "warp" it into the shape of the true data. This flexible mechanism makes it possible to replace our current MDN head with a Normalizing Flow-based predictive head, enabling the model to learn more expressive and accurate salary distributions. The principles of normalized flow, various model variants, and their applications in various fields are introduced in detail in [22]. [37] directly explores how to use normalized flows to solve the conditional density estimation problem.

A Major Notations

Table 3: Major notations in this paper

Symbol	Short Description
General and Instance-Level Concepts	
A	Set of attribute values for a single instance.
a_m	A specific attribute value for domain m.
M	The number of distinct attribute domains.
y	The target variable (salary).
S	The raw average expected salary.
S'	The transformed salary value used for training the model.
Graph Structure	
G_m	The graph for attribute domain m, $G_m = (V_m, E_m)$.
V_m	The set of all unique nodes in graph G_m.
E_m	The set of all edges in graph G_m.
$E_{\text{hier},m}$	The set of hierarchical edges in graph G_m.
$E_{\text{sim},m}$	The set of weighted similarity edges in graph G_m.
τ	The similarity threshold.
Model Architecture and Features	
f_θ	The complete parametric model parameterized by θ.
g_{θ_g}	The GAT-based sub-model.
d_{θ_d}	The MDN head sub-model.
X_m	The matrix of initial node features for graph G_m.
e_m	The matrix of learnable node embeddings for graph G_m.
a	The attention mechanism.
α_{ij}	The attention weight.
K_a	The number of independent attention mechanisms in multi-head attention.
H_m	The final node representation matrix from a GAT sub-module.
h_{comb}	The integrated node representation vector.
GMM and Prediction for a Single Instance	
$\hat{p}_\theta(y\|A)$	The model's predicted GMM distribution for instance A.
K	The number of Gaussian components in the GMM.
π_k, μ_k, σ_k	Weight, mean, and standard deviation of the k-th Gaussian.
$E[y\|A]$	The expected salary for instance A.
Querying and Aggregate Prediction	
$\mathcal{Q}_m$	A user-defined set of attribute values for a query in domain m.
$\mathcal{A}_Q$	The set of all combinations generated from a user query.
$W(A_j)$	Weight for a combination A_j in the query set.
$E[y\|\mathcal{A}_Q]$	The weighted average salary for the entire query set $\mathcal{A}_Q$.
Training	
D	The training dataset of $\{(A^{(i)}, y^{(i)})\}$ pairs.
N_{samples}	The number of samples in the training dataset.
$\mathcal{L}_{\text{NLL}}$	The Negative Log-Likelihood loss function.
$\mathcal{L}_{\text{MSE}}$	The Mean Squared Error loss function.
$\mathcal{L}_{\text{Combined}}$	The Combined loss function of $\mathcal{L}_{\text{NLL}}$ and $\mathcal{L}_{\text{MSE}}$.
α	The weight of $\mathcal{L}_{\text{MSE}}$ in $\mathcal{L}_{\text{Combined}}$.

B Data Schema

The following table provides a detailed schema of the key fields from the Jobdigger dataset used in this study. It outlines the original field name, its data type, a brief description, and its specific role within our GAT-MDN model.

Table 4: Data schema for the job posting dataset.

Field Name	Data Type	Description	Role in Model
`salary_from`	Numeric	The lower bound of the provided salary range. **All salary values represent post-tax annual salary in Euros (€).**	Input for Target Variable Calculation
`salary_to`	Numeric	The upper bound of the provided salary range.	Input for Target Variable Calculation
`S`	Numeric	The calculated average expected salary $((S_f + S_t)/2)$.	Intermediate Target Variable
`S'`	Numeric	The final log-transformed and standardized salary value.	**Final Target Variable**
`geo_level_2_string`	Categorical	The province where the job is located.	Hierarchical Attribute (Parent)
`geo_level_4_string`	Categorical	The city where the job is located.	Hierarchical Attribute (Child)
`occupation_level_1_name`	Categorical	The highest-level job category.	Hierarchical Attribute (Grandparent)
`occupation_level_2_name`	Categorical	The mid-level job category.	Hierarchical Attribute (Parent)
`occupation_level_6_name`	Categorical	The specific job title.	Hierarchical Attribute (Child)
`industry_1_name`	Categorical	The high-level industry sector.	Hierarchical Attribute (Parent)
`industry_2_name`	Categorical	The specific industry sub-sector.	Hierarchical Attribute (Child)
`job_id`	String	A unique identifier for the job posting.	Identifier
`job_description_text`	Text	The full text of the job description.	Future Work

References

[1] Lazar A. Income prediction via support vector machine. In *ICMLA*, pages 143–149, 2004.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. *CoRR*, abs/1907.10902, 2019.

[3] James Atwood and Don Towsley. Diffusion-convolutional neural networks, 2016.

[4] Smaranda Belciug and Dominic Gabriel Iliescu. Deep learning and gaussian mixture modelling clustering mix. a new approach for fetal morphology view plane differentiation. *Journal of Biomedical Informatics*, 143:104402, 2023.

[5] Christopher M. Bishop. Mixture density networks. Workingpaper, Aston University, 1994.

[6] Christian Borgelt and Michael R. Berthold. Finding relevant substructures of molecules: mining molecular fragments. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002)*, pages 51–58. IEEE Computer Society, 2002.

[7] Axel Brando Guillaumes. Mixture density networks for distribution and uncertainty estimation. Master's thesis, Universitat Politècnica de Catalunya, 2017.

[8] Long Chen, Yeran Sun, and Piyushimita Thakuriah. *Modelling and Predicting Individual Salaries in United Kingdom with Graph Convolutional Network*, pages 61–74. 01 2020.

[9] Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. Exploring the potential of large language models (llms) in learning on graphs, 2024.

[10] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints, 2015.

[11] Eurostat. Nace rev. 2 - statistical classification of economic activities. `https://ec.europa.eu/eurostat/web/nace-rev2`, 2008. Based on Regulation (EC) No 1893/2006 of the European Parliament and of the Council.

[12] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[13] David Ha and Jürgen Schmidhuber. World models. *CoRR*, abs/1803.10122, 2018.

[14] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In Gäel Varoquaux, Travis Vaught, and Jarrod Millman, editors, *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, Aug 2008.

[15] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018.

[16] J. Huan, W. Wang, and J. Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In *Third IEEE International Conference on Data Mining*, pages 549–552, 2003.

[17] Akihiro Inokuchi, Takashi Washio, and Hiroshi Motoda. An apriori-based algorithm for mining frequent substructures from graph data. In Djamel A. Zighed, Jan Komorowski, and Jan Żytkow, editors, *Principles of Data Mining and Knowledge Discovery*, pages 13–23, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[18] Yang Ji, Ying Sun, and Hengshu Zhu. Enhancing job salary prediction with disentangled composition effect modeling: A neural prototyping approach, 2025.

[19] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: A generative approach to clustering. *CoRR*, abs/1611.05148, 2016.

[20] Jobdigger. Real-time labour market data and analysis. `https://www.jobdigger.nl/`, 2025. Data provided for research purposes.

[21] Keisuke Kinoshita, Marc Delcroix, Atsunori Ogawa, Takuya Higuchi, and Tomohiro Nakatani. Deep mixture density network for statistical model-based feature enhancement. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 251–255, 2017.

[22] Ivan Kobyzev, Simon J.D. Prince, and Marcus A. Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(11):3964–3979, November 2021.

[23] Cheng-Te Li, Yu-Che Tsai, Chih-Yao Chen, and Jay Chiehen Liao. Graph neural networks for tabular data learning: A survey with taxonomy and directions, 2024.

[24] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. *arXiv preprint arXiv:1511.05493*, 2015.

[25] Yasser T. Matbouli and Suliman M. Alghamdi. Statistical machine learning regression models for salary prediction featuring economy wide activities and occupations. *Information*, 13(10), 2022.

[26] Geoffrey J. McLachlan and Suren Rathnayake. On the number of components in a gaussian mixture model. *WIREs Data Mining and Knowledge Discovery*, 4(5):361–372.

[27] Paul D. McNicholas and Thomas Brendan Murphy. Model-based clustering of microarray expression data via latent gaussian mixture models. *Bioinformatics*, 26(21):2705–2712, 08 2010.

[28] Qingxin Meng, Keli Xiao, Dazhong Shen, Hengshu Zhu, and Hui Xiong. Fine-Grained Job Salary Benchmarking with a Nonparametric Dirichlet Process–Based Latent Factor Model. *INFORMS Journal on Computing*, 34(5), 2022.

[29] Qingxin Meng, Hengshu Zhu, Keli Xiao, and Hui Xiong. Intelligent salary benchmarking for talent recruitment: A holistic matrix factorization approach. pages 337–346, 11 2018.

[30] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2014–2023. PMLR, 2016.

[31] Siegfried Nijssen and Joost N. Kok. A Quickstart in Frequent Structure Mining Can Make a Difference. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 647–652. ACM, 2004.

[32] Primož Skledar. Forecasting univariate time series salary data with machine learning models, 2019. Faculty of Civil and Geodetic Engineering, University of Ljubljana.

[33] C. Stauffer and W.E.L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, pages 246–252 Vol. 2, 1999.

[34] Rosanna Stofberg, Mark Bussin, and Calvin Mabaso. Pay transparency, job turnover intentions and the mediating role of perceived organizational support and organizational justice. *Employee Relations: The International Journal*, 44:162–182, 12 2022.

[35] Yujun Sun, Fuzhen Zhuang, Hengshu Zhu, Deqing Wang, and Hui Xiong. Market-oriented job skill valuation with cooperative composition neural network. *Nature Communications*, 12(1):1992, 2021.

[36] Qiaoyu Tan, Xin Zhang, Xiao Huang, Hao Chen, Jundong Li, and Xia Hu. Collaborative graph neural networks for attributed network embedding, 2023.

[37] Brian L Trippe and Richard E Turner. Conditional density estimation with bayesian normalising flows, 2018.

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[39] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[40] Zhongsheng Wang, Shinsuke Sugaya, and Dat PT Nguyen. Salary prediction using bidirectional-gru-cnn model. *Assoc. Nat. Lang. Process*, 2019.

[41] Xifeng Yan and Jiawei Han. gspan: graph-based substructure pattern mining. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 721–724, 2002.

[42] Yuping Wang Yihui Cai Jiajun Liao Yuping Yan, Xiaoli Li. Prediction model of salary dynamic fluctuation trends incorporating multivariate time series. *Vol. 20 No. 10s (2024)*, 2024.

[43] Quan T Z and Raheem M. Salary prediction in data science field using specialized skills and job benefits–a literature. *Journal of Applied Technology and Innovation*, 6(3):70–74, 2022.

[44] Gulnarida Zhalilova, Aliyma Mamatkasymova, Elnura Zhusupova, and Kunduz Zhalzhaeva. Forecasting data science professionals' salaries using machine learning methods based on real data. *AIP Conference Proceedings*, 3244:030034, 2024.

[45] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and N. Chawla. Heterogeneous graph neural network. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.