MULTIMEDIA BIG DATA ANALYTICS AND FUSION FOR DATA SCIENCE

A Dissertation
IN
Computer Science
and
Electrical and Computer Engineering

Presented to the Faculty of the University
of Missouri–Kansas City in partial fulfillment of
the requirements for the degree

DOCTOR OF PHILOSOPHY

by
TIANYI WANG

B.S., Tianjin University of Finance and Economics, Tianjin, China, 2009
M.S., Washington State University, Pullman, WA, USA, 2011
M. S., Florida International University, Miami, FL, USA, 2017

Kansas City, Missouri
2023

MULTIMEDIA BIG DATA ANALYTICS AND FUSION FOR DATA SCIENCE

Tianyi Wang, Candidate for the Doctor of Philosophy Degree

University of Missouri–Kansas City, 2023

ABSTRACT

Big data is becoming increasingly prevalent in people's everyday lives due to the enormous quantity of data generated from social and economic activities worldwide. As a result, extensive research has been undertaken to support the big data revolution. However, as data grows in volume, traditional data analytic methods face various challenges—especially when raw data comes in multiple forms and formats. This dissertation proposes a multimodal big data analytics and fusion framework that addresses several challenges in data science for handling and learning from multimodal big data.

The proposed framework addresses issues during a standard data science project workflow, including data fusion, spatio-temporal deep feature extraction, and model training optimization strategy. First, a hierarchical graph fusion network is presented to capture the inter-modality correlations among modalities. The network hierarchy models the modality-wise combinations with gradually increased complexity to explore all n-modality interactions. Next, an adaptive spatio-temporal graph network is proposed to

capture the hidden patterns from spatio-temporal data. It exploits local and global node correlations by improving the pre-defined graph Laplacian and automatically generates the graph adjacency matrix based on a data-driven method. In addition, a dynamic multi-task learning method is introduced to optimize the model training progress by dynamically adjusting the loss weights assigned to each task. It systematically monitors the sample-level prediction errors, task-level weight parameter changing rate, and iteration-level total loss to adjust the weight balance among tasks.The proposed framework has been evaluated on various datasets, including disaster event videos, social media, traffic flow, and other public datasets.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Graduate Studies, have examined a dissertation titled "Multimedia Big Data Analytics and Fusion for Data Science," presented by Tianyi Wang, candidate for the Doctor of Philosophy degree, and hereby certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Shu-Ching Chen, Ph.D., Committee Chair
School of Science and Engineering

Mei-Ling Shyu, Ph.D.
School of Science and Engineering

Yugyung Lee, Ph.D.
School of Science and Engineering

Dianxiang Xu, Ph.D.
School of Science and Engineering

CONTENTS

ILLUSTRATIONS

TABLES

ACKNOWLEDGEMENTS

First, I would like to express my sincere gratitude to my advisor, Dr. Shu-Ching Chen, for his continuous support of my Ph.D. study and related research and his patience, motivation, and immense knowledge. His guidance helped me in all the research and writing this dissertation. I would also like to thank my co-advisor, Dr. Mei-Ling Shyu, for her valuable advice and collaboration throughout my Ph.D. study. She provided me with insightful feedback and constructive criticism on my research papers and dissertation chapters. She also helped me improve my presentation skills and academic writing. I would like to thank my Supervisory Committee members, Dr. Yugyung Lee and Dr. Dianxiang Xu, for their help and suggestions at UMKC. I would like to thank my dissertation committee members in Florida International University, Dr. Xudong He, Dr. Jainendra K. Navlakha, Dr. Hyeyoung Hah and Dr. Sitharama S. Iyengar. I am especially grateful to Samira, Haiman, Yudong, Maria, and Daniel E. Martinez for their help with data collection, experiments, and proofreading. I also appreciate the fun times we had together during coffee breaks, lab meetings, and social events. Last but not least, I would like to thank my wife, son, and parents for their love, encouragement, and sacrifice. They have always been there for me in good times and bad times. They are the source of my strength and motivation.

This research is partially supported by NSF CNS-2125165, NSF CNS-1952089, Florida Public Hurricane Loss Model (FPHLM) and the "Real-Time Tracking of Intra-Regional Migration from the Caribbean to Puerto Rico after Extreme Events" project from Natural Hazards Center.

CHAPTER 1

INTRODUCTION

1.1 Background and introduction

The information revolution has had a continuing effect on people's daily lives. Automated devices, such as phones, cameras, and sensors, have evolved into critical components of contemporary society. Due to the ubiquity of these devices and the interactions between computers and human beings, massive amounts of data have been gathered and waited to be analyzed. There is an inherent requirement for scalable and practical data analytics procedures with such enormous datasets. Given the vast amount of data available in various formats and settings, traditional data analytic approaches are becoming obsolete. As a result, more sophisticated data science techniques are required to interpret these heterogeneous, massive data collections with varying degrees of quality and semantics.

Benefits from the advancement of high-performance computing, machine learning, and data mining have been widely applied in various domains to solve real-world problems involving large-scale multimedia data, which include autonomous driving [52, 72, 97, 125], language translation [4, 48, 54, 112], disaster management [41, 45, 140, 167], and traffic flow optimization [25, 69, 127].

In recent years, multimodal learning has attracted significant interest from the research community due to its benefit in utilizing the massive amount of real-world data, which often contains multiple data sources [29, 30, 142]. Compared to its single-modality

counterpart, multimodal learning is the technique that focuses on exploiting the rich information underlying various input modalities. Information in nature always comes with different modalities with a certain degree of relationships between them. Different modalities are characterized by their distinct statistical properties. For Artificial Intelligence (AI) and machine learning models to learn knowledge from real-world data, it is crucial to interpret such statistical properties of all modalities.

The motivation for multimodal learning can be reflected in multiple aspects. These can include obtaining a more unified concept and a global view of the ecosystem, discovering hidden patterns, mining interrelationships between different data sources, and extracting knowledge across modalities. Fusing multiple data sources could provide rich and complementary information that greatly enhances the model performance. For instance, the fusion of audio and visual features and textual features has become a widely adopted strategy in personality prediction [82]. Therefore, multimodal learning can be considered a systematic and comprehensive methodology to build and train models that could process and extract the joint representation from multiple data sources.

The scope of this doctoral dissertation is to present a comprehensive multimodal big data analytics and fusion framework for data science. The proposed framework addresses several challenges in data science for handling and learning from multimodal big data. More specifically, the proposed framework addresses issues that surfaced during the entire data science project roadmap, including data fusion, deep feature extraction, and model training optimization strategy. Major challenges in multimodal big data learning can be summarized as follows:

- **Data Fusion:** In multimodal learning, data from different sources are presented in different forms and formats. Combining data from various modalities with different spatial and temporal characteristics is crucial for multimodal learning. The correlation among modalities can be demonstrated either at the feature level, such as the raw or pre-processed data from each source, or at the semantic level, such as the decision score generated by each unimodal classifier [204]. The independent property can provide new information to exploit the hidden patterns that enhance the discriminative power of the model. It is crucial to consider correlation and independence when handling multimodal fusion since both provide equally valuable insights into the problem. Common strategy toward multimodal fusion tends to focus on individual modalities. Each modality is trained on a standalone network, and the intermediate results are integrated at various levels. Early fusion and late fusion [157] are the two most widely adopted fusion strategies based on this methodology. However, due to the heterogeneous nature of multimodal data and the disconnection among networks, the fused vector still falls short of representing the complex distribution among input sources.

- **Spatio-Temporal Data Modeling:** Many real-world problems can be represented by sequences of spatio-temporal data describing activities that occur in a range of locations and periods, such as videos, traffic flow data, and remote sensing imagery data. Take traffic flow data as an example. Each entry contains the geographic location and the corresponding timestamp. State-of-the-art deep learning approaches for processing spatio-temporal data combine convolutional neural networks (CNNs)

and recurrent neural networks (RNNs). In many studies, CNN is the default network structure for spatial feature extraction. However, conventional CNN is only effective on grid structures, such as images and videos, and fails to capture the spatial relationship between objects measured using non-Euclidean distance [192]. On the other hand, when using RNN-based approaches to extract temporal features from data, information is lost because the information in the hidden layers is transmitted down through the network. In recent studies, graph convolutional network (GCN) has seen extensive applications in data that demonstrate a strong relationship between objects. However, most existing studies solely rely on pre-defined network topology and do not consider modeling the node-specific patterns. The intuitive pre-defined graph is constructed on the neighboring connectivity or specific distance measurement. It is often insufficient to include such correlations since many trivial hidden patterns may be lost. In addition, adjacent nodes frequently show diverse patterns in real-world problems such as traffic forecasting. Typical graph convolution using shared weights on all neighboring nodes cannot capture such dynamic node-specific patterns.

- **Model Training Optimization with Multi-Task Learning:** Common machine learning/deep learning training strategies only involve a single objective. However, simultaneously solving several tasks could provide quite a few benefits. It reduces memory consumption and accelerates inferences by performing multiple inferences in a single forward pass. This strategy is also referred to as multi-task learning. When training with multi-task learning, the model utilizes the aggregated loss from each task to optimize the network parameters. The final loss is calculated as the

linear weighted sum of each task's loss. As a result, one major challenge in multi-task learning is to select the optimal weight assigned for each task so that the training process can be balanced among all objectives. The most commonly adopted approaches include equal weight assignment and manual weight allocation. On the other hand, an easier task with equal weights is likely to degrade overall model performance since the relatively small loss reduces the gradient of the aggregated loss during backpropagation. On the other hand, manually tweaking the weights is overly time-consuming and demands extensive domain knowledge.

1.2 Proposed Solutions

This dissertation proposes a multimodal data analytics and fusion framework to address the aforementioned limitations and challenges. The proposed framework is an end-to-end platform for gathering, organizing, and analyzing large-volume, heterogeneous data in different forms and formats. We developed a novel deep learning architecture to 1) pre-process and extract features from multi-modality data, 2) fuse features from different sources to exploit the inter-modality interactions, 3) capture the spatio-temporal dependency from complex real-world data that contains sophisticated relationships, and 4) explore the opportunity to optimize the model learning process by investigating the correlation between different tasks and objectives. The proposed framework's main components were evaluated using datasets from a wide range of applications, including a disaster video dataset containing video clips, audio, and text description [177], a social media multimedia disaster dataset containing multiple label concepts [182], a large-scale air

travel transaction dataset containing origin-destination and airfare price information [181], a flight operation dataset, and two ground transportation traffic flow datasets. The proposed solutions are:

- **Hierarchical Multimodal Fusion:** We developed a hierarchical multimodal fusion framework to address the aforementioned challenges. It is difficult to model the cross-modality relationship among modalities in multimodal fusion, and it is much more challenging to learn the joint representation among multiple modalities. The proposed model adopted a tree-based hierarchical graph structure that iteratively combines each modality on different levels to learn the cross-modal interactions among all their combinations. It also dynamically allocates the weights for each pair in a sample-aware fashion. The similarities between nodes are studied and utilized as the edge weights in generating nodes in the next level. A dynamic attention unit is also applied to determine the importance of each modality and assign it as the weights for the connecting edges. Furthermore, to preserve the independent characteristics of the source modality, the original feature from each modality is added back to the final fused feature. The proposed fusion network is flexible and highly modulated, which can be applied to various network structures.

- **Adaptive Spatio-Temporal Graph Network:** To address the challenge in spatio-temporal data modeling, we proposed an adaptive spatio-temporal graph network to utilize multiple graphs to capture the spatial dependency in the data. The local GCN focuses on the spatial connectivity between nodes with direct connections. A shifted graph Laplacian method was developed to expand the local GCN to

learn more trivial hidden patterns. The global GCN captures the network-wide correlation among nodes that share similar characteristics. In addition, an effective normalization technique is also applied to control the adjacency matrix's sparsity and reduce redundant node-wise correlation. Furthermore, a novel data-driven adaptive graph generation approach was designed to help the model learn a comprehensive network topology and produce the adaptive GCN. A bidirectional long-short-term memory (BiLSTM) based sequence-to-sequence fusion module was applied to extract the short-term temporal information in the data sequence. The sequence-to-sequence fusion network also combines the output from each graph to leverage the cross-modal interactions in modeling the temporal dependency. Last but not least, a spatio-temporal graph transformer module was developed to complement sequence-to-sequence fusion module by dynamically capturing the spatio-temporal dependencies in long-term predictions.

- **Dynamic Multi-task learning:** A novel dynamic multi-task learning strategy was proposed to address the challenge of optimizing the model training process in a multi-task learning scenario. The proposed model utilizes the shared network to learn general information that can be leveraged across all tasks. On the other hand, the task-specific networks help the model learn patterns related to each task. The proposed dynamic task balancing approach automatically adjusts the training progress on the sample and task levels. The sample-level dynamic balancing function focuses on difficult instances by allocating more resources during training. At the same time, the task-level dynamic balancing mechanism adjusts weight distribution by attending

to the learning rate of each task. In addition, extra cautions are paid to the task imbalance problem by introducing the task penalty term to the task-level balancing function. Finally, a loss weight re-balancing method was applied to automatically adjust each task's weighted scalar so that their losses are tuned to a similar scale. As a result, the model can obtain a more balanced loss distribution at the beginning of each training iteration.

1.3 Contributions

The major contributions of this dissertation can be summarized as follows.

- The proposed multimodal big data analytic and fusion framework is highly flexible and can be applied to various applications and domains. The proposed framework can extract spatio-temporal features with complex correlations, fuse different modalities, and retain the independent and correlated information, and improve the model training process by dynamically allocating resources and balancing the impact of each training task.

- A hierarchical graph fusion network is proposed to capture the inter-modality correlations among modalities and, at the same time, retain their independent properties. The structure of the hierarchical graph fusion network can be seen as a tree graph. Each modality is fused on different levels, in which the cross-modality interactions are learned based on various combinations. We use nodes to demonstrate different input signal combinations and edges to represent the similarity between each pair of signals. The proposed method is very flexible and can be applied to various

scenarios.

- A graph-based deep learning network is proposed to capture the spatio-temporal relationship in the data. The proposed framework learns the spatial dependency using multiple pre-defined and adaptively generated graph convolutional networks. Novel approaches such as shifted graph Laplacian, sparse matrix normalization, and adaptive graph adjacency matrix generation have been proposed to address limitations and challenges in the current literature. A BiLSTM-based sequence-to-sequence fusion model is applied to model the short-term temporal information and learn the cross-modal interactions between input sources. A transformer-based graph module is developed to capture the time-evolving long-term dependencies. The network also utilizes graph regularization to facilitate the loss function in minimizing the training loss. Results on real-world traffic data demonstrate the effectiveness of the proposed work.

- A dynamic multi-task learning method is proposed. It eliminates the need for manual loss weight tuning during the training process, which can be inaccurate and time-consuming. It automatically adjusts the training process during three model learning levels. On the sample level, it aims to prioritize resources to objects that produce more significant errors. On the task level, it adjusts the weight of the loss generated by each task during the training phase so that tasks producing more significant losses will be prioritized. Another mechanism is applied to the training iteration level, enabling the model to automatically adjust the loss weight for each task to improve performance.

- We have tested the proposed framework on various datasets and applications, including disaster damage assessment, disaster situation assessment, airfare price prediction, and traffic data forecasting.

1.4 Scope and Limitations

The assumptions and limitations that apply to the proposed framework are as follows.

- Some of the model parameters are empirically determined, such as the parameter used in calculating the graph edge weights that control the growth rate in the hierarchical graph fusion network.
- Without loss of generality, the proposed multimodal big data analytics and fusion framework is mainly evaluated on video, image, and numerical data. However, the proposed model, including the hierarchical graph fusion network, spatio-temporal graph network, and dynamic multi-task learning method, can be extended to other data types and domains.
- Several pre-trained models are incorporated in the proposed multimodal big data analytics and fusion framework to extract features from the raw image and audio data. However, the scope of the dissertation does not include image and audio feature extraction.

1.5 Outline

This dissertation is structured as follows. Chapter 2 summarizes the literature on multimodal fusion, spatio-temporal feature extraction, and multi-task learning. Chapter 3 introduces the proposed multimodal big data analytics and fusion framework and its major components. Chapter 4 introduces the proposed dynamic multi-task learning method. Chapter 5 discusses multimodal fusion solutions and the approaches to extract cross-modal correlations. Chapter 6 presents the deep learning model for extracting spatio-temporal information from real-world data using graph-based networks. Finally, Chapter 7 summarizes the current work and provides potential future directions.

CHAPTER 2

RELATED WORK

In this chapter, the literature in the area of multimodal fusion, deep learning for spatio-temporal graph data and multi-task learning are presented.

2.1 Multimodal Fusion

The benefit of multimodal learning can be demonstrated in various aspects. For instance, it helps to obtain a more unified concept and global view of the ecosystem, discover the hidden pattern and mine the interrelationships between different data sources. The fusion of multiple data source could provide richer and complementary information that greatly enhance the model performance. For instance, the fusion of audio, visual and textual features has become a widely adopted strategy in personality prediction [82]. Multimodal fusion techniques can be categorized based on where and how the fusion is conducted, namely feature-level fusion, intermediate fusion and decision-level fusion. Figure 1 illustrates the 3 types of fusion strategies.

Another classification method is based on whether the fusion technique utilizes a particular machine learning model, leading to free and model-based fusion. The following section will discuss fusion techniques based on the types mentioned above.

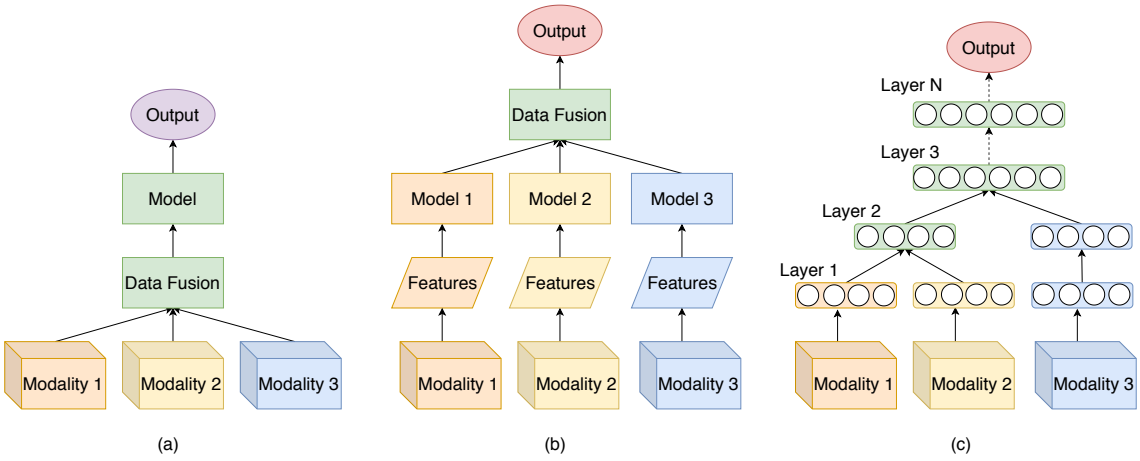


Figure 1: A demonstration of different types of fusion models for multimodal learning. (a) Early or feature-level fusion, (b) Late or decision-level fusion, and (c) Intermediate or hybrid fusion

2.1.1 Feature level fusion

Feature level fusion, known as early fusion, is the fusion strategy of integrating data from multiple modalities immediately after extraction. It is illustrated in figure 1(a). Early fusion is usually implemented by joining raw or pre-processed data from each modality. Some popular modalities include:

- **Visual features:** This feature type includes color, shape and texture. Visual features can be extracted using the bag of visual words (BOVW) and scale-invariant feature transform (SIFT) techniques. The recent popularity of deep learning started the trend of using features that are automatically generated by neural networks, such as the convolutional neural network (CNN).

- **Text features.** Textual features can be generated by counting word occurrences in

the document or using various text embedding techniques, such as word embedding [124], bag-of-words (BOW) and TF-IDF.

- **Audio features** Audio features may be generated by using handcrafted features such as Mel-frequency cepstral coefficients (MFCC) [114] or automatically by neural network-based feature extractor such as Soundnet [9].

- **Other multimedia features** Besides the features mentioned above, other features, such as various sensory data and depth images, have also been widely adopted in multimodal fusion.

Feature-level fusion exploits the correlation and interaction between low-level features from each modality. Additionally, if handcrafted features are adopted, since data from each source are merged in the early stage, only a single model needs to be trained, which leads to a much more compact model architecture.

Early fusion could be implemented by concatenating the features of each modality, which leads to a relatively large feature size. Dimensionality reduction techniques such as PCA are applied to project the high dimensional vectors into lower-dimensional space [81]. Auto-encoder is another technique that is widely used in deep learning-based multimodal tasks to reconstruct the original input based on its distributed representation [111]. Based on this theory, studies have been conducted to map the data from multiple sources into a common vector space. Wagner *et al.* [174] proposed a fusion model for multispectral pedestrian detection problems by utilizing early fusion to combine the two data sources at the pixel level.

2.1.2 Decision level fusion

Decision-level fusion, or late fusion, is the fusion strategy that utilizes the final score generated by each unimodal classifier. Such scores are analyzed and manipulated to obtain a final decision score for the main task. An illustration of decision-level fusion can be found in figure 1(b).

Decision-level fusion has multiple advantages over feature-level fusion. Features from each modality tend to have different semantic representations. Such discrepancy can be amplified further by issues like time synchronization and missing data. In comparison, the decision score used in decision-level fusion does not pose such a problem. Moreover, decision-level fusion allows domain-specific models and algorithms for each modality, such as applying CNN for visual data and RNN for temporal data, often producing better representation for each data type.

However, the decision-level fusion strategy does not consider the feature-level correlation among each modality since separate models are built for each modality. This will lead to the loss of useful inter-modality information. Moreover, the learning process may become time-consuming since each modality requires a separate model to generate the required unimodal score.

Zhu *et al.* [213] used late fusion to combine the scores of image and optical flow data generated by each unimodal network. Hou *et al.* [68] concatenated the output from the audio and video models to train the fusion network. Vielzeuf *et al.* [172] proposed a score tree method involving multiple score fusion layers. It starts with combining the final score of each model and concatenates it with the score from other modalities. Then, each

joint vector will be fed into another fully connected layer to get the prediction score. The top of the score tree is the final fully connected layer that will take the combined score and generate the final prediction.

2.1.3 Intermediate Level Fusion

Intermediate-level fusion is the strategy of transforming the raw inputs into their higher-level representation. It is easier to fuse different modalities and learn the joint multimodal representation. Neural networks are widely applied to extract the raw input features. The input data is passed through multiple layers, undergoing various linear and non-linear transformation that generates low or high-level feature vectors. Additionally, each modality can be trained on its modality-specific algorithm. Later on, the output from one layer can be joined with the output from other modality-specific layers. Figure 1(c) illustrates a simple network architecture utilizing intermediate-level fusion.

Farnadi *et al.* [49] proposed stacking and power-set combination strategy for multimodal fusion. Stacking uses the predicted result of one NN as input of another NN in the next epoch (for multi-task learning where there are multiple target variables). The power-set combination is the method of using the power-set combination of data sources from all unimodal. This method captures both the early and late-level meanings of multiple modalities. Hu *et al.* [71] stacked joint layers and the unimodal layers into one network to preserve the consistency of intra-modality and inter-modality, which then helped to establish the correlations in other layers. Vielzeuf *et al.* [171] proposed a network architecture that fuses multimodal features by calculating the weighted sum of the layers

from the corresponding unimodal networks and their previous layers. It utilizes unimodal hidden representations and a central joint representation at each layer.

2.1.4 Model Free Fusion

Most multimodal fusion tasks adopted model-free fusion methods. Such methods are easy to implement and do not depend on a specific prediction model. Furthermore, model-free fusion is often used to create the joint representation that is the input for many model-based fusion methods. Here we introduce several widely adopted methods, including linear weighted fusion, bilinear fusion and canonical correlation analysis fusion.

2.1.4.1 Linear weighted fusion

Linear weighted fusion is the method of combining information from multiple modalities in a linear way. It can be applied to both feature-level and decision-level fusion. The weight for each feature can be manually assigned or learned by machine learning models, such as support vector machine (SVM) or neural network.

Max fusion. Max fusion $y^{max} = f^{max}(x_1, x_2)$ takes the maximum between the two features in each dimension. It can be illustrated as:

$$y^i_{max} = max\{x^i_1, x^i_2\}$$

where $1 \leq i \leq D$ and $x_1, x_2, y_{max} \in \mathbb{R}^D$

Sum fusion. Sum fusion $y_{sum} = f_{sum}(x_1, x_2)$ computes the summation of the two features on the same vector space location i:

$$y^i_{sum} = x^i_1 + x^i_2$$

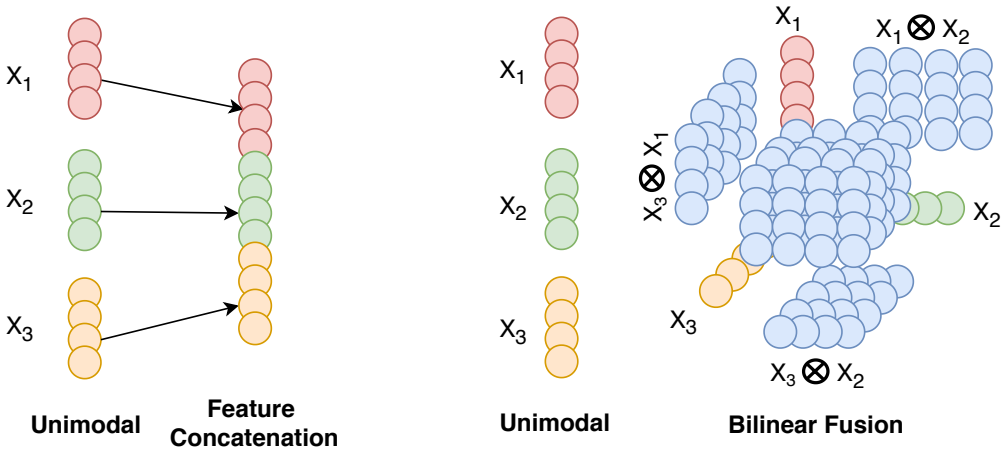


Figure 2: A demonstration of concatenation fusion and bilinear fusion. Left: early or feature-level fusion using the concatenation fusion method. Right: feature-level fusion using bilinear fusion

Concatenation fusion. Concatenation fusion $y_{cat} = f_{cat}(x_1, x_2)$ merge the two feature vector x_1 and x_2 by appending one feature after another. Unlike the previous methods, it will produce a joint feature with a dimension of $D' + D''$:

$$y_{concat} = [x_1, x_2]$$

Figure 2 (left) shows an example of concatenation fusion of 3 modalities.

Max and sum fusions are preferred when the scores for each modality can be easily computed. However, a potential drawback of this method is when outliers appear, which will outweigh the features or scores of other modalities. The concatenation method can retain most information at the cost of substantially increasing the joint vector dimension. It will be problematic when the task involves a large number of modalities. A possible solution to this issue is to use dimensionality reduction techniques such as PCA to shrink the feature size before fusion. Another option is to use a fully connected layer, pooling layer or

convolutional layer with a large filter size to lower the input vector size. However, all these methods will inevitably introduce information loss. Therefore, balancing performance and efficiency is a critical factor that needs consideration. Vielzeuf *et al.* [172] tackled the video emotion classification problem by concatenating the semantic score generated by image and optical flow modality. Then, several fully connected layers are applied to learn the weights for the joint decision vector. Hu *et al.* [71] used a hierarchical strategy to concatenate the unimodal feature with the joint feature throughout different network layers.

2.1.4.2 Bilinear (tensor) fusion

Recently, Tensor fusion has attracted the attention of many studies. Tensor fusion tackles the heterogeneous data distribution challenge in multimodal learning by fusing each modality at the tensor level. As a result, it enables the model to learn the granularity of cross-modal interactions. Tensor fusion has demonstrated promising results in multimodal deep learning for visual question answering [20] and sentiment analysis [194]. Ben-younes *et al.* proposed a framework to solve the visual question answering problem [20]. They extracted features from both visual images and textual questions using GRU (Gated Recurrent Unit) and ResNet [66]. Then, features are fused using the tensor fusion approach. During the fusion process, a tensor-based Tucker decomposition approach is utilized to parametrize the tensor correlation between visual and textual representations. Another work by Zhao *et al.* [208] used a multi-agent tensor layer and convolutional fusion to capture the cross-modal interactions.

Bilinear fusion has demonstrated great success in multimodal deep learning for visual question answering [20] and sentiment analysis [195]. The fusion function $y_{bi} = f_{bi}(x_1, x_2, x_3)$ computes the outer products of the three feature vectors from each modality. It can be illustrated as:

$$y_{bi} = x_1 \otimes x_2 \otimes x_3$$

Here $\otimes$ indicates the outer product between vectors, and the combined vector $y_{bi} \in \mathbb{R}^{D^3}$ captures the interaction between unimodal vectors at every spatial location. The multiplicative relationship in the feature representation helps bilinear fusion exploit the correlation among all unimodal feature representations. Figure 2 (right) shows an example of bilinear fusion for three modalities.

Ben-younes *et al.* [20] proposed a framework to solve the visual question-answering problem. They extracted features from visual images and textual questions using GRU (Gated Recurrent Unit) and Resnet. Then, features are fused using bilinear fusion. Multimodal tensor-based Tucker decomposition efficiently parameterized the bilinear interactions between visual and textual representations. Liu *et al.* [113] used the low-rank weight decomposition method to enhance the efficiency of the multimodal framework tested on various tasks, including multimodal sentiment analysis, emotion recognition and speaker trait analysis.

2.1.5 Model Based Fusion

Model-based fusion uses machine learning models to learn the interaction among modalities.

2.1.5.1 Attention based fusion

Contributes the most information to distinguish an object. Attention mechanism has been widely employed in NLP (Natural Language Processing). In multimodal analysis, the contribution of each modality is not the same. To prioritize the most effective modality, an attention mechanism could be applied to assign weight to encourage or punish a specific modality.

The attention layer uses the combined feature vector from different modalities as input and generates the attention weight vector. For instance, let X_1, X_2 and X_3 be the feature set from three modalities, $C = [X_1, X_2, X_3]$ be the joint feature representation. Then, the attention weight vector α_{fuse} and the fused multimodal feature vector F are computed as follows:

$$P_F = tanh(W_F \cdot C)$$

$$\alpha_{fuse} = softmax(w_F^T \cdot P_F)$$

$$F = C \cdot \ \alpha_{fuse}$$

Here, W_F and w_F^T is the learned weight parameter, P_F is the relevant score for feature F

2.1.5.2 Residual Based Fusion

Residual learning has been proven effective in training deep networks and has achieved state-of-the-art performance. It utilizes the skip-connection mechanism to reuse the activation from a previous layer to help the adjacent layer for weight learning. The residual layer significantly alleviates the vanishing gradient problem, in which the weights learned for each lay get too small that the network stops learning. A residual layer can be

adapted as a correction function in multimodal learning. The complementary channels are utilized to correct minor errors in the prediction results.

In practice, the residual unit takes the decision score generated by the network of each unimodal, then skips connecting them with the score generated by the joint representation of the same set of features. For instance, let N be the number of unimodal, Y_0 be the ground truth, Y_i be the model prediction, and ϵ be the error term between Y_i and Y_0. Our goal is to predict $Y^{'}$, which is the sum of the averaged predictions and the learned correction term c:

$$Y^{'} = Y_{avg} + c = \frac{1}{N}\sum_{i=1}^{N} Y_i + c = Y_0 + \frac{1}{N}\sum_{i=1}^{N} \epsilon_i + c$$

And the residual correction unit is forced to minimize the loss, we have:

$$||Y^{'} - Y_0|| \rightarrow 0$$

Which can be expressed into a constraint on c and ϵ_i:

$$||\frac{1}{N}\sum_{i=1}^{N} \epsilon_i - c|| \rightarrow 0$$

Here, the averaged prediction result serves as the identity mapping function. As the correction term c is learned through the training, the model could measure the confidence level of a specific input channel.

Audebart *et al.* [7] studied the semantic segmentation of earth observation data problem by using the residual unit as a correction term to improve the fusion performance. More specifically, the features generated by each model just before the final softmax layer are concatenated and fed into a 3-layer CNN. At the same time, the same feature from each

model is skip-connected to the fused feature. Audebart *et al.* [8] fused the RGB image and remote sensing data using the residual late fusion strategy. The core module includes a residual convolutional neural network that takes as input the last feature maps from two deep networks, and each fully connected network generates a prediction.

2.1.5.3 Deep Boltzmann Machine

A deep Boltzmann machine (DBM) is an undirected graphical model that can generate the joint representation of latent random variables from multiple modalities. The multimodal representation is trained using the variational approach [152]. Compared with multiple-layer perception (MLP), one of the big advantages of DBM for multimodal learning is the ability to generate missing data modalities due to its generative nature [152].

Liu *et al.* [111] used a bimodal deep autoencoder (BDAE) based on DBM to combine multiple types of physiological signals at the feature level for emotion recognition tasks. Wu *et al.* [184] combined DBM with fully connected and convolutional layers to extract high-level features from all modalities. Similarly, Radu *et al.* [145] applied DBM on mobile sensor data for the activity recognition task.

2.1.5.4 Graph Based Fusion

Graph-based fusion networks transform modalities and the interactions among them into fusion graphs. Features from each modality are considered vertices, and the relationships between them are implemented as the edges. Zadeh *et al.* tried to use a Dynamic Fusion Graph (DFG) to model the n-modal dynamics [196]. Compared to Tensor Fusion, DFG achieves better training efficiency where fewer learnable parameters

are introduced. It also uses learned parameters to control the activation of certain edges and thus dynamically changes the network structure. Multimodal metrics learning and graph-based fusion are combined to measure feature similarity between modalities [6]. Chen *et al.* [27] proposed a heterogeneous graph-based fusion network that focuses on the fusion of multimodal data with missing modalities. It uses a graph network to project the missing data into a joint embedding space with other modalities.

2.2 Multi-Task Learning

Multi-task learning aims to train a single model for multiple related tasks with better learning efficiency and model performance than training one model for one task. The intuition behind multi-task learning is that the knowledge learned from one task can complement learning from others.

One main approach in multi-task learning is integrating different task objectives and jointly training the model. Bischke *et al.* [22] followed this idea and trained an encoder-decoder model for both distance estimation and building segmentation on remote sensing imagery. The performance metrics of both tasks are improved by utilizing the multi-task learning framework. However, it remains challenging to tune the weights applied to each task manually.

Multi-task learning can be categorized into three general types based on how input sources are utilized to predict output targets. Figure 3 shows the three types of multi-task learning paradigms. Assume there are N unique input $X^1, X^2 ... X^N$ and N unique output targets $Y^1, Y^2 ... Y^N$, the first type of multi-task learning uses each input data source to

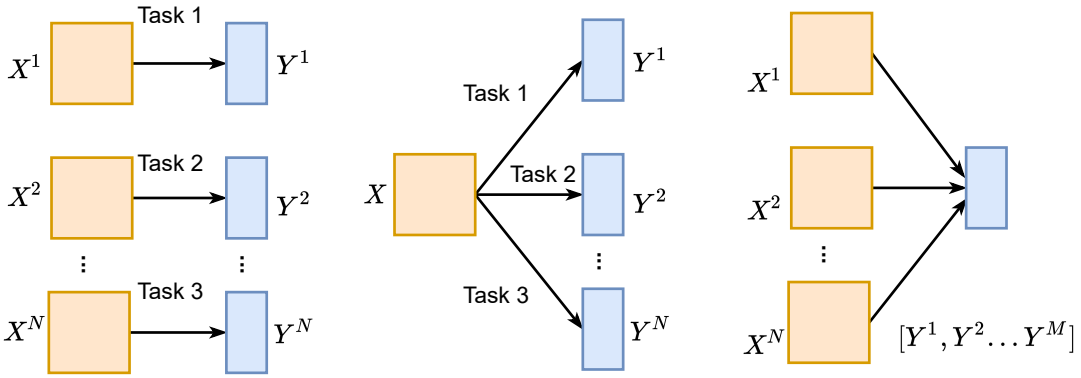


Figure 3: Left: each input source predicts its output target. Center: single input source predicts multiple output targets. Right: multiple input sources combined to predict multiple output targets.

predict its corresponding output target [187, 201]. The second type uses a single data source to predict all N output targets [77, 106–108, 134, 166]. The third multi-task learning type utilizes N input sources to collaboratively predict M output targets or each input source can be used to predict multiple targets [106, 108, 199].

Multi-task learning exploits the shared semantic information across tasks by training multiple tasks in the same model. Related tasks can complement each other to make the model more generalized. This leads to better training and inference efficiency and more robust model performance. Multi-task learning also shows excellent potential in the multimodal learning domain. Sener *et al.* [154] utilized multi-objective optimization to find the Pareto optimal solution to minimize the weighted combination of task losses. Vandenhende *et al.* [169] applied a multimodal distillation unit to model task correlation from various levels of the network. A more recent work by Hu and Singh [74] employed an encoder-decoder mechanism to encode each input modality and decode them into a shared

embedding space. The most prevalent multi-task learning approaches can be categorized into 1) network architecture engineering and 2) feature and task relation learning.

2.2.1 Network Architecture Engineering

A standard practice in multi-task learning is constructing a shared network structure to extract the common features among all tasks [37, 109, 118, 206, 209]. The task-wide features are then sent into each task-specific output head to produce the prediction score for each task. Zhao *et al.* [209] used a task-specific projection matrix as a module to process the feature map so that the feature map dimension is consistent with the rest of the network structure. In another study by Liu *et al.* [109], an attention module is applied to the shared network to produce task-specific context vectors that retain the critical information for each task. Instead of utilizing a single shared network to extract the global features, another type of model contains multiple task-specific networks [55, 129, 151]. Ruder *et al.* [151] proposed the Sluice network, where each task-specific network utilizes information from both shared and task-specific output from the previous layers. It divided the task-specific network into two components. Each is responsible for extracting task-specific or shared information.

Existing deep neural network architectures can be tweaked to handle multi-task learning problems. The cross-stitch network [129] uses the cross-stitch units to combine multiple networks where each is trained for a specific task. This helps the model to learn the optimal combination of shared and task-specific features. UberNet [92] builds the task-shared layer using a pyramid structure based on the VGG-NET [156]. It feeds a

series of down-sampled images of different resolutions into the task-shared layer, which is constructed on top of the task-specific layers. Using a technique named 12-in-1, Lu *et al.* [117]'s multi-tasking model simultaneously processes 12 distinct datasets. The proposed method achieves state-of-the-art performance compared to other techniques by pre-training the model using multi-task learning on all twelve tasks.

Sun *et al.* [163] introduces the Task Switching Networks (TSNs) that use a single set of network parameters for all tasks. This is achieved by applying a conditional network to generate task-specific vectors. The proposed network utilizes an encoder-decoder structure based on U-Net [149]. The encoder learns the shared representation across all tasks, and each decoder generates the task-specific embedding. In another work, Vandenhende *et al.* [169] proposes a network structure that models task interactions at various receptive field scales.

2.2.2 Feature and Task Relation Learning

Exploiting the underlying relationship between features and tasks can facilitate information sharing in multi-task learning. Lu *et al.* [116] uses a dynamic branching approach to construct a tree-like network structure automatically. It places certain concepts in the same branch by considering task correlation and complexity. Also, the use of weight uncertainty [85] models the task-dependent homoscedastic uncertainty to weigh different tasks. In another work by Chen *et al.*, the gradient norms of each task-specific layer are used to dynamically change the learning progress [32]. Other measurements can also be used to balance the task weights. Dynamic task prioritization [61] uses the key

performance indicator, such as accuracy or average precision, as the learning progress signal to adjust the task weight distribution.

Task relationship plays a vital role in multi-task learning since the relationship could be modeled as similarities using various evaluation metrics. Early works treat this relationship as prior knowledge. In a study conducted by Kato *et al.* [84] and Evgeniou *et al.* [46], the task similarities are used to construct regularizers to guide the learning of several tasks based on the idea that the similar the task, the closer the corresponding model parameters are located in the vector space. However, in many cases, the task relationship is absent from the beginning. The model must learn such a relationship in a data-driven way. Bonilla *et al.* [59] introduced a regularization-based method that learns the task relations hierarchically by developing a tree-based algorithm. Similarly, in a study by Nguyen and Okatani [133], dense co-attention layers are used to hierarchically group the tasks by searching over the layers for each task. In another work, bi-directional GRU layers with pairwise attention are used for generating the shared representation of all tasks [1].

Task grouping aims to separate tasks that could harm each other by selectively sharing the information among tasks. Task grouping is relatively tricky as finding the right task combination is time-consuming and requires relative domain knowledge. Several works have utilized empirical studies to find the optimal grouping strategy of tasks in several application domains and explore the relationship between main task and auxiliary task [5, 21]. Standley *et al.* [161] utilized a branch-and-bound algorithm and the performance of each task-specific network to group a subset of the task-specific network for maximum performance gain.

Transfer learning has been widely adopted in quite a few deep learning domains. It can also help to transfer the task relationship in a multi-task learning scenario. Zamir *et al.* [197] propose a task taxonomy strategy to build a hypergraphy to model the task relationship transferability. It contains several task-specific networks that learn the task transfer during training. Then, the task affinity score is acquired from a transfer function to limit the number of tasks with access to the data. In another work [44], a representation similarity analysis (RSA) calculates the correlations between task-specific networks. It is based on the assumption that task-specific networks will present similar representations if the two tasks have positively related. In addition, RSA only compares the representation among task-specific networks and does not perform transfer learning, making it much more efficient. Song *et al.* [158] adopted the same idea of RSA by comparing the similarity score between tasks. Instead of comparing the task-specific network representation, they used attribution maps containing each unit's relevance score to the network output.

2.3 Deep Learning for Spatio-Temporal Graph Data

Many real-world problems can be represented by sequences of spatio-temporal data that describe an activity that occurs in various locations and times, such as videos, traffic flow data and remote sensing imagery data. Much research has been spent on studying spatio-temporal data in recent decades. However, the traditional approach mainly relies on manual feature engineering, which requires extensive prior and domain knowledge of the data. Recently, deep learning models such as CNN and RNN significantly alleviated this effort by allowing automatic feature extraction due to their automatic feature representation

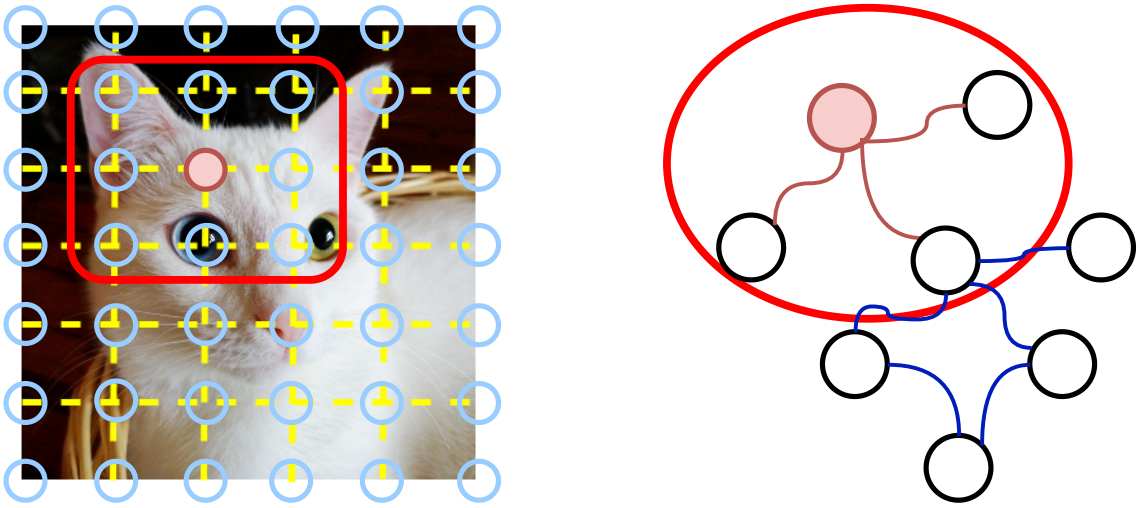


Figure 4: A comparison of 2D convolution and graph convolution. Left: 2D convolution used by most CNN. Right: graph convolution that applies to all neighboring nodes of the target node.

learning capability.

2.3.1 Graph Data

Graph data is a unique form that can exist in many domains. Unlike regular spatio data that a two-dimension matrix or grid can represent, nodes in graph data cannot be measured using Euclidean distance. Therefore, traditional deep learning models like CNN cannot be applied. Recently, graph convolutional network (GCN) has been widely applied to graph-structured data due to its ability to model data with non-Euclidean distance relationships, such as traffic flow and brain network data [79, 99]. Unlike CNN, which utilizes a "grid-like" fixed-size sliding window to calculate convolution across the 2D plane, graph convolution calculates the average value of all neighboring nodes connecting to the target node. The comparison of two convolutions is shown in Figure 4

Guo *et al.* [63] used a dual-attention network containing graph-level and sequence-level attention mechanisms to model the spatial and temporal dependency between nodes in the graph. The flight delay is predicted in 30-minute intervals using classification instead of traditional precise time using regression to improve the model efficiency.

The Autoencoder model can be utilized to learn the low-dimension feature encoding. For spatio-temporal graph data problems, autoencoders can be applied to extract the temporal representation from the input data. In Bao *et al.* [18]'s work, they predict the network-wide average delay by first categorizing US domestic airports into three types based on the number of average daily flights. Then K-means clustering was used to divide the airports into four groups based on their average hourly delay pattern.

2.3.2 Graph Convolutional Network for Spatio-Temporal Data Processing

GCN has been successfully applied in spatio-temporal data modeling when the correlation between objects can be constructed as a graph. Existing applications of GCN-based networks, such as [105], [80] and [67], only models the stationary spatial dependencies in the GCN layers. Although there have been efforts in incorporating the temporal node dependency into the GCNs, such as embedding input graph signals of immediate adjacent time steps in the same GCN layer [210], such approaches still fall short in learning the long-term temporal relations. Recent advance in transformer [26, 70, 147] enables the models to learn the temporal information from a more extended period. Instead of memorizing the hidden states of neighboring short-term input tokens, the transformer utilizes the self-attention mechanism to generate the contextual vectors based on the entire

input sequence. A deep neural network based on GCN and sequence to sequence model is used to extract the spatial-temporal features. Zeng *et al.* [198] used a weighted adjacency matrix based on the weighted sum of the spatial distance and flight frequency between each pair of airports for the GCN. Li *et al.* [98] proposed an adaptive GCN (AGCN) that captures the structural relations that are not specified by the graph's adjacency matrix. It generates a residual graph adjacency matrix using a learnable distance function and the characteristics of two nodes as inputs.

2.3.3 Traffic Forecasting as a Spatio-Temporal Data Processing Problem

Traffic forecasting is one domain that heavily involves spatio-temporal data. DCRN-N [102] makes multi-step traffic prediction by integrating graph convolution with diffusion process and recurrent models in an encoder-decoder architecture. STGCN [192] utilizes GCN to model spatial correlations and a temporal convolution network to capture temporal dependencies from the input data. A convolutional graph network AGCRN [15] learns the adjacency matrix from input data to model the spatial correlations and adopts a gated recurrent neural network to model the temporal correlations. Zheng *et al.* [211] use dilated causal spatio-temporal graph convolution layers to learn spatio-temporal correlations in multiple time intervals and adopted multi-range attention to help the model focus on different ranges. Li *et al.* [103] use static and dynamic graphs to learn short and long-term data patterns. A multi-head attention unit is leveraged to learn the correlation among multiple variables. In AdapGL [202], the model trains two GCNs back-to-back. The pre-defined adjacency matrix is used to optimize the learned adjacency matrix through each

training iteration. In another work, Bai *et al.* [15] applied a note adaptive parameter learning module to model the complicated correlations in traffic forecasting problems. It uses matrix factorization to reduce the computational complexity created by the high dimensional weight parameter matrix. Additionally, the graph's adjacency matrix containing the hidden spatial correlation between nodes is learned from the training process instead of using a pre-defined correlation matrix. Zheng *et al.* [211] combines the pre-defined graph and adaptively generated graph to learn the spatio-temporal dependency in the data. Instead of using a standard RNN-based network to learn the temporal representation, a spatio-temporal joint graph convolution (STJGC) is applied. STJGC performs the convolution operation by sliding and skipping over the input sequence based on certain intervals or time steps.

CHAPTER 3

OVERVIEW OF THE FRAMEWORK

3.1 Framework Overview

Massive volumes of data are generated daily from various sources, such as social and economic activities and social media. There is an inherent need for a scalable and meaningful data processing pipeline with such large datasets. Given the vast amount of data available in various formats and contexts, traditional data processing methods are becoming obsolete. As a result, more sophisticated data science techniques are required to process these heterogeneous, large datasets with varying degrees of quality and semantics. Additionally, high-performance computing has evolved into a necessary technology for processing this data in a scalable manner.

This dissertation proposes a multimodal big data analytics and fusion framework for data science, with applications in disaster information management and traffic forecasting, among others. The overall framework, shown in Figure 5, comprises three major components: hierarchical graph fusion, adaptive spatio-temporal graph network, and dynamic multi-task learning. These components are integrated coherently to address the challenges inherent in multimodal big data and support the various functionalities available in this field. Hierarchical graph fusion captures inter-modality correlations between modalities by fusing them hierarchically. The adaptive spatio-temporal graph network captures the spatio-temporal information from the data and learns the correlation between local

and global features. The dynamic multi-task learning method automatically adjusts the training progress on sample, task and iteration levels to improve the model performance. The proposed framework has been tested on various applications, including disaster damage assessment, disaster situation assessment, airfare price prediction, and flight delay prediction.

3.2 Hierarchical Graph Fusion

Multimodal learning has attracted significant interest from the research community due to its benefit in utilizing the vast amount of real-world data, which often contains multiple data sources [29] [30] [142]. Compared to single-modal learning, multimodal learning is a technique that focuses on utilizing the rich information contained within various input modalities. Multimodal fusion is a critical step in multimodal learning, as it combines the input features of each modality into a single vector. Therefore, how features are fused significantly impacts the model's ability to harvest information from multiple input sources. This study aims to develop a hierarchical graph fusion network to capture the complex cross-modality dependency among each input source. We first utilize several pre-trained deep learning models to extract intermediate features from the raw input of each modality. Then, the feature fusion is done by applying the proposed hierarchical graph fusion approach. The multimodal fusion network connects each modality hierarchically to create a graph structure with vertices representing joined modalities and edges representing cross-modality interactions. The relative importance of joined modalities within the same level is determined sample-to-sample and then used to formulate the joint embedding used

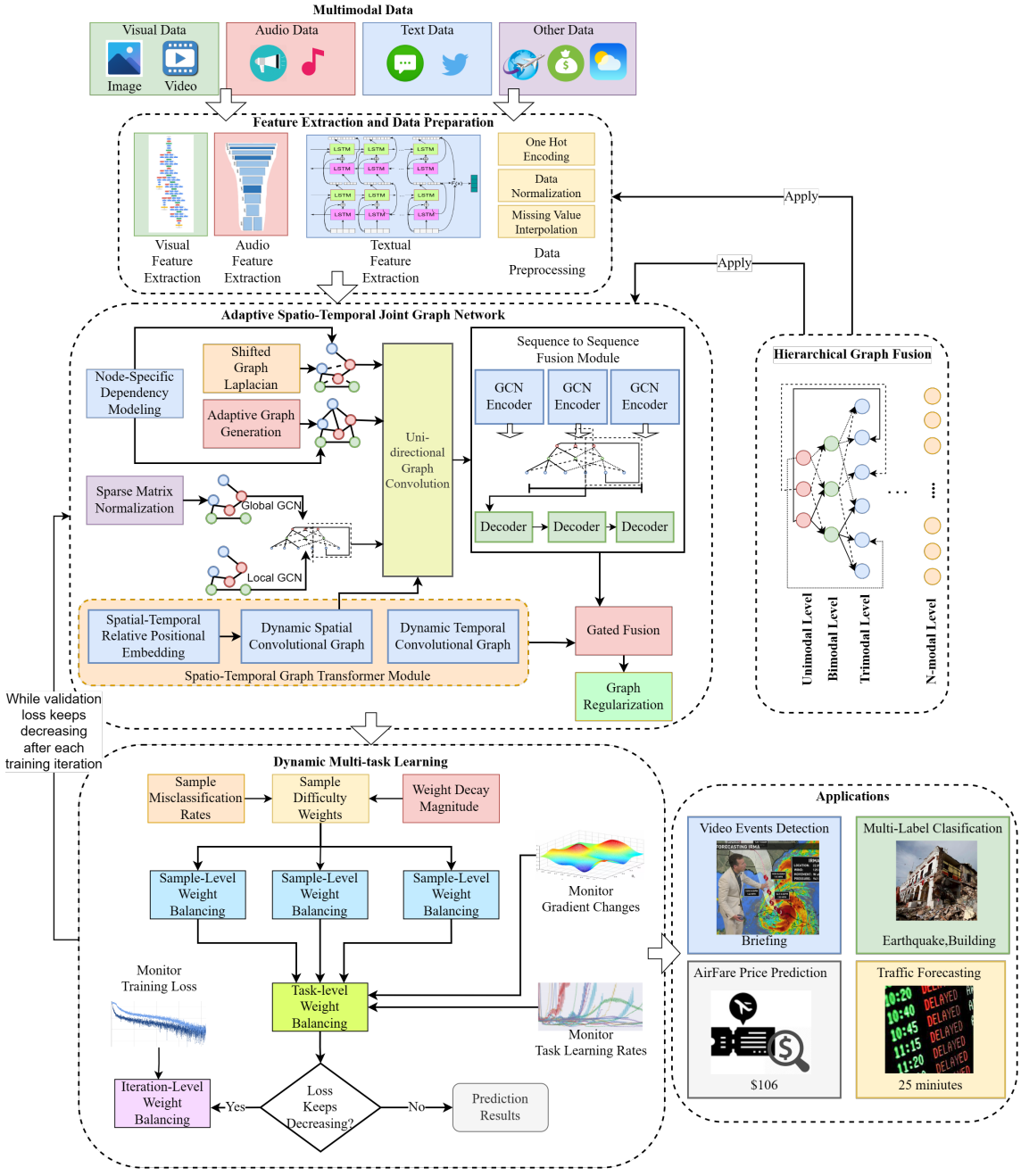


Figure 5: Overview of the dissertation's framework

at the subsequent level. In the proposed framework, this component can be applied in different stages during the training process:

- Early fusion [178], which combines the outputs of each domain-specific feature extraction network, helps the model learn the cross-modality correlation among all inputs from low-level features. In addition, early-stage fusion promotes the preservation of granular information that could otherwise be lost as it flows deeper into the network.

- Intermediate fusion where outputs generated by each convolutional graph are combined. Unlike the low-level features in the early stage, outputs from the different convolutional graphs represent high-level features, including the local spatial correlation, global property similarity, and other abstract patterns. Fusion applied at this stage assists the model in capturing the joint representation from different contexts in the graph networks.

The proposed hierarchical graph fusion aims to preserve the interactions among all possible combinations produced by each modality. A tree-based structure is applied to implement this method. The top-level branch consists of every single modality and is the building block for the entire graph. The nodes in the second level represent all bi-modal combinations, and the same method can be applied to build all n-modal combinations in deeper levels. The joint representation for each level is calculated as the weighted sum of all vertices. This work determines the weights by the similarity between the feature vectors forming the combination. The closer the two features in the vector space, the smaller

weights are carried since their interaction offers little information. The final output of the fusion network is the concatenation of combined features at each level.

3.3 Spatio-Temporal Graph Network

Large volumes of spatio-temporal data are being generated and analyzed daily from various domains, such as social media, remote sensing satellites, transportation and mobile GPS. The traditional approach for analyzing spatio-temporal data often utilizes CNN and RNN. However, conventional CNN can only be applied on grid-structured data, such as images and videos, and fails to capture the spatial relationship between individual objects measured via non-Euclidean distance [192]. Likewise, when using RNN-based methods for temporal feature extraction, information is frequently lost as hidden state information is passed down within the network. This work proposes an adaptive spatio-temporal graph network to model the complex spatial and temporal dependency in real-world problems. The proposed method contains the following main components:

- An effective sparse matrix normalization technique is developed to replace the regular normalized graph Laplacian adjacency matrix. It helps the graph retain essential node connections while keeping the adjacency matrix's sparsity to control the computational cost.
- Multiple GCNs are trained to utilize the pre-defined network topology, global feature similarity, and hidden dependencies between nodes.
- A static graph learning module with shifted graph Laplacian method to expand the

pre-defined graph topology so it can learn more trivial hidden patterns.

- An graph learning module with a data-driven adaptive graph generation approach that automatically learns the graph adjacency matrix. An adaptive graph regularization term is designed to enforce smoothness and sparsity in the learned adjacency matrix.
- A novel unidirectional graph convolution that can capture the information inflow and outflow in traffic forecasting problems.
- A BiLSTM-based sequence-to-sequence fusion model generates prediction results in multiple time steps. The attention mechanism is applied to generate the hidden context vector that contains the weighted sum of every hidden state in the encoder. It helps the model retain information when processing lengthy input sequences. In addition, the context vectors from multiple encoders or graphs are combined with leveraging the cross-modal interactions in modeling the temporal dependency.
- A spatio-temporal graph transformer model to jointly model the dynamic spatio-temporal dependencies. It consists of spatio-temporal relative positional encoding, which generates the time-evolving spatial embedding of the graph signals, and the dynamic spatial and temporal convolutional graphs that model the spatial and temporal node dependencies, respectively.

The proposed model first applies a shifted graph Laplacian approach to expand the sensitivity of the pre-defined graph. This way, the transformed graph Laplacian can capture more granular hidden patterns and still use the original graph structure's knowledge.

Second, an adaptive graph generation method and adaptive graph regularization are applied to automatically learn the network topology and control the sparsity and smoothness of the graph signals. Furthermore, a node-specific dependency modeling module modifies the original graph convolution operation so the model can learn node-specific patterns. The sequence-to-sequence fusion network encodes multiple graph layer signals in parallel and hierarchically combines them to retain the cross-modality interactions and capture the short-term temporal dependencies. Finally, the spatio-temporal graph transformer module complements the sequence-to-sequence fusion model by dynamically capturing the spatio-temporal dependencies in long-term predictions.

3.4 Dynamic Multi-Task Learning

Traditionally, machine learning is applied to solve a single task or optimize a single metric. Either one or more models in an ensemble are trained to do the needed task to achieve this aim. The hyperparameters and network structure are fine-tuned during the process to improve the model performance. While such an approach generally produces acceptable results, focusing exclusively on a single task may lead to the ignorance of crucial dependencies among related tasks that could help to enhance the model's performance. Multi-task learning exploits the characteristics and interactions across different learning objectives to help the model learn a better generalization of the original problem and reduces the risk of overfitting. In the proposed framework, the main parts of the multi-task learning method are as follows:

- Sample level weight balancing helps the model allocate more resources to difficult

cases. More specifically, as a sample is misclassified or produces a large error, the associated loss generated by the loss function is magnified to encourage the model to focus on these samples during the training process.

- Task level weight balancing utilizes the loss ratio between tasks as the metric to measure the task imbalances. The weight gradient from the first layer of the task-specific network is used to evaluate the current learning magnitudes. Therefore, the task-level loss function aims to minimize the difference between the weighted gradient of each task and the average gradient weighted by the training rate.

- After a complete training cycle, iteration level weight balancing adjusts the task loss weights. It employs a more aggressive loss updating procedure which helps the model quickly reach the optimal task loss weight and avoid the loss weights from falling into the local minimum or maximum.

The proposed dynamic multi-task learning method improves the model performance by balancing the training loss on different levels. It is also highly versatile and can be applied to other problem domains. For instance, a multi-label classification task can benefit from the dynamic multi-task learning approach by treating the classification of each label as a standalone task [182].

3.5 Datasets

3.5.1 CrisisMMD

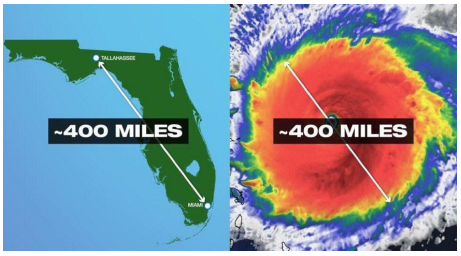
CrisisMMD [2] is a large scale public multimodal natural disaster dataset collected from Twitter. It contains 16,097 tweets and 18,126 images, each associated with at least one image. Seven natural disasters are included, specifically Hurricane Irma (2017), Hurricane Harvey (2017), Hurricane Maria (2017), Mexico Earthquake (2017), California Wildfire (2017), Iraq-Iran Earthquake (2017), and Sri Lanka Floods (2017). Figure 6 demonstrates some of the sample images from each disaster event. There are 3 main concepts: data informative indicator, humanitarian categories, and damage severity assessment. Samples are labeled as "Informative," "Not informative," or "Don't know or can't judge" based on the criterion of whether the given tweet or image is useful for humanitarian aid. Samples labeled as "Not informative" or "Don't know or can't judge" will not be further reviewed for other concepts. The humanitarian categories include concepts such as "Infrastructure and utility damage," "Vehicle damage," "Rescue, volunteering, or donation effort," "Injured or dead people," "Affected individuals," "Missing or found people," "Other relevant information," and "Not relevant or can't judge." The damage assessment concept is exclusive to the image. It labels the samples categorized as "Infrastructure and utility damage" as "Severe," "Mild," "Little or no damage," and "Don't know or can't judge."

CrissMMD has been used in many studies on disaster response and management using social media information [65, 94, 119]. A major challenge of this dataset is the multiple concepts/labels a single image-text pair can be assigned. The traditional machine learning model is designed to handle binary label classification and does not perform well

(a) California wild fire

(b) Hurricane Harvey

(c) Iran Earthquake



(d) Hurricane Irma

(e) Hurricane Maria

(f) Mexico Earthquake

(g) Sri Lanka Floods

Figure 6: Sample images of each disaster event in the CrisisMMD dataset

for multi-label classification tasks. Although some works have been done to transform the model or loss function so they can process multi-label data [53, 143, 159], the inter-relationship between concepts is mostly omitted.

In this dissertation, in order to utilize both text and image data to identify the humanitarian category concept, a multi-label multimodal classification task is included to predict the combined humanitarian category labels using both modalities. A similar multimodal approach is applied to the infrastructure damage level classification task, but we only did this for single-label classification.

3.5.2 Youtube Disaster Video Dataset

The Youtube Disaster Video Dataset is a natural disaster video dataset [141] collected from YouTube. It contains 1,540 video clips and seven concepts (shown in Figure 7) that are related to the 2017 hurricanes Harvey and Irma. The dataset contains seven complex semantic concepts: demonstration, emergency response, flood/storm, human relief, damage, victim, and speak/briefing/interview. The raw video footage is processed using keyframe extraction and audio track generation. Each video is labeled both at the frame level and video level. Human annotation is done on the keyframes, and a video-level concept is created based on the frame-level concept distribution.

One major challenge of this dataset is the class imbalance problem. Table 1 demonstrates the number of instances for each concept and the corresponding P/N ratio. The table shows that the number of instances is skewed across all concepts. Extreme data imbalance can create critical issues during the model training process since the small

Table 1: The statistical summary of the disaster video dataset

Concepts	Number of Instances	P/N Ratio
Demonstration	150	0.047
Emergency Response	338	0.105
Flood/Storm	971	0.301
Human Relief	273	0.085
Damage	371	0.115
Victim	311	0.096
Speak/Briefing/Interview	811	0.251
Total	3,225	

sample size in some concepts may cause the model to under-fit and makes poor predictions. In addition, it is challenging to properly harness the full potential of multimedia data due to their heterogeneous characteristics. Last but not least, this dataset also contains multiple labels on the same sample, which makes the prediction task even more challenging.

This dissertation utilizes the proposed hierarchical graph fusion to combine all modalities and applies a dynamic multi-task learning strategy to convert the multi-label classification task into a multi-task learning problem. The hierarchical graph fusion network could fully explore the inter-modality correlations among modalities by hierarchically modeling all modality-wise combinations. The dynamic multi-task learning approach automatically balances the learning progress of each task to benefit the minority concepts.

(a) demonstration

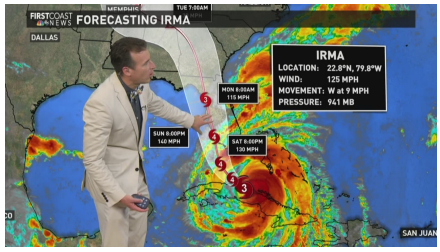
(b) emergency response

(c) flood and storm

(d) human relief

(e) damage

(f) victim



(g) briefing

Figure 7: Sample images of all concepts in the disaster video dataset

3.5.3 Airline Origin and Destination Survey (DB1B) and Air Carrier Statistics database (T-100)

The DB1B and T-100 datasets are published by the U.S. Bureau of Transportation Statistics (BTS). BTS regularly updates and releases both the DB1B and the T-100. The DB1B dataset provides quarterly-aggregated information about airline tickets in the United States from reporting carriers, and consists of 10% randomly sampled ticket data from each reporting carrier. The information in DB1B is organized into three parts, namely "Coupon," "Market," and "Ticket." A "Coupon" is an atomic unit of an airline ticket, indicating a passenger's itinerary that is directly transported from one airport to another. In contrast, each ticket could contain multiple coupons and multiple passengers. Therefore, "Coupon" in DB1B includes information about each leg, "Market" provides information on the market segment, such as the distance between two airports, and "Ticket" provides additional information at the ticket level, such as airfare price. All the records in "Coupon" are bounded to a "Market" record and a "Ticket" record.

Table 2 shows the layout of the ticket and coupon database tables and the sample

Table 2: Structure of the DB1B ticket and coupon table with sample records

DB1B Ticket Table							
ITIN_ID	QUARTER	ORIGIN	ITIN_FARE	DISTANCE	PAX	DOLLAR_CRED	-
2018112	1	ABE	340	1384	1	1	-
...	...	...	...	...	..	...	-
DB1B Coupon Table							
ITIN_ID	QUARTER	ORIGIN	DEST	REPORTING_CARRIER	PAX	SEAT_CLASS	DISTANCE
2018112	1	ABE	ATL	9E	1	X	692
2018112	1	ATL	ABE	9E	1	X	692
...	...	...	...	...	...	...	...

Table 3: Summary of data in DB1B and T-100 used in the proposed framework

Entity	Availability	Data
Ticket	DB1B	fare price, total distance, and total number of passengers
Coupon	DB1B	market segment, time of the itinerary, carrier, and seat class
Market segment	DB1B&T-100	original airport, destination airport, and segment distance
Market segment by carrier	T-100	number of passengers, and number of available seats by aircraft type

Table 4: Structure of the T-100 dataset with a sample record

SEATS	PAX	CARRIER	ORIGIN	DEST	QUARTER
150	140	9E	ABE	ATL	1
...	...	...	...	...	...

records with the same itinerary ID (ITIN_ID) in the DB1B dataset.

T-100, unlike DB1B, includes monthly domestic non-stop segment statistics from domestic and foreign airlines. It presents the number of passengers of each airline and each market segment by aircraft type. Table 4 shows an example record in the T-100 database, and Table 3 summarizes the information of these two datasets.

A significant challenge in this dataset is cleaning the data so the machine learning model can use them. Various data types and formats co-exist in the same record, including categorical, discrete, continuous, numerical, etc. A systematic data preprocessing pipeline is crucial in handling such heterogeneous data. Another challenge is to select the most relevant feature set so the input dimension can be controlled in a reasonable size and the

Table 5: Selected attributes and their corresponding data type in the BTS Reporting Carrier On-Time Performance data

Attribute Name	Data Type
Year	Categorical
Quarter	Categorical
Month	Categorical
Day of Month	Categorical
Day of Week	Categorical
Reporting Airline	Categorical
Origin Airport ID	Categorical
Destination Airport ID	Categorical
CRS Departure Time	Continuous
CRS Arrival Time	Continuous
Arrival Delay	Continuous
Departure Delay	Continuous
Distance	Continuous

model performance can be improved.

Our proposed work applies a comprehensive machine learning framework to handle the DB1B and T-100 datasets to predict the quarterly average airfare price. Data preprocessing has been applied, including data cleaning, transformation, and feature engineering. A feature selection approach is developed to choose the most optimal features that help the model achieve the best performance. The hierarchical graph fusion method also combines each data source, which exploits their inter-modality interactions.

3.5.4 Reporting Carrier On-Time Performance Data

BTS also publishes the Reporting Carrier On-Time Performance Data, which contains comprehensive flight record data from over 20 domestic airlines and 400 airports in the United States. Each flight record contains a specific flight date and time, origin and destination airports, scheduled arrival/departure time, actual arrival/departure time, and other information. Table 5 shows the selected attributes and their corresponding data type used in this study. The research community has widely used this dataset to study flight delay patterns and causes.

One major challenge when using this dataset for flight delay prediction problems is properly learning the traffic patterns from hundreds of airports of various sizes and thousands of flight routes that cover distinctive locations. The spatio-temporal nature of the flight record data requires elaborately designed methods to exploit fully.

Our proposed adaptive spatio-temporal graph network utilizes this dataset to construct several convolutional graphs that model existing network connectivity, correlation among properties shared with all airports and hidden patterns that the pre-defined network topology cannot explain.

3.5.5 PeMSD Datasets

The PeMSD datasets are collected and published by the California CalTrans Group's Performance Measurement System (PeMS). The data is collected from all major metropolitan areas in California using over 40,000 sensors. In this dissertation, we use the PeMS4 and PeMS8 datasets.

- **PeMSD4**. This dataset includes historical traffic condition data in the San Francisco Bay area. The data is collected in 5-minute intervals using 307 sensors on seven major roads. The period covered by PeMSD4 ranges from January 2018 to February 2018. Three types of measurements are used, which include average speed, average occupancy and traffic flow. In this study, we are focusing on traffic flow and traffic speed prediction.

- **PeMSD8**. This dataset contains the traffic information in the San Bernardino area from July 2016 to August 2016. The 170 sensors used 5 minutes intervals on eight roads to collect the average speed, average occupancy and traffic flow information. Same as PeMSD4, we are targeting traffic flow and traffic speed prediction in this study.

CHAPTER 4

DYNAMIC MULTI-TASK LEARNING

Typical machine learning/deep learning training strategies focus exclusively on a single task. However, solving multiple using the same model through a single training pass could yield several benefits. For instance, performing multiple inferences in a single forward pass reduces memory consumption and speeds up inferencing. As a result, multi-task learning has been adopted to help improve the learning process of one objective by simultaneously learning other related tasks. When training with a multi-task learning strategy, the model optimizes the network parameters based on the aggregated loss from each objective. More specifically, the total loss can be considered a linear weighted sum of the losses from each task. As a result, a significant challenge in multi-task learning is determining the appropriate weight to assign to each task to balance the training process across all objectives. Equal weight assignment and manual weight allocation are the two most applied strategies. A more straightforward task will almost certainly degrade the overall model performance with equal weights assignment, as the slight loss reduces the gradient of the aggregated loss during backpropagation. On the other hand, manual fine-tuning requires considerable effort to determine the proper weighting and extensive domain knowledge.

This chapter proposes a novel dynamic multi-task learning approach to address the challenge of optimizing the model training process in a multi-task learning scenario [177,

182]. The dynamic task balancing approach automatically adjusts the training progress at the sample and task levels. By allocating additional resources to difficult instances, the sample-level dynamic balancing function amplifies the loss generated by these samples. Simultaneously, the task-level dynamic balancing mechanism adjusts weight distribution based on each task's training rate. Additionally, a loss weighting method named automatic loss weighting is proposed to automatically adjust the weighted scalar for each task after each training iteration. The loss for each task is tuned to a similar scale. As a result, the model can begin the subsequent training iteration with a more balanced loss distribution.

4.1 Automatic Loss Weighting

Due to the increase in human activities and the expansion of human habitat, the damages caused by natural disasters have been dramatically increased [91]. This poses a significant challenge for disaster response management, demanding a more efficient and effective emergency response and recovery plan. To make accurate evaluations of human injury, death, and property damages, the responders have to collect and analyze large amounts of data promptly. However, the traditional information gathering channels lack the capability of quick information delivery as well as large areas of coverage [86].

The rise of social media platforms has dramatically changed the way of sharing and acquiring information. In combination with the prevalence of smartphones, eyewitnesses can produce content right at the disaster scene. During disastrous events, it can provide valuable situation updates, which help the emergency response personnel to make faster and more accurate decisions. With the help of artificial intelligence (AI) techniques such

as machine learning and deep learning, the emergency response team can analyze the social media data in real time to determine the impact in different regions and prioritize the resource distribution accordingly. Many studies have been conducted to utilize social media data and deep learning methods to identify humanitarian activities, assess infrastructure damages and enhance emergency awareness [36] [100] [189].

Regardless of the applications and methods used, many existing disaster situation assessment studies focus on a single task and use a single data modality. One major limitation of such approaches is that the shared representations among different tasks and input modalities are not fully utilized. To address such challenges, an automatic loss weighting method is first introduced. The proposed method solves multiple tasks on a single model with a better generalization ability on the problem. The model can automatically adjust the loss weights for each task to improve performance. Moreover, automatic loss weighting is more efficient and avoids the need for tedious manual weight tuning. We demonstrate the effectiveness of our proposed loss weighting method and compare its performance with models trained without loss weighting. Figure 8 shows the architecture of the proposed framework, which includes an image module, a text module, and a multi-task classification module.

4.1.1 Image Feature Extraction

The VGG-19 [156] model pre-trained on ImageNet [39] is used to extract the low-level features from the imagery data. ImageNet is a large-scale imagery collection dataset that contains over 14 million images and 20,000 categories. In this case, the

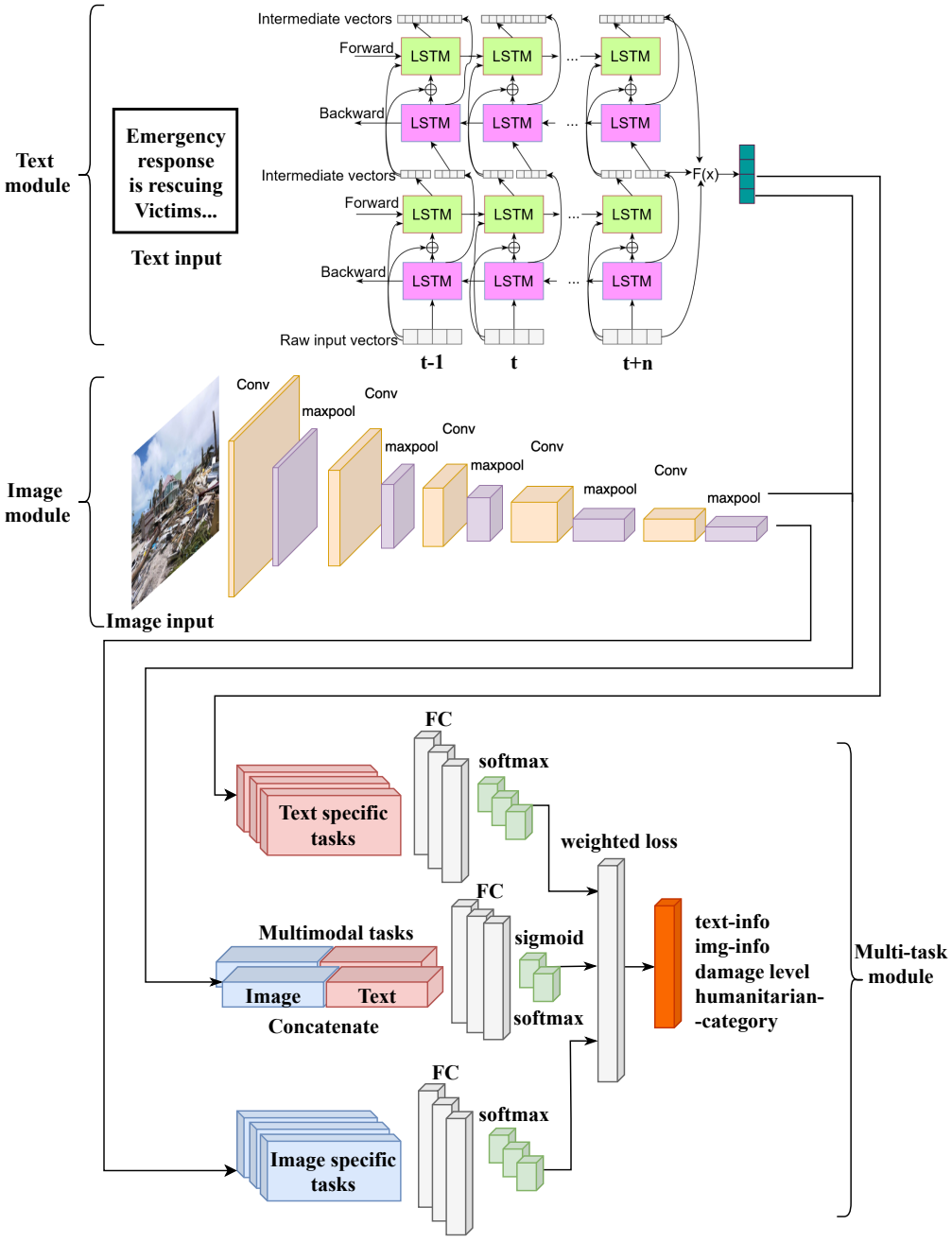


Figure 8: Illustration of the automatic loss weighting framework for disaster damage assessment based on social media data

main advantage of transfer learning is the benefit of adopting the learned knowledge from ImageNet to our problem domain. The pre-trained VGG-19 model serves as a feature extractor to generate the low-level features, which the rest of the layers can use in the model. We remove the last 2 fully connected layers in the original VGG-19 model and obtain the intermediate results from the last convolutional layer. During the training process, the weights of all the layers are kept fixed in the pre-trained model and fine-tuned on the rest of the fully connected layers. The image module in Figure 8 demonstrates the detailed network structure used in our framework.

4.1.2 Texture Feature Extraction

Traditional word embedding methods like Glove [137] and Word2vec [128] do not consider the underlying semantic meaning of the entire text corpus, and they do not perform well on out of vocabulary words. In this paper, the proposed text embedding module is built based on the work of Peters *et al.* [139]. First, each character in the text corpus is tokenized, and a temporal convolutional neural network (CNN) [203] is used to generate the CNN character embedding. The temporal convolutional module computes the 1-D convolution on the input data and generates the raw word vector of the input token. This process is demonstrated in the text module according to Figure 8. This helps the model capture the morphological features the word-level embedding does not contain. Also, by combining the vector of each character, the vector representation for the unseen words can be formed. Second, the raw text vector will be fed into two consecutive bi-directional long-short-term machines (BiLSTMs). The forward and backward passes of the BiLSTM

layer will learn the context information before and after the target word. The first BiLSTM layer will generate a set of intermediate vectors and send them into the next BiLSTM. In addition, a skip connection is added between the two BiLSTM layers. Finally, the weighted sum of the raw input vectors and the two intermediate vectors is calculated to form the final representation of the target word (as shown in Equation (4.1)).

$$V_k = \beta_k \cdot (s_0 \cdot x_k + s_1 \cdot h_k^1 + s_2 \cdot h_k^2), \tag{4.1}$$

where V_k is the final word representation, β_k is a task-specific scaling factor to help the model optimization, s_i is the softmax-normalized weights, x_k is the raw input vector for word k, h_k^1 and h_k^2 is the intermediate vectors generated by the two BiLSTM layers.

4.1.3 Feature Fusion

In many tasks, data from different modalities could provide complementary information that helps better generalize the problem. Multimodal learning is utilized by fusing the early output from the text and image modules. The intuition of this approach is to preserve the semantic correlations from each data source. Features generated by the text and image modules are flattened and concatenated to form a joint vector.

4.1.4 Multi-task Learning

Multi-task learning is used to improve humanitarian activity and infrastructure damage classification tasks. Multi-task learning is best suited for problems that contain related tasks. By sharing the representations among all tasks, our model can learn a better generalization across the problem domain. This paper analyzes the humanitarian activities

and infrastructure damage levels using text and image data from the same Twitter tweet. When the text and image data appear together, they tend to carry shared information or can complement each other. Therefore, tasks involving text and image data can be related. In Figure 8, the multi-task module implements the hard-parameter sharing methodology, in which all tasks share the same set of hidden layers. As a result, our model becomes more resilient to overfitting because the network is forced to learn multiple tasks simultaneously.

In multi-task learning, the loss function is defined in Equation (4.2).

$$L_{total}(x; W) = \sum_{i=1}^{N} w_i L_i, \tag{4.2}$$

where L_{total} is the final loss that the model tries to optimize, x and W are the input and weight parameter to the model, L_i and w_i are the loss and corresponding weight scalar for task i, and N is the total number of tasks. By default, the loss weights are uniformly assigned, which means all tasks are weighted at the same scale. However, this may cause easier tasks to dominate the learning process. Also, when tasks use different loss functions, the loss for each task can significantly differ due to how each function is mathematically defined.

This paper proposes a novel loss weighting method that automatically adjusts the weighted scalar for each task. Our goal is to adjust the loss of each task to a similar scale so that they can be optimized more equally during the training. Our proposed method uses the mean training loss to calculate the loss weights. The rationale is that tasks with a higher total loss should also significantly impact the final weighted sum loss. The model can obtain a more balanced loss distribution at the beginning of the following training iteration by suppressing these tasks. The details are illustrated in Algorithm 1. In each iteration,

Algorithm 1 The proposed loss weighting algorithm

Input: Training dataset D_t and validation dataset D_v

$J^0 \leftarrow \infty, t \leftarrow 1$

$\hat{L}^t \leftarrow [1, 1, ..., 1]$

$J^1, \{L^t_{i,j}\} \leftarrow \text{ModelTrain}(\hat{L}^t, D_t, D_v)$

while $J^t < J^{t-1}$ **do**

  $t \leftarrow t + 1$

  for $i = 1, 2, \ldots, |\mathbb{L}|$ **do**

    $\overline{L}^t_i \leftarrow \frac{\sum_{j=1}^{|E|} L^{t-1}_{i,j}}{|E|}$

  end for

  for $i = 1, 2, ..., |\mathbb{L}|$ **do**

    $\hat{L}^t_i \leftarrow \frac{\max_i(\overline{L}^t_i)}{\overline{L}^t_i}$

  end for

  $\hat{L} \leftarrow [\hat{L}^t_1, \hat{L}^t_2, \ldots, \hat{L}^t_{|\mathbb{L}|}]$

  $J^t \leftarrow \text{ModelTrain}(\hat{L}^t, D_t, D_v)$

end while

t, the training loss $L_{i,j}^t$ of task i at epoch j is averaged to generate the mean training loss $\overline{L}_i^t$ over all epochs. Then, the weighted loss $\hat{L}_i^t$ for each task is calculated by dividing the maximum of $\overline{L}^t$ using the mean loss of each task $\overline{L}^t$. As a result, we can acquire a loss weight scalar for all tasks after the model has converged. In the next iteration, the model will be trained with the new loss weight setting. Specifically, all tasks will be assigned a uniform loss weight in the first iteration. We keep track of the validation loss J^t generated by the training process ($\mathrm{ModelTrain}()$) at iteration t. The iteration will automatically terminate when the validation loss stops improving. In practice, we also set a tolerance factor that allows the model to keep training even when the loss does not improve above a threshold for a certain number of iterations.

4.1.5 Experiments and Analysis

4.1.5.1 Dataset Description

To test the proposed framework, the CrisisMMD dataset [2] is used. This dataset contains 16,097 tweets and 18,126 images, each associated with at least one image. Seven natural disasters are included, specifically Hurricane Irma (2017), Hurricane Harvey (2017), Hurricane Maria (2017), Mexico Earthquake (2017), California Wildfire (2017), Iraq-Iran Earthquake (2017), and Sri Lanka Floods (2017).

There are 3 main concepts: data informative indicator, humanitarian categories, and damage severity assessment. Samples are labeled as “Informative”, “Not informative”, or “Don’t know or can’t judge” based on the criterion of whether the given tweet or image is helpful for humanitarian aid. Samples labeled as “Not informative” or “Don’t

know or can't judge" will not be further reviewed for other concepts. The humanitarian categories include concepts such as "Infrastructure and utility damage", "Vehicle damage", "Rescue, volunteering, or donation effort", "Injured or dead people", "Affected individuals", "Missing or found people", "Other relevant information", and "Not relevant or can't judge". The damage assessment concept is exclusive to the image. It labels the samples categorized as "Infrastructure and utility damage" as "Severe", "Mild", "Little or no damage", and "Don't know or can't judge". To utilize text and image data for identifying the humanitarian category, a multi-label multimodal classification task is included to predict the combined humanitarian category labels from the two modalities. A similar multimodal approach is applied to the infrastructure damage level classification task, but we only did this for single-label classification.

4.1.5.2 Experimental Setup

The dataset is randomly split into 60% for training, 20% for validation, and 20% for testing. The validation set is used to tune the model hyperparameters. Stop words and URLs are removed from the text data. For the image data, the images are resized and cropped to 224 by 224 pixels using bilinear interpolation. Categorical cross-entropy is the loss function for all single-modality classification tasks, and binary cross-entropy is used for the multi-label classification task. Adam algorithm is chosen as the optimizer, and the initial learning rate is set to 0.001. Each fully connected layer uses ReLu as the activation function with 50% dropout. Regarding the last fully connected layer, softmax is used for the single-modal tasks, and sigmoid is used for the multimodal tasks. The model is trained

using early stopping with a batch size of 256. Precision, recall, F1 measure, and Hamming loss (HL) are used as the evaluation metrics. Our proposed multi-task multimodal model is trained on text informative classification, image informative classification, multimodal-multilabel humanitarian category classification, and multimodal infrastructure damage classification tasks.

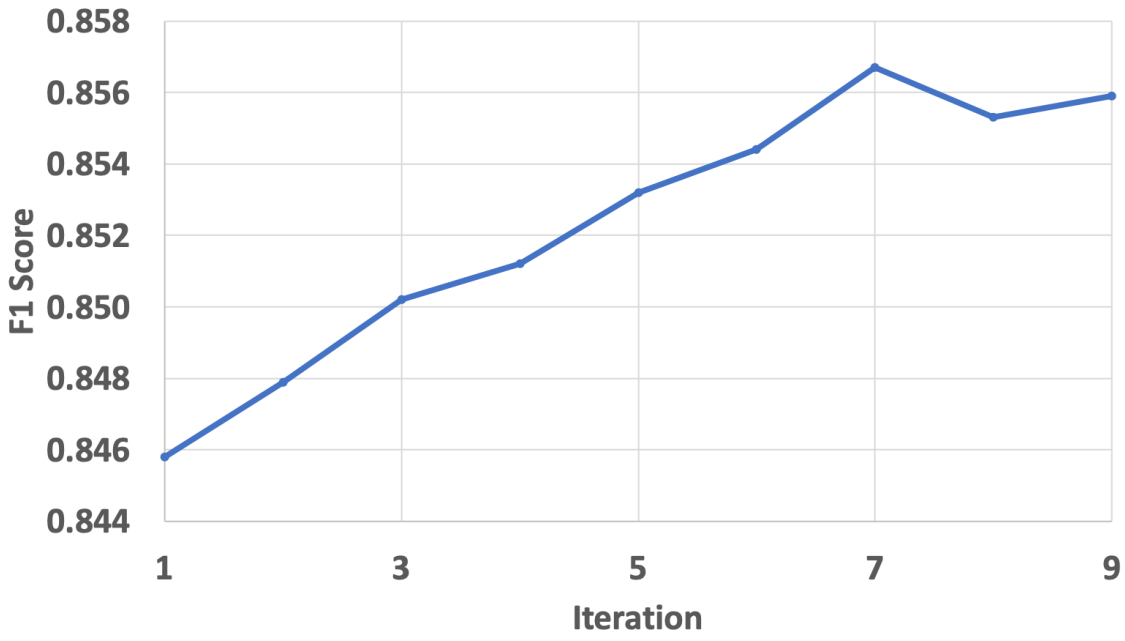


Figure 9: Multimodal damage classification results using the loss weighting algorithm

4.1.5.3 Experimental Results

To demonstrate the effectiveness of our proposed model, it is compared with several baselines. Each baseline is a model trained on a specific task only. They include models trained on informative, humanitarian, and damage classification tasks using either text or image. Additionally, two baseline models trained on humanitarian and damage tasks using

Table 6: Performance comparison between single tasks and our method. Columns T (text) and I (image) indicate the involved modality. Column O indicates the applied task, in which "i" is informative classification, "h" is humanitarian classification, and "d" is damage classification. Column P gives the results of our proposed model.

P	O	T	I	Precision	Recall	F1	HL
	i	x		0.81	0.81	0.81	0.12
x	i	x		**0.83**	**0.83**	**0.83**	**0.11**
	i		x	0.66	0.62	0.64	0.22
x	i		x	**0.81**	**0.81**	**0.81**	**0.13**
	h	x		0.72	0.57	0.63	0.08
	h		x	0.74	0.27	0.39	0.09
	h	x	x	0.72	0.57	0.63	0.11
x	h	x	x	**0.73**	**0.58**	**0.64**	**0.10**
	d		x	0.82	0.79	0.80	0.05
	d	x	x	0.85	0.80	0.82	0.05
x	d	x	x	**0.88**	**0.82**	**0.86**	**0.05**

image and text are added to compare the effectiveness of multimodal learning. Table 6 shows the performance comparison between the baselines and our proposed multi-task multimodal model. The models are grouped based on the task, and their performance is presented. In the first two groups of comparisons, our approach outperforms every baseline model on their corresponding tasks. These results illustrate the effectiveness of multi-task learning in disaster situation assessment applications. In the following two groups of comparisons, all multimodal tasks achieved better performance when compared to their corresponding single-modality tasks. This shows that multimodal learning can leverage the complementary information from the text and image inputs. Furthermore, the performance of the proposed loss weighting method is also evaluated. Figure 9 shows the F1 score of the multimodal damage level classification task at each iteration. After applying the loss weighting approach, the target task has shown a gradually improved F1 score from 0.846 to 0.857, illustrating its effectiveness. Note that the scores started to oscillate after 7 iterations, mainly caused by the model becoming saturated.

A 5-point sensitivity analysis test is conducted on the multimodal damage classification task to further investigate the effects of the loss weighting method. This test aims to observe the task performance changes under different loss weight settings. The loss weight for the damage task is set to 10, 5, 1, 0.2, and 0.1 during the 5 runs. On the other hand, the loss weights of the other 3 tasks are set to 1 and did not change during the test. The results are shown in Figure 10. It can be observed that the performance of the damage task is consistently better with a higher loss weight, which is not surprising. On the contrary, the other 3 tasks get a slight performance boost when the loss weight of the damage task

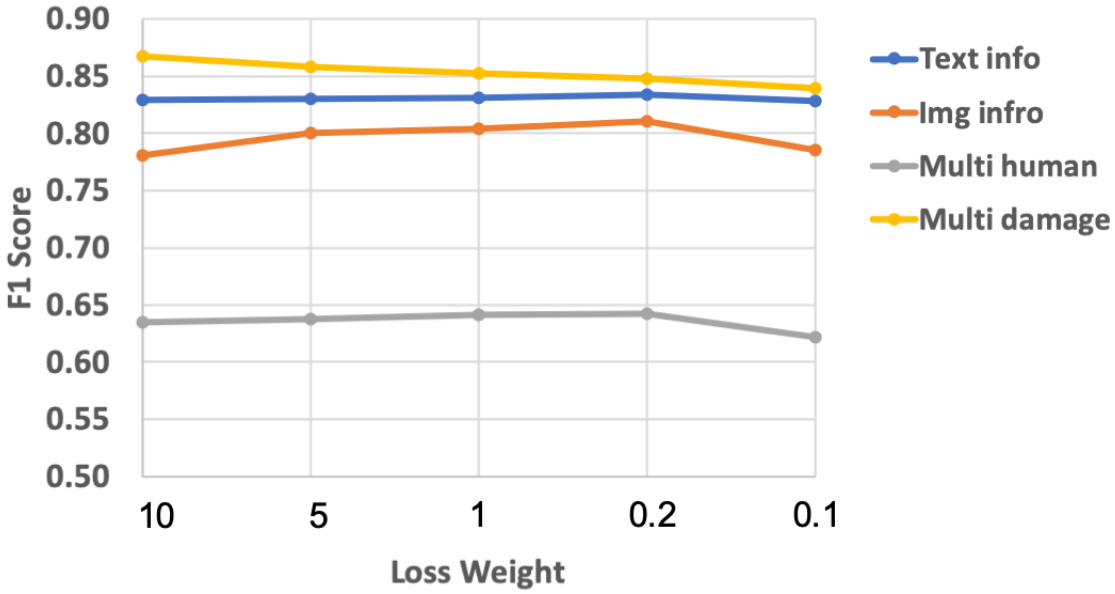


Figure 10: Performance comparison of all tasks when different loss weight settings for the multimodal damage level classification task

decreases. However, their performance starts to drop when the weight of the damage task becomes too small (0.1). The initial performance improvement can be explained by the less loss domination from damage task to the total loss so that the model can generalize better on the entire problem domain. In the end, when the loss weight of the damage task is set to a very small value, the model can barely learn from this task, which causes the overall performance to be degraded.

4.1.6 conclusion

This work introduces a novel automatic loss weighting deep learning framework based on multi-task and multimodal learning for social media disaster situation assessment. The proposed model is evaluated on a multimedia natural disaster dataset collected from

Twitter. The experimental results demonstrated that multi-task multimodal learning could improve the model performance by simultaneously learning the related tasks. Moreover, the proposed automatic loss weighting method can further improve the model performance.

4.2 Dynamic Task Balancing

In a real-world scenario, visual data often contain rich information describing a specific environment containing multiple objects and their interactions [28]. The classic single-label classification deep neural networks are designed to detect the existence of a single object or action. Therefore, to model the rich semantic information in visual data, deep neural networks should be adapted to model multiple objects. The specific task that aims to accomplish this goal is named multi-label classification. One important property that distinguishes multi-label classification from the standard multi-class classification is that the labels in multi-label classification are not mutually exclusive. Recently, multi-label classification has attracted attention on a wide variety of domains [175] [141].

Cost functions such as ranking loss, cross-entropy, and mean-squared error loss commonly used in single-label classification tasks cannot be directly applied to multi-label classification problems. A widely adopted approach is to transform the problem into a single-label classification. A classic method is one-vs-rest or one-vs-all. One-vs-rest splits the multi-label classification problem into several binary classification subproblems. Each subproblem has a classifier trained on one of the single labels. Then, the final prediction result is the ensemble of the output from all classifiers [12]. However, one

major disadvantage of the native one-vs-rest method is disregarding inter-label dependency. It is well-studied that strong co-occurrence exists in most multi-label classification tasks [57] [126]. A prevalent case can be observed in natural disaster images, where victims and building debris frequently appear together. Many recent studies focus on learning the underlying correlation among labels so that the inter-label dependency can be retrieved [214] [188]. Furthermore, to infer the joint label probability from the latent space, the final loss functions should be able to assess the optimal confidence thresholds for separating each label [101]. However, such approaches require precisely formulated modeling of the co-occurrence dependencies between labels, which can vary extensively among tasks and input sources [207]. Moreover, inevitable trade-offs still need to be made between the model complexity and the training time since the pair-wise correlation strategy adopted in these studies inevitably creates a large number of parameters [183].

This section proposes a novel multi-label multi-task attention network (MTMLAN) that utilizes the temporal and spatial information from the input data for video information retrieval tasks. An attention mechanism is applied to facilitate the training process by putting more weight on a specific segment of the input sequence. We applied a novel dynamic weighting method that can automatically adjust the task weights based on the sample and task-level learning complexity to address the task weighting problem challenge. We evaluated our framework on a multi-label natural disaster video dataset, which can be expanded to almost any domain. The dynamic weighting method can be applied to all deep neural networks, greatly expanding its usability. The key contributions of this work are:

- A novel deep learning framework MTMLAN that utilizes multi-task learning to

solve multi-label video information retrieval tasks.

- A novel dynamic task balancing method for multi-task learning problems based on the sample and task-level learning complexities.

The proposed dynamic task balancing could be applied to any deep neural network (DNN) and problem domains. In this work, we constructed recurrent neural networks (RNNs) based DNN named multi-task multi-label attention network (MTMLAN) for demonstration.

4.2.1 Architecture Design

The main architecture of MTMLAN follows the hard parameter sharing schema [150]. It consists of two main components: shared and task-specific networks. Figure 13 shows the overview of our proposed framework. The shared network learns the shared feature representation of all tasks, which can significantly reduce the risk of overfitting. In this paper, we construct the shared network based on the Inception V3 [164] model. The network consists of multiple small convolutional filters (3×3), and a batch normalized fully connected layer of the auxiliary classifier. The original Inception V3 is truncated after the last average pooling layer to generate the spatial features.

The outputs of the shared network are then fed into each task-specific network. The task-specific network contains two bidirectional gated recurrent units (BiGRU) and an attention module. The BiGRUs extract temporal information from the sequential video frames. The attention module enables task-specific networks to learn task-specific features. In other work, it functions as a feature selection mechanism by helping the

model focus on the most relevant part of the input sequence. In this paper, the attention module is implemented based on self-attention [34], a particular variant of the attention mechanism. While in regular attention, the model looks at multiple sequence inputs at adjacent time steps to determine the weight to put on each location [13], self-attention helps the model to focus at different locations of the current input sequence to get a more in-depth representation of the subject matter.

We denote the input sequence as I and the number of features in I as n. Then, the input vector can be described as:

$$I = (\omega_1, \omega_2, ..., \omega_n) \tag{4.3}$$

where I is the output of the shared network and ω_i is the i_{th} feature for an n-dimensional input vector. Then, the BiGRUs takes the input sequence I and generates the hidden state h_m for feature m:

$$\overrightarrow{h_m} = \overrightarrow{GRU}(\omega_m, \overrightarrow{h_{m-1}}) \tag{4.4}$$

$$\overleftarrow{h_m} = \overleftarrow{GRU}(\omega_m, \overleftarrow{h_{m-1}}) \tag{4.5}$$

The vectorized hidden state H is a constituent of each feature-level hidden state h_i, which is the concatenation of the feature-level hidden states from the two unidirectional GRUs:

$$H = (h_1, h_2, ...h_n) \tag{4.6}$$

Then, the attention weight matrix M is calculated by using the hidden state vector H:

$$M = softmax(W_{s2}tanh(W_{s1}H)) \tag{4.7}$$

where W_{s1} and W_{s2} are two trainable weight matrices, $tanh()$ represents the hyperbolic tangent function. The softmax function ensures the sum of the weight matrix is 1.

The attention weight matrix helps the model to focus on a specific location on the input sequence by assigning corresponding weights to each feature. To get the final weighted output A, we apply the attention weight matrix to the hidden state vector. Here the matrix multiplication operation is used:

$$A = MH \tag{4.8}$$

The weighted output is then fed into the last step of the task-specific network, which is the final fully connected layer.

4.2.2 Dynamic task balancing

Multi-task learning requires carefully balancing the training progress between tasks. The proposed dynamic task balancing method comprises two components: sample-level dynamic balancing and task-level dynamic balancing.

4.2.2.1 Sample-level dynamic balancing

The traditional solution for the class imbalance problem is to assign a penalty factor to the majority class in the loss function. While effective, this method only considers the problem at a class level. The truth is sample difficulty also has a substantial impact on the learning process. For instance, the cross-entropy (CE) loss function for binary classification tasks can be described as:

$$CE(p_k) = -\log(p_k) \tag{4.9}$$

where

$$CE(p_k) = \begin{cases} p, & if\ y = 1 \\ 1 - p, & otherwise \end{cases} \tag{4.10}$$

where $y \in \{-1, 1\}$ is the ground-truth label, $p \in [0, 1]$ represents the probability that the target class has label $y = 1$. Based on [104], we define the sample-level loss function $SL()$ as:

$$SL(p_k) = -(1 - p_k)^{\beta} \log(p_k) \tag{4.11}$$

where α_t is β is the sample level focusing parameter.

As the sample is misclassified and p_k is small, $(1-p_k)^{\beta}$ is very close to 1. Therefore, the loss is not affected. In comparison, as p_k gradually turns to 1, the impact on loss will increase, which means the weight for correctly classified samples decreases. β controls the magnitude of how the weight of the easy samples decreases. As a result, the sample-level dynamic balancing method effectively helps the model to adjust the resources to difficult samples.

4.2.2.2 Task-level dynamic balancing

One of the most prominent issues with multi-task learning is finding suitable weights for each task so the weighted linear sum of all losses could be optimized. Inspired by [32], we propose a novel task-level dynamic balancing (TDB) method that is capable of handling the task imbalance problem. TDB uses the loss ratio between tasks as the metric to measure task imbalances. The weight gradient from the first layer of the task-specific network is used to evaluate the current learning magnitudes. Therefore, the goal of the

task-level loss function $TL(t)$ is to minimize the difference between the weighted gradient of each task and the average gradient weighted by the training rate.

The task-level losses $TL(t)$ at training step t is defined as:

$$TL(t) = \sum_j \frac{N}{n_j} \left| G_W^{(j)}(t) - \overline{G}_W(t) \times \left[r_j(t)^\theta \right] \right|_1 \tag{4.12}$$

where N is the total number of training instances, n_j is the number of instances in task j. $\frac{N}{n_j}$ is the inverted class/task distribution, which serves as a penalty term to suppress the majority class/task. W contains weight parameters from the last layer of the shared network, θ is a hyperparameter that governs how rapidly the training rate will be restored to the average scale, $G_W^{(j)}(t)$ represents the L_2 norm of the gradient of the weighted single-task loss $w_j(t)L_j(t)$ for task j for the chosen weights W:

$$G_W^{(j)}(t) = \left\| \nabla_W w_j(t) L_j(t) \right\|_2 \tag{4.13}$$

$\overline{G}_W(t)$ defines the average gradient norm among all tasks T at time step t:

$$\overline{G}_W(t) = E_T \left[G_W^{(j)}(t) \right] \tag{4.14}$$

The relative inverse training rate of task j $r_j(t)$ is defined as:

$$r_j(t) = \frac{\hat{L}_j(t)}{E_T \left[\hat{L}_j(t) \right]} \tag{4.15}$$

where $\hat{L}_j(t)$ is the loss ratio for task j at time step t to time step 0:

$$\hat{L}_j(t) = \frac{L_j(t)}{L_j(0)} \tag{4.16}$$

After upgrading the weight parameters in the training steps, the task losses are normalized so that the global training rate will not affect the gradient. The task weight for the next

training set is then defined as:

$$w_j(t+1) = \lambda(t) w_j(t+1) \tag{4.17}$$

where

$$\lambda(t+1) = \frac{T}{\sum_j w_j(t+1)} \tag{4.18}$$

The steps to implement TDB for each training step can be described as 1) Perform a forward pass at the beginning of each training step, 2) extract the gradients of the first layer in each one of the task-specific networks G_W^j, and their corresponding L_2 norms are calculated, 3) calculates the average gradient $\overline{G}_W(t)$, 4) calculate the relative loss $\hat{t}$ for each task, 5) calculate the relative inverse training rates $r_j(t)$ for each task, 6) calculate the $\overline{G}_W(t) \times \left[r_j(t)^\theta\right]$ in equation 10, 7) calculate the gradient loss $TL(t)$, 8) update the task loss weights from $w_j(t) \rightarrow w_j(t+1)$, 9) update the model weights $W(t) \rightarrow W(t+1)$, 10) re-normalize the task loss weights $w_j(t+1)$

4.2.3 Experiments and Analysis

4.2.3.1 Dataset

In this work, we used a natural disaster video dataset [141] collected from YouTube. It contains 1,540 video clips and seven concepts (shown in Figure 7) related to the 2017 hurricanes Harvey and Irma. Following our previous work [140], each video clip is sub-sampled to 40 frames.

Table 7: The statistical summary of the disaster video dataset

Concepts	Number of Instances	P/N Ratio
Demonstration	150	0.047
Emergency Response	338	0.105
Flood/Storm	971	0.301
Human Relief	273	0.085
Damage	371	0.115
Victim	311	0.096
Speak/Briefing/Interview	811	0.251
Total	3,225	

4.2.3.2 Experimental setup

The dataset is randomly split into 60% for training, 20% for validation, and 20% for testing. All hyperparameters are tuned on the validation set. The Inception V3-based shared-network is pre-trained on ImageNet [38] and the output of the last average pooling layer is used as the input for the task-specific networks. The proposed sample-level balancing loss function is used for each task-specific network, and the task-level balancing loss function is used on the final aggregated loss. Based on our empirical study, setting the hyperparameter α in the task-level loss function to 1 returns the best results. During the training, a batch size of 20 is used for the input. The learning rate is set to 0.001

Table 8: The per-concept accuracy results on the disaster video dataset

Approach	Demonstration	Emergency Response	Flood/Storm	Human Relief	Damage	Victim	Briefing
CMLC	0.8136	0.8066	0.8466	0.8123	0.7566	0.7452	0.8574
EWMTC	0.8249	0.8516	0.8779	0.8346	0.8010	0.7947	0.8552
GradNorm	0.8469	0.8711	0.9024	0.8753	0.8719	0.8540	0.8807
WU	0.8441	0.8597	0.9108	0.8776	0.8697	0.8600	0.8791
MTMLAN w/o SL Balancing	0.8740	0.9124	0.9137	0.9145	0.8913	0.8654	0.8985
MTMLAN w/o TIP	0.8948	0.9116	0.9194	0.9159	0.9083	0.8712	0.9001
MTMLAN	**0.9335**	**0.9487**	**0.9331**	**0.9410**	**0.9215**	**0.8833**	**0.9221**

and Adam [88] is used as the optimizer during the training. We report the results in Micro Averaged F-measure (MicroF1), Hamming Loss (HL), and Mean Average Precision (MAP).

4.2.3.3 Experimental Results

To demonstrate the effectiveness of our approach, several baseline methods are also tested on the disaster video dataset: 1) A common multi-label classification model (CMLC). It has a similar network structure as the proposed MTMLAN before the task-specific networks. This baseline model replaces the task-specific networks with a single 2-layer Bidirectional GRU. The sigmoid activation function is applied on the last fully connected layer, and cross-entropy is used as the loss function; 2) A equal weight multi-task classification model (EWMTC). It has the same network structure as MTMLAN

Table 9: Performance evaluation results on the disaster video dataset

Approach	Weight Balancing	MicroF1	HL	MAP
CMLC	N/A	0.7267	0.1277	0.6848
EWMTC	Equal task weight	0.8015	0.1129	0.7341
GradNorm	Task-level	0.8569	0.0788	0.7822
WU	Task-level	0.8441	0.0793	0.7463
MTMLAN w/o SL Balancing	Task-level	0.8740	0.0661	0.8233
MTMLAN w/o TIP	Sample-level & task-level	0.8889	0.0634	0.8245
MTMLAN	Sample-level & task-level	**0.9135**	**0.0512**	**0.8559**

without the sample-level and task-level dynamic balancing mechanism. Therefore, the final loss is simply the equal weight linear sum of all task losses, 3) GradNorm [32] is applied on MTMLAN to replace the proposed sample-level and task-level dynamic balancing mechanism, 4) Weight Uncertainty (WU) [85] method. The sample-level and task-level dynamic balancing mechanism in MTMLAN are replaced by the homoscedastic uncertainty approach, 5) MTMLAN without sample-level dynamic balancing (MTMLAN w/o SL Balancing), 6) MTMLAN without task imbalance penalty term (MTMLAN w/o

TIP).

Table 9 shows the detailed performance results of the baselines and the proposed MTMLAN method. The "weight balancing" column shows which type of task weight balancing the corresponding method applies. It can be seen from the table that the common multi-label classification (CMLC) method has the worst performance regarding all three metrics. The equal weight multi-task classification (EWMTC) method performs better. This illustrates the effectiveness of multi-task learning in solving multi-label problems.

The results of the two state-of-the-art multi-task learning techniques, namely Weight Uncertainty (WU) and GradNorm, demonstrate further improved performance compared to the vanilla EWMTC approach, with GradNorm having a slight edge over WU. It should be noted that both methods only focus on optimizing task-level weight balance. Next, we compare the performance of the three variants of the proposed MTMLAN method. It can be seen from the table that both of them outperformed GradNorm and WU. The model performance did suffer when purposely excluding the sample-level balancing function or the task imbalance penalty term in the task-level balancing function. This demonstrates the effectiveness of the two components.

Table 8 shows the detailed classification accuracy for each task/concept of the baselines and the proposed MTMLAN method. It can be seen from the table that the trend for task-level classification accuracy performance is entirely consistent with the overall performance of each technique. The standard multi-label classification (CMLC) method shows the worst performance among all seven tasks/concepts, especially on tasks/concepts with fewer samples. Table 7 shows that the P/N ratio of flood/storm and

speak/briefing/interview concepts are significantly higher than the rest of the concepts in the dataset. This partly explains the worse performance on the minority concepts. In comparison, the equal weight multi-task classification (EWMTC) method performs better, proving that the model could generalize better on the whole problem domain by learning the shared representation across all tasks. The same outcome applies to GradNorm and Weight Uncertainty, improving accuracy across all tasks/concepts. However, none of these approaches shows a noticeable improvement in narrowing the performance gap between the minority and majority tasks/concepts.

In contrast, components in MTMLAN, such as the sample-level balancing function and the task imbalance penalty term, force the model to allocate more resources to difficult samples and minority tasks/concepts while training. As a result, minority tasks/concepts observed much higher performance gain than their majority counterparts. For instance, when using the results of CMLC as a benchmark, the accuracy of demonstration, damage, and victim concepts have improved by 14.74%, 21.79% and 18.53%, respectively. This is significantly higher than the improvements on majority concepts, such as flood/storm and briefing, which account for 10.22% and 7.55%.

We further demonstrate the effectiveness of the proposed method in Figure 11, which shows the training loss history of MTMNAN against the other two state-of-the-art methods. It can be seen from the figure that MTMNAN constantly produces lower losses compared to the other two methods.

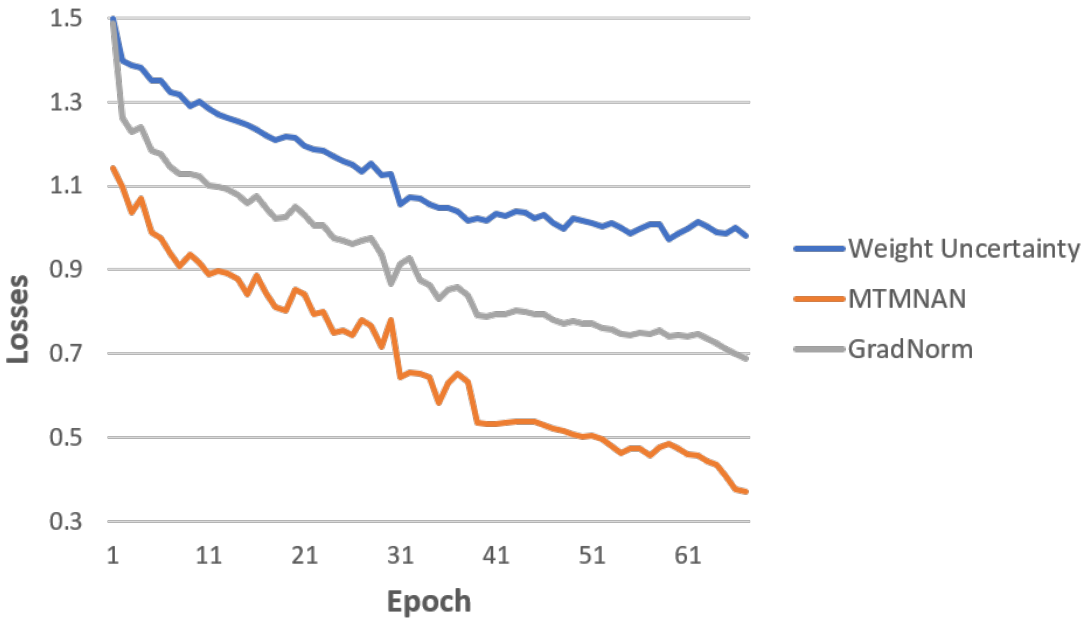


Figure 11: Training loss comparison among Weight Uncertainty, GradNorm, and the proposed MTMNAN methods

4.3 Conclusion

This work presents the novel dynamic task-balancing approach and a multi-label multi-task deep learning framework for disaster video classification. The proposed MTMNAN model utilizes the shared network to learn general information that can be shared across all tasks. On the other hand, task-specific networks help the model learn patterns related to each task. The proposed dynamic task balancing approach automatically adjusts the training progress on both the sample and task levels. The sample-level dynamic balancing function focuses on difficult instances by allocating more resources. At the same time, the task-level dynamic balancing mechanism adjusts weight distribution by

attending to the training rate of each task. In addition, extra cautions are paid to the task imbalance problem by introducing the task penalty term to the task-level balancing function. In conclusion, we showed that the proposed MTMNAN could perform better than other state-of-the-art techniques. The next chapter will extend the proposed framework to accommodate multimodal data inputs.

CHAPTER 5

HIERARCHICAL GRAPH FUSION

Multimodal learning has attracted significant interest from the research community due to its benefit in utilizing the massive amount of real-world data which often contain multiple data sources [29] [30] [142]. Compared to its single-modality counterpart, multimodal learning is the technique that focuses on exploiting the rich information underlying various input modalities. One essential step in multimodal learning is multimodal fusion, where the input features of each modality are combined to form a single vector. Therefore, the feature fusion strategy substantially impacts the model's effectiveness in harvesting the information provided by multiple input sources. Contrary to traditional belief, merely increasing the number of input modalities does not always yield better results [16]. The main cause that leads to the subpar performance is due to the oversight of cross-modality interactions.

How to effectively fuse the representations of diverse modalities has become a pressing issue in multimodal learning and therefore attracted much attention from the research community. The heterogeneous nature of multimodal data creates an emerging barrier in harnessing comprehensive information across all modalities, which is the key to fully understanding and utilizing the rich multimedia information [73]. Early attempts at multimodal fusion tend to work on each modality separately. Each modality is trained on its own network with the resulting intermediate features combined in different stages

of the processing chain, such as early fusion and late fusion [157]. However, due to the heterogeneous nature of multimodal data and the disconnection among networks, the fused vector still falls short of representing the complex distribution among modalities.

This chapter introduces the proposed hierarchical multimodal fusion network and its applications. The multimodal fusion network aims to capture the inter-modality correlations among modalities and, at the same time, retain their independent properties. The proposed method has been applied to two applications: a hierarchical multimodal fusion network with dynamic multi-task learning for disaster situation assessment and a machine learning framework for airfare price prediction.

5.1 Hierarchical Multimodal Fusion Network with Dynamic Multi-Task Learning

In this section, we propose a novel hierarchical multimodal fusion network with dynamic multi-task learning [178]. The multi-task learning strategy applied in this work combines the automatic loss weighting and dynamic loss balancing introduced in Chapter 4. The multimodal fusion network hierarchically joins each modality to form a graph structure where the vertices represent joined modalities and the edges contain the cross-modality interactions. The relative importance among joined modalities at the same level is learned in a sample-to-sample fashion and applied to formulate the joint embedding that will be used in the next level. We also propose a dynamic multi-task learning approach that disintegrates the multi-label classification problem into various single-label binary classification tasks. By monitoring the training complexity in each task, the dynamic multi-task learning unit automatically adjusts the weighting of the task loss so that the

optimal weight balance can be achieved. The dynamic multi-task learning unit also assigns a set of initial task loss weights at the beginning of the training cycles and keeps updating them throughout the training process to ensure the task loss weights are not caught in the local minimum/maximum.

In summary, the major contributions of this work are listed below:

- We propose a novel hierarchical multimodal fusion network that exploits the cross-modal interactions.
- A dynamic multi-task learning approach that automatically optimizes the model training process based on both task level and sample level training complexities. It also re-balances the loss weights for each task at the onset of the training cycles to minimize the chance of task weights being caught in the local minimum/maximum.

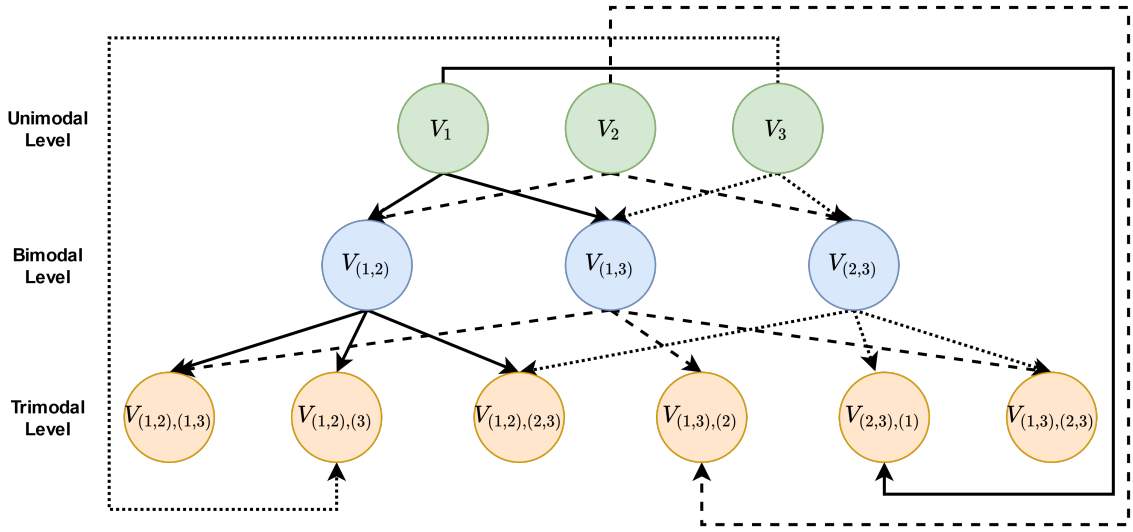


Figure 12: Hierarchical Graph Fusion Network (HGFN) with 3 input modalities

5.1.1 Architecture Design

In this section, we present the architecture design of the hierarchical multimodal fusion multi-task learning framework. The framework is composed of two main components: a Hierarchical Graph Fusion Network (HGFN) and a dynamic MTL (DMTL) module. In the first step, the feature representations of each modality are fused by the HGFN. In step two, the joint feature produced in step one will be used by the DMTL module to optimize the model training progress by dynamically adjusting the loss weights assigned to each task

5.1.2 Hierarchical Graph Fusion Network

Inspired by [120], we form the HGFN by exploiting the n-modal interactions. HGFN combines all modalities on unimodal, bimodal, and trimodal levels and models the interactions and relationships between each pair of combinations. An overview of the HGFN is shown in Figure 12.

The first level contains all unimodal and their interactions. We define the unimodal input feature vector V_i, where M is the total number of modalities and $i = [1, M]$. Although HGFN can be applied to any number of modalities, here we consider $M = 3$ in the rest of the paper. To model the relative importance of each modality and assign weights to the edges, we apply a Dynamic Attention Unit (DAU) to learn the importance of each modality and assign it the weights of the connected edges. More specifically, features from each modality are first concatenated together and then pass to a network composed of 2 convolutional layers with 5 by 5 and 1 by 1 kernel size and LeakyRelu activation.

Padding is performed to ensure the features of each modality have the same dimension. This process can be described as follows.

$$w_1 \oplus w_2 ... \oplus w_M = DAU(V_1 \oplus V_2 ... \oplus V_M) \tag{5.1}$$

where $\oplus$ is the concatenation operation, $V_1, V_2, ...V_M$ are the unimodal vectors of the M modalities, and $w_1, w_2, ...w_M$ are the corresponding weights. DAU learns the dynamic importance score that should be assigned to each vector in a sample-based fashion. Such an importance score will be used as the foundation to form the edge weights at higher levels.

In the next step, the final unimodal level vector can be obtained as the weighted average of vectors from all unimodal level vertices:

$$F_{unimodal} = \frac{1}{M} \sum_{i=1}^{M} w_i \cdot V_i \tag{5.2}$$

where $F_{unimodal}$ is the combined unimodal vector.

In the bimodal level, each pair of unimodal vectors are combined to form the vertices in this level. A neural network $CONV$ with one 1D convolutional layer and one dense layer with LeakyRelu activation is used to combine the unimodal vectors and produce all bimodal level vertices. This procedure can be described as follows:

$$\begin{gathered} V_{(a,b)} = CONV(V_a \oplus V_b) \\ a = 1, 2, ...M; b = 1, 2, ...M; a \neq b \end{gathered} \tag{5.3}$$

where $V_{(a,b)}$ is the bimodal vector. Regarding the edges that connect the vertices between unimodal and bimodal levels, we assume that the closer the two features in the vector space, the more homogeneous the information they possess. Therefore, the combination

of such two features will not provide as much information as the two distinct features do. Based on this assumption, we calculate the similarity between each pair of nodes at the bimodal level. The calculation can be described as:

$$S_{a,b} = COS(\widetilde{V}_a, \widetilde{V}_b) \tag{5.4}$$

where $S_{a,b}$ represents the similarity score between vertices a and b, COS is the cosine similarity function, and $\widetilde{V}_a$ and $\widetilde{V}_b$ are the softmax normalized form of vector V_a and V_b. The purpose of softmax normalization is to constrain the values of both vectors to be between 0 and 1. According to our assumption, the more similar two vectors are, the less weight they should carry when combined. In other words, the edge weight between the two vertices should grow in inverse proportion to the similarity score. Therefore, the edge weight that connects vertex a in the unimodal level and vertex ab in the bimodal level is calculated as $\frac{w_a}{S_{a,b}+\theta}$. Similarly, the edge weight that connects vertex b and ab is defined as $\frac{w_b}{S_{a,b}+\theta}$. Term θ is an adjustable factor that controls the growth rate with a value between 0 and 1. Based on the empirical study, $\theta = 0.5$ is used in this paper. Consequently, the vertex weight in the bimodal level is formulated as:

$$\begin{aligned} q_{a,b} &= \frac{w_a + w_b}{S_{a,b} + \theta} \\ w_{a,b} &= \frac{e^{q_{a,b}}}{\sum_{j=1}^{M} \sum_{k=1, j\neq k}^{M} e^{q_{j,k}}} \end{aligned} \tag{5.5}$$

where $q_{a,b}$ is the vertex weight for $V_{(a,b)}$ in the bimodal level, and $w_{a,b}$ represents the softmax normalized form of $q_{a,b}$. Then, the final combined bimodal level vector can be described as:

$$F_{bimodal} = \sum_{a=1}^{M} \sum_{b=1, a\neq b}^{M} w_{a,b} \cdot V_{(a,b)} \tag{5.6}$$

where $F_{bimodal}$ is the combined bimodal vector

At the trimodal level, all calculations are similar to the procedure illustrated in the bimodal part. Equations (5.4), (5.5), and (5.6) are used to calculate the trimodal level similarity scores, vertex weight, and combined trimodal vector. The trimodal level contains two types of vertices: 1) the combination of bimodal vertices; and 2) each bimodal vertex is combined with the unimodal vertex that is not included in the formation of this bimodal vertex. Therefore, for a dataset with 3 input modalities, there will be a total of 6 vertices in the trimodal level.

In the last step, the combined vectors from unimodal, bimodal, and trimodal levels are concatenated to form the final combined vector $F_{combined}$:

$$F_{combined} = F_{unimomdal} \oplus F_{bimodal} \oplus F_{trimodal} \tag{5.7}$$

5.1.3 Multi-task learning Module

Multi-task learning helps the model become more generalized by having multiple tasks training at the same time, which also reduces the potential of overfitting. In this work, the dynamic multi-task learning (DMTL) [178] strategy introduced in Chapter 4 is applied to train the model to predict arrival and departure delays simultaneously. DMTL works on both sample and task levels. Sample-level DMTL focuses on prioritizing resources to samples that produce more significant errors. On the other hand, task-level DMTL automatically adjusts the weight on the loss generated by each task during the training phase so that tasks producing more significant losses will be prioritized. Additionally, automatic loss weighting [182] is applied on the training iteration level, enabling the model

to automatically adjust the loss weights for each task to improve performance.

5.1.4 Experiments and Analysis

5.1.4.1 Datasets

CrisisMMD [3] is a multimedia Twitter dataset with more than 16,000 tweets and 18,000 images that are related to seven major natural disaster events. Each sample is labeled with 3 groups of concepts: data informative level, humanitarian category, and damage level. The data informative level represents the amount of information carried, the humanitarian category covers the type of humanitarian crisis and relieving efforts that occurred at the scene, and the damage level is the severity of damage to infrastructures and utilities. We report the F1 score, Hamming Loss (HL), and Mean Average Precision (MAP) on this dataset. For F1 and MAP, the higher the score the better, whereas for HL the lower the score the better.

YouTube Disaster dataset [141] is a multi-label YouTube hurricane disaster video dataset that contains more than 1,500 video clips and the corresponding text descriptions. Each sample is manually labeled with 7 concepts based on the elements present in the scene. These concepts include demonstration, emergency response, flood/storm, human relief, damage, victim, and speak/briefing/interview. We report the model performance in F1 score, Hamming Loss, and Mean Average Precision on this dataset as well.

5.1.4.2 Experimental Setup

Visual Feature Extraction: We use ImageNet [38] pretrained Inception V3 [164] model as the feature extractor for the visual data. Regarding the YouTube Disaster dataset,

Table 10: Data informative concept performance evaluation on the CrissMMD dataset

Method	F1	HL	MAP
CFC + LSEW	0.623	0.237	0.587
MATF + LSEW	0.774	0.151	0.738
GFN + LSEW	0.813	0.104	0.762
HGFN + LSEW	0.839	0.097	0.794
CFC + MOO	0.685	0.202	0.638
CFC + MTI-NET	0.673	0.214	0.625
CFC + DMTL	0.736	0.164	0.709
HGFN + DMTL	0.862	0.041	0.825

each video clip is subsampled into 40 frames and resized and cropped into 224 by 224 pixels.

Textual Feature Extraction: Embeddings from Language Models (ELMo) representation [139] is used to generate the word embedding for textual data. Compared to traditional text embedding techniques such as Word2vec [128] and Glove [138], ELMo can capture the morphological information and also excel in handling out of vocabulary words.

Audio Feature Extraction: A pre-trained SoundNet [10] is used to extract the audio features.

For the CrissMMD dataset, features generated by each pre-trained model are

Table 11: Humanitarian category concept performance evaluation on the CrissMMD dataset

Method	F1	HL	MAP
CFC + LSEW	0.527	0.293	0.496
MATF + LSEW	0.681	0.207	0.649
GFN + LSEW	0.677	0.209	0.642
HGFN + LSEW	0.712	0.181	0.695
CFC + MOO	0.603	0.246	0.571
CFC + MTI-NET	0.614	0.237	0.588
CFC + DMTL	0.686	0.194	0.660
HGFN + DMTL	0.762	0.153	0.749

directly passed to HGFN to perform the multimodal fusion. To exploit the temporal information in the YouTube Disaster dataset, features generated by the pre-trained models are first fed into a small neural network with 2 Bidirectional Gated Recurrent Unit (Bi-GRU) layers with attention enabled. Then, the intermediate vectors are processed by HGFN, which is similar to the process applied to CrissMMD.

For both datasets, 60% of the data is used for training, 20% for validation, and 20% for testing. The validation set is used to tune all hyperparameters, and the term α in the TDB loss function is set to 1 based on the empirical study. Adam [88] is used for optimizing the training process and the initial learning rate is set to 0.01.

Table 12: Damage level concept performance evaluation on the CrissMMD dataset

Method	F1	HL	MAP
CFC + LSEW	0.634	0.229	0.607
MATF + LSEW	0.781	0.148	0.745
GFN + LSEW	0.819	0.117	0.793
HGFN + LSEW	0.852	0.080	0.839
CFC + MOO	0.693	0.181	0.664
CFC + MTI-NET	0.688	0.186	0.650
CFC + DMTL	0.746	0.149	0.715
HGFN + DMTL	0.913	0.029	0.897

The DMTL module is applied to 3 concept groups of the CrissMMD dataset, in which each concept is modeled as a distinct task. In comparison, we consider each label in the YouTube Disaster dataset as a single task. This converts the original multi-label classification problem into an MTL problem.

5.1.4.3 Experimental Results

Several baselines, including state-of-the-art methods, are selected to demonstrate the performance of our proposed framework. The multimodal fusion baselines include 1) a standard fuse by concatenation (CFC) approach that simply concatenates each modality immediately after the initial feature extraction step; 2) tensor-based fusion method

Table 13: Performance evaluation on the YouTube Disaster dataset

Method	F1	HL	MAP
CFC + LSEW	0.769	0.157	0.722
MATF + LSEW	0.865	0.053	0.818
GFN + LSEW	0.889	0.041	0.805
HGFN + LSEW	0.931	0.024	0.890
CFC + MOO	0.874	0.040	0.828
CFC + MTI-NET	0.882	0.035	0.831
CFC + DMTL	0.922	0.027	0.904
HGFN + DMTL	0.987	0.011	0.958

Table 14: Per-concept classification accuracy on YouTube Disaster dataset

Approach	Demonstration	Emergency Response	Flood/Storm	Human Relief	Damage	Victim	Briefing
CFC + LSEW	0.823	0.812	0.866	0.829	0.787	0.780	0.875
MATF + LSEW	0.853	0.841	0.897	0.854	0.811	0.804	0.905
GFN + LSEW	0.866	0.851	0.902	0.865	0.831	0.824	0.880
HGFN + LSEW	0.914	0.909	0.960	0.927	0.913	0.895	0.923
CFC + MOO	0.841	0.835	0.887	0.853	0.819	0.804	0.890
CFC + MTI-NET	0.866	0.852	0.875	0.841	0.833	0.812	0.906
CFC + DMTL	0.933	0.908	0.932	0.931	0.942	0.903	0.915
HGFN + DMTL	0.955	0.971	0.989	0.973	0.952	0.917	0.982

MAFT [208]; and 3) Graph Fusion Network (GFN) [120]. The baselines for MTL include 1) a linear sum of all task loss with equal weights (LSEW); 2) Multi-Objective Optimization (MOO) [154]; and 3) Multi-scale Task Interaction NETwork (MTI-NET) [169]. For comparison purposes, we replace the model low-level layers of each baseline with the aforementioned pre-trained models.

Multimodal fusion strategies: Table 10, Table 11, and Table 12 demonstrate the performance of the proposed HGFN and DMTL approaches, as well as other baseline methods on the data informative, humanitarian category, and damage level concepts of the CrissMMD dataset. Table 13 shows the experimental results on the YouTube Disaster dataset. It can be observed that for both datasets, the CFC+LSEW combination yields the lowest score in all metrics. This is unsurprising since a simple concatenation of features in the early stage often fails to reflect the heterogeneous distribution of different modalities. Moreover, an equal weight linear sum of task loss in MTL has very limited effectiveness or even a negative impact when a few tasks dominate the training process.

Tensor-based fusion method MATF and graph-based fusion method GFN both demonstrate performance improvements compared to the CFC+LSEW vanilla approach. GFN exhibits a clear edge over MATF, especially on data with more input modalities, such as the YouTube Disaster dataset. This is partly because common tensor fusion approaches like MATF only model the joint embedding representation after the fusion operation, whereas GFN fills this gap by learning the inter-modality interaction during the early stage.

Our proposed HGFN outperforms all baselines and beat the 2nd best performer by 4.2% in F1 score and 8.5% in MAP. We argue that this can partly be attributed to the DAU

that learns the relative importance of each modality and integrates it at the beginning of the graph fusion network. We also report the per-concept results on the YouTube Disaster dataset in Table 14, which shows the classification accuracy of all 7 concepts. It can be observed that our proposed approach outperforms GFN+LSEW (the second-best result) by up to 8.2% in the damage concept. Our model exhibits consistent performance on both datasets regarding the multimodal fusion results.

Multi-task learning strategies: Table 10, Table 11, Table 12, and Table 13 also illustrate the results of MTL methods on both CrissMMD and YouTube Disaster datasets. MOO and MTI-NET exhibit more robust performance than the equal-weight linear sum MTL approach. However, the overall improvement is not quite significant. A probable explanation is in the situation of severe class imbalance, where there will be a substantial performance hit on both methods. Our proposed approach handles the class imbalance issue by introducing the inverted task sample distribution ratio term in the DMTL loss function, which helps the model further penalize the majority classes by allocating more resources to the minority classes. We also argue that re-balancing the task loss weight at the beginning of a training cycle helps our model continue reducing the total training loss; while this mechanism is absent in the other two methods.

For the CrissMMD dataset, our proposed DMTL approach outperforms the 2nd best method by 7.2% in F1 score and 7.3% in MAP. Regarding the YouTube Disaster dataset, our approach also leads the 2nd best performer in classification accuracy by 9.1% in the victim concept.

Our proposed model with the hierarchical graph fusion network and dynamic

MTL achieves the best performance among all baselines in both CrissMMD and YouTube Disaster datasets. Furthermore, the modularity design of HGFN and DMTL modules makes them very flexible and easy to apply to other data types and model structures.

5.1.5 Conclusion

This work proposes a hierarchical multimodal multi-task learning framework that learns the joint embedding space for all cross-modality interactions and handles input data with multiple non-exclusive labels. We first analyzed the challenges of multimodal fusion and designed a novel hierarchical graph fusion network that is capable of exploiting joint embedding among all cross-modality interactions. Then, the DMTL module that automatically adjusts the loss weight for each task based on their learning complexity is applied. The DMTL module also takes into account the sample difficulty factors by allocating more resources to the hard samples. A task loss weight re-balancing mechanism is in place to ensure an optimal weight distribution at the beginning of the training cycle, which effectively prevents the weight from falling into the local minimum/maximum. Experimental results on two multimedia datasets show that our method outperforms baseline approaches by a clear margin. Moreover, our proposed framework can be applied to other data domains and network structures with little effort due to its modular nature.

5.2 Hierarchical Fusion Network for Airfare Price Prediction

Since the deregulation of the airline industry, airfare pricing strategy has developed into a complex structure of sophisticated rules and mathematical models that drive the pricing strategies of airfare [162] [123] [121]. Although still primarily held in secret,

studies have found that these rules are widely known to be affected by a variety of factors [135] [23]. Although still significant, traditional variables such as distance are no longer the sole factor that dictates the pricing strategy. Elements related to economic, marketing and societal trends have played increasing roles in dictating airfare prices.

Most studies on airfare price prediction have focused on either the national level or a specific market. Research at the market segment level, however, is still very limited. We define the term market segment as the market/airport pair between the flight origin and the destination. Predicting the airfare trend at the specific market segment level is crucial for airlines to adjust their strategy and resources for a specific route. However, existing studies on market segment price prediction use heuristic-based conventional statistical models, such as linear regression [173] [146], and are based on the assumption that there exists a linear relationship between the dependent and independent variables, which in many cases, may not be valid.

Recent advances in Artificial Intelligence (AI) and Machine Learning make it possible to infer rules and model variations on airfare prices based on a large number of features, often uncovering hidden relationships amongst the features automatically. To the best of our knowledge, all existing work leveraging machine learning approaches for airfare price prediction are based on: 1) proprietary datasets that are not publicly available [40] [131] and 2) transaction records data crawled from online travel booking sites like Kayak.com [168] [31] [110]. The problem with the former lies in the difficulty of gaining access to the data, making reproducing the results and extending the work nearly impossible. The issue with the latter is that the transaction records from each online

booking site are a small fraction of the total ticket sales from the entire market, making the acquired data likely to be skewed and, thus, not representing the true nature of the entire market.

This work proposes a novel hierarchical fusion network applied to two public data sources in the domain of air transportation: the Airline Origin and Destination Survey (DB1B) and the Air Carrier Statistics database (T-100) [181]. The proposed framework combines the two databases and macroeconomic data and uses machine learning algorithms to model the quarterly average ticket price based on different origin and destination pairs, known as the market segment. The framework achieves a high prediction accuracy with a 0.869 adjusted R squared score on the testing dataset.

5.2.1 Architecture Design

The proposed framework utilizes both the DB1B and T-100 datasets and macroeconomic data to predict the quarterly average airfare at the market segment level. Figure 13 shows an overview of the major components of the proposed framework. In the data preprocessing step, all datasets are cleaned to exclude possible erroneous samples, transformed and combined based on the market segment. The feature extraction module serves to extract and generate handcrafted features that aim to characterize the market segment. The goal of the feature selection module is to optimize the prediction model's performance by analyzing the features' effectiveness and removing any irrelevant features. Finally, the hierarchical graph fusion strategy is applied to the selected features, which are used as the input to the prediction model.

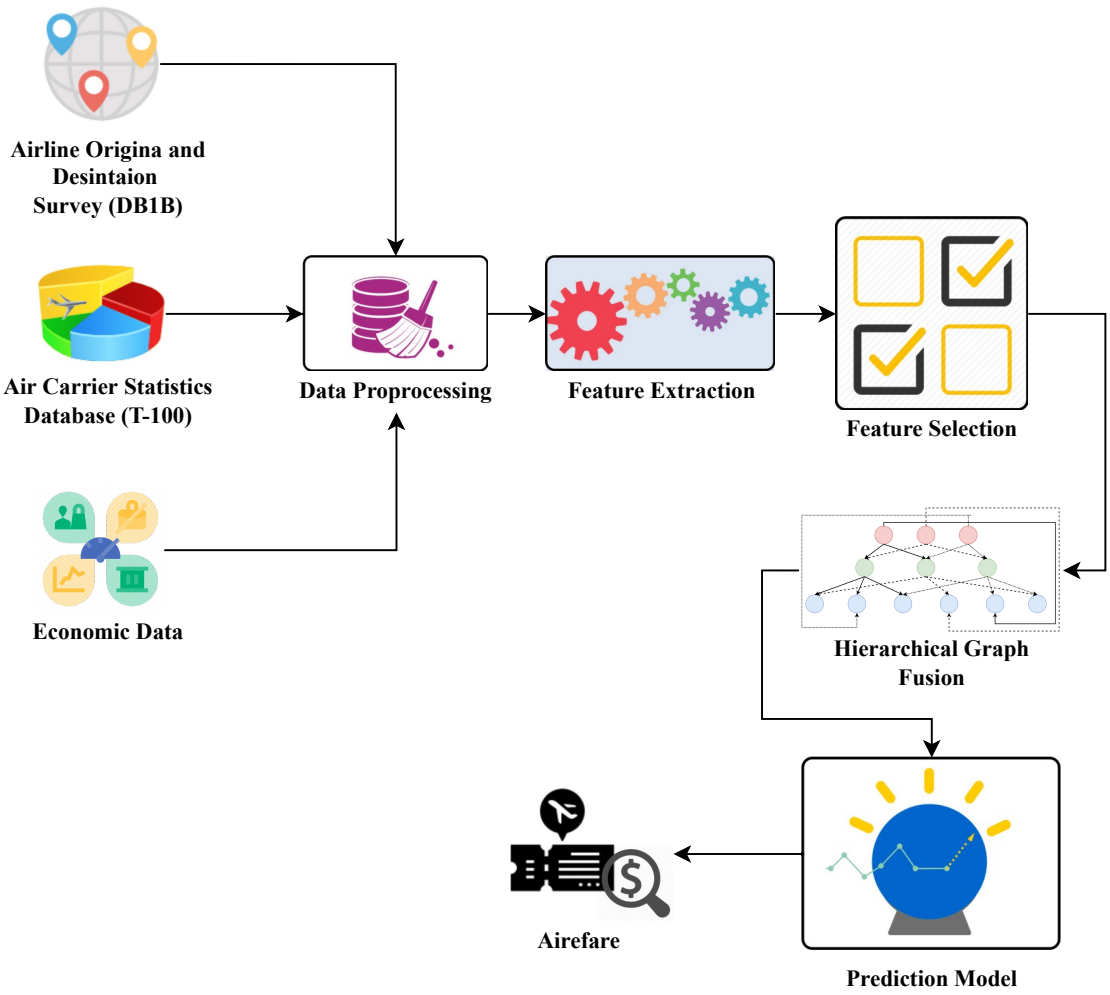


Figure 13: Proposed framework for airfare price prediction using public data sources

5.2.2 Datasets

To build the airline ticket price model at the market segment level, information about both the airline traffic and passenger volume for each market segment is required. Therefore, two public datasets (DB1B and T-100) are used in our proposed framework. Data collected during 2018 are used to train and evaluate the proposed model.

Many attributes contain the same information in the DB1B and T-100 datasets. Directly merging the tables creates many duplicate fields. Also, the airline data may include erroneous values caused by human error, currency conversion error, etc. Therefore, a properly designed data preprocessing workflow is crucial to generate accurate input data to build the machine learning model. For our proposed framework, a subset of most related data is used, including the origin airport (ORIGIN), the destination airport (DEST), time of the itinerary (QUARTER), carrier information (REPORTING_CARRIER), seat class (SEAT_CLASS) (e.g., first, business, economic, etc.), total flight distance for a ticket (DISTANCE), airfare price (ITIN_FARE), and the number of passengers in a ticket (PAX).

First, the DB1B ticket and coupon tables are merged based on the ITIN_ID. The ITIN_ID is the primary key for the ticket table. In the coupon table, all entries belonging to the same ticket share the same ITIN_ID. Samples in the DB1B ticket table with the itinerary value (ITIN_FARE) less than $50, or distance field (DISTANCE) less than 100 miles in the Coupon table are removed because those samples in practice, are considered reporting errors. Samples with the price credibility field (DOLLAR_CRED) equal to 0 are unreliable carrier reports, which are also disregarded. Since only the ticket table contains the ticket price, the price for each market segment is calculated based on the ITIN_FARE in

the ticket table and the distance ratio. The distance ratio measures the proportion between the distance of each leg in the coupon table and the entire length of the itinerary in the ticket table. Finally, the quarterly average fare value for each SEAT_CLASS on each specific market segment is generated.

Similar to the DB1B, the "SEATS" and "PAX" fields in T-100 are aggregated based on the origin and destination airports pair for each quarter. In the final stage, the two data sources consisting of the cleaned attributes are merged based on the market segment and every quarter.

5.2.2.1 Feature Extraction

Several features have been extracted from the DB1B and T-100 datasets to represent a specific aspect of the market segment. Furthermore, several macroeconomic features are added to the feature set to exploit the relationship between the airline industry and the overall economic conditions. Table 15 describes all the features that are identified during feature extraction.

The Load Factor (LF) is a primary metric used in the transportation industry. It represents the supply and demand relationship in a given market, strongly influencing an airline's pricing strategy. The T-100 dataset includes two features, the number of available seats and the number of actual passengers carried, that allow us to calculate the LF of a market by dividing the total passenger volume (P) by the total number of Available Seats (AS) in that market segment:

$$LF = \frac{P}{AS} \tag{5.8}$$

The effect of competition among airlines in a given market segment has been shown to affect the pricing strategy of the airlines [56]. In a less competitive market, the market power of a given airline is more substantial, and thus, it is more likely to engage in price discrimination. On the other hand, the higher the level of competition, the weaker the market power of an airline, and then the less likely the chance of the airline fare increases. The competition factor in the proposed model is based on the Herfindahl-Hirschman Index (HHI) [148], which measures the level of competition in a given market. It is the sum of the squared fraction of the market share of each top company:

$$HHI = \sum_{a=1}^{C} s^a, \tag{5.9}$$

$$s^a = \frac{v^a}{P}, \tag{5.10}$$

where C is the total number of companies, s^a is the market share of company a, v^a is the number of passenger carried by company a, and P is the total number of passenger in the market. We used the T-100 dataset to extract the market share of each airline in a specific market segment by calculating the ratio of the number of passengers carried by that airline to the total passenger volume of the market segment.

The emergence of Low-Cost Carriers (LCC) has revolutionized the entire operating model of the airline industry. The presence of LCC in a market has substantially impacted the total passenger volume and air ticket price [51]. A "LCC Presence" field is added to indicate whether the "Carrier" field in the DB1B coupon table contains the International Air Transport Association (IATA) code [76] related to one of the LCCs operating in the U.S. domestic markets. The six LCCs are Allegiant Air, Frontier Airlines, JetBlue, Southwest

Airlines, Spirit Airlines, and Sun Country Airlines.

Macroeconomic data, such as crude oil prices and Consumer Price Index (CPI), can also be utilized to uncover the hidden trend in airline fares. Fuel costs can take up to 50% of the total operating cost of an airline [93]. Hence, the level of crude oil price plays an essential role in formulating the airline's pricing strategy. It is a common practice for airlines to pass the cost of aviation fuel to the customer by adjusting the fare to compensate for the fluctuation of crude oil prices. In this paper, we used the West Texas Intermediate (WTI) crude oil price data and calculated its quarterly average value. Furthermore, the CPI measures the weighted average prices of various types of consumer goods and services, which include the prices in the transportation industry [96]. Therefore, we exploit its potential to measure the current level of air travel cost. The monthly CPI data is acquired from the Organization for Economic Co-operation and Development. Similar to the crude oil price, we calculate the quarterly average value. Figure 14 depicts the quarterly value trend of crude oil price, CPI, and airfare from 2006 to 2017. It demonstrates a clear relationship between the three types of data.

5.2.2.2 Feature Selection

A feature selection technique is applied to improve the model performance by investigating the degree of impact of each feature on the prediction result. We utilize the random forests model to construct an automated feature selection module. random forests is a tree-based ensemble learning algorithm that builds multiple decision tree classifiers during the training phase and outputs the predicted results based on either the majority vote

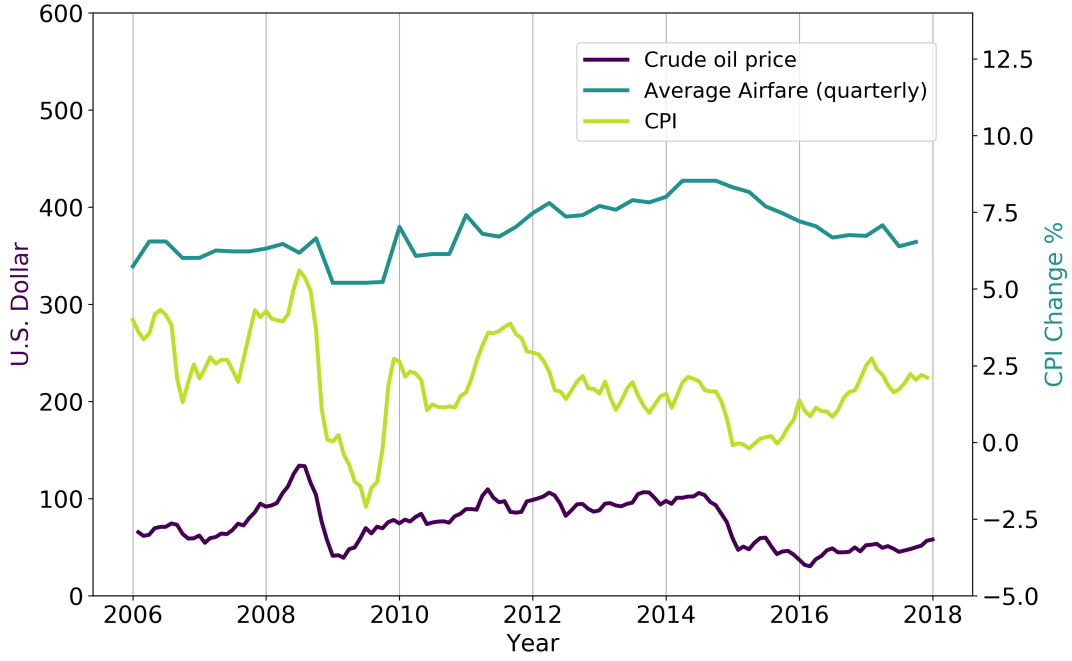


Figure 14: A comparison between the crude oil price, CPI and the quarterly averaged airfare from 2006 to 2017

(classification) or the average (regression) of the predictions of all decision trees. After training the random forests model with the entire feature set, it ranks all the features by their importance. A feature's importance is measured by the average decrease in impurity. It is the total decrease in the node's impurity caused by the corresponding feature, weighted by the chance that the decision path includes this node. There are several ways of choosing the impurity metric, and because our target is a continuous value, the sum of squared errors (SSE) is chosen as the impurity metric. The SSE for node o can be calculated as:

$$SSE_o = \sum_{j=1}^{S} \epsilon_j^2, \tag{5.11}$$

where S is the number of splits from the node, and ϵ is the error between the true value and the predicted value. The Node Importance (NI) for node o can be calculated as (assuming the parent node splits into two child nodes):

$$NI_o = w_o SSE_o - w_l SSE_l - w_r SSE_r, \tag{5.12}$$

where w_o, w_l and w_r are the weighted number of samples pass through node o and it's left and right child node. Then, the Feature Importance (FI) for feature x can be calculated as

$$FI_x = \frac{\sum_{b, b \in nodes\ split\ on\ feature\ x} NI_b}{\sum_{k, k \in all\ nodes} NI_k}. \tag{5.13}$$

Generally, a feature gains more importance when it has a greater effect of reducing the prediction error.

In the next step, the feature selection module applies Recursive Elimination (RE) to select the best set of features for the prediction model. More specifically, for each iteration, the feature with the lowest feature importance is eliminated, and the model will be retrained on the updated input. This process terminates when removing more features does not improve the model's performance.

5.2.2.3 Hierarchical Graph Fusion

Factors that affect airfare can originate from various aspects. Therefore, it is natural to include as many input variables as possible to improve the prediction accuracy. When dealing with multimodal problems, a common practice is simply concatenating all variables alone in one dimension. Although simple and effective in some cases, this approach ignores the cross-modality interactions among each input source. In order to

exploit the inter-modality dependency among all features, we adopted the hierarchical graph fusion approach (HGF) in the model [178]. The structure of HGF is a tree graph. Each modality is combined on different levels, in which the cross-modality interactions are learned based on different combinations. Similar to GNN, we use nodes to represent each feature combination and edges to represent the similarity between each pair of nodes. Therefore, the value of each child node is the result of both parent nodes and the edge weights. The final output of HGF is the concatenation of the weighted sum of each level.

5.2.2.4 Machine Learning Model

When developing the machine learning model, we chose random forests as the learner for the airfare price prediction task. Based on our empirical study, the random forests model demonstrates the best performance on the data as compared to several ML techniques including LR, SVM, and neural networks. Comparison results are explained in Section 5.2.3.

5.2.3 Experiments and Analysis

5.2.3.1 Experimental Setup

For our experiments, we collected 16,577,497 and 41,360,566 samples from the 2018 DB1B ticket table and coupon table, respectively. The T-100 dataset contains 329,426 samples. We tested several well-known machine learning models as baselines to compare with the random forests (RF) model. In particular, linear regression(LR), support vector machine(SVM), Multilayer Perceptrons (MLPs), and XGBoost Tree are adopted for the evaluation. For the SVM model, the radial basis function kernel is used, the tolerance

for the stopping criterion is set to 0.001, and the penalty parameter for the error term is set to 0.1. For the MLPs, three hidden layers are used with 30 neurons per layer. The Rectified Linear Unit (ReLU) [132] is used as the activation function and Adam is the optimization function [89]. The learning rate is set to 0.0001 with momentum enabled set to 0.9. For the XGBoost model, the number of estimators is set to 100 with a learning rate of 0.1, and the max depth equals 5. For the RF model, the number of estimators is also set to 100 with the minimum number of samples to split set to 2. To evaluate the proposed price prediction model, two popular metrics for regression analysis are used: the Root Mean Square Error (RMSE) and the Adjusted R Squared. RMSE calculates the differences between the observed values, y, and predicted values, $\hat{y}$. This difference for each data point is also called the residual. Thus, RMSE is calculated as follows:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{5.14}$$

where N is the total sample size. The lower the RMSE value, the higher the performance of the regression model.

The Adjusted R Squared, (R^2_{adj}), is usually used to explain how well the independent variables fit a curve or line. Adjusted R^2 also adjusts for the number of variables in a model. The higher the Adjusted R Squared is, the better the result of regression is. It is calculated as follows:

$$R^2_{adj} = 1 - \left[\frac{(1 - R^2)(N - 1)}{N - p - 1}\right] \tag{5.15}$$

where p is the number of predictors and R^2 is:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y_i})^2}{\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{5.16}$$

where $\bar{y}$ is the mean value of y.

5.2.3.2 Experimental Results

To demonstrate the importance of each feature for airfare price prediction, we extracted the importance scores generated by the feature selection module. Figure 15 depicts the importance value for each feature. As shown in the figure, "Distance" and "Seat Class" (Economy or business) are the most important factors for airfare price estimation followed by "Passenger Volume", "Load factor", and "Competition Factor". Although the "CPI" and "Crude Oil Price" do not have very high importance scores, they can still help the model predict a more accurate estimation of the airfare price. However, based on our experiments, "Quarter" does not help the regression model. Including the variable "Quarter" does not reduce the error during the training phase. Thus, it is automatically removed by the RF feature selection module. The goal is to identify the features that improve the model's performance and adding irrelevant features deteriorates the model's performance, as the model learns an irrelevant pattern.

The results comparing various regression models with feature selection and without feature selection are shown in Table 16. As seen from this table, LR and SVM have the lowest performance compared to other ML methods concerning the RMSE and R^2_{adj} metrics. The performance of all models improves after applying feature selection, which illustrates the importance of this module. XGBoost performs better than MLP, SVM, and

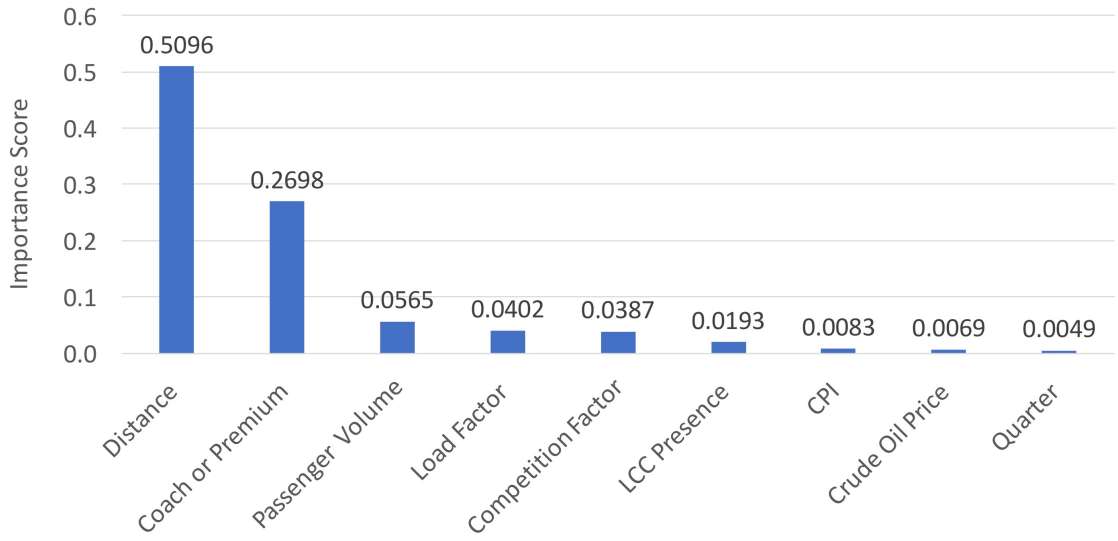


Figure 15: Importance score value for each feature generated by the random forests model

LR but does not outperform RF for airfare price prediction. Therefore, we utilize RF in the proposed framework, which achieves the highest performance compared to other baselines for this dataset. Specifically, it reaches 62.753 and 0.869 RMSE and R^2_{adj}, respectively. In other words, it improves the R^2_{adj} by 40% compared to the LR model, which is extensively used in previous studies for airfare price prediction.

Another experiment was conducted to demonstrate the importance of features specifically employed for our regression model. In this experiment, we only used standard features with high importance scores, such as "Distance", "Seat Class", and "Passenger Volume". The results are presented in Table 17. Again, we find that LR and SVM have lower performance than other models, and RF reaches the highest performance for RMSE and R^2_{adj}. However, the performance (R^2_{adj}) dropped by almost 7% for the RF model when

the less important factors were removed. Similarly, the performance for other models dropped as well. Although the less important factors may not contribute significantly to the performance, these results show that to achieve the best-performing model, one should include the "Load factor", "Competition Factor", "CPI", and "Crude Oil Price" as features. Consequently, the proposed framework utilizes all these features to achieve the highest airfare price prediction performance.

Table 18 demonstrates the impact of HGF on model performance. As shown in the table, the prediction scores for all models received noticeable improvement by applying the hierarchical fusion strategy. For instance, the adjusted R^2 score increased by 4.08% for the random forest model. Table 19 presents the prediction results of the proposed framework on all tested machine learning models. Random forests received an adjusted R^2 score of 0.903 and RMSE of 62.642, the best results among all models.

5.2.4 Conclusion

In this study, a hierarchical graph fusion framework was developed to predict the quarterly average airfare price at the market segment level. We combined the U.S. domestic airline ticket sales data and non-stop segment data from two public datasets (DB1B and T-100). Several features were extracted from the datasets and combined with macroeconomic data to model the air travel market segments. With the help of the feature selection techniques and the HGF strategy, our proposed model can predict the quarterly average airfare price with an adjusted R-squared score of 0.903. To the best of our knowledge, most previous studies on airfare price prediction using the DB1B dataset

have focused on conventional statistical approaches, which have limitations for problem estimations and assumptions. Also, to our knowledge, no other studies have integrated the information from DB1B, T-100, and macroeconomic data to model the air travel market segment. Thus, our study demonstrates the effectiveness of the proposed hierarchical graph fusion method, compares the performance of various machine learning classifiers and finds the best one for the airfare price prediction task by leveraging the information from the DB1B and T-100 datasets.

Table 15: The list of features generated during the feature extraction stage with explanations

Feature Name	**Description**
Distance	Market distance between the origin and destination airports
Seat Class	Indicator for economy or premium seat type
Passenger Volume	Total number of passengers traveled between the origin and destination airports
Load Factor	The ratio of the total number of passenger to the total number of seats in a market
Competition Factor	The market HHI
LCC Presence	Indicator of LCC operating in the market
Crude Oil Price	Quarterly average of crude oil price
CPI	Quarterly average of Consumer Price Index
Quarter	Indicates the three-month period of the year

Table 16: Performance comparison (before applying HGF) for different regression models with and without feature selection

Method	Without feature selection		With feature selection	
	RMSE	R^2_{adj}	RMSE	R^2_{adj}
LR	111.000	0.612	110.284	0.618
SVM	112.963	0.587	108.358	0.626
MLP	88.447	0.754	85.832	0.766
XGBoost	83.481	0.778	80.447	0.797
RF	**66.584**	**0.858**	**62.753**	**0.869**

Table 17: Performance comparison for different regression models without Load factor, Competition Factor, CPI, and Crude Oil Price features

Method	Without additional features		With additional features	
	RMSE	R^2_{adj}	RMSE	R^2_{adj}
LR	112.039	0.599	111.000	0.612
SVM	109.914	0.615	112.963	0.587
MLP	94.569	0.715	88.447	0.754
XGBoost	90.419	0.739	83.481	0.778
Random Forests	**70.575**	**0.804**	**66.584**	**0.858**

Table 18: Performance comparison (before applying feature selection) for different regression models with and without hierarchical graph fusion

Method	Without HGF		With HGF	
	RMSE	R^2_{adj}	RMSE	R^2_{adj}
LR	111.000	0.612	106.560	0.636
SVM	112.963	0.587	109.574	0.610
MLP	88.447	0.754	85.832	0.782
XGBoost	83.481	0.778	85.705	0.806
Random Forests	**66.584**	**0.858**	**63.92**	**0.893**

Table 19: Performance comparison for different regression models with the proposed framework

Method	RMSE	R^2_{adj}
LR	109.031	0.623
SVM	107.382	0.644
MLP	83.991	0.792
XGBoost	82.857	0.816
RF	**62.642**	**0.903**

CHAPTER 6

SPATIO-TEMPORAL GRAPH NETWORK

Many real-world problems can be represented by sequences of spatio-temporal data that describe an activity that occurs in various locations and times, such as videos, traffic flow data, and remote sensing imagery data. For example, each data sample contains the geographic location and the corresponding timestamp in traffic flow data. State-of-the-art deep learning approaches for processing spatio-temporal data combine convolutional neural network (CNN) and recurrent neural network (RNN). CNN has been the standard network structure for extracting local spatial characteristics in many studies [83, 136]. However, regular CNN only works on grid structures, such as images and videos, and falls short of capturing the spatial relationship among all objects [192]. On the other hand, when applying RNN-based methods to extract temporal features from the data, information tends to be lost due to how the information in the hidden states is passed down inside the network.

In this chapter, we first introduce the adaptive spatio-temporal graph network. The proposed framework takes advantage of both local and global spatial-temporal relationships between nodes in the network. The local spatial features are learned by connecting nearby nodes. The global spatial features are learned by constructing the sparse adjacency matrix representing the similarity among all nodes in the network. A BiLSTM-based sequence-to-sequence model is applied to capture the temporal dependency among features from the

input sequence. Then, we will introduce another graph network, the adaptive joint spatio-temporal graph learning network (AJSTGL). AJSTGL utilizes static and adaptive graph learning modules to capture the pre-defined and hidden spatial traffic patterns. We further adopt an auxiliary convolutional graph to complement the flight record data in uncovering more complex contextual node correlations. The standard graph convolutional layer is transformed to capture the unidirectional data flow. A sequence-to-sequence fusion model is also proposed to exploit the temporal correlation and combine the output of multiple parallelized encoders. we also develop the spatio-temporal graph transformer module to complement the sequence-to-sequence fusion module by dynamically capturing the time-evolving spatial node relations in long-term prediction. Experiments on three large-scale traffic flow datasets demonstrate that our model could outperform other state-of-the-art baselines.

6.1 Multitask Local-Global Graph Network

In recent decades, flight delay prediction has been defined as a spatio-temporal issue and has been intensively explored. [18, 63, 144, 165, 198]. Traditional approaches include using mathematical and statistical/probabilistic tools to capture the correlation between contributing variables [19, 42, 130] and create simulations that allow the study of the effect of certain factors under different scenarios [115, 153, 200]. Recently, data-driven approaches such as deep learning have gained ground due to the exponential growth of data availability and computing power [60, 87, 191]. Most works on flight prediction focus on a single airport or airline [24, 95, 122]. However, a more significant challenge

than working at the airport or airline level is to predict the flight delay at a network-wide level due to the added complexity of modeling the correlation among airports. Previous works on network-wide level delay prediction struggled to strike a balance between model generalization and complexity if they wished to adapt to different types of airports [17, 18].

This work proposes a novel Graph Convolutional Neural Networks (GCN)-based model with a multi-task learning strategy (MTLG-Net) for network-wide flight delay prediction [179]. MTLG-Net takes advantage of both local and global spatial-temporal relationships among nodes in the network. The local spatial features are learned through the connectivity between nearby network nodes. The global spatial features are learned by constructing the sparse adjacency matrix representing the similarity among all nodes in the network. A BiLSTM network is added to capture the temporal dependency among features from the input sequence. In order to take advantage of different input data sources, a hierarchical multimodal fusion network is adopted to learn their inter and intra-modality correlations. Due to the close correlation between arrival and departure delays, we adopt a dynamic multi-task learning strategy to train the model on these two tasks concurrently, which could significantly improve the model's generalization. The main contribution of our proposed MTLG-Net can be summarized as follows:

- A novel GCN-based network that learns both local and global spatial features in the graph. The local GCN focuses on the spatial relationship between nodes with direct connections. The global GCN captures the network-wide correlation among nodes that shares similar characteristics. In addition, an effective normalization technique is applied to the global GCN graph to control the sparsity of the adjacency matrix

and reduce redundant node-wise correlation.

- A hierarchical multimodal fusion network that exploits cross-modal interactions among flight delay data, weather data, flight volume, etc.
- A dynamic multi-task learning method uses sample and task level training complexity to adjust the training progress. In addition, it also automatically optimizes the task loss weight to further improve the model performance.

Figure 16 demonstrates the architectural design of the proposed framework.

6.1.1 Graph Convolutional Network

Similar to CNN, GCN applies convolutions by sliding the filters across the graph network to extract the spatial relationship among neighboring nodes. Otherwise, unlike CNNs, GCNs have been frequently applied to problems that contain data with non-Euclidean patterns [193]. We use GCNs to model both the local and global characteristic correlations among each airport in the flight network. The constructed graph can be represented as $G = (X, y, E, A)$, where X represents the node feature, y is the average hourly flight delay, $x \in \mathbb{R}^N$ and N is the number of airports, E is the set of edges in the graph, and A represents the adjacency matrix, $A \in \mathbb{R}^{N \times N}$. Then, a forward pass in the GCN layer can be expressed as:

$$H^{[l+1]} = \sigma(H^l \widetilde{A} W^l) \tag{6.1}$$

where $H^{[l+1]}$ and H^l are the feature vectors at layer $l+1$ and l, σ represents any non-linear activation function, A is the adjacency function that represents the relationship between

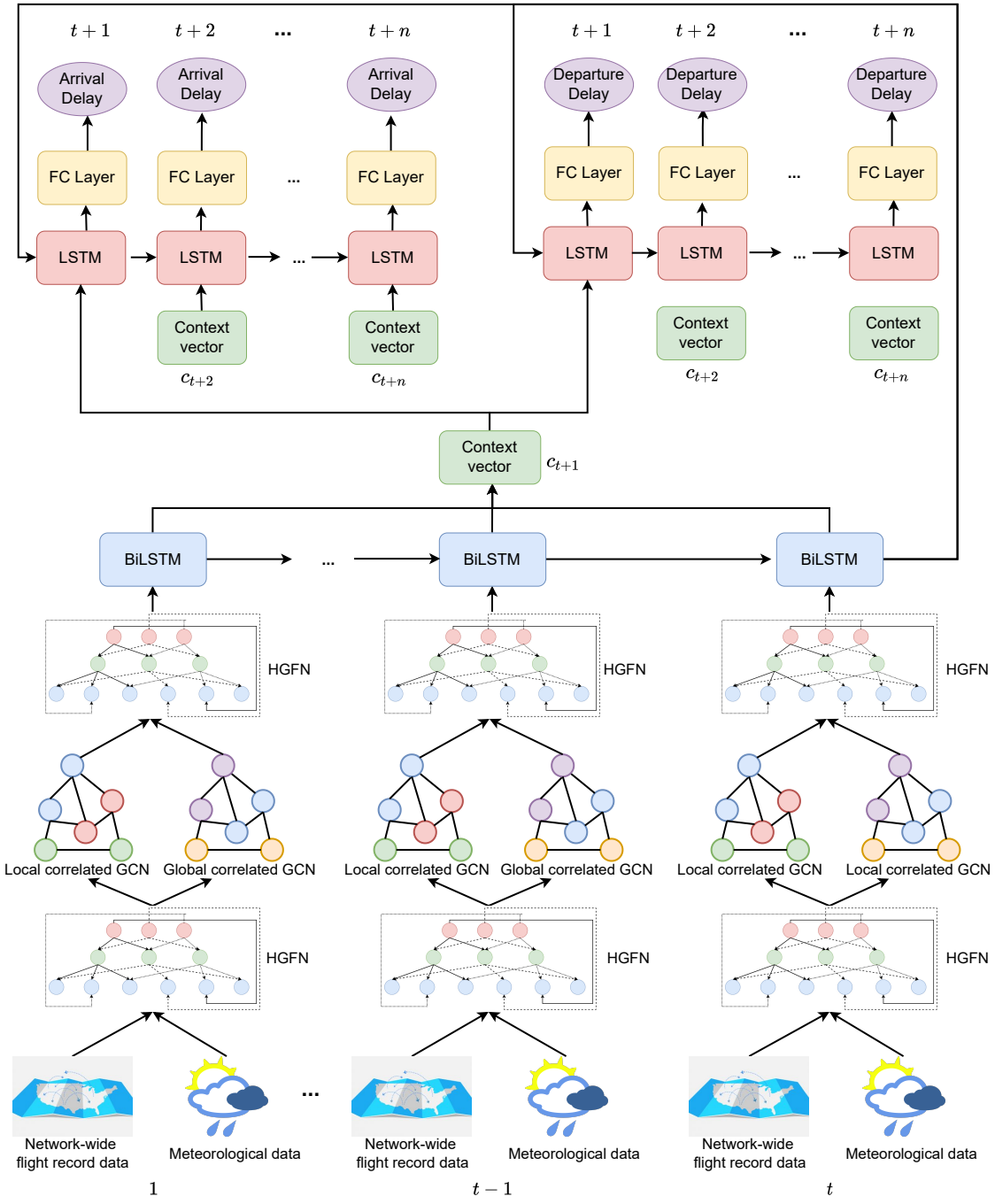


Figure 16: An overview of the proposed MTLG-Net.

each pair of nodes, and W^l is the weight parameters of layer l. The normalized adjacency matrix $\widetilde{A}$ is intended to prevent numerical instability and can be expressed as follows:

$$\widetilde{A} = D^{-\frac{1}{2}} A D^{-\frac{1}{2}} + I \tag{6.2}$$

where D is the degree matrix and I represents the identity matrix, which functions as a self-loop to each node.

This study utilizes two types of GCNs to model the correlations among airports from both local and global levels. The local correlated GCN is a directed graph based on each airport's flight route connectivity. On the other hand, the global GCN uses an undirected graph to represent the node similarity.

The local correlated GCN uses adjacency matrix $A_{local} \in A \in \mathbb{R}^{N \times N}$ to represent the connectivity of the airports. If there exists a flight route between airport i and j, then entry a_{ij} in A_{local} is 1, otherwise, 0. Therefore, the normalized adjacency matrix for local correlated GCN A_local is shown as follows:

$$\widetilde{A}_{local} = D^{-\frac{1}{2}} A_{local} D^{-\frac{1}{2}} + I \tag{6.3}$$

and the graph $G_local = (X, y, E, \widetilde{A}_{local})$ is obtained for the local correlated GCN.

The global correlated GCN uses an adjacency matrix that constitutes the similarity matrix embedding between each pair of nodes. The similarity between airports is calculated based on each airport's annual average flight and passenger volumes. The two factors are concatenated to form the similarity vector. The similarity score between two airports can be expressed as the cosine similarity between the two vectors:

$$S_{ij} = \frac{v_i \cdot v_j}{\|v_i\| \, \|v_j\|} \tag{6.4}$$

where v_i and v_j are the similarity vectors for airport i and j. Therefore, the similarity matrix S that contains the similarity scores for all nodes can be represented as:

$$S = \begin{bmatrix} s_{11} & s_{12} & \dots & s_{1,N} \\ . & . & . & . \\ . & . & . & . \\ s_{N,1} & s_{N,2} & \dots & s_{N,N} \end{bmatrix} \tag{6.5}$$

We first must solve the matrix sparsity problem before applying the similarity matrix as the global correlated graph adjacency matrix. In the local correlated graph, its adjacency matrix is generated based on the flight connectivity between airports. As a result, the adjacency matrix contains many zero elements representing airport pairs with no connecting flight. In comparison, the original similarity matrix does not contain zero values due to how the score is calculated. The computational complexity becomes much higher when the adjacency matrix is too dense.

On the other hand, if the matrix becomes sparse, some useful information may be lost. To address this issue, we developed a new approach named sparse matrix normalization to replace the default normalized graph Laplacian adjacency matrix to control the sparsity in the global correlated GCN adjacency matrix. The intermediate adjacency

matrix can be expressed as:

$$\widetilde{A}^{ij}_{global} = \begin{cases} \frac{(e^{s_{ij}-\sigma}-1)^2}{\sum \sum_{i,j}^{N}(e^{s_{ij}-\sigma}-1)^2+\varepsilon}, & i \neq j \ and \ s_i - \sigma > 0 \\ \\ 0, & otherwise \end{cases} \tag{6.6}$$

where $s_{i,j}$ is the similarity score between airport i and j, if we define the flattened similarity matrix as a 1D array $\hat{S} = [S_1, S_2, S_3, ...S_{N\times N}]$,$\hat{S} \in \mathbb{R}^{N\times N}$, then σ is the standard deviation of $\hat{S}$ and ε is a small constant to reduce the numerical instability.

The normalized adjacency matrix with self-loop can be expressed as:

$$\widetilde{A}_{global} = A_g\hat{lobal} + I \tag{6.7}$$

and the final graph $G_global = (X, y, E, \widetilde{A}_{global})$ is obtained for the global correlated GCN.

By utilizing the local and global correlated GCN, the model could capture the local and global correlations between airports without sacrificing much on the computation complexity.

6.1.2 Temporal Feature Extraction

Recurrent neural network (RNN) has been extensively involved in modeling sequential data by extracting temporal features that can be used for various tasks, including time series prediction, machine translation and speech recognition. Flight records, weather forecasts and other related sequential data contain valuable information that RNN can use to mine the temporal dependency among these variables. In this work, we apply a

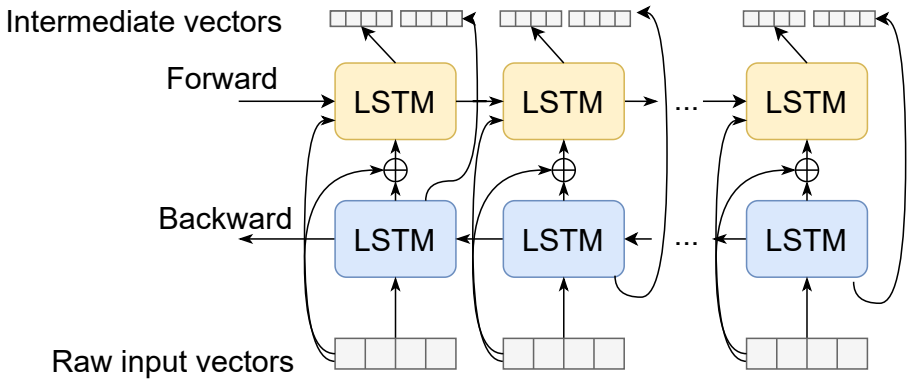


Figure 17: Network structure of the BiLSTM used in our model. Each arrow indicates the direction of the data flow, and $\oplus$ is the concatenation operation.

sequence-to-sequence model based on BiLSTM to predict flight delay through multiple time steps. Figure 17 demonstrates the structure of our BiLSTM network. LSTM is often considered superior to the vanilla RNN in capturing long-term information in the sequence. This is achieved by implementing an input gate, forget gate and output gate to control the passage of information:

$$i_t = \sigma(U_i x_t + W_i H_{t-1} + b_i) \tag{6.8}$$

$$f_t = \sigma(U_f x_t + W_f H_{t-1} + b_f) \tag{6.9}$$

$$o_t = \sigma(U_o x_t + W_o H_{t-1} + b_o) \tag{6.10}$$

where i_t,f_t and o_t represent the output of input, forget and output gates function, σ is the sigmoid activation function, U_i,U_f and U_o are the weight parameter for input x_t at

time step t, W_i, W_f and W_o are the weight parameter for hidden state H_{t-1} at time step t-1 and b_i, b_f and b_f are the bias terms.

Then, cell state c_t at time step t can be illustrated as a function based on the output of forget gate and input gate:

$$c_t = i_t \odot \tanh(U_c x_t + W_c H_{t-1} + b_t) + f_t \odot c_{t-1} \tag{6.11}$$

where c_t and c_{t-1} are the memory state at time step t and $t-1$, Uc and W_c are the weight parameters and H_{t-1} is the output hidden state, $\odot$ stands for the Huffman product.

Finally, the hidden unit H_t at the current time step can be calculated as:

$$H_t = o_t \odot \tanh(c_t), \tag{6.12}$$

A basic BiLSTM structure contains two LSTM units that process the same sequential data from both ends. By combining forward pass and backward pass in the training process, the model could better utilize context information to help make decisions. Flight arrival and departure patterns are often highly correlated, making BiLSTM a good candidate for handling this task. Figure 17 demonstrates the structure of our BiLSTM network.

6.1.3 Multiple Time Step Ahead Prediction

In practice, a good flight delay prediction model should be able to make long-term ahead-of-time predictions. This study uses the sequence-to-sequence (Seq2Seq) model [35] to encode the input flight delay and weather information into an internal context vector. The decoder will generate the prediction results in various time steps. The structure of

the Seq2Seq model is shown in Figure 18. The graph shows that the encoder comprises one layer of BiLSTM, where the sequential input signal gets passed down to produce the intermediate context vector. The context vector is then fed into two separate decoders, namely the arrival delay decoder and the departure delay decoder. Each decoder comprises one LSTM unit that outputs results at different intervals.

Attention mechanism [13] is applied to the Seq2Seq model. The final hidden state output by the encoder is replaced by the attention context vector that contains the weighted sum of every hidden state in the encoder. It helps the model retain more information when processing long input sequences.

6.1.4 Feature Fusion Module

The cause of flight delay is complex and can be attributed to various factors. As a result, it is essential to leverage contextual information to complement flight record data. Variables impacting air traffic may include flight date and time, flight volume and weather conditions. When multimodal data is involved, a common practice is to concatenate all feature vectors. However, it ignores the cross-modality interactions between input sources. To exploit the inter-modality correlations among all features, we adopted a hierarchical graph fusion (HGF) approach in the proposed model [178]. HGF utilizes a tree-based graph to combine each modality on different levels. The nodes on each level represent different modality combinations, and the edge weight measures the similarity between each pair of nodes. The complexity of the combination increases as the graph level gets deeper. By concatenating the level representation vector, we can get the final output of

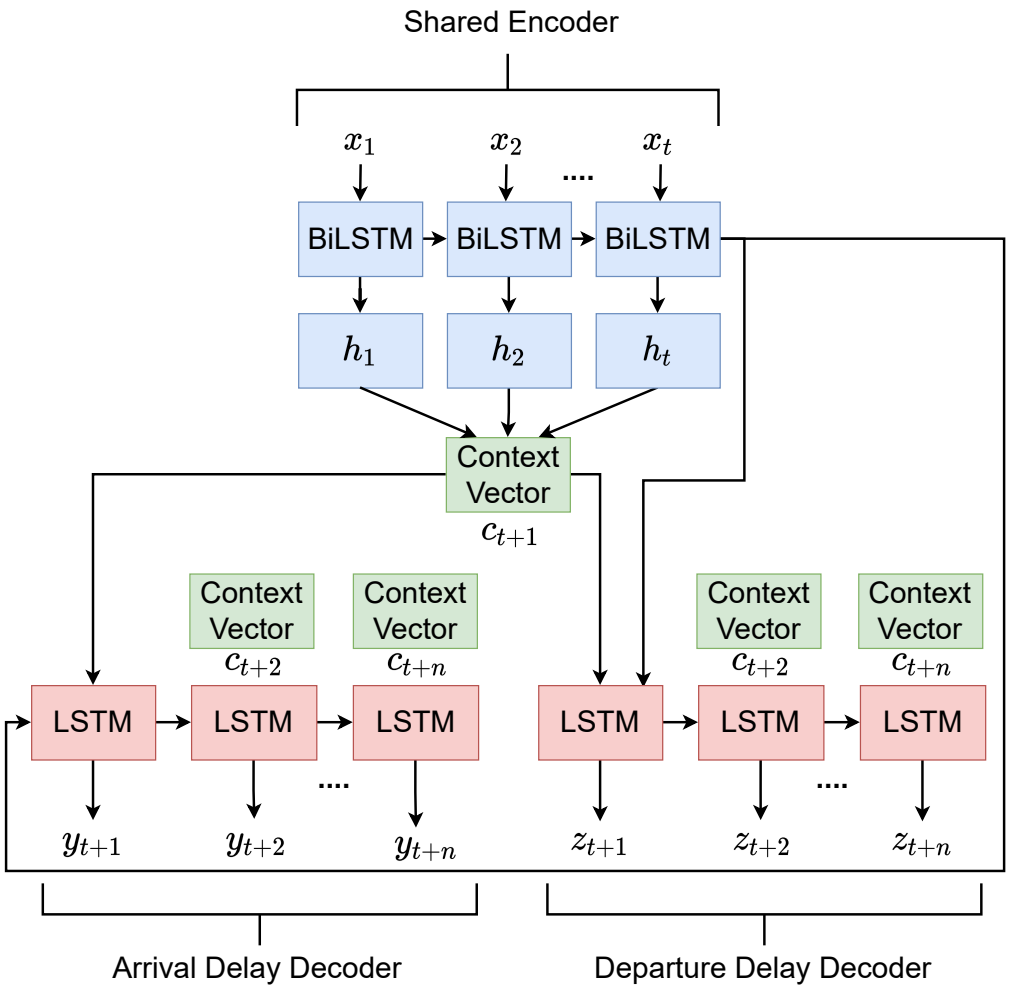


Figure 18: Overall structure of the BiLSTM based Seq2Seq model used in this model. x_1, x_2 and x_t are the input sequence from different time steps; $y_t + 1$, $y_t + 2$ and $y_t + n$ are the output of arrival delay task decoder, $z_t + 1$, $z_t + 2$ and $z_t + n$ are the output of departure delay task decoder The context vector generated by the attention module serves as the initial input for the decoder. The two tasks share the same encoder.

HGF.

6.1.5 Multi-task learning Module

Multi-task learning helps the model become more generalized by training multiple tasks simultaneously, reducing the potential of overfitting. In MTLG-Net, a dynamic multi-task learning (DMTL) [178] strategy is applied to train the model to simultaneously predict arrival and departure delays. DMTL works on both sample-level and task level. Sample-level DMTL focuses on prioritizing resources to samples that produce more significant errors. On the other hand, task-level DMTL automatically adjusts the weight on the loss generated by each task during the training phase so that tasks producing more significant losses will be prioritized.

6.1.6 Experiments and Analysis

6.1.6.1 Datasets

Reporting Carrier On-Time Performance Data: the reporting carrier on-time performance data published by BTS is a large-scale dataset that contains the on-time performance information of flights from all reporting carriers. Relevant attributes include flight date, origin and destination (O.D.), airport id, actual arrival time, actual departure time, arrival delay, departure delay and flight distance. Data from January 2017 to December 2021 are collected, including 30,940,455 entries that cover 433 airports and 8102 origin and destination pairs. During the data cleaning process, airports with less than 50 average daily flights, O.D. pairs with less than ten average daily flights and data entries with abnormal values (flight distance less than 100 miles and longer than 3000 miles except for O.D. pairs including Hawaii) are dropped. The cleaned dataset contains 25,312,665

entries, 87 airports and 294 OD pairs. Furthermore, we aggregated the data at the airport level, split the data into 24-hour intervals based on actual arrival time and obtained the average hourly arrival/departure delays for each airport.

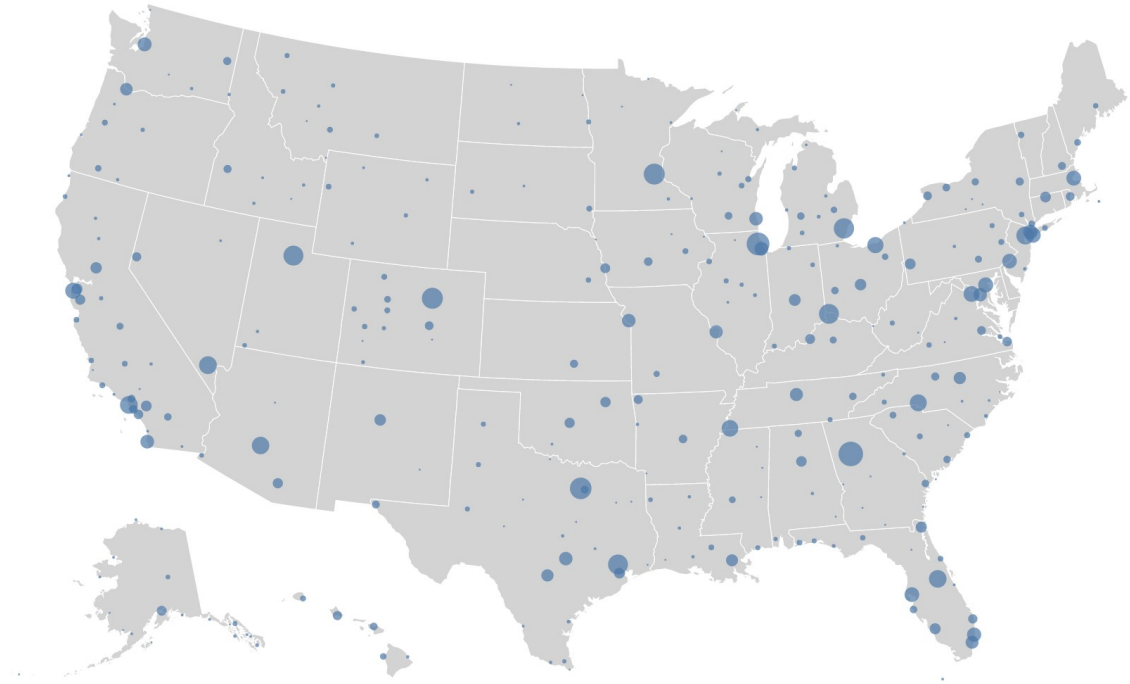


Figure 19: Map showing major U.S. airports based on the number of connecting flights. The size of the blue dot indicates the relative connection flight volume an airport receives compared to other airports.

Passenger Volume Data: we collected the passenger volume data for all U.S. commercial airports from the most recent passenger boarding (enplanement) and all-cargo data [47] published by the Federal Aviation Administration (FAA). This dataset contains the Enplanements (passenger boarding) information for 446 domestic and commercial

airports in the U.S. It also categorized commercial airports into large, medium, small, non-hub and non-primary, based on the annual passenger volume. Table 20 illustrates the categorization criteria and the number of airports in each category.

Table 20: U.S. airports categorization based on passenger volume

	Criteria	# of airports
Large Hub	Serviced more than 1% of total annual U.S. commercial passengers	28
Medium Hub	Serviced 0.25% to 1% of total annual U.S. commercial passengers	36
Small Hub	Serviced 0.05% to 0.25% of total annual U.S. commercial passengers	80
Non-hub	Serviced more than 10,000 passengers to 0.25% of total annual U.S. commercial passengers	195
Non-primary	Serviced more than 2,500 to 10,000 annual U.S. commercial passengers	107

Meteorological Data: the National Oceanic and Atmospheric Administration (NOAA) provides historical meteorological data collected at weather stations across the U.S. It contains the average hourly data regarding air temperature, precipitation, sky covers and clouds, sunshine, water, weather type and wind speed. In this work, we collected data from weather stations near the 87 airports and selected air temperature, precipitation and

wind speed as part of the input variables.

6.1.6.2 Experimental Setup

The dataset is separated into training, validation and test sets with a 60%, 20% and 20% split. Model hyperparameters are tuned using the validation set. Adam [88] is used to optimize the training process, using an initial learning rate of 0.01. Early drop is applied to prevent overfitting. Input vectors from different modalities are padded with 0 to ensure consistent dimension before being fused by HGF. We use Mean Average Error (MAE) as the metric for evaluation purposes.

6.1.6.3 Experimental Results

We conducted experiments using several baseline methods that have been commonly applied to study sequential data. ARIMA (autoregressive integrated moving average) is a statistical analysis model that predicts future value based on value from the previous time step. It has been extensively used to tackle traffic flow challenges. LSTM and Seq2Seq are two deep learning network models that have been heavily involved in time series data analysis. In addition, we selected two state-of-the-art models that have appeared in recent flight delay prediction studies. AG2S-Net [18] is a deep learning model that utilizes GCN and Seq2Seq to predict multi-step flight delays. DGLSTM [198] is another graph-based deep learning model that employs two adjacency matrices to represent the spherical distance and demand relationship among all airports.

We first report the experimental results on the network level to demonstrate the overall model performance and generalization capability. In addition, we divided the

testing dataset based on airport passenger volume defined in Table 20 to illustrate the prediction outcomes on the large, medium and small hub subgroups.

Table 21: Performance comparison of MTLG-Net with different baselines for average hourly arrival delay prediction

Method	MAE									
	1h	2h	3h	4h	5h	6h	7h	8h	9h	10h
ARIMA	9.331	10.021	11.263	12.834	13.097	14.765	15.932	16.283	17.727	18.316
LSTM	8.915	9.244	10.035	11.037	12.543	13.297	14.504	15.801	16.424	17.688
Seq2Seq	8.524	8.980	9.453	10.041	11.224	12.875	13.662	14.017	15.239	16.433
AG2S-Net	6.323	6.847	7.225	9.988	10.320	12.089	13.095	14.228	14.276	14.301
DGLSTM	5.013	5.877	7.044	9.635	10.098	11.767	12.237	12.768	12.980	13.176
MTLG-Net	**4.573**	**4.842**	**5.925**	**7.852**	**9.374**	**10.237**	**11.535**	**11.570**	**11.335**	**11.659**

Network-level: Table 21 and Table 22 contains the MAE of arrival and departure delay prediction in ten hourly intervals. It can be seen from the table that all methods consistently perform better on shorter time intervals. ARIMA performed the worst among other methods. The problem with ARIMA is that the model's effectiveness heavily relies on selecting its parameters, and the tuning process can be quite time-consuming. As a relatively simple model, ARIMA also lacks in capturing more complex patterns in the data. In comparison, results from LSTM and Seq2Seq models are considerably better, with Seq2Seq2 beating LSTM, especially in the longer time ahead prediction. AG2S and DGLSTM utilize GCN in their network structure and perform noticeably better than the relative basic models. The proposed MTLG-Net outperforms all baseline methods. It

Table 22: Performance comparison of MTLG-Net with different baselines for average hourly departure delay prediction

Method	MAE									
	1h	2h	3h	4h	5h	6h	7h	8h	9h	10h
ARIMA	9.223	10.010	11.152	12.836	13.041	14.460	15.892	16.290	17.517	18.411
LSTM	8.925	9.237	10.031	11.997	12.493	13.195	14.204	15.752	16.372	17.851
Seq2Seq	8.525	8.930	9.517	10.031	11.324	12.759	13.709	14.136	15.231	16.333
AG2S-Net	6.401	6.848	7.210	9.979	10.293	12.067	13.126	14.231	14.273	14.310
DGLSTM	5.011	5.863	7.041	9.584	10.123	11.742	12.347	12.744	12.982	13.241
MTLG-Net	**4.499**	**4.831**	**5.917**	**7.858**	**9.291**	**10.239**	**11.485**	**11.573**	**11.258**	**11.536**

achieves 29.71% lower MAE for one-hour ahead delay prediction than AG2S-Net and 10.22% than DGLSTM. For two-hour ahead prediction, Our model outperforms AG2S-Net by 29.45% and DGLSTM by 17.6%. Even for the extreme ten-hour-ahead prediction, our model still pulls 19.38% and 14.7% leads compared to AG2S-NET and DGLSTM.

Large, medium and small hubs: Table 23 - 28 compares the model performance of our model with other baseline methods on large, medium and small hub groups for the flight arrival and departure delay prediction. As shown in the tables, all methods demonstrate better results on large hubs, and the accuracy decreases as hub size is reduced. This outcome is as expected since large hubs generate more flight volumes that accurately represent the actual delay pattern. The proposed MTLG-Net also demonstrates higher consistency in arrival and departure delay prediction across the three airport categories. For instance, in one-hour-ahead delay prediction, the prediction accuracy difference between

Table 23: Performance comparison of MTLG-Net with different baselines for the average hourly arrival delay prediction on large hubs

Method	MAE									
	1h	2h	3h	4h	5h	6h	7h	8h	9h	10h
ARIMA	5.820	6.513	7.742	11.834	11.587	14.274	14.439	14.876	16.247	16.805
LSTM	5.404	5.732	6.523	10.521	13.012	13.743	14.002	14.297	14.919	16.154
Seq2Seq	5.021	5.469	5.942	8.020	8.713	11.364	11.151	11.506	11.723	11.946
AG2S-Net	4.810	5.347	5.824	8.3098	8.817	11.142	11.534	12.713	12.790	12.212
DGLSTM	4.202	5.065	6.223	8.824	10.343	10.255	10.813	11.246	11.458	11.654
MTLG-Net	**4.062**	**4.331**	**5.414**	**7.341**	**8.863**	**9.026**	**11.024**	**10.059**	**10.124**	**10.148**

large and small hubs for GCNTL-Net is only 15.9%. The following two best-performing methods, DGLSTM and AG2S-Net, had the difference increased to 30.14% and 95.16%. This same pattern can also be observed throughout the rest of the nine-time interval prediction results. Therefore, our model is more capable of learning the delay pattern in medium to small hubs. The global correlation graph captures the similarity between airports based on their flight and passenger volume properties. By connecting airports of comparable size, the model partly mitigates the drawback of having fewer flight data on smaller air hubs.

6.1.7 Ablation Study

Table 29 demonstrates the ablation study of each component's impact on the overall model performance.

Table 24: Performance comparison of MTLG-Net with different baselines for the average hourly departure delay prediction on large hubs

Method	MAE									
	1h	2h	3h	4h	5h	6h	7h	8h	9h	10h
ARIMA	5.990	6.238	7.332	10.014	11.230	13.649	14.081	14.679	15.306	16.600
LSTM	5.631	5.826	6.220	10.174	12.653	13.362	13.611	13.907	14.515	16.010
Seq2Seq	5.233	5.431	5.996	8.620	9.613	11.200	11.832	12.819	12.829	12.903
AG2S-Net	4.993	5.037	5.990	8.133	9.483	10.667	10.314	12.420	12.467	12.539
DGLSTM	4.311	4.752	5.830	7.872	9.015	10.031	10.125	11.148	11.461	11.763
MTLG-Net	**4.187**	**4.520**	**5.606**	**7.547**	**8.980**	**9.828**	**9.974**	**10.462**	**11.147**	**11.425**

Global Correlation GCN o/w global GCN test is configured by only including the local correlated GCN, which uses flight route connectivity to build the adjacency matrix. This test determines how a GCN constructed on global features, such as the hourly passenger volume, will help the model learn the additional variable dependency. As shown in the table 29, for w/o global GCN, the 1-hour ahead prediction MAE increased by 18.38% for arrival delay and 10.8% for departure delay, yielding the most significant impact on the model performance, followed by not applying HGF (w/o HGF). A common goal for the global correlation GCN and HGF is to capture the more complex dependency between variables. The patterns and relationships that the local connection fails to capture will be lost in a graph network if the global features are not adequately learned, which explains the poor model performance.

Multitask Learning Under the w/o MTL scenario, the model is only trained

Table 25: Performance comparison of MTLG-Net with different baselines for the average hourly departure arrival prediction on medium hubs

Method	MAE									
	1h	2h	3h	4h	5h	6h	7h	8h	9h	10h
ARIMA	8.452	9.132	10.351	12.945	14.153	16.843	16.853	17.357	17.327	18.416
LSTM	8.003	8.414	9.149	13.134	15.462	16.343	16.615	16.912	17.535	18.794
Seq2Seq	7.754	8.011	8.432	11.152	11.335	13.986	13.773	14.128	14.360	14.531
AG2S-Net	7.312	7.824	8.313	10.914	11.415	13.100	14.032	15.345	15.343	15.259
DGLSTM	5.526	6.334	7.563	10.106	10.591	12.251	12.759	13.218	13.463	13.607
MTLG-Net	**4.784**	**5.053**	**6.136**	**8.049**	**9.544**	**10.453**	**11.741**	**11.799**	**11.520**	**11.817**

to predict one delay type. We would like to know how much performance gain can be achieved by training two related tasks simultaneously. We also would like to compare the result of w/o with w/o DMTL, the next test that adopts the common MTL strategy by allocating equal weight to both tasks' losses when calculating the final training loss. It will be interesting to learn the impact of dynamically allocating resources on sample and task levels during training on a graph-based model architecture. Completely removing MTL (w/o MTL) produces a lesser impact on the model's performance (0.6% decrease in arrival delay and 2.53% decrease in departure delay prediction accuracy) when compared to w/o DMTL (3.3% decrease in arrival delay and 4.93% decrease in departure delay prediction accuracy). This may be counter-intuitive at first glance. However, treating tasks equally in an MTL scenario may not be beneficial or harm the model performance if one task dominates the training process. The DMTL strategy automatically allocates resources to

Table 26: Performance comparison of MTLG-Net with different baselines for the average hourly departure delay prediction on medium hubs

Method	MAE									
	1h	2h	3h	4h	5h	6h	7h	8h	9h	10h
ARIMA	8.222	9.104	10.135	12.845	14.032	16.449	16.881	17.281	18.526	19.402
LSTM	7.920	8.199	9.209	13.015	15.501	16.207	16.230	16.731	17.337	18.734
Seq2Seq	7.534	8.004	8.524	11.020	11.316	13.679	13.733	14.227	14.350	14.237
AG2S-Net	7.423	7.638	8.133	10.954	11.366	13.156	14.213	15.360	15.353	15.207
DGLSTM	5.524	6.346	7.531	10.063	10.614	12.234	12.834	13.233	13.463	13.737
MTLG-Net	**4.708**	**5.020**	**6.134**	**9.046**	**9.489**	**10.451**	**11.694**	**11.763**	**11.447**	**11.743**

samples and tasks with a more challenging time learning during the training process.

Multimodal Fusion w/o HGF test applies the standard concatenation operation when fusing features from different data sources or combining intermediate outputs from the two GCNs. The hierarchical graph fusion strategy adopted by HGF could capture more complex cross-modality dependencies among input channels. HGF aims to exploit the cross-modal correlation between input modalities. Removing HGF limits the model's ability to capture complex joint feature representations among input sources. Consequently, the prediction accuracy decreased by 7.65% in arrival delay and 9.24% in departure delay.

Meteorological Data Table 30 illustrates the impact of meteorological input variables on the model performance. As can be observed in the table, weather variables such as average hourly wind speed, precipitation, temperature, and visibility substantially impacted the prediction results. For arrival delay prediction, the one-hour-head prediction

Table 27: Performance comparison of MTLG-Net with different baselines for the average hourly arrival delay prediction on small hubs

Method	MAE									
	1h	2h	3h	4h	5h	6h	7h	8h	9h	10h
ARIMA	10.678	1.242	12.463	15.043	16.353	18.643	18.837	18.997	18.916	18.856
LSTM	10.171	10.457	11.203	15.031	17.250	18.316	18.356	18.501	18.247	18.520
Seq2Seq	9.764	10.133	10.410	13.248	13.367	15.835	15.737	16.119	16.273	16.330
AG2S-Net	9.387	9.728	10.433	12.945	13.427	15.113	16.101	17.336	17.297	17.259
DGLSTM	5.501	6.323	7.312	10.120	11.013	12.440	13.122	14.738	14.952	15.007
MTLG-Net	**4.873**	**5.142**	**6.245**	**8.134**	**9.435**	**10.542**	**11.653**	**11.700**	**11.434**	**11.830**

accuracy decreased by 5.23% without including the meteorological variables and decreased by 6.25% for departure delay. Overall, the average accuracy decreased by 4.19% across the entire 10 hours of time steps.

6.2 Adaptive Joint Spatio-Temporal Graph Learning Network

Recently, graph convolutional network (GCN) has seen extensive applications on graph-structured data [176, 190, 205]. GCN applies convolution on neighboring nodes based on the adjacency/correlation matrix forming the network topology. Most existing GCN-based models only use pre-defined node relationships to construct the adjacency matrix [62, 75, 212]. The pre-defined graph network is often inferred from the physical route connection and specific distance measurements. Due to the complex nature of traffic forecasting problems, such an intuitive graph structure cannot capture diverse traffic

Table 28: Performance comparison of MTLG-Net with different baselines for the average hourly departure delay prediction on small hubs

Method	MAE									
	1h	2h	3h	4h	5h	6h	7h	8h	9h	10h
ARIMA	10.235	11.258	12.015	14.865	16.712	17.047	17.963	18.762	18.856	19.602
LSTM	9.940	10.345	11.015	14.545	16.031	16.537	17.357	18.317	18.335	18.434
Seq2Seq	9.525	10.103	10.452	13.035	13.425	15.691	15.745	16.221	16.417	16.217
AG2S-Net	9.516	9.525	10.135	12.963	13.353	15.205	16.191	17.295	17.302	17.200
DGLSTM	5.516	6.335	7.510	10.041	10.908	12.221	12.815	12.728	12.942	13.262
MTLG-Net	**4.735**	**5.101**	**6.124**	**9.532**	**10.754**	**11.983**	**12.026**	**12.061**	**12.156**	**12.341**

patterns. Furthermore, domain knowledge is often required to develop a high-quality pre-defined graph topology, significantly limiting its application to other problem domains. Some studies [15, 43, 210] attempted to learn the adjacency matrix using data-driven methods to circumvent the drawbacks of a pre-defined graph. However, most of these adjacency matrices are too dense to be optimized efficiently. Furthermore, the learned adjacency matrix is prone to noise and fine-scale roughness, which increases the difficulty of learning the hidden traffic patterns from the data.

This section addresses the aforementioned challenges by proposing AJSTGL, a novel adaptive joint spatio-temporal graph learning network [180]. First, we apply a shifted graph Laplacian approach to expand the sensitivity of the pre-defined graph. The transformed graph Laplacian can capture more granular hidden patterns and still leverage the original graph structure's knowledge. Second, adaptive graph learning and

graph regularization methods are applied to learn the network topology in a data-driven manner and improve the graph quality. Furthermore, we adopt a node-specific dependency modeling module to replace the conventional graph convolutional layer so that the model can learn node-specific patterns. A sequence-to-sequence fusion module is developed to encode multiple graph signals in parallel and hierarchically combine them to learn the short-term temporal node dependencies. We also develop the spatio-temporal graph transformer module to complement the sequence-to-sequence fusion module by dynamically capturing the time-evolving spatial node relations in long-term predictions.

The main contribution of this work can be summarized as follows.

- We propose a novel graph learning network for traffic data forecasting. It utilizes static, adaptive, auxiliary, and dynamic spatial convolutional graphs to capture the spatio-temporal dependencies.
- We present a new adaptive graph learning method to learn the network topology in a data-driven manner and capture the graph signals in both directions.
- We design a sequence-to-sequence fusion module and a dynamic temporal convolutional graph for jointly learning the short- and long-term temporal relations.
- We develop several graph learning techniques, such as shifted graph Laplacian and node-specific dependency modeling, to help the model capture more complex hidden patterns and improve the graph quality.
- We evaluate our model on three large-scale traffic datasets and compare it with several baseline methods. The experimental results demonstrate the excellent performance

of our approach compared to other state-of-the-art techniques.

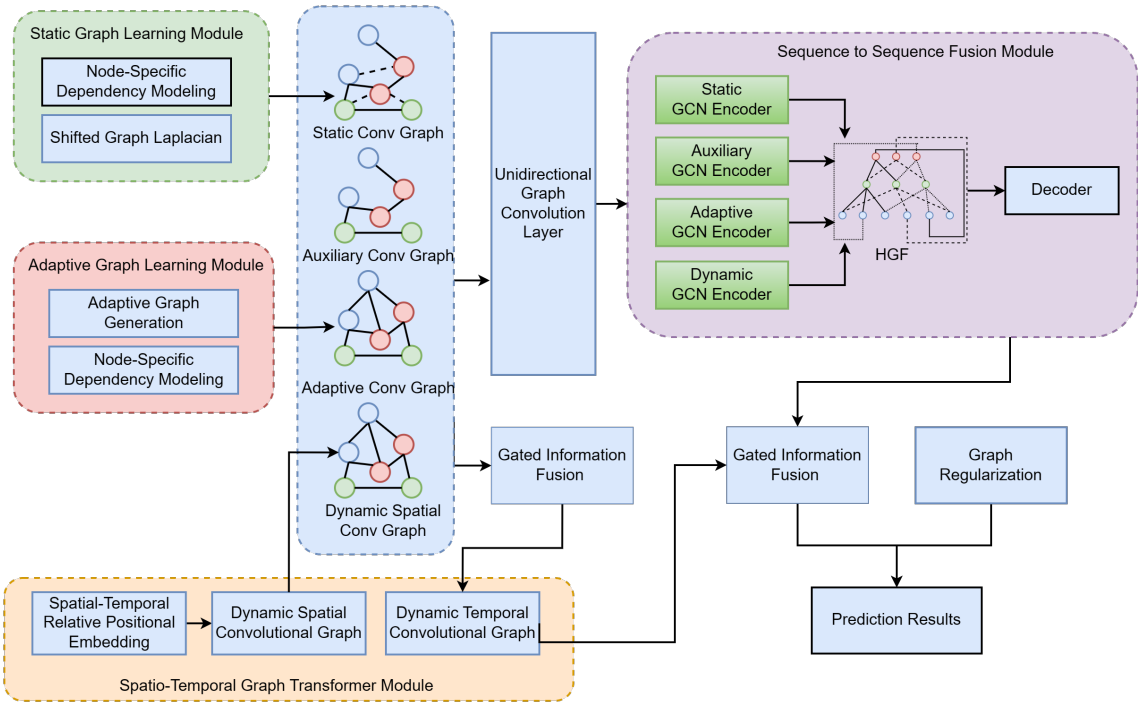


Figure 20: An overview of the proposed framework.

6.2.1 Architecture Design

In this section, we introduce the proposed AJSTGL and its main components. Figure 20 illustrates the overall structure of the framework. AJSTGL mainly consists of the static graph learning module, adaptive graph learning module, spatio-temporal graph transformer module, and sequence-to-sequence fusion module. In addition, we apply several techniques, such as unidirectional graph convolution, gated information fusion, and graph regularization, to enhance the model's ability to model spatio-temporal dependencies and produce higher-quality graphs.

6.2.1.1 Preliminaries

Given a graph $G = (V, E, X, A)$, where $|V| = N$ represents the set of nodes, E represents the edges, $X \in \mathbb{R}^{N \times Q}$ is the node feature with Q as the vector size, and $A \in \mathbb{R}^{N \times N}$ is the adjacency matrix that defines the topology of the graph network. The normalized graph Laplacian with self-loop can be represented as follows.

$$L = I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{6.13}$$

where $I_N \in \mathbb{R}^{N \times N}$ is the identity matrix and D represents the degree matrix of A. To reduce the computation complexity on a large graph, the first-order Chebyshev polynomial approximation [90] of the graph convolution can be described as follows.

$$X^{l+1} = (I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) X^l \Theta + b \tag{6.14}$$

where $X^l \in \mathbb{R}^{N \times Q}$ and $X^{l+1} \in \mathbb{R}^{N \times Z}$ are the input and output at layers l and $l+1$ with Q and Z as the corresponding vector sizes, and $\Theta \in \mathbb{R}^{Q \times Z}$ and $b \in \mathbb{R}^Z$ are the weight and bias terms, respectively.

6.2.1.2 Static Graph Learning Module

The static graph learning module combines node connectivity and geographical distance to model the spatial node dependencies.

Node-Specific Dependency Modeling

In a standard graph convolution layer, the weights are shared among all nodes when the convolution operation is applied. However, in many real-world problems, the shared

weights could become a constraint since some heterogeneous spatial patterns also exist among nearby nodes. For instance, airports with connected flight routes or are geographically closely located may exhibit diverse traffic patterns due to the influence of weather conditions and special events. In this case, it is essential to incorporate parameters to model additional node-specific properties.

We conduct node-specific dependency modeling by adding an extra dimension to the weight parameter, which results in a node-specific weight parameter $\hat{\Theta} \in \mathbb{R}^{N \times Q \times Z}$. However, the expanded weight parameter significantly increases the computational cost during optimization, especially for graphs with a large node count N. To solve this problem, we apply graph parameter decomposition to factorize Θ into δ_s and W_s, where $\delta_s \in \mathbb{R}^{N \times d_s}$ with $d_s << N$ and $W \in \mathbb{R}^{d_s \times Q \times Z}$. The updated graph convolution layer can be expressed as:

$$X^{l+1} = (I_N + D^{-\frac{1}{2}} A D^{-\frac{1}{2}}) X^l \delta_s W_s + \delta_s b_s \tag{6.15}$$

where $b_s \in \mathbb{R}^{d_s \times Z}$ is the updated bias term.

Shifted Graph Laplacian

Intuitive node relations are commonly used to model the spatial node dependency in a pre-defined network topology. Nevertheless, hidden patterns exist that the pre-defined graph schema cannot capture. Thus, we develop a strategy to introduce a shifted graph Laplacian L_s on top of the intrinsic graph Laplacian L to learn the hidden spatial node dependencies.

The adjacency matrix in a shifted graph Laplacian uses the node-wise Gaussian smoothed similarity score as the values. In this work, the similarity score is measured by

the cosine similarity between each pair of nodes. The shifted graph A_s adjacency matrix is represented as follows.

$$A_s^{ij} = \begin{cases} exp(-\frac{cos(x_i,x_j)^2}{2\sigma^2}), & i \neq j \; and \; cos(x_i, x_j) \geqslant \varepsilon_s \\ \\ 0, & otherwise \end{cases} \tag{6.16}$$

where $cos(x_i, x_j)$ is the cosine similarity score between nodes x_i and x_j, σ is the standard deviation, and ε_s is the threshold parameter that controls the matrix sparsity. We empirically set ε_s to 0.5 to avoid the matrix from becoming overly sparse. Then, the shifted graph Laplacian L_s can be expressed as follows.

$$L_s = I_N + D_s^{-\frac{1}{2}} A_s D_s^{-\frac{1}{2}} \tag{6.17}$$

where D_s is the degree matrix of A_s. Finally, the graph Laplacian for the static convolutional graph can be calculated as:

$$\widetilde{L} = L + \alpha L_s \tag{6.18}$$

where α is a learnable parameter that controls the shifting scale on the original graph Laplacian. Based on Equation 6.18, Equation 6.15 can be transformed to:

$$X^{l+1} = \widetilde{L} X^l \delta_s W_s + \delta_s b_s \tag{6.19}$$

6.2.1.3 Auxiliary Convolutional Graph

We develop an auxiliary convolutional graph to exploit a traffic network's contextual information such as wind, humidity, and temperature. The adjacency matrix of the auxiliary

convolutional graph is generated based on the node's geographic proximity. To ensure the matrix symmetrical property, a Gaussian kernel function is applied to produce the edge weights. Similar to Equation 6.16, the adjacency matrix A_a can be expressed as follows.

$$A_a^{ij} = \begin{cases} exp(-\frac{dist(x_i,x_j)^2}{2\sigma^2}), & i \neq j \ and \ dist(x_i, x_j) \leqslant \varepsilon_c \\ \\ 0, & otherwise \end{cases} \tag{6.20}$$

where $dist(x_i, x_j)$ is the spherical distance between nodes x_i and x_j, σ is the standard deviation, and ε_c is the threshold parameter that controls the matrix sparsity. We empirically set ε_c to 0.4 for optimal model performance.

6.2.1.4 Adaptive Graph Learning Module

Many existing studies only use pre-defined adjacency matrices to represent the node relations. This limits the model's ability to learn more complex spatio-temporal graph patterns. Another disadvantage of the conventional graph convolutional layer is that it assumes bidirectional node correlations. However, in many real-world problems, such as traffic data forecasting, the changes in traffic patterns may only transmit in a single direction.

To address the aforementioned limitations, we propose an adaptive graph learning approach. It automatically learns the hidden graph topology from the input data. Furthermore, the learned graph adjacency matrix considers the unidirectional correlations between each pair of nodes. First, we use an anti-symmetric matrix to construct the normalized

graph Laplacian of the new graph L_a:

$$L_a = I_N + softmax(ReLU(M_p M_q^T - M_q M_p^T)) \tag{6.21}$$

where $M_p, M_q \in \mathbb{R}^{N \times d_a}$, $d_a << N$ are the learned node embedding that formulates the adjacency matrix through the training process. Being the product of the two anti-symmetric matrices, $M_p M_q^T - M_q M_p^T$ has zero values in all of its diagonal elements. The $ReLU()$ function further transforms another half of the negative matrix elements to zeroes. The $softmax()$ function normalizes the adjacency matrix. Additionally, we apply node-specific dependency modeling from Equation 6.15 to enable the model to learn node-specific dependencies. Finally, the GCN layer in the adaptive convolutional graph can be expressed as follows:

$$X^{l+1} = L_a X^l \delta_a W_a + \delta_a b_a \tag{6.22}$$

6.2.1.5 Unidirectional Graph Convolution Layer

A unique graph convolution layer transformation approach is developed to further enhance the model's capability of learning the heterogeneous unidirectional data patterns. The proposed unidirectional graph convolution layer can be represented as follows.

$$X^{l+1} = (\hat{L} X^l \hat{\delta} \hat{W} + \hat{\delta} \hat{b}) \oplus (\hat{L}^T X^l \hat{\delta} \hat{W} + \hat{\delta} \hat{b}) \tag{6.23}$$

The transformed graph convolutional layer adds a second term $\hat{L}^T X^l \hat{\delta} \hat{W} + \hat{\delta} \hat{b}$ with transposed graph Laplacian $\hat{L}^T$ to model the reversed data flow pattern.

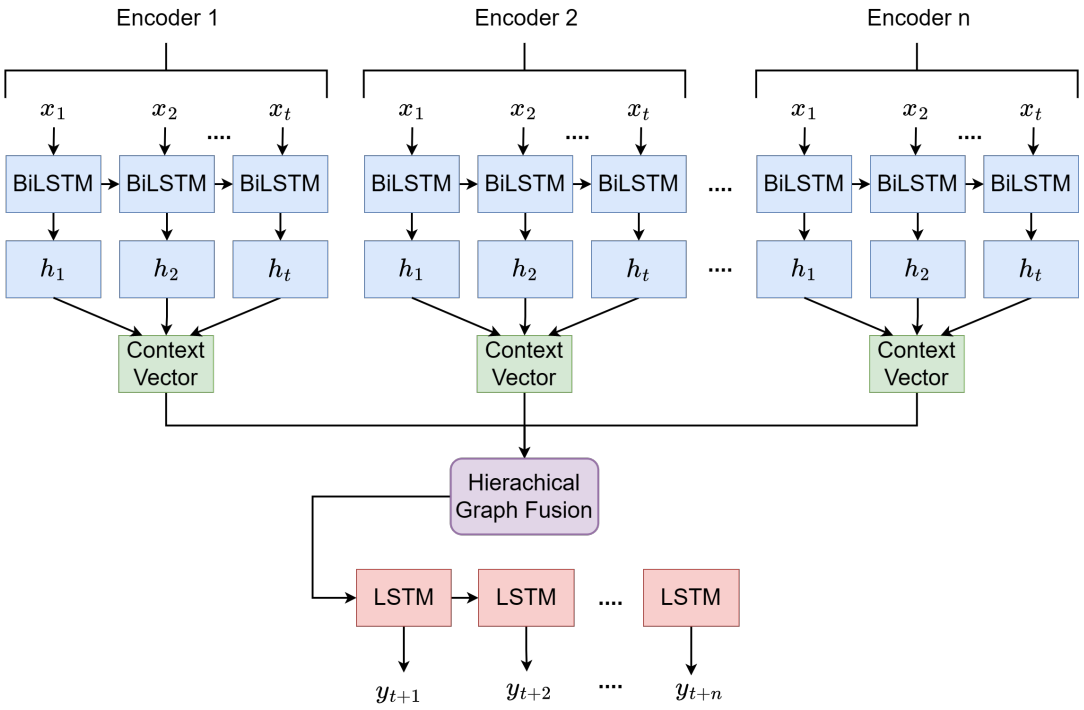


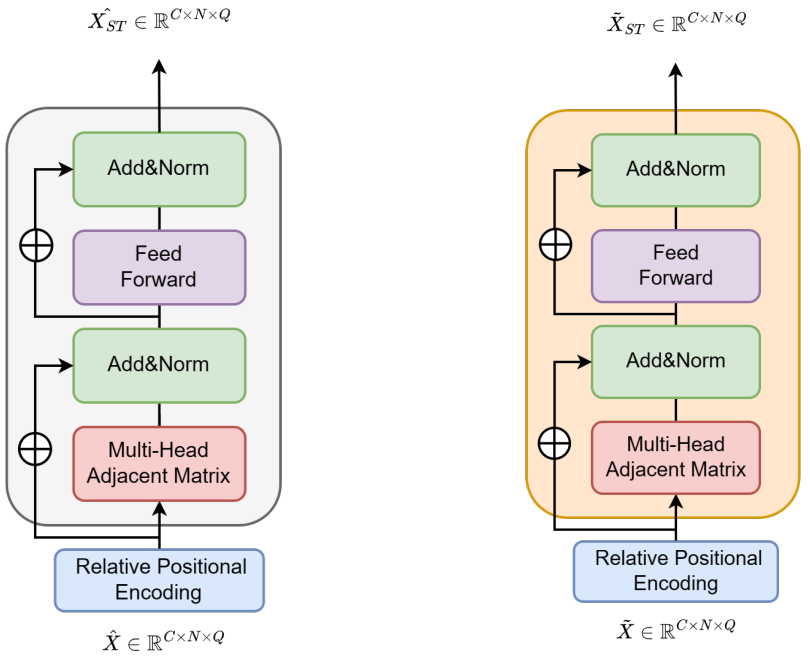
Figure 21: Illustration of the overall structure of S2SFM.

6.2.1.6 Sequence-to-Sequence Fusion Module

We develop a sequence-to-sequence fusion Module (S2SFM) to capture the short-term temporal node dependencies. Figure 21 demonstrates the overall structure of the proposed S2SFM. Unlike typical sequence-to-sequence models with a single encoder, S2SFM utilizes parallelized Bi-LSTM based encoders to concurrently process the graph signals from all GCNs. The context vectors of all encoders are combined using the hierarchical graph fusion (HGF) approach [178] to capture the n-modal cross-modality interactions among all graph signals. HGF utilizes a tree-based graph structure to join the input signals on different levels. It learns the unique joint-modality representations, with lower-level nodes representing basic interactions and higher-level nodes modeling more complex correlations. The final output of HGF can be represented as follows.

$$F_{combined} = F_{uni-modal} \oplus F_{bi-modal} \oplus F_{tri-modal} ... \oplus F_{n-modal} \tag{6.24}$$

where $F_{combined}$ is the final combined context vector, $F_{uni-modal}$, $F_{bi-modal}$, $F_{tri-modal}$, and $F_{n-modal}$ are the representation vectors for each level, and $\oplus$ is the concatenation operation. In this study, we use HGF to combine the output signals of four GCNs. Therefore, the first level contains the uni-modal interactions, the second level learns the bi-modal interactions, the third level models tri-modal interactions, and the fourth level captures the quad-modal interactions. The combined context vector is then passed into the decoder to produce the output sequence.



(a) Dynamic Spatial Convolutional Graph

(b) Dynamic Temporal Convolutional Graph

Figure 22: Illustration of the architecture of the proposed spatio-temporal graph transformer module (STGTM). The relative positional encoding learns the node's spatial and temporal dependency and generates the spatio-temporal aware embedding vectors. The dynamic spatial and temporal convolutional graphs are stacked together to model the node relations jointly.

6.2.1.7 Spatio-Temporal Graph Transformer Module

To complement S2SFM in long-term prediction, we develop the spatio-temporal graph transformer module (STGTM) to jointly model the dynamic spatio-temporal dependencies. Its overall structure is demonstrated in Figure 22. STGTM consists of spatio-temporal relative positional encoding, which generates the time-evolving spatial embedding of the graph signals, and the dynamic spatial and temporal convolutional graphs that model the spatial and temporal node dependencies, respectively.

Spatio-Temporal Relative Positional Encoding

Transformer-based model applies positional encoding to embed the spatial relations of each token in the input sequence. Static positional encoding approaches, such as the widely adopted sinusoidal wavelength method [170], rely on fixed-length input sequences and do not consider their relative positional relations. In STGTM, we apply a relative positional encoding layer with trainable parameters to learn the dynamic spatial and temporal dependencies. More specifically, matrices $PE_S \in \mathbb{R}^{N \times N}$ and $PE_T \in \mathbb{R}^{C \times C}$ in the encoding layer hold the learned spatial and temporal relative positional embedding, where C represents the number of time steps in the input sequence. The embedded input feature vector can be represented as:

$$X_E = conv(\hat{X} \oplus PE_S^{'} \oplus PE_T^{'}) \tag{6.25}$$

where $\hat{X} \in \mathbb{R}^{C \times N \times Q}$ is a 3-D vector that contains the features of all nodes across the entire historical time sequence, and $PE_S^{'} \in \mathbb{R}^{C \times N \times N}$ and $PE_T^{'} \in \mathbb{R}^{C \times N \times C}$ are the expanded matrices of PE_S and PE_T along the spatial and temporal dimensions, respectively. $conv()$

is a 1×1 convolutional layer that converts the original input vector $\hat{X}$ into its spatio-temporal embedded form $X_E \in \mathbb{R}^{C \times N \times Q}$.

Dynamic Spatial Convolutional Graph

The dynamic spatial convolutional graph (DSCG) learns the adjacency matrix from the time-evolving hidden patterns in the positional embedded input sequence. We apply a multi-head self-attention mechanism to capture the pattern representations from different subspaces and form the adjacency matrix. Figure 23 demonstrates the structure of the multi-head adjacency matrix that applies the self-attention mechanism. The subspaces used in DSCG contains the query $Q_S \in \mathbb{R}^{N \times d_k}$, key $K_S \in \mathbb{R}^{N \times d_k}$, and value $V_S \in \mathbb{R}^{N \times d_v}$ spaces, where d_k is the query size and key vector size, and d_v is the value vector size. Each matrix is calculated as the product of the input feature vector and its corresponding weight parameter:

$$Q_S = X_E^{'} W_Q,\ K_S = X_E^{'} W_K,\ V_S = X_E^{'} W_V \tag{6.26}$$

where $X_E^{'} \in \mathbb{R}^{N \times Q}$ represents a single time step in X_E, $W_Q \in \mathbb{R}^{Q \times d_k}$, $W_K \in \mathbb{R}^{Q \times d_k}$, and $W_V \in \mathbb{R}^{Q \times Q}$ are the learnable weight parameters for Q_S, K_S, and V_S, respectively.

The adjacency matrix of DSCG for each attention head unit can be derived from the scaled dot-product among the query, key, and value matrices:

$$A_d^i = softmax(\frac{Q_S^i K_S^{iT}}{\sqrt{d_k}}) V_S^i \tag{6.27}$$

where $softmax()$ is used to obtain the normalized spatio-temporal node-wise dependency in the ith attention head A_d^i, and $\sqrt{d_k}$ serves as the scaling factor to stabilize the gradients during the training. The multi-head attention unit is generated by concatenating each

attention head unit:

$$\hat{A}_d = concat(A_d^1, A_d^2, ..., A_d^i)W_O \tag{6.28}$$

where $A_d^1, ..., A_d^i$ are the single-head attention units, and W_O is the learnable weight parameter. The output of the node input feature through the multi-head attention unit is then calculated as follows.

$$\hat{X_E} = \hat{A}_d X_E \tag{6.29}$$

We add residual connection and layer normalization to help improve the model stability and generalization performance:

$$\hat{X_E'} = LN(X_E + \hat{X_E}) \tag{6.30}$$

where $LN()$ represents layer normalization [11]. Then the output is fed into two feed-forward layers with the ReLU activation function:

$$X_{ST} = ReLU(ReLU(\hat{X_E'}W_a + b_a)W_b + b_b) \tag{6.31}$$

where W_a, W_b, b_a, and b_b are the weight parameters. In the last step, we apply residual connection and layer normalization again on the output of the two feed-forward layers to generate the final feature vector $\hat{X_{ST}} \in \mathbb{R}^{C \times N \times Q}$ of STGTM:

$$\hat{X_{ST}} = LN(X_{ST} + \hat{X_E'}) \tag{6.32}$$

Dynamic Temporal Convolutional Graph

The dynamic temporal convolutional graph (DTCG) shares a similar network structure to DSCG. We first apply residual connection between the original input vector X and

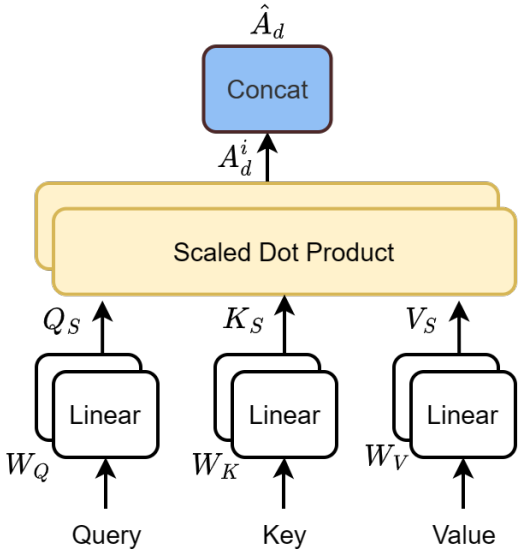


Figure 23: The multi-head adjacency matrix structure.

DSCG output $\hat{X_{ST}}$ to get $\tilde{X} \in \mathbb{R}^{C \times N \times Q}$. The relative positional encoding layer is applied to produce embedding results using the temporal encoding PE_T. $\tilde{X}$ and PE_T are concatenated and passed through a 1×1 convolutional layer to generate the temporal encoded vector $\tilde{X}_E$. Similar to DSCG, the multi-head self-attention mechanism is applied to capture the pattern representations from $\tilde{Q}_S$, $\tilde{K}_S$, and $\tilde{V}_S$ subspaces:

$$\tilde{Q}_S = \tilde{X}'_E \tilde{W}_Q, \ \tilde{K}_S = \tilde{X}'_E \tilde{W}_K, \ \tilde{V}_S = \tilde{X}'_E \tilde{W}_V \tag{6.33}$$

where $\tilde{X}'_E \in \mathbb{R}^{C \times Q}$ represents an arbitrary node in $\tilde{X}_E$, $\tilde{W}_Q \in \mathbb{R}^{Q \times \tilde{d}_k}$, $\tilde{W}_K \in \mathbb{R}^{Q \times \tilde{d}_k}$, and $\tilde{W}_V \in \mathbb{R}^{Q \times Q}$ are the trainable weight parameters for $\tilde{Q}_S$, $\tilde{K}_S$, and $\tilde{V}_S$. Then, the attention-aware adjacency matrix based on a single attention unit can be expressed as:

$$\tilde{A}_d^i = softmax(\frac{\tilde{Q}_S^i \tilde{K}_S^{iT}}{\sqrt{d_k}}) \tilde{V}_S^i \tag{6.34}$$

Similar to DSCG, the final multi-head attention context vector is generated by concatenating each single-head unit:

$$\tilde{A}_d = concat(\tilde{A}_d^1, \tilde{A}_d^2, ..., \tilde{A}_d^i)\tilde{W}_O \tag{6.35}$$

Next, the intermediate output $\tilde{X}_E$ is multiplied with the multi-head unit to produce the attention weighted vector:

$$\bar{X}_E = \tilde{A}_d \tilde{X}_E \tag{6.36}$$

The attention weighted vector is fed into two feed-forward layers with residual connection and layer normalization:

$$\bar{X}_E^{'} = LN(\tilde{X}_E + \bar{X}_E) \tag{6.37}$$

$$\bar{X}_{ST} = ReLU(ReLU(\bar{X}_E^{'}\bar{W}_a + \bar{b}_a)\bar{W}_b + \bar{b}_b) \tag{6.38}$$

$$\tilde{X}_{ST} = LN(\bar{X}_{ST} + \bar{X_E^{'}}) \tag{6.39}$$

where $\tilde{X}_{ST} \in \mathbb{R}^{C \times N \times Q}$ is the final output of STGTM, which will be combined with the output of S2SFM in the gated information fusion module.

6.2.1.8 Gated Information Fusion

We apply a gating mechanism to adaptively combine the output features of S2SFM and STGTM. More specifically, a set of learnable parameters are used as a gate to control the relative weighting between the two input vectors:

$$\mathbb{G} = sigmoid((\gamma(\tilde{X}_{ST}) \oplus \gamma(X_{S2SFM}))W_g + b_g) \tag{6.40}$$

where the sigmoid activation function is used to limit the value of gate $\mathbb{G}$ within range $[0, 1]$, $\gamma()$ is a linear projection function that transforms the feature vectors from both

modules into a 1-D vector, and W_g and b_g are the weight parameters. As a result, we can use $\mathbb{G}$ to fuse the two feature vectors as:

$$X_f = \mathbb{G} \odot \tilde{X}_{ST} + (1 - \mathbb{G}) \odot X_{S2SFM} \tag{6.41}$$

where $\odot$ is element-wise multiplication. $X_f \in \mathbb{R}^{C \times N \times Q}$ represents the joint spatio-temporal dependencies that will be used for the final prediction.

6.2.1.9 Graph Regularization

We apply graph regularization [78] to improve the quality of the learned graphs. The regularization function is expressed as follows.

$$J_{gr} = \sum_{i=1}^{N} \sum_{j=1}^{N} \|x_i - x_j\|_2^2 \, A_a^{ij} + \beta (A_a^{ij})^2 \tag{6.42}$$

where A_a^{ij} is the corresponding adjacency matrix element for nodes x_i and x_j, and β is a scaling factor controlling the matrix sparsity. Term $\|x_i - x_j\|_2^2$ enforces the graph proximity property by encouraging larger A_a^{ij} values when x_i and x_j are close and smaller A_a^{ij} values when the two nodes move away in the latent space.

6.2.1.10 Prediction Layer and Loss Function

To generate the multi-step prediction, the output of the gated information fusion module X_f is passed into a double-layered convolutional network:

$$Y = conv(conv(X_f)) \tag{6.43}$$

X_f is transformed into a 2-D vector $Y \in \mathbb{R}^{N \times \tau}$, where τ is the number of time steps in the output sequence.

The loss function adopted in this study minimizes the mean absolute error (MAE) between the ground truth value and the predicted value. The graph regularization introduced in Equation 6.42 is added to the MAE loss function as a regularization term. The final loss function is expressed as follows:

$$L_{loss} = \left\| Y - \hat{Y} \right\|_1 + \lambda J_{gr} \tag{6.44}$$

where $\hat{Y}$ is the ground truth value, and λ is a scaling factor that controls the degree of regularization. We empirically set λ to 0.4 to achieve the best model performance.

6.2.2 Experiments and Analyses

6.2.2.1 Datasets

We evaluate AJSTGL on several large scale real-world datasets: Reporting Carrier On-Time Performance, PeMSD4, and PeMSD8.

- **Reporting Carrier On-Time Performance (RCOTP)**. Released by the United States Bureau of Transportation Statistics (BTS), this dataset contains flight operation information for all reporting airlines in the U.S. domestic market. Records from January 2017 to December 2021 were collected, which include 433 airports and 30,940,455 records. Among them, 8102 origin and destination (OD) pairs are retrieved. Figure 19 visualizes the major airports based on the volumes of their connecting flights. In the experiment, we predict the multi-step flight arrival delays at the airport level.

- **PeMSD4**. This dataset includes historical traffic condition data in the San Francisco Bay area published by the Caltrans Group using the Performance Measurements

System (PeMS). The data was collected in 5-minute intervals using 307 sensors on seven major roads. The period covered by PeMSD4 ranges from January 2018 to February 2018. Three types of measurements are used, which include average speed, average occupancy, and traffic flow. In this study, we focus on traffic flow and traffic speed predictions.

- **PeMSD8**. Also published by Caltrans Group, this dataset contains the traffic information in the San Bernardino area from July 2016 to August 2016. The 170 sensors used 5-minute intervals on eight roads to collect the average speed, average occupancy, and traffic flow information. Like PeMSD4, we focus on traffic flow and speed predictions in this study.

We also utilize real-time weather forecasting records from National Climate Data Center (NCDC). Weather conditions collected by nearby weather stations at each traffic network node in the same period are used as the input for the auxiliary convolutional graph network.

6.2.2.2 Experimental Setup

The RCOTP data is aggregated at the airport level to produce the average hourly flight delay. For PeMSD4 and PeMSD8, we aggregate the traffic speed at 5-minute intervals. Data preprocessing is done on all datasets: 1) missing values are interpolated as the mean value of the previous and later time steps; 2) categorical and discrete values are one-hot encoded; and 3) continuous values are normalized using min-max normalization. For RCOTP, we conduct arrival delay prediction in the next ten-hour horizons. For PeMSD4

and PeMSD8, we perform traffic flow prediction and traffic speed forecast in the next 15, 30, 45, and 60 minutes horizons.

All datasets are split into 60% for training, 20% for validation, and 20% for testing. Hyperparameters including the threshold parameter ε_s in the shifted graph Laplacian adjacency matrix, ε_c in the auxiliary GCN adjacency matrix, node embedding size d_s, β in the graph regularization, and λ in the final loss function are tuned on the validation set.

The model is trained using the Adam optimizer [88] with a training rate of 0.01 and batch size of 128. Early stopping is applied to prevent the model from overfitting.

6.2.2.3 Evaluation Metrics and Baseline Methods

To evaluate the proposed AJSTGL, we adopt mean absolute error (MAE) and root mean squared error (RMSE). Several baselines are utilized to compare with our proposed method:

- **Autoregressive Integrated Moving Average (ARIMA)** [155]: a statistical analysis model that uses previous time step values to predict future values.
- **DCRNN** [102]: a deep neural network with an encoder-decoder structure that combines graph convolution with diffusion operation and RNNs for multi-step prediction.
- **STGCN** [192]: a graph convolutional network utilizing GCN to model spatial correlations and a temporal convolution network to capture temporal dependencies.

- **ASTGCN** [62]: a spatio-temporal convolutional graph network leveraging an attention mechanism to model spatial and temporal dependencies.
- **Graph WaveNet** [185]: a spatio-temporal convolutional graph network that applies diffusion convolution to capture spatial dependencies and leverages dilated convolution to model the temporal correlations.
- **AGCRN** [15]: a convolutional graph network learning the correlation matrix from data to capture the spatial dependencies and utilizing a recurrent neural network to learn the temporal correlation from the input data.

6.2.2.4 Experimental Results

Overall Comparison

Table 31 illustrates the model performance of AJSTGL on the three datasets against all baselines. As shown in the table, we report the average arrival delay for RCOTP and traffic flow prediction for PeMSD4 and PeMSD8. The traffic speed prediction results on PeMSD4 and PeMSD8 are presented in Table 33 and Table 34, respectively.

RCOTP. It can be observed that ARIMA performs the worst among all methods as it only captures the temporal dependency in the data. Other spatio-temporal graph models demonstrate more robust performance since they model both the spatial and temporal correlations from the data. Our proposed AJSTGL outperforms all baseline methods and leads the second-best performer (AGCRN) by 15.74% in MAE. It implies that AJSTGL is very effective at long-term forecasting and including auxiliary GCN facilitates the model to capture additional spatial node dependencies.

Furthermore, Table 32 and Figure 24 demonstrate the detailed ten horizon prediction results. AJSTGL performs noticeably better from mid to long-term time steps (sixth to eighth horizons). As we discussed earlier, long-term prediction suffers from diminishing return effects while utilizing historical observations. Therefore, it is essential to leverage contextual information and construct the graph adjacency matrix based on dynamic long-term spatio-temporal dependencies.

PeMSD4 and PeMSD8. The result patterns for AJSTGL and other baselines are similar on PeMSD4 and PeMSD8. Our model achieves the lowest MAE and RMSE scores in both datasets and leads the second-best model AGCRN by 9% in MAE on PeMSD4 and Graph WaveNet by 11.67% in MAE on PeMSD8. In Table 33 and Table 34, AJSTGL once again produces the lowest MAE and RMSE scores among the baselines. It beats the second-best method (STGCN) by 11.9% in MAE on PeMSD4 and the second-best model (DCRNN) by 10.4% in MAE on PeMSD8.

Ablation Study

We conduct an ablation study to further investigate each main component's effectiveness in AJSTGL and their impacts on the model performance. Table 35 presents the component-wise impact on all three datasets. Table 36 and Figure 25 demonstrate the ablation study of arrival delay prediction results in the next ten horizons on RCOTP.

Effect of shifted graph Laplacian (w/o SGL): In this scenario, the static convolutional graph only utilizes the explicit network topology for the graph adjacency matrix. As demonstrated in Table 35, the MAE of w/o SGL variant increases by 5.18% on average. It implies that the shifted graph Laplacian can effectively help the static GCN capture the

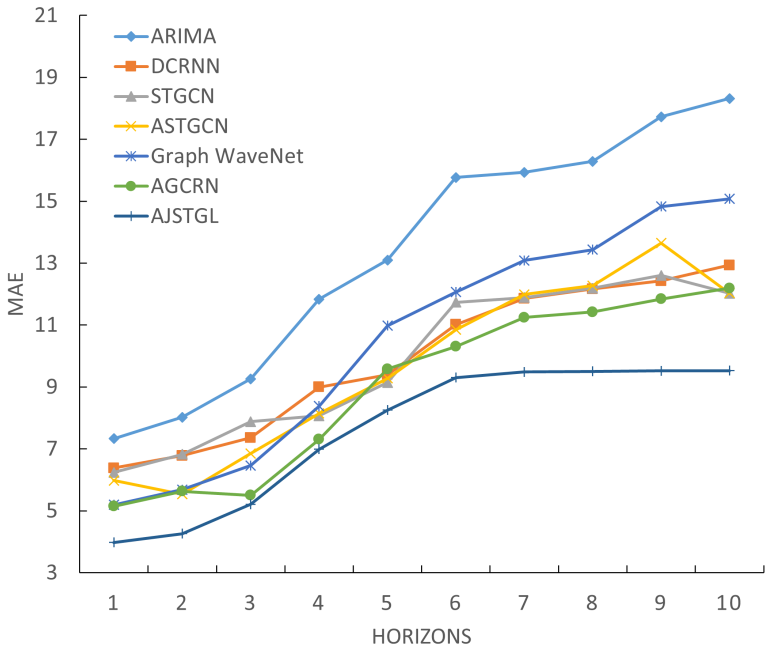


Figure 24: Visulization MAE for the RCOTP dataset obtained by AJSTGL and other baselines. Arrival delay in the next ten horizons

hidden patterns in the graph signal.

Effect of node-specific dependency modeling (w/o NSDM): To investigate the effect of node-specific dependency modeling, we remove the node-specific embedding in the weight parameters of all GCN layers. Results in Table 35 show that the MAE of w/o NSDM increases by 8.78% on average across all datasets. In Table 36 and Figure 25, it can also be observed that the performance gain from NSDM is substantially higher in mid to late horizons. We argue that NSDM helps the model learn the node-related patterns that could compensate for the lack of historical information in long-term prediction.

Effect of adaptive graph learning (w/o AGG): In the w/o AGG test, the GCN

created by the adaptive graph learning module is removed. We want to study the impact of the hidden patterns captured by the adaptive GCN and observe whether it could complement the intrinsic pre-defined adjacency matrix. As illustrated in Table 35, the MAE of w/o AGG increases by 9.5% on average across all datasets. The substantial performance impact suggests that the pre-defined graph structure could benefit significantly from the granular node dependencies captured by AGG.

Effect of graph regularization (w/o GR): Graph regularization enforces the smoothness of the learned graph and further controls the matrix sparsity. The w/o GR test removes the regularization term (set λ to zero in the loss function) and uses only the MAE loss function to optimize the model parameters. As shown in Table 35, the MAE score of w/o GR test increases by 6.84% compared to the baseline. Since AJSTGL heavily relies on learning the spatio-temporal dependencies, it indicates that graph regularization could improve the learned graph quality.

Effect of Unidirectional graph convolution (w/o UC): In AJSTGL, we transform the standard Chebyshev polynomials-based graph convolution layer by concatenating two convolutional layers with transposed graph Laplacian to capture the unidirectional data flow patterns. To evaluate its effectiveness, we use the standard graph convolution layer in all GCNs. Table 35 shows that the MAE score increases by 4.31% on average. It implies that modeling the unidirectional data flow on complex real-world data enables the model to capture the in-flow and out-flow patterns.

Effect of auxiliary GCN (w/o AGCN): We remove the auxiliary GCN, which adopts the spherical distance between nodes as the adjacency matrix to model the contextual

weather conditions. As shown in Table 35, w/o AGCN produces an increase of 6.65% in average MAE score, substantially impacting the overall prediction accuracy. As a result, we observe a substantial correlation between the contextual weather condition and traffic patterns.

Effect of sequence-to-sequence fusion module (w/o S2SFM): We further validate that the proposed S2SFM is capable of learning short-term temporal dependencies. All GCNs are combined with the gated fusion mechanism in this test and passed into DTCG. We remove S2SFM so that AJSTGL solely relies on DTCG to model the temporal node dependency. From Table 35, we can observe a 12.37% increase in average MAE score, which produces the second largest hit on the model performance. Table 36 and Figure 25 further illustrate the impact of S2SFM on short-term prediction. Compared to STGTM, removing S2SFM creates a much greater penalty in short-term prediction performance (first to fourth horizons).

Effect of spatio-temporal graph transformer module (w/o STGTM): The main goal of STGTM is to complement S2SFM on long-term prediction. To evaluate the effectiveness of STGTM, we completely remove it from AJSTGL. Table 35 demonstrates a significant 15.13% increase in average MAE from all datasets. Table 36 and Figure 25 provide more insight from the ten horizons arrival delay prediction results. It can be observed that STGTM excels at mid to long-term predictions (fifth to tenth horizons) when compared to S2SFM. This outcome can be explained from several aspects: 1) instead of relying on the absolute position, the relative positional encoding generates the spatial and temporal embedding of the input sequence in a data-aware manner. It helps the model

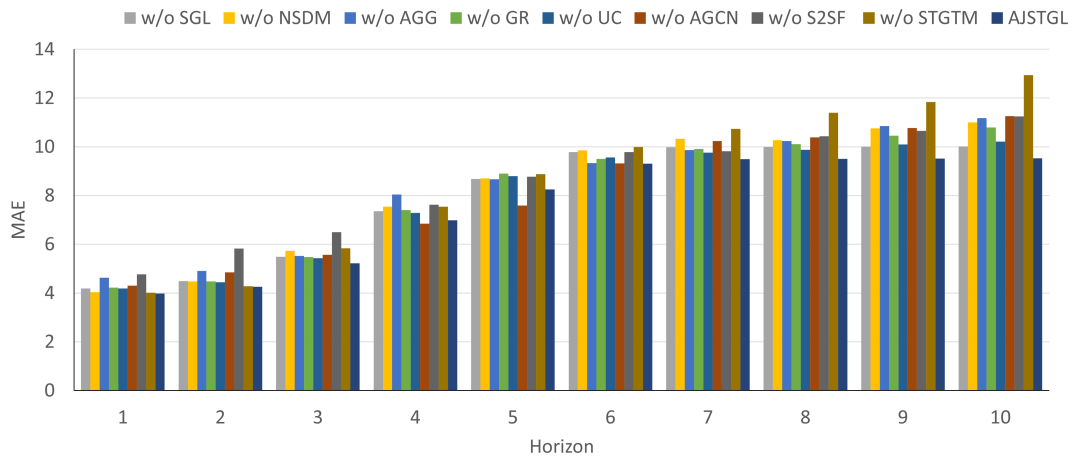


Figure 25: Ablation study MAE for arrival delay on RCOTP for the next ten horizons

capture the dynamic hidden patterns from the graph signals; 2) the adjacency matrix constructed by the multi-head attention mechanism learns the spatial node dependencies from high dimensional latent subspaces, which extend the model's capacity in modeling the hidden spatio-temporal relations; 3) AJSTGL balances short-term and long-term predictions by adopting both S2SFM and STGTM. The gated information fusion module ensures the model learns the optimal weighting when combining the features from the two modules.

6.2.3 Conclusion

This chapter presents a novel deep learning framework for spatio-temporal data processing. In the first work, a local and global correlated GCN is created to capture each airport's regional connectivity and global similarity with other airports. A hierarchical multimodal fusion network is applied to fuse flight and meteorological data and exploit

their cross-modality dependency. During the training phase, a dynamic multi-task learning strategy is used to predict flight arrival and departure delays at the same time to boost the model's generalization. The proposed model is evaluated on a large-scale carrier on-time performance dataset against several baselines and two state-of-the-art methods. The experimental results demonstrated that our model could effectively forecast short to medium-term flight delays.

In the second work, we use static and adaptive graph learning modules to improve the pre-defined graph and adaptively learn new graphs to capture more trivial hidden patterns. An auxiliary convolutional graph is adopted to leverage contextual information. We further transform the standard graph convolutional layer to allow the model to learn unidirectional traffic flow patterns. The sequence-to-sequence fusion module hierarchically combines the parallelized encoders and learns the short-term temporal node dependencies. We also develop the spatio-temporal graph transformer module to complement S2SFM by dynamically capturing the spatio-temporal dependencies in long-term prediction. Experimental results on three real-world datasets demonstrate the excellent performance of our approach compared to other state-of-the-art baselines.

Table 29: The ablation study of each component's impact on flight arrival and departure delay prediction

Method	MAE									
	1h	2h	3h	4h	5h	6h	7h	8h	9h	10h
	Arrival Delay									
w/o global GCN	4.988	5.237	6.310	8.247	9.801	10.645	11.984	11.973	12.712	12.680
w/o MTL	4.601	4.875	6.011	7.903	9.408	10.288	11.594	11.583	12.425	12.394
w/o DMTL	4.725	5.023	6.176	8.062	9.603	10.471	11.754	11.772	12.548	12.609
w/o HGF	4.923	5.230	6.297	8.216	9.779	10.639	11.980	11.947	12.700	12.698
MTLG-Net	**4.573**	**4.842**	**5.925**	**7.852**	**9.374**	**10.237**	**11.535**	**11.570**	**12.335**	**12.359**
	Departure Delay									
w/o global GCN	4.985	5.231	6.296	8.239	9.821	10.651	11.990	11.965	12.689	12.688
w/o MTL	4.613	4.882	6.001	7.913	9.385	10.297	11.604	11.579	12.412	12.386
w/o DMTL	4.721	5.021	6.161	8.045	9.517	10.450	11.800	11.792	12.537	12.601
w/o HGF	4.915	5.221	6.283	8.210	9.815	10.613	11.914	11.951	12.717	12.683
MTLG-Net	**4.499**	**4.831**	**5.917**	**7.858**	**9.291**	**10.239**	**11.485**	**11.573**	**11.258**	**11.536**

Table 30: The ablation study of meteorological input variable's impact on arrival and departure delay

Method	MAE									
	1h	2h	3h	4h	5h	6h	7h	8h	9h	10h
	Arrival Delay									
w/o Met Var	4.812	5.107	6.210	8.134	9.662	10.545	11.913	11.940	12.744	12.738
with Met Var	**4.573**	**4.842**	**5.925**	**7.852**	**9.374**	**10.237**	**11.535**	**11.570**	**12.335**	**12.359**
	Departure Delay									
w/o Met Var	4.780	5.139	6.224	8.183	9.637	10.653	11.872	11.993	11.701	12.015
with Met Var	**4.499**	**4.831**	**5.917**	**7.858**	**9.291**	**10.239**	**11.485**	**11.573**	**11.258**	**11.536**

Table 31: Overall performance comparison of AJSTGL and baselines on three datasets: a) RCOTP: average ten hours arrival delay, b) PeMSD4: traffic flow, c) PeMSD8: traffic flow.

Model	Dataset	RCOTP		PeMSD4		PeMSD8	
	Metrics	MAE	RMSE	MAE	RMSE	MAE	RMSE
ARIMA		13.36	24.04	40.05	66.08	36.11	59.85
DCRNN		9.93	16.39	21.25	33.49	16.81	26.34
STGCN		9.86	16.86	21.2	35.01	17.52	27.14
ASTGCN		9.66	16.22	22.94	35.41	18.28	28.44
Graph WaveNet		10.52	17.88	20.01	31.11	15.61	24.44
AGCRN		9.02	15.24	19.85	32.28	15.97	25.55
AJSTGL		**7.60**	**12.84**	**18.07**	**29.37**	**13.79**	**22.48**

Table 32: Overall performance comparison of AJSTGL and baselines on average hourly flight arrival delay prediction in ten hours horizons using RCOTP dataset

Method	MAE									
	1h	2h	3h	4h	5h	6h	7h	8h	9h	10h
ARIMA	7.33	8.02	9.26	11.83	13.10	15.77	15.93	16.28	17.73	18.32
DCRNN	6.38	6.78	7.36	8.10	9.39	11.02	11.86	12.17	12.42	12.94
STGCN	6.24	6.83	7.88	8.07	9.14	11.73	11.89	12.19	12.60	12.02
ASTGCN	5.98	5.54	6.85	8.15	9.28	10.85	11.99	12.27	13.65	12.03
GW	5.19	5.68	6.46	8.38	10.98	12.07	13.09	13.43	14.83	15.07
AGCRN	5.15	5.63	5.50	7.31	9.58	10.31	11.24	11.42	11.84	12.19
AJSTGL	**4.02**	**4.26**	**5.21**	**6.99**	**8.44**	**9.32**	**9.59**	**9.73**	**9.95**	**10.21**

Table 33: MAE and RMSE of traffic speed prediction on PeMSD4 for the next four horizons

Model	Dataset	PeMSD4 (15/30/45/60 min)	
	Metrics	MAE	RMSE
ARIMA		2.8	5.43
DCRNN		1.34/1.79/2.06/2.27	2.95/4.08/4.82/5.36
STGCN		1.48/1.95/2.23/2.61	3.01/4.22/5.03/5.66
ASTGCN		2.11/2.45/2.62/2.74	3.94/4.58/4.94/5.18
Graph WaveNet		1.45/1.92/2.15/2.55	2.97/4.15/4.89/5.52
AGCRN		2.04/2.39/2.58/2.66	3.86/4.47/4.80/5.12
AJSTGL		**1.26/1.62/1.78/1.91**	**2.65/3.40/3.74/3.82**

Table 34: MAE and RMSE of traffic speed prediction on PeMSD8 for the next four horizons

Model	Dataset	PeMSD8 (15/30/45/60 min)	
	Metrics	MAE	RMSE
ARIMA		2.22	4.57
DCRNN		1.16/1.52/1.69/1.88	2.55/3.53/4.10/4.47
STGCN		1.21/1.62/1.95/2.28	3.21/3.74/3.94/4.19
ASTGCN		1.53/1.71/1.85/1.94	3.22/3.75/3.98/4.25
Graph WaveNet		1.19/1.58/1.91/2.26	3.17/3.69/3.92/4.13
AGCRN		1.51/1.66/1.82/1.88	3.21/3.68/3.86/4.31
AJSTGL		**1.13/1.32/1.51/1.64**	**2.37/2.77/3.17/3.28**

Table 35: Ablation study MAE and RMSE for average arrival delay on RCOTP and traffic flow on PeMSD4 and PeMSD8 datasets.

Method	Dataset	RCOTP		PeMSD4		PeMSD8	
	Metrics	MAE	RMSE	MAE	RMSE	MAE	RMSE
w/o SGL		7.99	15.83	19.01	30.88	14.50	23.64
w/o NSDM		8.27	16.37	19.66	31.94	15.00	24.45
w/o AGG		8.32	16.47	19.78	32.14	15.10	24.61
w/o GR		8.12	16.08	19.31	31.37	14.73	24.02
w/o UC		7.93	15.70	18.85	30.63	14.39	23.45
w/o AGCN		8.11	16.05	19.27	31.32	14.71	23.97
w/o S2SFM		7.84	15.52	18.64	30.29	14.23	23.19
w/o STGTM		8.01	15.86	19.04	30.95	14.53	23.69
AJSTGL		**7.60**	**12.84**	**18.07**	**29.37**	**13.79**	**22.48**

Table 36: Ablation study MAE for arrival delay on RCOTP for the next ten horizons

Method	MAE									
	1h	2h	3h	4h	5h	6h	7h	8h	9h	10h
w/o SGL	4.19	4.49	5.49	7.35	8.68	9.79	9.98	9.99	10.00	10.01
w/o NSDM	4.04	4.47	5.74	7.55	8.71	9.85	10.33	10.27	10.75	11.00
w/o AGG	4.63	4.90	5.53	8.04	8.67	9.33	9.86	10.23	10.85	11.18
w/o GR	4.23	4.47	5.47	7.41	8.90	9.51	9.90	10.11	10.45	10.79
w/o UC	4.19	4.44	5.43	7.29	8.80	9.56	9.76	9.87	10.09	10.21
w/o AGCN	4.30	4.85	5.57	6.84	7.59	9.32	10.23	10.38	10.77	11.25
w/o S2SFM	4.77	5.82	6.50	7.62	8.77	9.79	9.81	10.43	10.66	11.25
w/o STGTM	4.02	4.28	5.83	7.55	8.88	9.99	10.74	11.39	11.84	12.95
AJSTGL	**3.98**	**4.26**	**5.21**	**6.99**	**8.25**	**9.30**	**9.49**	**9.50**	**9.52**	**9.52**

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

7.1 Conclusions

This dissertation proposes a comprehensive framework of multimodal big data analytics and fusion for data science. The main components included in this work are (1) hierarchical graph fusion, (2) adaptive spatio-temporal graph network, and (3) dynamic multi-task learning. These components are systematically integrated to provide new solutions for multimodal big data analytics problems. The following is a summary of each element:

- A novel dynamic multi-task learning approach is proposed to address the challenge of optimizing the model training process in a multi-task learning scenario. The dynamic task balancing adjusts the training progress at the sample and task levels. By allocating additional resources to difficult instances, the sample-level dynamic balancing function amplifies the loss generated by these samples. Simultaneously, the task-level dynamic balancing mechanism adjusts weight distribution based on each task's training rate. Additionally, a loss weighting method named automatic loss weighting is designed to automatically adjust the weight scalar for each task following every training iteration. As the loss for each task is tuned to a similar scale, the model can begin the subsequent training iteration with a more balanced loss distribution.

- A novel hierarchical graph fusion network is proposed to capture the inter-modality correlations among modalities and retain their independent properties. Like a tree-based graph, each modality is combined on different levels, in which the cross-modality interactions are learned based on various combinations. Graph nodes represent each input signal combination, and edges represent the similarity among modality combinations. Therefore, the value of each child node is represented as the integration of both parent nodes and the edge weights.

- A novel adaptive spatio-temporal graph network with a sequence-to-sequence fusion model is proposed. Multiple pre-defined and adaptively generated GCNs are utilized to capture the spatial dependency in the data. The locally correlated GCN focuses on the spatial connectivity between nodes with direct connections. The globally correlated GCN captures the network-wide correlation among nodes with similar characteristics. Finally, the adaptive GCN comprehensively learns the hidden patterns between nodes. In addition, an effective normalization technique is also applied to control the adjacency matrix's sparsity and reduce redundant node-wise correlation. A bidirectional long-short-term memory (BiLSTM) based sequence-to-sequence fusion module is applied to extract the short-term temporal information in the data sequence. The sequence-to-sequence fusion network also combines the output from each graph to leverage the cross-modal interactions in modeling the temporal dependency. Lastly, a spatio-temporal graph transformer module is developed to complement the sequence-to-sequence fusion module by dynamically capturing the spatio-temporal dependencies in long-term predictions.

- The proposed framework has been tested on various datasets and applications, including disaster damage assessment, disaster situation assessment, airfare price prediction, flight delay prediction, traffic speed prediction, and traffic flow forecasting.

7.2 Future Works

Despite the various solutions introduced in the previous chapters, many challenges in multimodal big data analytics still need to be tackled to improve the proposed multimedia big data analytics and fusion framework.

7.2.1 Task Relation Learning in Multi-Tasking Learning

As discussed in Chapter 4, multi-task learning is an important research domain that enhances the modeling performance by learning multiple related tasks. In this dissertation, we focus on optimizing the training process, such as adjusting the learning rate based on weight gradient to improve the efficiency of multi-task learning. In future work, we would also like to target task relation learning [14]. Unlike neural network architecture design or learning optimization methods, task relation learning aims to capture the representation of tasks based on their similarities. The model performance could be further improved by grouping correlated tasks and adapting similar training strategies. Initial attempts on task relation learning try to group a primary task with multiple correlated auxiliary tasks [50, 160]. However, this requires empirical knowledge in choosing the optimal task groups or significant effort in testing the modeling results on all task combinations.

Our future work will focus on modeling the task similarity representation by

comparing the attention maps across tasks' input and output data. The attention map contains the correlation pattern between input and output data based on their specific format. For example, in text data, the attention map represents the scoring of each input token relevant to tokens in the output. It is reasonable to assume that similar tasks will also demonstrate similar attention map patterns. Another advantage of the proposed approach is that the attention map can be generated on a simple model since learning the relative relations among tasks can be separated from the primary task. This greatly reduces the computation cost as there is no need to train a more complex main model.

7.2.2 Adversarial Joint-Modality Embedding Learning for Multimodal Fusion

In this dissertation, we proposed the hierarchical graph fusion approach to capture the inter-modality correlations among modalities for multimodal fusion. It uses a tree-based graph structure to combine the vectors of modalities by learning their interactions. However, the heterogeneous data distributions among different sources still pose a great challenge for multimodal fusion [16]. Most existing approaches use multiple modality-specific networks to extract the vector representation from each source and directly combine them using various fusion methods [64, 171, 186]. However, due to the challenge of heterogeneous data distributions, the complementary information across modalities may not be fully exploited without first learning a joint embedding among the modalities before the fusion step. Generative adversarial networks (GANs) [58] have been widely applied to computer vision and natural language processing to map the distributions of source domains to that of the target domain based on a prior domain distribution using the adversarial training

technique.

Our future work will use an encoder-decoder framework with adversarial learning to re-construct each modality into a transformed distribution by embedding all modalities' inputs into a common vector space. In other words, the encoder generator produces the embedding space vector for each source modality and tries to fool the discriminator into classifying the embedded source modality vector as the target modality vector. On the other hand, the discriminator's goal is to distinguish the target modality vector from the embedded source modality representation produced by the generator. Such an adversarial learning process will enable the model to learn the modality-invariant joint embedding of all modalities. Combining the hierarchical graph fusion approach discussed in Chapter 5 and the adversarial joint-modality embedding learning method, the model could more efficiently capture the inter-modality interactions and alleviate the issue caused by the heterogeneous data distributions among distinct source modalities.

7.2.3 Graph Convolutional Networks for Multi-Label Classification

Chapter 6 presents various novel techniques of applying GCNs for spatio-temporal data modeling. In future work, we will continue to explore the potential of GCNs in handling multi-label classification problems. Chapter 4 examines the effectiveness of applying multi-task learning for multi-label classification tasks in the later stage of a modeling pipeline. Specifically, we convert the multi-label classification problem into a multi-task learning job by treating the prediction of each label as a single task. Such an approach focuses on balancing the training dynamic by adjusting the learning rate for each

task. However, it is equally important to consider the label correlation during the early stage. The low-level label relations could easily be lost in the training process as the model gradually learns more abstract input representations.

In our future work, we propose to leverage GCNs to capture the inter-label relations by modeling the prior label representations using a unique correlation mapping function. Take the disaster video classification task as an example. The graph nodes represent the word embedding of the labels, and the adjacency matrix contains the correlations for each label pair. Unlike conventional graph data, where the adjacency matrix can be constructed using the pre-defined network topology, there is no easy way to identify the label relations in an arbitrary multi-label dataset based on prior knowledge [33]. To solve this issue, we define the label adjacency matrix by learning the co-occurrence patterns from the data. The co-occurrence matrix contains the conditional probabilities of the occurrences of each label with regard to all other labels. To avoid overfitting the co-occurrence matrix on the training dataset, we can leverage the sparse matrix normalization technique discussed in Chapter 6 as a means of regularization. Sparse matrix normalization removes trivial node connections to enhance the generalization ability of the graph network. In practice, GCNs containing the learned label correlations are used as classifiers for feature representations extracted by other domain-specific sub-networks. The proposed approach can be applied to various real-world application domains where complex label correlations exist.

REFERENCE LIST

[1] Akhtar, M. S., Chauhan, D. S., Ghosal, D., Poria, S., Ekbal, A., and Bhattacharyya, P. Multi-task learning for multi-modal emotion recognition and sentiment analysis. *arXiv preprint arXiv:1905.05812* (2019).

[2] Alam, F., Ofli, F., and Imran, M. CrisisMMD: Multimodal Twitter Datasets from Natural Disasters. In *12th International Conference on Web and Social Media* (2018), pp. 465–473.

[3] Alam, F., Ofli, F., and Imran, M. Crisismmd: Multimodal twitter datasets from natural disasters. In *Proceedings of the International AAAI Conference on Web and Social Media* (2018), vol. 12.

[4] Ali, M., Yousuf, N., Rahman, M., Chaki, J., Dey, N., Santosh, K., et al. Machine translation using deep learning for universal networking language based on their structure. *International Journal of Machine Learning and Cybernetics 12*, 8 (2021), 2365–2376.

[5] Alonso, H. M., and Plank, B. When is multitask learning effective? Semantic sequence prediction under varying data conditions. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*

(2017), M. Lapata, P. Blunsom, and A. Koller, Eds., Association for Computational Linguistics, pp. 44–53.

[6] Angelou, M., Solachidis, V., Vretos, N., and Daras, P. Graph-based multimodal fusion with metric learning for multimodal classification. *Pattern Recognition 95* (2019), 296–307.

[7] Audebert, N., Le Saux, B., and Lefèvre, S. Semantic segmentation of earth observation data using multimodal and multi-scale deep networks. In *Asian Conference on Computer Vision* (2016), Springer, pp. 180–196.

[8] Audebert, N., Le Saux, B., and Lefèvre, S. Beyond RGB: Very high resolution urban remote sensing with multimodal deep networks. *ISPRS Journal of Photogrammetry and Remote Sensing 140* (2018), 20–32.

[9] Aytar, Y., Vondrick, C., and Torralba, A. Soundnet: Learning sound representations from unlabeled video. In *Advances in neural information processing systems* (2016), pp. 892–900.

[10] Aytar, Y., Vondrick, C., and Torralba, A. SoundNet: Learning Sound Representations from Unlabeled Video. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems* (2016), pp. 892–900.

[11] Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450* (2016).

[12] Babbar, R., and Schölkopf, B. Dismec: Distributed sparse machines for extreme multi-label classification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* (2017), pp. 721–729.

[13] Bahdanau, D., Cho, K., and Bengio, Y. Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (2015).

[14] Bai, G., and Zhao, L. Saliency-Regularized Deep Multi-Task Learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (2022), pp. 15–25.

[15] Bai, L., Yao, L., Li, C., Wang, X., and Wang, C. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in Neural Information Processing Systems 33* (2020), 17804–17815.

[16] Baltrušaitis, T., Ahuja, C., and Morency, L.-P. Multimodal machine learning: A survey and taxonomy. *IEEE Transactions on Pattern Analysis and Machine Intelligence 41*, 2 (2018), 423–443.

[17] Bao, J., Liu, P., and Ukkusuri, S. V. A spatiotemporal deep learning approach for citywide short-term crash risk prediction with multi-source data. *Accident Analysis & Prevention 122* (2019), 239–254.

[18] Bao, J., Yang, Z., and Zeng, W. Graph to sequence learning with attention mechanism for network-wide multi-step-ahead flight delay prediction. *Transportation Research Part C: Emerging Technologies 130* (2021), 103323.

[19] Beatty, R., Hsu, R., Berry, L., and Rome, J. Preliminary evaluation of flight delay propagation through an airline schedule. *Air Traffic Control Quarterly 7*, 4 (1999), 259–270.

[20] Ben-Younes, H., Cadene, R., Cord, M., and Thome, N. Mutan: Multimodal tucker fusion for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 2612–2620.

[21] Bingel, J., and Søgaard, A. Identifying beneficial task relations for multi-task learning in deep neural networks. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 2: Short Papers* (2017), M. Lapata, P. Blunsom, and A. Koller, Eds., Association for Computational Linguistics, pp. 164–169.

[22] Bischke, B., Helber, P., Folz, J., Borth, D., and Dengel, A. Multi-Task Learning for Segmentation of Building Footprints with Deep Neural Networks. *CoRR abs/1709.05932* (2017).

[23] Burger, B., and Fuchs, M. Dynamic pricing – A future airline business model. *Journal of Revenue and Pricing Management 4*, 1 (2005), 39–53.

[24] Chakrabarty, N. A data mining approach to flight arrival delay prediction for american airlines. In *2019 9th Annual Information Technology, Electromechanical Engineering and Microelectronics Conference (IEMECON)* (2019), IEEE, pp. 102–107.

[25] Chen, C., Liu, B., Wan, S., Qiao, P., and Pei, Q. An edge traffic flow detection scheme based on deep learning in an intelligent transportation system. *IEEE Transactions on Intelligent Transportation Systems 22*, 3 (2020), 1840–1852.

[26] Chen, C., Liu, Y., Chen, L., and Zhang, C. Bidirectional spatial-temporal adaptive transformer for Urban traffic flow forecasting. *IEEE Transactions on Neural Networks and Learning Systems* (2022).

[27] Chen, J., and Zhang, A. HGMF: Heterogeneous Graph-based Fusion for Multimodal Data with Incompleteness. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020), pp. 1295–1305.

[28] Chen, S., and Kashyap, R. L. A Spatio-Temporal Semantic Model for Multimedia Database Systems and Multimedia Information Systems. *IEEE Trans. Knowl. Data Eng. 13*, 4 (2001), 607–622.

[29] Chen, S.-C., and Kashyap, R. L. A spatio-temporal semantic model for multimedia database systems and multimedia information systems. *IEEE Transactions on Knowledge and Data Engineering 13*, 4 (2001), 607–622.

[30] Chen, S.-C., Shyu, M.-L., Chen, M., and Zhang, C. A decision tree-based multimodal data mining framework for soccer goal detection. In *IEEE International Conference on Multimedia and Expo (ICME)* (2004), vol. 1, pp. 265–268.

[31] Chen, Y., Cao, J., Feng, S., and Tan, Y. An ensemble learning based approach for building airfare forecast service. In *the IEEE international conference on big data* (2015), pp. 964–969.

[32] Chen, Z., Badrinarayanan, V., Lee, C.-Y., and Rabinovich, A. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning* (2018), pp. 794–803.

[33] Chen, Z.-M., Wei, X.-S., Wang, P., and Guo, Y. Multi-label image recognition with graph convolutional networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 5177–5186.

[34] Cheng, J., Dong, L., and Lapata, M. Long Short-Term Memory-Networks for Machine Reading. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing, Austin, Texas, USA, November 1-4, 2016* (2016), The Association for Computational Linguistics, pp. 551–561.

[35] Chiu, C.-C., Sainath, T. N., Wu, Y., Prabhavalkar, R., Nguyen, P., Chen, Z., Kannan, A., Weiss, R. J., Rao, K., Gonina, E., et al. State-of-the-art speech recognition with sequence-to-sequence models. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), IEEE, pp. 4774–4778.

[36] Cresci, S., Cimino, A., Dell'Orletta, F., and Tesconi, M. Crisis mapping during natural disasters via text analysis of social media messages. In *International Conference on Web Information Systems Engineering* (2015), Springer, pp. 250–258.

[37] Dai, J., He, K., and Sun, J. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 3150–3158.

[38] Deng, J., Dong, W., Socher, R., Li, L., Li, K., and Li, F. ImageNet: A large-scale hierarchical image database. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)* (2009), IEEE Computer Society, pp. 248–255.

[39] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition* (2009), pp. 248–255.

[40] Derudder, B., and Witlox, F. An appraisal of the use of airline data in assessing the world city network: a research note on data. *Urban Studies 42*, 13 (2005), 2371–2388.

[41] Devaraj, J., Ganesan, S., Elavarasan, R. M., and Subramaniam, U. A Novel Deep Learning Based Model for Tropical Intensity Estimation and Post-Disaster Management of Hurricanes. *Applied Sciences 11*, 9 (2021), 4129.

[42] Ding, Y. Predicting flight delay based on multiple linear regression. In *IOP conference series: Earth and environmental science* (2017), vol. 81, IOP Publishing, p. 012198.

[43] Duan, Z., Yang, Y., and Zhou, W. Multi-View Spatial-Temporal Adaptive Graph Convolutional Networks for Traffic Forecasting. In *2021 16th International Conference on Intelligent Systems and Knowledge Engineering (ISKE)* (2021), IEEE, pp. 35–41.

[44] Dwivedi, K., and Roig, G. Representation similarity analysis for efficient task taxonomy & transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 12387–12396.

[45] Eligüzel, N., Çetinkaya, C., and Dereli, T. Application of named entity recognition on tweets during earthquake disaster: a deep learning-based approach. *Soft Computing 26*, 1 (2022), 395–421.

[46] Evgeniou, T., Micchelli, C. A., Pontil, M., and Shawe-Taylor, J. Learning multiple tasks with kernel methods. *Journal of machine learning research 6*, 4 (2005).

[47] FAA. Passenger Boarding (Enplanement) and All-Cargo Data for U.S. Airports, 2022. `https://www.faa.gov/airports/planning_capacity/passenger_allcargo_stats/passenger/`.

[48] Fan, A., Bhosale, S., Schwenk, H., Ma, Z., El-Kishky, A., Goyal, S., Baines, M., Celebi, O., Wenzek, G., Chaudhary, V., et al. Beyond english-centric multilingual machine translation. *Journal of Machine Learning Research 22*, 107 (2021), 1–48.

[49] Farnadi, G., Tang, J., De Cock, M., and Moens, M.-F. User profiling through deep multimodal fusion. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (2018), ACM, pp. 171–179.

[50] Fifty, C., Amid, E., Zhao, Z., Yu, T., Anil, R., and Finn, C. Efficiently identifying task groupings for multi-task learning. *Advances in Neural Information Processing Systems 34* (2021), 27503–27516.

[51] Francis, G., Fidato, A., and Humphreys, I. Airport–airline interaction: the impact of low-cost carriers on two European airports. *Journal of Air Transport Management 9*, 4 (2003), 267–273.

[52] Fujiyoshi, H., Hirakawa, T., and Yamashita, T. Deep learning-based image recognition for autonomous driving. *IATSS research 43*, 4 (2019), 244–252.

[53] Ganda, D., and Buch, R. A survey on multi label classification. *Recent Trends in Programming Languages 5*, 1 (2018), 19–23.

[54] Gao, F., Zhu, J., Wu, L., Xia, Y., Qin, T., Cheng, X., Zhou, W., and Liu, T.-Y. Soft contextual data augmentation for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (2019), pp. 5539–5544.

[55] Gao, Y., Ma, J., Zhao, M., Liu, W., and Yuille, A. L. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 3205–3214.

[56] Gerardi, K. S., and Shapiro, A. H. Does competition reduce price dispersion? New evidence from the airline industry. *Journal of Political Economy 117*, 1 (2009), 1–37.

[57] Ghamrawi, N., and McCallum, A. Collective multi-label classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management* (2005), pp. 195–200.

[58] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM 63*, 11 (2020), 139–144.

[59] Görnitz, N., Widmer, C., Zeller, G., Kahles, A., Rätsch, G., and Sonnenburg, S. Hierarchical multitask structured output learning for large-scale sequence segmentation. *Advances in Neural Information Processing Systems 24* (2011).

[60] Gui, G., Liu, F., Sun, J., Yang, J., Zhou, Z., and Zhao, D. Flight delay prediction based on aviation big data and machine learning. *IEEE Transactions on Vehicular Technology 69*, 1 (2019), 140–150.

[61] Guo, M., Haque, A., Huang, D., Yeung, S., and Fei-Fei, L. Dynamic Task Prioritization for Multitask Learning. In *15th European Conference on Computer Vision* (2018), pp. 282–299.

[62] Guo, S., Lin, Y., Feng, N., Song, C., and Wan, H. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI conference on artificial intelligence* (2019), vol. 33, pp. 922–929.

[63] Guo, Z., Mei, G., Liu, S., Pan, L., Bian, L., Tang, H., and Wang, D. SGDAN—A Spatio-Temporal Graph Dual-Attention Neural Network for Quantified Flight Delay Prediction. *Sensors 20*, 22 (2020), 6433.

[64] Han, W., Chen, H., and Poria, S. Improving multimodal fusion with hierarchical mutual information maximization for multimodal sentiment analysis. *arXiv preprint arXiv:2109.00412* (2021).

[65] Hao, H., and Wang, Y. Leveraging multimodal social media data for rapid disaster damage assessment. *International Journal of Disaster Risk Reduction 51* (2020), 101760.

[66] He, K., Zhang, X., Ren, S., and Sun, J. Deep Residual Learning for Image Recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), IEEE Computer Society, pp. 770–778.

[67] Hou, F., Zhang, Y., Fu, X., Jiao, L., and Zheng, W. The prediction of multistep traffic flow based on AST-GCN-LSTM. *Journal of Advanced Transportation 2021* (2021), 1–10.

[68] Hou, J.-C., Wang, S.-S., Lai, Y.-H., Tsao, Y., Chang, H.-W., and Wang, H.-M. Audio-visual speech enhancement using multimodal deep convolutional neural networks. *IEEE Transactions on Emerging Topics in Computational Intelligence 2*, 2 (2018), 117–128.

[69] Hou, Y., Deng, Z., and Cui, H. Short-term traffic flow prediction with weather conditions: based on deep learning algorithms and data fusion. *Complexity 2021* (2021).

[70] Hrinchuk, O., Popova, M., and Ginsburg, B. Correction of automatic speech recognition with transformer sequence-to-sequence model. In *Icassp 2020-2020 ieee international conference on acoustics, speech and signal processing (icassp)* (2020), IEEE, pp. 7074–7078.

[71] Hu, D., Nie, F., and Li, X. Dense Multimodal Fusion for Hierarchically Joint Representation. *arXiv preprint arXiv:1810.03414* (2018).

[72] Hu, J., Zhang, X., and Maybank, S. Abnormal driving detection with normalized driving behavior data: a deep learning approach. *IEEE transactions on vehicular technology 69*, 7 (2020), 6943–6951.

[73] Hu, P., Huang, Y.-A., Chan, K. C., and You, Z.-H. Learning multimodal networks from heterogeneous data for prediction of lncRNA–miRNA interactions. *IEEE/ACM Transactions on Computational Biology and Bioinformatics 17*, 5 (2019), 1516–1524.

[74] Hu, R., and Singh, A. UniT: Multimodal Multitask Learning with a Unified Transformer. *arXiv preprint arXiv:2102.10772* (2021).

[75] Huang, S., Wang, D., Wu, X., and Tang, A. DSANet: Dual self-attention network for multivariate time series forecasting. In *Proceedings of the 28th ACM international conference on information and knowledge management* (2019), pp. 2129–2132.

[76] International Civil Aviation Organization. List of Low-Cost-Carriers (LCCs), cited July 2018.

[77] Jebara, T. Multi-task feature and kernel selection for SVMs. In *Proceedings of the twenty-first international conference on Machine learning* (2004), p. 55.

[78] Jiang, B., Zhang, Z., Lin, D., Tang, J., and Luo, B. Semi-supervised learning with graph learning-convolutional networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 11313–11320.

[79] Jiang, H., Cao, P., Xu, M., Yang, J., and Zaiane, O. Hi-GCN: A hierarchical graph convolution network for graph embedding learning of brain network and brain disorders prediction. *Computers in Biology and Medicine 127* (2020), 104096.

[80] Jiang, M., Chen, W., and Li, X. S-GCN-GRU-NN: A novel hybrid model by combining a Spatiotemporal Graph Convolutional Network and a Gated Recurrent Units Neural Network for short-term traffic speed forecasting. *Journal of Data, Information and Management 3* (2021), 1–20.

[81] Jolliffe, I. *Principal component analysis*. Springer, 2011.

[82] Kampman, O., Barezi, E. J., Bertero, D., and Fung, P. Investigating Audio, Visual, and Text Fusion Methods for End-to-End Automatic Personality Prediction. *CoRR abs/1805.00705* (2018).

[83] Karatzoglou, A., Schnell, N., and Beigl, M. A convolutional neural network approach for modeling semantic trajectories and predicting future locations. In *International Conference on Artificial Neural Networks* (2018), Springer, pp. 61–72.

[84] Kato, T., Kashima, H., Sugiyama, M., and Asai, K. Multi-task learning via conic programming. *Advances in Neural Information Processing Systems 20* (2007).

[85] Kendall, A., Gal, Y., and Cipolla, R. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 7482–7491.

[86] Kim, J., and Hastak, M. Social network analysis: Characteristics of online social networks after a disaster. *International Journal of Information Management 38*, 1 (2018), 86–96.

[87] Kim, Y. J., Choi, S., Briceno, S., and Mavris, D. A deep learning approach to flight delay prediction. In *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)* (2016), IEEE, pp. 1–6.

[88] Kingma, D. P., and Ba, J. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (2015).

[89] Kingma, D. P., and Ba, J. Adam: A Method for Stochastic Optimization. In *the 3rd international conference on learning representations* (2015).

[90] Kipf, T. N., and Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings* (2017), OpenReview.net.

[91] Klomp, J. Economic development and natural disasters: A satellite data analysis. *Global Environmental Change 36* (2016), 67–88.

[92] Kokkinos, I. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 6129–6138.

[93] Koopmans, C., and Lieshout, R. Airline cost changes: To what extent are they passed through to the passenger? *Journal of Air Transport Management 53* (2016), 1–11.

[94] Kumar, A., Singh, J. P., Dwivedi, Y. K., and Rana, N. P. A deep multi-modal neural network for informative Twitter content classification during emergencies. *Annals of Operations Research* (2020), 1–32.

[95] Lambelho, M., Mitici, M., Pickup, S., and Marsden, A. Assessing strategic flight schedules at an airport using machine learning-based flight delay and cancellation predictions. *Journal of Air Transport Management 82* (2020), 101737.

[96] Lee, S., Seo, K., and Sharma, A. Corporate social responsibility and firm performance in the airline industry: The moderating role of oil prices. *Tourism Management 38* (2013), 20–30.

[97] Li, G., Yang, Y., Qu, X., Cao, D., and Li, K. A deep learning based image enhancement approach for autonomous driving at night. *Knowledge-Based Systems 213* (2021), 106617.

[98] Li, R., Wang, S., Zhu, F., and Huang, J. Adaptive graph convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2018), vol. 32.

[99] Li, W., Wang, X., Zhang, Y., and Wu, Q. Traffic flow prediction over muti-sensor data correlation with graph convolution network. *Neurocomputing 427* (2021), 50–63.

[100] Li, X., Caragea, D., Zhang, H., and Imran, M. Localizing and Quantifying Damage in Social Media Images. In *IEEE/ACM 2018 International Conference on Advances in Social Networks Analysis and Mining* (2018), pp. 194–201.

[101] Li, Y., Song, Y., and Luo, J. Improving pairwise ranking for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 3617–3625.

[102] Li, Y., Yu, R., Shahabi, C., and Liu, Y. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings* (2018), OpenReview.net.

[103] Li, Z., Zhang, G., Xu, L., and Yu, J. Dynamic Graph Learning-Neural Network for Multivariate Time Series Modeling. *CoRR abs/2112.03273* (2021).

[104] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision* (2017), pp. 2980–2988.

[105] Liu, D., Hui, S., Li, L., Liu, Z., and Zhang, Z. A method for short-term traffic flow forecasting based on GCN-LSTM. In *2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL)* (2020), IEEE, pp. 364–368.

[106] Liu, J., Ji, S., and Ye, J. Multi-task feature learning via efficient l2, 1-norm minimization. *arXiv preprint arXiv:1205.2631* (2012).

[107] Liu, J., Ji, S., Ye, J., et al. SLEP: Sparse learning with efficient projections. *Arizona State University 6*, 491 (2009), 7.

[108] Liu, J., and Ye, J. Efficient Euclidean projections in linear time. In *Proceedings of the 26th Annual International Conference on Machine Learning* (2009), pp. 657–664.

[109] Liu, S., Johns, E., and Davison, A. J. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 1871–1880.

[110] Liu, T., Cao, J., Tan, Y., and Xiao, Q. ACER: An adaptive context-aware ensemble regression model for airfare price prediction. In *the international conference on progress in informatics and computing* (2017), pp. 312–317.

[111] Liu, Y., Feng, X., and Zhou, Z. Multimodal video classification with stacked contractive autoencoders. *Signal Processing 120* (2016), 761–766.

[112] Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., Lewis, M., and Zettlemoyer, L. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics 8* (2020), 726–742.

[113] Liu, Z., Shen, Y., Lakshminarasimhan, V. B., Liang, P. P., Zadeh, A., and Morency, L.-P. Efficient low-rank multimodal fusion with modality-specific factors. *arXiv preprint arXiv:1806.00064* (2018).

[114] Logan, B., et al. Mel Frequency Cepstral Coefficients for Music Modeling. In *ISMIR* (2000), vol. 270, pp. 1–11.

[115] Long, D., and Hasan, S. Improved predictions of flight delays using LMINET2 system-wide simulation model. In *9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO) and Aircraft Noise and Emissions Reduction Symposium (ANERS)* (2009), p. 6961.

[116] Lu, Yongxi hand Kumar, A., Zhai, S., Cheng, Y., Javidi, T., and Feris, R. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 5334–5343.

[117] Lu, J., Goswami, V., Rohrbach, M., Parikh, D., and Lee, S. 12-in-1: Multi-task vision and language representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 10437–10446.

[118] Ma, J., Zhao, Z., Yi, X., Chen, J., Hong, L., and Chi, E. H. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), pp. 1930–1939.

[119] Madichetty, S., and Sridevi, M. Detecting informative tweets during disaster using deep neural networks. In *2019 11th International Conference on Communication Systems & Networks (COMSNETS)* (2019), IEEE, pp. 709–713.

[120] Mai, S., Hu, H., and Xing, S. Modality to modality translation: An adversarial representation learning and graph fusion network for multimodal fusion. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), vol. 34, pp. 164–172.

[121] Malighetti, P., Paleari, S., and Redondi, R. Has Ryanair's pricing strategy changed over time? An empirical analysis of its 2006–2007 flights. *Tourism Management 31*, 1 (2010), 36–44.

[122] Manna, S., Biswas, S., Kundu, R., Rakshit, S., Gupta, P., and Barman, S. A statistical approach to predict flight delay using gradient boosted decision tree. In *2017 International Conference on Computational Intelligence in Data Science (ICCIDS)* (2017), IEEE, pp. 1–5.

[123] Mantin, B., and Koo, B. Dynamic price dispersion in airline markets. *Transportation Research Part E: Logistics and Transportation Review 45*, 6 (2009), 1020–1029.

[124] Mao, J., Xu, J., Jing, K., and Yuille, A. L. Training and Evaluating Multimodal Word Embeddings with Large-scale Web Annotated Images. In *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 442–450.

[125] Maqueda, A. I., Loquercio, A., Gallego, G., García, N., and Scaramuzza, D. Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 5419–5427.

[126] Mensink, T., Gavves, E., and Snoek, C. G. Costa: Co-occurrence statistics for zero-shot classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2014), pp. 2441–2448.

[127] Miglani, A., and Kumar, N. Deep learning models for traffic flow prediction in autonomous vehicles: A review, solutions, and challenges. *Vehicular Communications 20* (2019), 100184.

[128] Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient Estimation of Word Representations in Vector Space. In *1st International Conference on Learning Representations* (2013).

[129] Misra, I., Shrivastava, A., Gupta, A., and Hebert, M. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 3994–4003.

[130] Mofokeng, T. J., and Marnewick, A. Factors contributing to delays regarding aircraft during A-check maintenance. In *2017 ieee technology & engineering management conference (temscon)* (2017), IEEE, pp. 185–190.

[131] Mottini, A., and Acuna-Agost, R. Deep choice model using pointer networks for airline itinerary prediction. In *the 23rd ACM SIGKDD international conference on knowledge discovery and data mining* (2017), pp. 1575–1583.

[132] Nair, V., and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *the 27th international conference on machine learning* (2010), pp. 807–814.

[133] Nguyen, D.-K., and Okatani, T. Multi-task learning of hierarchical vision-language representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 10492–10501.

[134] Obozinski, G., Taskar, B., and Jordan, M. Multi-task feature selection. *Statistics Department, UC Berkeley, Tech. Rep 2*, 2.2 (2006), 2.

[135] Oum, T. H., Zhang, A., and Zhang, Y. Inter-firm rivalry and firm-specific price elasticities in deregulated airline markets. *Journal of Transport Economics and Policy 7*, 2 (1993), 171–192.

[136] Ouyang, X., Zhang, C., Zhou, P., and Jiang, H. DeepSpace: An Online Deep Learning Framework for Mobile Big Data to Understand Human Mobility Patterns. *CoRR abs/1610.07009* (2016).

[137] Pennington, J., Socher, R., and Manning, C. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing* (2014), pp. 1532–1543.

[138] Pennington, J., Socher, R., and Manning, C. D. Glove: Global Vectors for Word Representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014), ACL, pp. 1532–1543.

[139] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep Contextualized Word Representations. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, (NAACL-HLT)* (2018), Association for Computational Linguistics, pp. 2227–2237.

[140] Pouyanfar, S., Tao, Y., Tian, H., Chen, S.-C., and Shyu, M.-L. Multimodal deep learning based on multiple correspondence analysis for disaster management. *World Wide Web 22*, 5 (2019), 1893–1911.

[141] Pouyanfar, S., Wang, T., and Chen, S.-C. A multi-label multimodal deep learning framework for imbalanced data classification. In *2019 IEEE conference on multimedia information processing and retrieval (MIPR)* (2019), IEEE, pp. 199–204.

[142] Pouyanfar, S., Yang, Y., Chen, S.-C., Shyu, M.-L., and Iyengar, S. Multimedia big data analytics: A survey. *ACM Computing Surveys (CSUR) 51*, 1 (2018), 1–34.

[143] Prajapati, P., Thakkar, A., and Ganatra, A. A survey and current research challenges in multi-label classification methods. *International Journal of Soft Computing and Engineering (IJSCE) 2*, 1 (2012), 248–252.

[144] Qu, J., Zhao, T., Ye, M., Li, J., and Liu, C. Flight delay prediction using deep convolutional neural network based on fusion of meteorological data. *Neural Processing Letters 52*, 2 (2020), 1461–1484.

[145] Radu, V., Lane, N. D., Bhattacharya, S., Mascolo, C., Marina, M. K., and Kawsar, F. Towards multimodal deep learning for activity recognition on mobile devices. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct* (2016), ACM, pp. 185–188.

[146] Rama-Murthy, K. *Modeling of United States Airline Fares–Using the Official Airline Guide (OAG) and Airline Origin and Destination Survey (DB1B)*. PhD thesis, Virginia Tech, 2006.

[147] Reza, S., Ferreira, M. C., Machado, J., and Tavares, J. M. R. A multi-head attention-based transformer model for traffic flow forecasting with a comparative analysis to recurrent neural networks. *Expert Systems with Applications 202* (2022), 117275.

[148] Rhoades, S. A. The herfindahl-hirschman index. *Federal Reserve Bulletin 79* (1993), 188.

[149] Ronneberger, O., Fischer, P., and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention* (2015), Springer, pp. 234–241.

[150] Ruder, S. An Overview of Multi-Task Learning in Deep Neural Networks. *CoRR abs/1706.05098* (2017).

[151] Ruder, S., Bingel, J., Augenstein, I., and Søgaard, A. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2019), vol. 33, pp. 4822–4829.

[152] Salakhutdinov, R., and Hinton, G. Deep boltzmann machines. In *Artificial intelligence and statistics* (2009), pp. 448–455.

[153] Schaefer, L., and Millner, D. Flight delay propagation analysis with the detailed policy assessment tool. In *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat. No. 01CH37236)* (2001), vol. 2, IEEE, pp. 1299–1303.

[154] Sener, O., and Koltun, V. Multi-Task Learning as Multi-Objective Optimization. In *Annual Conference on Neural Information Processing Systems* (2018), pp. 525–536.

[155] Shumway, R. H., and Stoffer, D. S. ARIMA models. In *Time series analysis and its applications*. Springer, 2017, pp. 75–163.

[156] Simonyan, K., and Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (2015), Y. Bengio and Y. LeCun, Eds.

[157] Snoek, C. G., Worring, M., and Smeulders, A. W. Early versus late fusion in semantic video analysis. In *Proceedings of the 13th Annual ACM International Conference on Multimedia* (2005), pp. 399–402.

[158] Song, J., Chen, Y., Wang, X., Shen, C., and Song, M. Deep model transferability from attribution maps. *Advances in Neural Information Processing Systems 32* (2019).

[159] Sorower, M. S. A literature survey on algorithms for multi-label learning. *Oregon State University, Corvallis 18* (2010), 1–25.

[160] Standley, T., Zamir, A., Chen, D., Guibas, L., Malik, J., and Savarese, S. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning* (2020), PMLR, pp. 9120–9132.

[161] Standley, T., Zamir, A. R., Chen, D., Guibas, L. J., Malik, J., and Savarese, S. Which Tasks Should Be Learned Together in Multi-task Learning? In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event* (2020), vol. 119 of *Proceedings of Machine Learning Research*, PMLR, pp. 9120–9132.

[162] Stavins, J. Price discrimination in the airline market: The effect of market concentration. *Review of Economics and Statistics 83*, 1 (2001), 200–202.

[163] Sun, G., Probst, T., Paudel, D. P., Popović, N., Kanakis, M., Patel, J., Dai, D., and Van Gool, L. Task Switching Network for Multi-Task Learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2021), pp. 8291–8300.

[164] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 2818–2826.

[165] Tian, H., Presa-Reyes, M., Tao, Y., Wang, T., Pouyanfar, S., Miguel, A., Luis, S., Shyu, M.-L., Chen, S.-C., and Iyengar, S. S. Data analytics for air travel data: a survey and new perspectives. *ACM Computing Surveys (CSUR) 54*, 8 (2021), 1–35.

[166] Titsias, M., and Lázaro-Gredilla, M. Spike and slab variational inference for multi-task and multiple kernel learning. *Advances in neural information processing systems 24* (2011).

[167] Tran, D. Q., Park, M., Jung, D., Park, S., et al. Damage-map estimation using uav images and deep learning algorithms for disaster management system. *Remote Sensing 12*, 24 (2020), 4169.

[168] Tziridis, K., Kalampokas, T., Papakostas, G. A., and Diamantaras, K. I. Airfare prices prediction using machine learning techniques. In *the 25th IEEE European signal processing conference* (2017), pp. 1036–1039.

[169] Vandenhende, S., Georgoulis, S., and Van Gool, L. Mti-net: Multi-scale task interaction networks for multi-task learning. In *European Conference on Computer Vision* (2020), Springer, pp. 527–543.

[170] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems 30* (2017).

[171] Vielzeuf, V., Lechervy, A., Pateux, S., and Jurie, F. Centralnet: a multilayer approach for multimodal fusion. In *European Conference on Computer Vision* (2018), Springer, pp. 575–589.

[172] Vielzeuf, V., Pateux, S., and Jurie, F. Temporal multimodal fusion for video emotion classification in the wild. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction* (2017), ACM, pp. 569–576.

[173] Vowles, T. M. Airfare pricing determinants in hub-to-hub markets. *Journal of Transport Geography 14*, 1 (2006), 15–22.

[174] Wagner, J., Fischer, V., Herman, M., and Behnke, S. Multispectral pedestrian detection using deep fusion convolutional neural networks. In *24th European*

Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN) (2016), pp. 509–514.

[175] Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., and Xu, W. Cnn-rnn: A unified framework for multi-label image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 2285–2294.

[176] Wang, S.-H., Govindaraj, V. V., Górriz, J. M., Zhang, X., and Zhang, Y.-D. Covid-19 classification by FGCNet with deep feature fusion from graph convolutional network and convolutional neural network. *Information Fusion 67* (2021), 208–229.

[177] Wang, T., and Chen, S.-C. Multi-Label Multi-Task Learning with Dynamic Task Weight Balancing. In *IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)* (2020), pp. 245–252.

[178] Wang, T., and Chen, S.-C. Hierarchical Multimodal Fusion Network with Dynamic Multi-task Learning. In *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)* (2021), IEEE, pp. 208–214.

[179] Wang, T., and Chen, S.-C. Multi-Task Local-Global Graph Network for Flight Delay Prediction. In *2022 IEEE 23rd International Conference on Information Reuse and Integration for Data Science (IRI)* (2022), IEEE, pp. 49–54.

[180] Wang, T., and Chen, S.-C. Adaptive joint cpatio-temporal graph learning network for traffic data forecasting. *ACM Transactions on Spatial Algorithms and Systems (TSAS)* (submitted).

[181] Wang, T., Pouyanfar, S., Tian, H., Tao, Y., Alonso, M., Luis, S., and Chen, S.-C. A framework for airfare price prediction: a machine learning approach. In *2019 IEEE 20th international conference on information reuse and integration for data science (IRI)* (2019), IEEE, pp. 200–207.

[182] Wang, T., Tao, Y., Chen, S.-C., and Shyu, M.-L. Multi-task multimodal learning for disaster situation assessment. In *2020 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)* (2020), IEEE, pp. 209–212.

[183] Weng, W., Lin, Y., Wu, S., Li, Y., and Kang, Y. Multi-label learning based on label-specific features and local pairwise label correlation. *Neurocomputing 273* (2018), 385–394.

[184] Wu, D., Pigou, L., Kindermans, P.-J., Le, N. D.-H., Shao, L., Dambre, J., and Odobez, J.-M. Deep dynamic neural networks for multimodal gesture segmentation and recognition. *IEEE transactions on pattern analysis and machine intelligence 38*, 8 (2016), 1583–1597.

[185] Wu, Z., Pan, S., Long, G., Jiang, J., and Zhang, C. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019* (2019), S. Kraus, Ed., ijcai.org, pp. 1907–1913.

[186] Xu, S., Zhou, D., Fang, J., Yin, J., Bin, Z., and Zhang, L. Fusionpainting: Multimodal fusion with adaptive attention for 3d object detection. In *2021 IEEE*

International Intelligent Transportation Systems Conference (ITSC) (2021), IEEE, pp. 3047–3054.

[187] Xue, W., Brahm, G., Pandey, S., Leung, S., and Li, S. Full left ventricle quantification via deep multitask relationships learning. *Medical image analysis 43* (2018), 54–65.

[188] Yeh, C.-K., Wu, W.-C., Ko, W.-J., and Wang, Y.-C. F. Learning deep latent space for multi-label classification. In *Thirty-First AAAI Conference on Artificial Intelligence* (2017).

[189] Yin, J., Karimi, S., Lampert, A., Cameron, M. A., Robinson, B., and Power, R. Using Social Media to Enhance Emergency Situation Awareness: Extended Abstract. In *Twenty-Fourth International Joint Conference on Artificial Intelligence* (2015), pp. 4234–4239.

[190] Yoon, Y., Yu, J., and Jeon, M. Predictively encoded graph convolutional network for noise-robust skeleton-based action recognition. *Applied Intelligence 52*, 3 (2022), 2317–2331.

[191] Yu, B., Guo, Z., Asian, S., Wang, H., and Chen, G. Flight delay prediction for commercial air transport: A deep learning approach. *Transportation Research Part E: Logistics and Transportation Review 125* (2019), 203–221.

[192] Yu, B., Yin, H., and Zhu, Z. Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden* (2018), J. Lang, Ed., ijcai.org, pp. 3634–3640.

[193] Yu, E.-Y., Wang, Y.-P., Fu, Y., Chen, D.-B., and Xie, M. Identifying critical nodes in complex networks via graph convolutional networks. *Knowledge-Based Systems 198* (2020), 105893.

[194] Zadeh, A., Chen, M., Poria, S., Cambria, E., and Morency, L. Tensor Fusion Network for Multimodal Sentiment Analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2017), Association for Computational Linguistics, pp. 1103–1114.

[195] Zadeh, A., Chen, M., Poria, S., Cambria, E., and Morency, L.-P. Tensor fusion network for multimodal sentiment analysis. *arXiv preprint arXiv:1707.07250* (2017).

[196] Zadeh, A. B., Liang, P. P., Poria, S., Cambria, E., and Morency, L.-P. Multimodal language analysis in the wild: Cmu-mosei dataset and interpretable dynamic fusion graph. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (2018), pp. 2236–2246.

[197] Zamir, A. R., Sax, A., Shen, W., Guibas, L. J., Malik, J., and Savarese, S. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 3712–3722.

[198] Zeng, W., Li, J., Quan, Z., and Lu, X. A Deep Graph-Embedded LSTM Neural Network Approach for Airport Delay Prediction. *Journal of Advanced Transportation 2021* (2021).

[199] Zhang, D., Shen, D., Initiative, A. D. N., et al. Multi-modal multi-task learning for joint prediction of multiple regression and classification variables in Alzheimer's disease. *NeuroImage 59*, 2 (2012), 895–907.

[200] Zhang, H., Wu, W., Zhang, S., and Witlox, F. Simulation analysis on flight delay propagation under different network configurations. *IEEE Access 8* (2020), 103236–103244.

[201] Zhang, J., Liang, J., and Hu, H. Multi-view texture classification using hierarchical synthetic images. *Multimedia Tools and Applications 76*, 16 (2017), 17511–17523.

[202] Zhang, W., Zhu, F., Lv, Y., Tan, C., Liu, W., Zhang, X., and Wang, F.-Y. AdapGL: An adaptive graph learning algorithm for traffic prediction based on spatiotemporal neural networks. *Transportation Research Part C: Emerging Technologies 139* (2022), 103659.

[203] Zhang, X., Zhao, J. J., and LeCun, Y. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems* (2015), pp. 649–657.

[204] Zhang, Y., Sidibé, D., Morel, O., and Mériaudeau, F. Deep multimodal fusion for semantic image segmentation: A survey. *Image and Vision Computing 105* (2021), 104042.

[205] Zhang, Y.-D., Satapathy, S. C., Guttery, D. S., Górriz, J. M., and Wang, S.-H. Improved breast cancer classification through combining graph convolutional network and convolutional neural network. *Information Processing & Management 58*, 2 (2021), 102439.

[206] Zhang, Z., Luo, P., Loy, C. C., and Tang, X. Facial landmark detection by deep multi-task learning. In *European conference on computer vision* (2014), Springer, pp. 94–108.

[207] Zhao, R.-W., Li, J., Chen, Y., Liu, J.-M., Jiang, Y.-G., and Xue, X. Regional Gating Neural Networks for Multi-label Image Classification. In *BMVC* (2016), pp. 1–12.

[208] Zhao, T., Xu, Y., Monfort, M., Choi, W., Baker, C., Zhao, Y., Wang, Y., and Wu, Y. N. Multi-agent tensor fusion for contextual trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 12126–12134.

[209] Zhao, X., Li, H., Shen, X., Liang, X., and Wu, Y. A modulation module for multi-task learning with applications in image retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 401–416.

[210] Zheng, C., Fan, X., Pan, S., Wu, Z., Wang, C., and Yu, P. S. Spatio-Temporal Joint Graph Convolutional Networks for Traffic Forecasting. *arXiv preprint arXiv:2111.13684* (2021).

[211] Zheng, C., Fan, X., Pan, S., Wu, Z., Wang, C., and Yu, P. S. Spatio-Temporal Joint Graph Convolutional Networks for Traffic Forecasting. *CoRR abs/2111.13684* (2021).

[212] Zheng, C., Fan, X., Wang, C., and Qi, J. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), vol. 34, pp. 1234–1241.

[213] Zhu, G., Zhang, L., Shen, P., and Song, J. Multimodal gesture recognition using 3-D convolution and convolutional LSTM. *IEEE Access 5* (2017), 4517–4524.

[214] Zhu, Q., Lin, L., Shyu, M.-L., and Chen, S.-C. Effective supervised discretization for classification based on correlation maximization. In *2011 IEEE International Conference on Information Reuse & Integration* (2011), IEEE, pp. 390–395.

VITA

Tianyi Wang is a Ph.D. candidate in the School of Science and Engineering at the University of Missouri-Kansas City (UMKC) under the supervision of Dr. Shu-Ching Chen and Dr. Mei-Ling Shyu. His research interests span the fields of data science, multimedia big data analytics, machine learning and deep learning. Previously, Tianyi received his B.S. degree in Accounting from Tianjin University of Finance and Economics in 2009, his M.S. degree in Accounting from Washington State University in 2011, and his second M.S. degree in Computer Science from FIU in 2017. He is the student leader in Florida Public Hurricane Loss Model (FPHLM) project. He published 13 research papers at top venues such as ACM Computing Surveys, IEEE Computer Vision and Pattern Recognition, Software: Practice and Experience, and IEEE Multimedia Information Processing and Retrieval.