# Knowledge Amalgamation from Heterogeneous Pre-Trained Models

**Jidapa Thadajarassiri**
Degree of Doctor of Philosophy
Data Science Program
Worcester Polytechnic Institute

**April 27, 2023**

**Committee Members:**
**Dr. Elke Rundensteiner, Professor, WPI. Advisor.**
**Dr. Xiangnan Kong, Associate Professor, WPI. Advisor.**
**Dr. Jian Zou, Associate Professor, WPI.**
**Dr. Supasorn Suwajanakorn, Lecturer, VISTEC (Thailand). External member.**

**Acknowledgments**

Throughout the journey of my Ph.D. studies, I would like to deeply thank many people who support me at every single step in this journey. First, my deepest gratitude goes to my wonderful advisor, Professor Elke Rundensteiner, and co-advisor, Professor Xiangnan Kong. I would like to thank Prof. Rundensteiner not only for her support on my work but also her positive attitude to encourage an enthusiastic and friendly atmosphere to our lab. I feel very fortunate to have her as my role model for my life and my career. I also thank deeply Prof. Kong for his sensible advice and patience which guides and inspires me how to contribute great research to our community. Without his support, I could not imagine how I could have gotten this far. Also special thank you to Professor Jian Zou and Professor Supasorn Suwajanakorn for accepting to serve on my dissertation committee.

I am also grateful for all amazing students who have always supported me: Thomas Hartvigsen, Cansu Sen, Walter Gerych, and all other daisy lab mates. My special thanks goes to Thomas Hartvigsen and Walter Gerych who devote time and effort to work very closely with me.

Lastly, I am absolutely speechless for unconditional and eternal love from my family. Thank you for their long patience and being there for me always.

**Abstract**

With the rapid advances in deep learning, many pre-trained models have been released for reuse to researchers and practitioners. These pre-trained models are invaluable, as they often perform better than models that can be trained using the limited resources and data available to the majority of the parties who utilize deep learning. These models typically contain unique knowledge derived from the tasks and datasets on which they were pre-trained. In consequence, reusing them may be difficult in practice because a downstream task may either require a knowledge base that is not completely captured by any single pre-trained model, or may not exactly match the task that had previously been solved by a specific model. This motivates the use of multiple pre-trained models simultaneously to expand the knowledge scope beyond any individual model. However, pre-trained models are usually large, leading the size of their ensemble to be huge and causing scalability problems on applications with limited resources such as mobile devices. This dissertation thus studies *knowledge amalgamation*, which is the problem of how best to combine complementary knowledge from *multiple heterogeneous pre-trained models*, referred to as *teachers*, into a lightweight *student* model. We propose four challenging knowledge amalgamation tasks corresponding to four different aspects of complementary knowledge inherent among teachers.

*Task 1: Meta-Embedding.* This task studies the combination of complementary knowledge from pre-trained *word embedding* models (teachers) in NLP such as Word2Vec or GloVe that contain representations for different words and encode distinct relationships between words. Often times, each individual teacher cannot fully meet the language needs for downstream text mining tasks, *e.g.*, neither teacher contains all the words used in a task, typically forcing a user to choose which teacher to use. Many works tackle this ambiguity by learning a meta-embedding model (student) to extend the word coverage captured in the union of multiple teachers into one model. However, they learn a meta-embedding for each word by reconstructing its corresponding representations in each source model without regarding the key information of the relations between the encodings of each model. In this task, we thus propose a meta-embedding method that directly preserves word-pair relations from multiple teachers.

*Task 2: Amalgamating Discriminative Knowledge.* A popular type of teacher model is a pre-trained *multi-class* classifier. Each teacher typically contains unique discriminative knowledge on a specific class set. This creates some restrictions when reusing an individual teacher on some downstream task, as this task may cover more classes than the teacher's specialized classes. For example, a downstream activity recognition task may contain more activities than were included in any individual teacher. Several studies in knowledge amalgamation thus

combine multiple teachers into one student model that becomes an expert on the union of the teachers' classes. Unfortunately, recent works focus exclusively on image classification and develop methods purely under the unsupervised setting that suffer heavily when an overconfident teacher provides inaccurate predictions with high confidence. In this task, we extend this study to the open problem of semi-supervised knowledge amalgamation, broadening the setting to a more realistic case in which we also explore sequence classification for the first time.

*Task 3: Amalgamating Label Dependencies.* In this task, we study pre-trained *multi-label* teacher models. These teachers usually contain complementary knowledge about the inter-class dependencies among class labels informed by the particular label sets that were given during their training. Unlike traditional supervised methods for multi-label classification that require huge labeled datasets, knowledge amalgamation approaches combine such multiple teachers into a student model to tackle multi-label problems without using any labeled data. However, the existing works train each label as an independent binary classification problem. They overlook the information on label inter-dependency, which is crucial for multi-label classification. To this end, we seek a solution to amalgamate heterogeneous teachers into a student model that aims to capture dependencies among the union of labels between all teachers.

*Task 4: Multi-Tasking Knowledge Amalgamation.* This study focuses on amalgamating knowledge from pre-trained multi-task teacher models. The key knowledge contained in these multi-task teachers is the shared representation that benefits the generalization of all tasks simultaneously. However, each teacher encodes different generalized knowledge as they aim to learn the shared representation to be applied to the different sets of tasks. Thus, some existing knowledge amalgamation works aim to combine this complementary knowledge in order to improve the common knowledge to be used for all tasks handled across all teachers. Unfortunately, they make the unrealistic assumption that all teachers and the student have an identical architecture and develop the layer-to-layer approaches to train the student. In practice, teachers often come with different architectures as they are pre-trained separately. Therefore, this task aims to develop the student model that can effectively amalgamate the complementary shared knowledge captured across multi-task teachers with heterogeneous architectures in order to learn high-quality common feature used for all tasks across all teachers.

# Contents


**1 Introduction** **7**
- 1.1 Motivation . . . 7
- 1.2 Knowledge Amalgamation . . . 8
- 1.3 Dissertation Tasks . . . 11
- 1.4 Publication List . . . 13

**2 Meta-Embedding** **15**
- 2.1 Motivation . . . 15
- 2.2 Related Works . . . 16
- 2.3 Problem Definition . . . 17
- 2.4 Challenges . . . 18
- 2.5 Proposed Method: **Sim**ilarity-Preserving **Me**ta-Embedding (SimME) . . . 19
  - 2.5.1 Relation-Encoder . . . 19
  - 2.5.2 Maskout . . . 21
- 2.6 Evaluation . . . 22
- 2.7 Conclusions . . . 26

**3 Amalgamating Discriminative Knowledge** **27**
- 3.1 Motivation . . . 27
- 3.2 Related Works . . . 28
- 3.3 Problem Definition . . . 29
- 3.4 Challenges . . . 30
- 3.5 Proposed Method: **T**eacher **C**oordinator (TC) . . . 31
  - 3.5.1 Teacher Trust Learner . . . 32
  - 3.5.2 Knowledge Amalgamator . . . 33
- 3.6 Evaluation . . . 34
- 3.7 Conclusions . . . 38

**4 Amalgamating Label Dependencies** **40**
- 4.1 Motivation . . . 40
- 4.2 Related works . . . 41
- 4.3 Problem Definition . . . 42
- 4.4 Challenges . . . 43
- 4.5 Proposed Method: **A**daptive K**n**owledge **T**ransfer (ANT) . . . 44
  - 4.5.1 Transfer Indicator (TI) . . . 45
  - 4.5.2 Knowledge Transfer Module . . . 46
  - 4.5.3 Prediction Integrator . . . 48
- 4.6 Evaluation . . . 50
- 4.7 Conclusions . . . 56

**5 Multi-Tasking Knowledge Amalgamation** **57**
5.1 Motivation . . . 57
5.2 Related works . . . 58
5.3 Problem Definition . . . 60
5.4 Challenges . . . 61
5.5 Proposed Method: **Ve**rsatile Common Feat**u**re Con**s**olidator (VENUS) . . . 62
5.5.1 Backbone Model. . . . 64
5.5.2 Feature Consolidator. . . . 65
5.5.3 Task-Specific Layers. . . . 65
5.6 Evaluation . . . 67
5.7 Conclusions . . . 73

**6 Conclusions** **74**
6.1 Summary of Contributions . . . 74
6.2 Future Direction . . . 75

# 1 Introduction

## 1.1 Motivation

Deep learning has received significant attention in the past decade due to its unprecedented success on several real-world tasks such as sentiment analysis [36, 122, 115], object detection [40, 54, 99], activity recognition [74, 111, 116], and clinical diagnosis [84, 76, 39]. These advances have mostly occurred using conventional supervised learning approaches that require substantial computing resources and huge datasets to train large models. Unfortunately, acquiring labeled data in practice is very expensive and several domains rarely have publicly-available data due to privacy concerns. For example, health records and customer activity data often contain sensitive private information, and are thus often unavailable to outside practitioners. This dramatically hinders the training of robust models from scratch, a problem that is compounded by limitations on computational resources.

Several research groups attempt to support other researchers and practitioners in overcoming these burdens by using their powerful computing resources to train a model on some large dataset and releasing pre-trained models [65, 77, 39, 40, 45, 114, 5, 94, 81, 122] for reuse in various domains. Each of these models, referred to as a *teacher* model, is typically trained to solve a specific problem, which leads it to have unique characteristics and knowledge specific to its own task. Therefore, reusing individual teachers for a downstream task in practice may be limited due to: (1) The knowledge base required for a downstream task may not be completely captured by any arbitrary teacher. For instance, a text-mining task of using Twitter to support clinical diagnosis needs word representations that can capture language use by both Twitter users and clinicians. However, the existing off-the-shelf word representations (teachers) are pre-trained on only either source which cannot fully cover the knowledge needed for the task. (2) A reuse task may not exactly match the task solved by any individual teacher. For example, a task to identify human

activities on a set of classes {*standing*, *sitting*, *walking*} but the two available teachers are trained on {*standing*, *sitting*} and {*sitting*, *walking*}, respectively.

These restrictions encourage the use of multiple teachers simultaneously, which can expand the knowledge scope to solve a broader set of tasks. Unfortunately, pre-trained teachers usually come with a very large size *e.g.,* 117M parameters for ResNet152 [40], 138M for VGG16 [94], or 88M for YOLOv5 [48] which could drive the size of their ensembles to be prohibitively huge. This significantly restricts their use for some applications with limited resources such as mobile devices, smart watches, or smart home appliances.

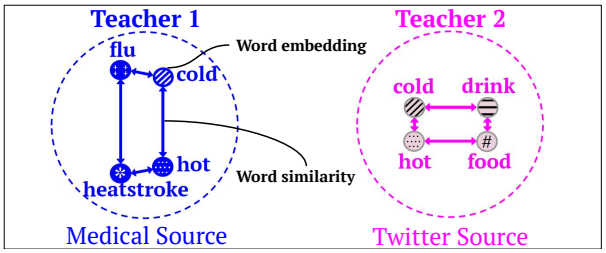
## 1.2 Knowledge Amalgamation

To maximize benefit from such available pre-trained models, researchers have studied a variety of directions for their reuse. The simplest method is ensemble learning [49, 25] that combines the outputs from multiple models by averaging their predicted scores or using majority voting. The latest ensemble methods apply deep learning techniques such as Drop Connection [109], Stochastic Depth [46], and Swapout [96]. These approaches require all models to tackle exactly the same task and to be run simultaneously which could cause scalability problem.

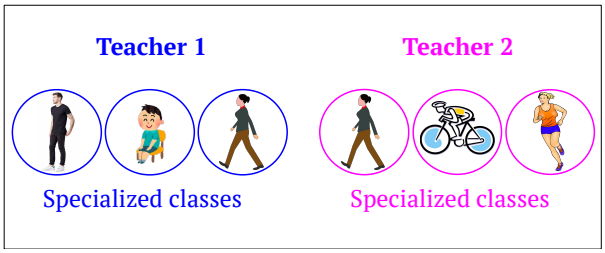
Later, with original goal to address such the scalability issue, knowledge distillation [42] propose to compress a large teacher model into a compact student model by using the teacher's soft predictions (logits) to supervise the student model. [1, 110] extend this idea to learning also from the other intermediate layers. However, these works focus mainly on single teacher - single student manner that assumes the teacher and the student solve the same task.

Inspired by these works, several researchers have been established numerous works [120, 9, 10, 119, 93, 108, 92, 60] towards *Knowledge Amalgamation* or *KA* to extend the study towards a more comprehensive objective. Knowledge amalgamation aims to combine *complementary knowledge* from multiple pre-trained models, referred to as *heterogeneous*

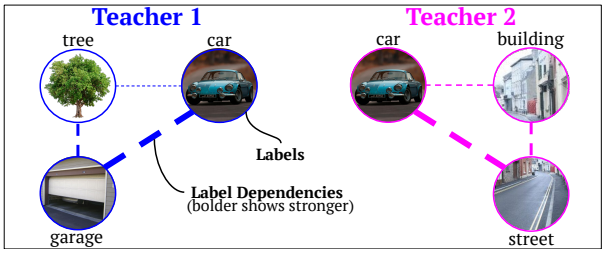
*teachers*, into one *student* model that is significantly smaller than the ensemble of original large teachers. It advances beyond knowledge distillation by combining teachers that specialize in different tasks, which is a more general form of the knowledge distillation problem. In this dissertation, as illustrated in Figure 1, we cover four types of complementary knowledge existing among teachers which can be summarized as follows:



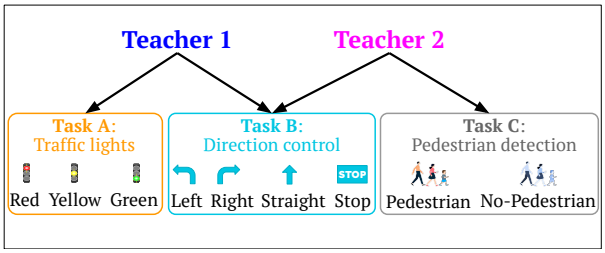

(a) *Instance Relation Knowledge.* Each teacher learns representations for different instances (words) and captures their relation differently.




(b) *Discriminative Knowledge.* Individual teacher has specific knowledge to discriminate a different set of classes.




(c) *Label Dependency Knowledge.* Teachers capture unique dependencies among class labels regarding the label sets they have experienced differently.




(d) *Multi-Tasking Knowledge.* Each teacher contains different generalized knowledge shared between different sets of related tasks.

Figure 1: Four types of knowledge inherent complementarily among pre-trained models (*teachers*) that are studied in this dissertation.

**Instance Relation Knowledge.** Several research groups [63, 77, 82, 124, 45] invest their resources to publish a pre-trained model that aims to obtain powerful instance representations to be used across a broad range of downstream tasks. As these representations are not specific to a task, these models usually learn high-quality representations that capture inherent relationship among instances. However, each of them is trained on a different dataset, leading them to represent different perspectives on how instances relate to each other. For example, in Natural Language Processing (NLP), there are many available

pre-trained word embedding models. Each captures a different vocabulary and encodes word relations differently depending on their context in their training corpus. As shown in Figure 1a, a teacher may be trained on a medical corpus capturing a set of words including *flu*, *heatstroke*, *cold*, and *hot* while another teacher may be trained on data collected from Twitter, capturing a different set of words including *drink*, *food*, *cold*, and *hot*. In this case, the two teachers encode a relation between *cold* and *hot* differently, being far apart in Teacher 1 while similar in Teacher 2, which are induced from their distinct context.

**Discriminative Knowledge.** Many other researchers [39, 40, 44, 72, 51] pre-train multi-class classification teachers. These models are trained on different datasets and typically solve different sets of classes. This leads each to contain discriminative information on a unique class set, such that their complement may contain knowledge beyond any individual teacher. For instance, consider a human activity recognition task as illustrated in Figure 1b. As shown, a set of three classes (*standing*, *sitting*, *walking*) might be the target in Teacher 1 while the other set of classes (*walking*, *cycling*, *running*) might be all that is known by Teacher 2. Their discriminative knowledge on such different activity sets complements each other to support the broader task of identifying all five activities: *standing*, *sitting*, *walking*, *cycling*, *running*.

**Label Dependency Knowledge.** Multi-label classification is another task that has gained increased attention from groups that publish pre-trained models [114, 48] since this task is applicable to a large variety of real-world problems [123, 55, 90, 31, 107]. Depending on which training data are used, each pre-trained model has been trained on a different label set and consequently captures different key knowledge of the dependencies among the labels. Figure 1c depicts an example of this setting. A dataset, used for Teacher 1, may cover the labels of *tree*, *car*, and *garage*. On the other hand, Teacher 2 may use another dataset containing the other labels including *car*, *building*, and *street*. The two teachers capture label dependencies limited to their experience which introduces different knowl-

edge among them. Some label dependencies may appear only in one teacher, for example *car* and *garage* in Teacher 1 or *car* and *street* in Teacher 2. Moreover, their predictions on the shared labels (*e.g., car*) may be different depending on their own particular context.

**Multi-Tasking Knowledge.** Other research groups [91, 68, 2] release their pre-trained models for multi-task learning. Each of these models is trained to learn the shared feature that benefits the generalization across a set of tasks. However, the task set handled by each teacher is typically different depending on their particular study. As illustrated in Figure 1d, Teacher 1 may learn to detect the traffic lights (task A) and predict the direction (task B) while Teacher 2 may learn another set of tasks including predicting the direction (task B) and detecting pedestrians (task C). Therefore, the two teachers capture the generalized knowledge specifically to the different related tasks. Combining their complementary knowledge would ideally improve the shared feature that could boost the performance of the three tasks simultaneously.

### 1.3 Dissertation Tasks

In this dissertation, we study four tasks of knowledge amalgamation according to the previously described types of complementary knowledge captured among multiple teachers.

**Task 1.** ***Meta-Embedding.*** This task focuses on pre-trained representation learning models that capture complementary knowledge of relationship among instances. We particularly focus on the NLP domain as this problem is prevalent among the available *pre-trained word embedding models*, which we consider as *teachers*. Our goal is to combine these pre-trained models into one student model, called a *meta-embedding*. A good solution to learn meta-embeddings should retain word relations that are encoded thoroughly in each pre-trained teacher. Also, it should be able to handle the words that only have embeddings in some but not all of the teachers.

**Task 2.** ***Amalgamating Discriminative Knowledge.*** In many cases, pre-trained models, or teachers, handle distinct and complementary sets of specialized classes for *multi-class classification*. This task aims to combine such heterogeneous teachers into a student model the can predict over the *union of their classes*. In this study, we assume access to few labeled samples along with many unlabeled samples. Additionally, we devote our study to sequence classification, which has never been explored in this context. An ideal solution should effectively fuse heterogeneous knowledge from teachers that may contain unrelated information as they are trained on different problems. Moreover, such a solution should be robust to the challenging case when a teacher provides inaccurate predictions with high confidence.

**Task 3.** ***Amalgamating Label Dependencies.*** In this task, we study knowledge amalgamation on *multi-label classification*. Each teacher is trained on a different label set and thus encodes label dependencies limited only to its known labels. Our task is to fuse the distinct knowledge from teachers into a student model that captures all *dependencies* among labels from all teachers, trained from only unlabeled data. Without any supervision, it is very challenging to learn such dependencies between all label-pairs in the union label set from teachers. This is especially true for label-pairs that have never been observed by any teacher, *e.g.*, learning dependency between *garage* and *street* that no teacher has knowledge about, as depicted in Figure 1c.

**Task 4.** ***Multi-Tasking Knowledge Amalgamation.*** Lastly, we explore the open problem of knowledge amalgamation for *multi-task learning*. In this setting, each teacher captures different *shared representations* that benefit the different sets of related tasks. Existing works assume strongly that the teachers and the student have an identical architecture while, in practice, they are pre-trained separately and may have different architectures. Therefore, in this task, we aim to train a student that can effectively combine knowledge from multi-task teachers with heterogeneous architectures. The ultimate goal of the student model is

to fuse their knowledge into the rich common feature that is effectively generalizable for all tasks across teachers.

### 1.4 Publication List

- **Jidapa Thadajarassiri**, Walter Gerych, Xiangnan Kong, Elke Rundensteiner. *Knowledge Amalgamation for Multi-Task Models with Heterogeneous Architectures.* In preparation for submission to CIKM 2023.
- Thomas Hartvigsen, **Jidapa Thadajarassiri**, Xiangnan Kong, Elke Rundensteiner. *Continuous-Time Attention Networks for Irregularly-Sampled Time Series Classification.* In submission to ICML 2023.
- **Jidapa Thadajarassiri**, Thomas Hartvigsen, Walter Gerych, Xiangnan Kong, Elke Rundensteiner. *Knowledge Amalgamation for Multi-Label Classification via Label Dependency Transfer.* In *Proceedings of AAAI*, 2023.
- Thomas Hartvigsen, Walter Gerych, **Jidapa Thadajarassiri**, Xiangnan Kong, Elke Rundensteiner. *Stop&Hop: Early Classification of Irregular Time Series.* In *Proceedings of CIKM*, 2022.
- ML Tlachac, Walter Gerych, Kratika Agrawal, Nicholas Jurovich, Benjamin Litterer, Saitheeraj Thatigotla, **Jidapa Thadajarassiri**, Elke Rundensteiner. *Conditional Sequence Generative Adversarial Networks for Text Generation.* In *BigData workshop* 2022.
- Dongyu Zhang, Cansu Sen, **Jidapa Thadajarassiri**, Thomas Hartvigsen, Xiangnan Kong, Elke Rundensteiner. *Human-like Explanation for Text Classification With Limited Attention Supervision.* In *Proceedings of IEEE BigData*, 2021.
- **Jidapa Thadajarassiri**, Thomas Hartvigsen, Xiangnan Kong, Elke Rundensteiner. *Semi-Supervised Knowledge Amalgamation for Sequence Classification.* In *Proceedings of AAAI*, 2021.

- **Jidapa Thadajarassiri**, Cansu Sen, Thomas Hartvigsen, Xiangnan Kong, Elke Rundensteiner. *Learning Similarity-Preserving Meta-Embedding for Text Mining.* In *Proceedings of IEEE BigData*, 2020.
- Dongyu Zhang, **Jidapa Thadajarassiri**, Cansu Sen, Elke Rundensteiner. *Time-Aware Transformer-based Network for Clinical Notes Series Prediction.* In *Proceedings of MLHC*, 2020.
- **Jidapa Thadajarassiri**, Cansu Sen, Thomas Hartvigsen, Xiangnan Kong, Elke Rundensteiner. *Comparing General and Locally-Learned Word Embeddings for Clinical Text Mining.* In *Proceedings of IEEE BHI*, 2019.

# 2 Meta-Embedding

This task is published at IEEE BigData 2020 [102].

## 2.1 Motivation

The progress of deep learning methods generally depends on how good a model can generate data representations to be used for a task. However, powerful representations should express fundamental information from data that is most useful for general problems, i.e. not task-specific. Thus, data representations should be able to capture inherent relation or similarity among instances so as to most benefit any downstream tasks. To achieve this, massive training data along with substantial computational resources are required which practically prohibit most practitioners to acquire.

To accelerate advances to deep learning community, many research groups [63, 77, 82, 124, 40, 45] invest their resources for such persistent training and release the powerful representation models for publicly use. However, each organization pre-trains its model from different data that consequently each of them encodes relations among instances differently regarding the unique context inherent in a specific training data. In this task, we focus particularly on Natural Language Processing (NLP) domain where the problem presents outstanding across various choices of pre-trained word embedding

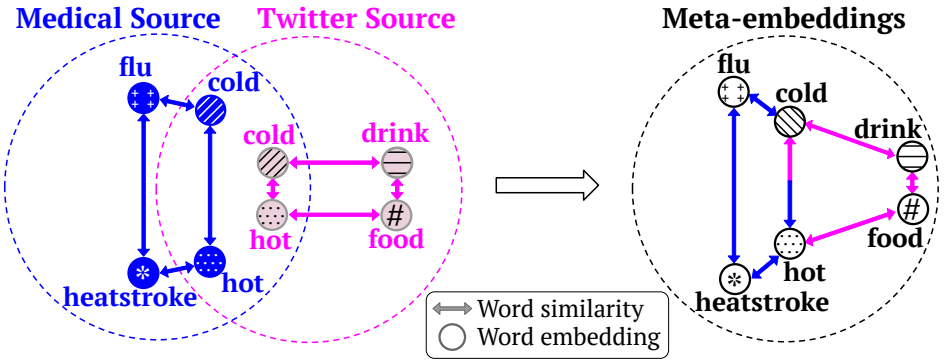



Figure 2: Learning meta-embeddings from multiple pre-trained embedding models that each source contains different vocabulary and endoce different relations among words.

models. Each covers different words and encodes their relations differently which suffers the reuse purpose significantly when a downstream task requires knowledge not limited to only individual pre-trained model but instead across multiple of them.

For example, as shown in Figure 2, assume our task involves a set of 6 words (*flu, heat-stroke, cold, hot, drink,* and *food*) while there are two available embedding models trained from medical corpus and Twitters respectively. Neither model can fully support the need in our task as only a subset of the 6 words is provided in each source. However, their complement could support us perfectly in this case.

To this end, our goal is to amalgamate complementary knowledge from multiple pre-trained embedding models, referred to as *teachers*, into a student model which is called *meta-embedding*. We aim to learn a meta-embedding model to cover words needed for a downstream task. More importantly, such an integrated model should effectively retain all informative relations among words that are carefully captured by each teacher model.

### 2.2 Related Works

Among many existing pre-trained word embeddings [64, 77, 82, 124], several studies [18, 113] show that a task performance varies dramatically across such pre-trained models while identifying which one is the best choice for which task remains unclear.

Recent works on *meta-embedding* [120, 20, 9, 10, 69] aim to overcome the choice issue by integrating multiple pre-trained embeddings into a new integrated space to be used across many downstream tasks. Most of them focus mainly on word-encoder methods by reconstructing encoded values of individual *word* from multiple pre-trained models. The two simplest approaches are averaging [20] and concatenating [120] while the latter increases dimensionality issue. More powerful methods, named 1toN and 1toN+ [120], train a shallow network to learn meta-embedding for each word by optimizing its values to be most similar to the corresponding raw values in original embedding sources. [10] consider neighbor words into account by training a meta-embedding for a word from lin-

ear weighted sums of its k-nearest neighboring words. This requires a hand-selection on the number of the neighbors to be counted. Most recently, [9] propose to apply autoencoder approach to encode the source embeddings to a meta-embedding that is decoded to the source with three operations including averaging, concatenating and decoupling.

None of these methods explicitly capture informative knowledge about the *relations between words* that are carefully encoded in each pre-trained embedding models. Moreover, most of these works approach words that not appear in only some sources (out-of-vocabulary or OOV) ineffectively. They develop random embeddings for these OOV words in each source which lead some bias information to learning meta-embeddings for them.

### 2.3 Problem Definition

Considering a task related to a vocabulary set $\mathcal{V}$ that consists of $t$ unique words, *i.e.* $\mathcal{V} = \{w_k\}_{k=1}^{t}$, the goal is to learn a *meta-embedding space* $\mathbf{M}$ that generates meta-embeddings corresponding to each word in $\mathcal{V}$ from complementary knowledge encoded in multiple pre-trained embedding sources. Let $d$ be a pre-defined dimension of meta-embeddings in $\mathbf{M}$. For each $w_k \in \mathcal{V}$, we denote the $d$-dimensional meta-embedding of $w_k$ as $\mathbf{e}_k$. Thus, the target space $\mathbf{M} = \{\mathbf{e}_k\}_{k=1}^{t}$ and its size is $t \times d$ in corresponding to the target vocabulary size, $|\mathcal{V}| = t$.

Let $\mathcal{P} = \{p_i\}_{i=1}^{n}$ denote a set of word-pairs composed from all words in $\mathcal{V}$. Given a set of pre-trained sources $\mathcal{S} = \{\mathbf{S}^j\}_{j=1}^{m}$, the number of words in each source may differ denoted by $\mathcal{V}^1, \mathcal{V}^2, \ldots, \mathcal{V}^m$ respectively. Note that the embedding dimension in each source may also differ. For each embedding source $\mathbf{S}^j$, if both words in a word-pair $p_i$ appear in $\mathbf{S}^j$, we measure their relation by computing cosine similarity score between their corresponding embeddings encoded in $\mathbf{S}^j$. We notate such score as $y_{p_i}^j$.

Our goal is to learn the $t$ meta-embeddings in $\mathbf{M}$ that preserve their relations as observed across embedding sources in $\mathcal{S}$. Therefore, we train meta-embeddings that the

similarity score of each word-pair $p_i$ in M, denoted as $y_{p_i}$, is most similar to its corresponding scores in the sources $\{y_{p_i}^j\}_{j=1}^m$. For the case of $p_i$ that does not appear in all sources, its similarity score should be optimized by only the scores from sources that truly observe such word-pair.

### 2.4 Challenges

Amalgamating knowledge from multiple pre-trained embedding models associates with two main challenges as follows:

- *Preservation of word similarity*. The most informative knowledge captured by each pre-trained embedding model is *relative distance between words* encoding relation among words from their training corpus. By directly projecting actual values of word embeddings from original sources into a meta-embedding space does not guarantee to preserve their distances. The successful meta-embedding learning should instead consider carefully on how to preserve such meaningful distances. This may challenge further as the observed relations of a word-pair across pre-trained sources may contradict one another, leading conflicts to be handled by the learning process such as the relation between *hot* and *cold*, shown in Figure 2, is captured to be close in the pre-trained model from Twitter while far apart in the other model trained on medical corpus.

- *Out-of-vocabulary (OOV) words*. Word coverage by each pre-trained embedding model depends entirely on its training corpus. Thus, several words may not exist in all embedding sources, referred to as out-of-vocabulary or OOV words. For example, {*flu* and *heatstroke*} or {*drink* and *food*} appear only in one source as illustrated in Figure 2. Meta-embeddings for these OOV words might become biased if they are learned from the source that does not really contain information regarding these words. It is therefore challenging to train a meta-embedding space that could extract only such

accurate knowledge from multiple sources.

## 2.5 Proposed Method: Similarity-Preserving Meta-Embedding (SimME)

In this work, we propose the *relation-based* learning method for meta-embeddings named **Sim**ilarity-Preserving **M**eta-**E**mbedding (SimME). Our approach trains a joint space M that directly preserves relations among words encoded in multiple pre-trained embedding models. Figure 3 depicts the overall architecture of the proposed SimME that consists of two key ingredients: the *Relation-Encoder* and the *Maskout*.

### 2.5.1 Relation-Encoder

This network maps a word-pair through the meta-embedding space M to encode their similarity. As illustrated in Figure 3, it first feeds a word-pair $p_i$ consisting of two words

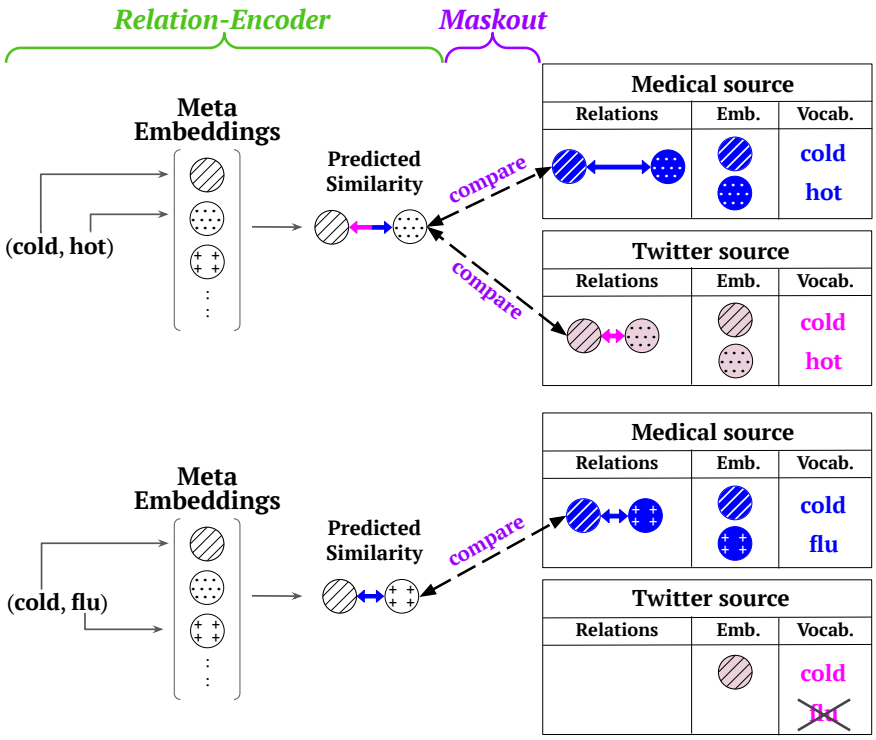



Figure 3: The proposed relation-based meta-embedding learning.

$w_k$ and $w_l$ into the network simultaneously. The Relation-Encoder then extracts the $d$-dimensional meta-embeddings for both words by applying an affine transformation on their one-hot vectors, denoted as $\mathbf{x}_k$ and $\mathbf{x}_l$ respectively, via the latent space $\mathbf{M}$ as shown in Equation 1.

$$\mathbf{e}_k = \mathbf{M}\mathbf{x}_k + \mathbf{b} \tag{1}$$

where $\mathbf{e}_k$ is the meta-embedding of $w_k$ and $\mathbf{b}$ is a bias vector. Extracting $\mathbf{e}_l$ for $w_l$ takes place simultaneously with the same operation. Then the cosine similarity score between their meta-embeddings, notated by $y_{p_i}$, is measured as follows:

$$y_{p_i} = \frac{{\mathbf{e}_k}^\top \cdot \mathbf{e}_l}{\|\mathbf{e}_k\| \cdot \|\mathbf{e}_l\|} \tag{2}$$

where $\|\|$ indicates cardinality and $\cdot$ is the dot product.

Similarly for each original pre-trained source that contain both $w_k$ and $w_l$ in a pair $p_i$, the cosine similarity score between their corresponding embeddings is each pre-trained model is computed. This yield a collection of similarity scores from the $m$ pre-trained sources which are denoted as $y^1_{p_i}, y^2_{p_i}, \ldots, y^m_{p_i}$. To preserve this collection of similarities, SimME is trained to minimize the mean squared error between such a collection and the similarity score in the meta-embedding space as shown in Equation 2. All parameters in $\mathbf{M}$ are thus updated through the following loss function:

$$J(\mathbf{M}) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(y_{p_i} - y^j_{p_i}\right)^2 \tag{3}$$

where $m$ is the number of pre-trained models and $n$ is the number of word-pairs.

### 2.5.2 Maskout

Regarding the prevalent existence of OOV words across pre-trained sources, several word-pairs related to them are observed sparsely and not presented in all sources. For example in Figure 2, *flu* and *heatstroke* exist only in the medical source that are considered as OOV words with respect to the pre-trained embeddings trained from Twitter. These two words involve with several word-pairs including (*flu*, *heatstroke*), (*hot*, *heatstroke*) and (*cold*, *flu*) that are also observed from one source only. We refer to these pairs as *nonmutual* relation.

In contrast to other state-of-the-art approaches [120, 9] that create artificial embeddings for these OOV words to be used for their training process, we propose a new loss term for meta-embedding, called *maskout*. *Maskout* steers the learning to adaptively train meta-embeddings based solely on the sources that actually observe such *nonmutual* relations. SimME, therefore, succeeds to avoid biased information from synthetic embeddings.

To achieve this, we employ an indicator function to modify the loss function in Equation 3. This allow SimME to be flexibly undated based on the sources that contain accurate information. The new loss function is:

$$J(\mathbf{M}) = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} \mathbb{1}_{p_i \in \mathbf{S}^j} \left( y_{p_i} - y_{p_i}^j \right)^2 \tag{4}$$

where $p_i$ is a word-pair in $\mathcal{P}$ and $\mathbb{1}$ is an indicator function:

$$\mathbb{1}_{p_i \in \mathbf{S}^j} = \begin{cases} 1 & \text{if } p_i \in \mathbf{S}^j \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

For the other word-pairs that their relations observed across all sources, SimME is optimized to preserve all similarity scores with equal weight which yields the consensus of word-pair relations among multiple pre-trained embedding sources.

## 2.6 Evaluation

We investigate amalgamating three popular pre-trained word embeddings which could form four possible meta-embedding spaces. The proposed SimME is trained to obtain such four spaces to be evaluated on four text mining tasks against six alternative methods.

**Pre-trained Word Embeddings.** We conduct our experiments using three word embeddings pre-trained from different corpus as follows:

- *Wikipedia* [77] covers 400,000 word embeddings that are trained on Wikipedia 2014 and Gigaword 5.
- *BioMed* [82] contains word embeddings for 2.4 million words that are trained from documents in PubMed and PubMed Central.
- *Twitter* [77] provides 1.2 million word embeddings that are trained from tweets.

**Meta-Embedding Spaces.** There are four possible meta-embedding spaces to be considered by amalgamating the three pre-trained embedding sources described above.

- WIKI-BIO combines word relations encoded in medical context from BioMed and general use from Wikipedia. While the two sources have only 124K words in common, their union improves the coverage to 2.6M words.
- BIO-TWITTER integrates word relations captured in social communication from Twitter and clinical style from BioMed. The two sources contain 63K words in their overlap while 3.5M words in their union.
- WIKI-TWITTER fuses word relations used for general purpose in Wikipedia and socializing use in Twitter. Combining these sources drive vocabulary coverage to 1.4M words, although the number of words covered in both sources is low as 145K words.

- WIKI-BIO-TWITTER amalgamates all three pre-trained sources promoting their vocabulary union to 3.7M words while there are only 59K appear in all sources.

**Compared Methods.** We compare SimME against seven alternative methods as follows:

- *Individual Sources*. This baseline uses each pre-trained embedding model in isolation. There is no incremental information from knowledge amalgamation.
- *Average* [20]. The embeddings of the same words in all sources are averaged, requiring them to have equal dimension.
- *Concat: Concatenation* [120]. This method simply concatenates the embeddings of the same words in all sources, increasing the dimension of meta-embeddings proportionally to the number of sources.
- *LLE: Locally Linear Meta-Embedding* [10]. This learns meta-embedding for each word from a linearly weighted sum of its k-nearest neighbor embeddings which may not capture relations to other words outside the neighborhood but are needed to be used in a downstream task.
- *AAEME: Averaged Auto-encoded Meta-Embedding* [9]. This method encodes raw embeddings of each word from each pre-trained source. Such encoded outputs are averaged to form the word's meta-embedding that is trained to decode or reconstruct each raw embedding.
- *CAEME: Concatenated Auto-encoded Meta-Embedding* [9]. Similar to AAEME, a meta-embedding is now formed by concatenating the encoded outputs.
- *DAEME: Decoupled Auto-encoded Meta-Embedding* [9]. Similar to CAEME, the encoded outputs are concatenated to form a meta-embedding. However, instead of using this meta-embedding to decode each raw embedding, each encoded output as

a part of the meta-embedding is separately trained to decode its corresponding raw embedding.

Original sources assign the special unknown-token embedding for OOV words. This influences to the average and concatenation methods that feed such unknown-token into their operations when a word does not exist in any source. LLE does not have the OOV issue as every word can be constructed from its own neighbor words. The other methods create synthetic embeddings for OOV words in the sources that are used to train meta-embeddings but also are self-trained simultaneous.

**Experiments and Results.** Our experiments are conducted to prove the key idea that an effective learning process for meta-embeddings should improve the quality of word representations that are beneficial to broad tasks in text mining. Thus, we evaluate SimME on four text mining tasks: (1) Semantics similarity prediction evaluated on Card [80], RareWord [61], and WordNet [66]. (2) Synonym detection evaluated on WordRep [33], and WordNet [66]. (3) Text classification evaluated on movie reviews [75], ProCon reviews [32], and the public Kaggle Dataset: Twitter US Airline Sentiment. (4) Concept categorization evaluated on ESSLLI [7].

Table 1 reports the performance on four tasks compared across all methods. The different appropriate metrics are used on each task as described in Table 1. These results show strong performance of SimME that achieves the top average rank across all tasks in all combinations. The success of SimME on semantics similarity prediction shows that it successfully learns meta-embeddings that capture semantic similarities of word-pairs to be most analogous to human-rated scores. Moreover, SimME achieves in not only projecting the meta-embeddings of synonymous words into the same neighborhoods in the learned space but also projecting words in the same category closed together. This is shown by the impressive performance on the experiments on synonym detection and concept categorization. Lastly, the results on text classification indicate that SimME suc-

| Datasets / Methods | *Semantics prediction* | | | *Synonym detection* | | *Text Classification* | | | *Concept* | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Card | RareWord | WordNet | WordRep | WordNet | Airline | Movie | ProCon | ESSLLI | Ave. Rank |
| **WIKI-BIO** | | | | | | | | | | |
| Wiki | .10 (7) | .33 (5) | .49 (4) | .66 (8) | .79 (5) | .83 (4) | **.59** (1) | .87 (5) | .17 (8) | 5.2 |
| Biomed | .12 (2) | .24 (7) | .18 (8) | .70 (4) | .67 (8) | .81 (6) | .56 (5) | .86 (6) | .19 (5) | 5.7 |
| Average | .10 (5) | .33 (6) | .51 (3) | .68 (5) | .80 (4) | .83 (2) | .56 (5) | .87 (4) | .19 (6) | 4.4 |
| Concat | .10 (4) | .34 (4) | .51 (2) | .68 (6) | .80 (3) | .83 (3) | .57 (3) | .88 (2) | .16 (9) | 4.0 |
| LLE | .03 (9) | .21 (9) | .09 (9) | .65 (9) | .54 (9) | .81 (7) | .54 (7) | .85 (8) | **.22** (1) | 7.6 |
| AAEME | .09 (8) | .22 (8) | .28 (7) | **.72** (1) | .74 (7) | .81 (7) | .50 (9) | .88 (3) | .20 (2) | 5.8 |
| CAEME | .10 (3) | **.37** (1) | .47 (5) | .70 (3) | .83 (2) | .80 (9) | .56 (4) | .74 (9) | .19 (4) | 4.4 |
| DAEME | .10 (6) | .36 (2) | .47 (6) | .67 (7) | .75 (6) | .82 (5) | .50 (8) | .85 (7) | .18 (7) | 6.0 |
| SimME | **.14** (1) | .34 (3) | **.57** (1) | .71 (2) | **.88** (1) | **.85** (1) | .58 (2) | **.89** (1) | .20 (3) | **1.7** |
| **BIO-TWITTER** | | | | | | | | | | |
| Biomed | .12 (2) | .24 (3) | .45 (5) | .70 (4) | .83 (4) | .81 (5) | .56 (4) | .86 (3) | .19 (5) | 3.9 |
| Twitter | .05 (8) | .15 (8) | .31 (8) | .62 (9) | .61 (8) | .83 (3) | .56 (3) | .87 (2) | .19 (4) | 5.9 |
| Average | .04 (9) | .09 (9) | .37 (7) | .64 (7) | .68 (7) | .80 (7) | .55 (5) | .84 (4) | .19 (6) | 6.8 |
| Concat | .07 (6) | .20 (5) | .40 (6) | .65 (6) | .69 (6) | .83 (2) | .57 (2) | .75 (7) | .18 (8) | 5.3 |
| LLE | .07 (7) | .15 (6) | .04 (9) | .63 (8) | .51 (9) | .79 (9) | .50 (7) | .83 (5) | **.21** (1) | 6.8 |
| AAEME | .07 (5) | .15 (7) | .51 (3) | **.72** (1) | .83 (3) | .80 (8) | .50 (8) | .72 (8) | .21 (2) | 5.0 |
| CAEME | .09 (3) | .25 (2) | .55 (2) | .70 (3) | .83 (2) | .82 (4) | .50 (9) | .60 (9) | .18 (9) | 4.8 |
| DAEME | .09 (4) | .23 (4) | .49 (4) | .69 (5) | .77 (5) | .80 (6) | .52 (6) | .77 (6) | .18 (7) | 5.2 |
| SimME | **.14** (1) | **.27** (1) | **.56** (1) | .70 (2) | **.85** (1) | **.84** (1) | **.59** (1) | **.89** (1) | .21 (2) | **1.2** |
| **WIKI-TWITTER** | | | | | | | | | | |
| Wiki | .10 (2) | **.33** (1) | .56 (2) | .66 (3) | .86 (2) | .83 (4) | .59 (4) | .87 (6) | .17 (8) | 3.6 |
| Twitter | .05 (8) | .15 (9) | .04 (8) | .62 (9) | .49 (9) | .83 (5) | .56 (5) | .87 (5) | .19 (4) | 6.9 |
| Average | .09 (5) | .30 (4) | .41 (7) | .65 (5) | .75 (7) | .83 (3) | **.60** (1) | .85 (7) | .16 (9) | 5.3 |
| Concat | .09 (4) | .29 (5) | .41 (6) | .65 (6) | .75 (6) | .84 (2) | **.60** (1) | .85 (7) | .17 (7) | 4.9 |
| LLE | .03 (9) | .20 (8) | -.02 (9) | .63 (7) | .50 (8) | .80 (8) | .43 (9) | .83 (9) | .19 (5) | 8.0 |
| AAEME | .08 (7) | .30 (3) | .54 (3) | **.68** (1) | .84 (3) | .79 (9) | .50 (7) | **.91** (1) | .19 (3) | 4.1 |
| CAEME | .09 (3) | .28 (6) | .52 (4) | .66 (4) | .82 (4) | .82 (6) | .44 (8) | .89 (4) | **.20** (1) | 4.4 |
| DAEME | .09 (6) | .28 (7) | .50 (5) | .63 (8) | .81 (5) | .80 (7) | .56 (6) | **.91** (1) | .17 (6) | 5.7 |
| SimME | **.15** (1) | .31 (2) | **.59** (1) | .66 (2) | **.91** (1) | **.84** (1) | **.60** (1) | **.91** (1) | **.20** (1) | **1.2** |
| **WIKI-BIO-TWITTER** | | | | | | | | | | |
| Wiki | .10 (5) | .33 (4) | .56 (2) | .66 (8) | .86 (2) | .83 (2) | .59 (3) | .87 (7) | .17 (10) | 4.8 |
| Biomed | .12 (2) | .24 (8) | .34 (8) | .70 (4) | .74 (8) | .81 (7) | .56 (6) | .86 (8) | .19 (2) | 5.9 |
| Twitter | .05 (9) | .15 (10) | .06 (9) | .62 (10) | .49 (10) | .83 (3) | .56 (5) | .87 (6) | **.19** (1) | 7.0 |
| Average | .09 (7) | .30 (7) | .42 (7) | .67 (6) | .76 (7) | .82 (5) | .59 (4) | .88 (5) | .18 (9) | 6.3 |
| Concat | .09 (6) | .30 (6) | .42 (6) | .67 (7) | .76 (6) | .82 (4) | .61 (2) | .83 (9) | .19 (4) | 5.6 |
| LLE | .04 (10) | .19 (9) | -.01 (10) | .64 (9) | .50 (9) | .80 (9) | .54 (7) | .81 (10) | .19 (3) | 8.4 |
| AAEME | .11 (4) | **.37** (1) | .51 (3) | **.72** (1) | .82 (3) | .81 (8) | .51 (8) | .91 (4) | .18 (5) | 4.1 |
| CAEME | .11 (3) | .35 (2) | .46 (4) | .71 (3) | .78 (4) | .82 (6) | .50 (9) | .93 (3) | .18 (5) | 4.3 |
| DAEME | .09 (7) | .31 (4) | .44 (4) | .69 (4) | .77 (4) | .79 (9) | .44 (9) | **.94** (1) | .18 (5) | 5.2 |
| SimME | **.18** (1) | .35 (3) | **.61** (1) | .71 (2) | **.92** (1) | **.85** (1) | **.64** (1) | **.94** (1) | .18 (5) | **1.8** |

Table 1: Compared results on 4 text mining tasks: (1) Semantics prediction (metric: correlation). (2) Synonym detection (metric: AUC). (3) Text classification (metric: Accurary). (4) Concept categorization (metric: NMI). The number in parenthesise shows ranking performance (rank 1 indicates the best performance against the other methods). Average rank shows the overview compared performance across all experiments. **Bold**: best scores.

ceeds in generating meta-embeddings to support the most common task in text mining. All in all, these results indicate that SimME successfully amalgamates complementary knowledge about word relations among multiple pre-trained embedding models into one powerful meta-embedding space.

## 2.7 Conclusions

In this work, we propose SimME, a novel learning method for meta-embeddings, to preserve the most meaningful information of word relations encoded differently across multiple pre-trained embedding models. Our proposed method overcomes the pervasive challenge of out-of-vocabulary (OOV) words by adopting *maskout* optimization that steers the model to learn adaptively from the only pre-trained models that truly observe information regarding OOV words. The meta-embeddings trained by our approach offer an integrated source of high quality word representations that combine knowledge from multiple sources which could support the broaden use in text mining tasks.

# 3 Amalgamating Discriminative Knowledge

This task is published at AAAI 2021 [101].

## 3.1 Motivation

Many advances on multi-class classification have been propelled by conventional supervised learning approaches which require a significant large amount of labeled data to train for a robust model. Unfortunately, data labeling in practice is very expensive and proprietary in many cases such as health records or customer activities. Often times, only few labels are obtainable that seriously hinder training a robust model from scratch.

Fortunately, several research groups owning access to huge confidential data train a model on their private data and release such pre-train model offering reuse for other users [39, 40, 45]. These pre-trained models are obviously trained on different private datasets and mostly solve different specific tasks, leading each to obtain discriminative knowledge on a unique class set. Thus, reusing these pre-trained models are practically limited as a

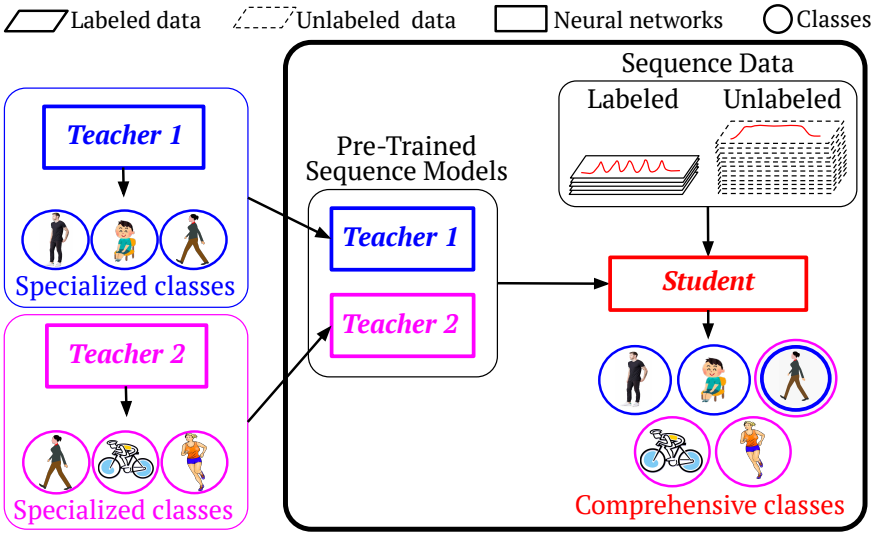



Figure 4: Amalgamating multiple pre-trained multi-class classifiers (*teachers*), that specialize on different sets of classes, into a *student* model to comprehend on the union of classes covered in all teachers. Assuming only few labeled data are available along with some large unlabeled data.

user's task may be different, even slightly, than the task solved by a pre-trained model.

Recently, knowledge amalgamation proposes to combine different discriminative expertise from multiple pre-trained models (*teachers*) into a student model. The goal of this student model is to comprehend on the union of their specialized classes that would support reusing on a broaden task. However, recent works [92, 119, 60, 108] study mainly on the unsupervised setting that suffer heavily when an overconfident teacher provides inaccurate prediction with high confidence for instances belonging to classes which it had never actually experienced on. For example as shown in Figure 4, due to similarities between *cycling* and *sitting*, Teacher 1 may confidently predict *sitting* for a person that in fact *cycling* since the teacher has no experience on detecting *cycling*. Moreover, these works focus exclusively on images with no works to-date explore this problem for sequences.

In this work, we explore this problem to sequence classification and broaden its setting to more practical cases when few labeled data are available. We define the new setting as the problem of semi-supervised knowledge amalgamation (SKA) for sequence classification, as depicted in Figure 4.

### 3.2 Related Works

There are several knowledge amalgamation (KA) works aiming to combine multiple teachers that specialize in different class sets. [92] have a strong assumption on the architecture of all teachers and students to be identical. This work proposes the student model to imitate the compressed features from multiple teachers layer by layer. At each layer, they first concatenates the features from all teachers then applies autoencoder technique to extract their compact feature set which then used to supervise the corresponding feature layer in the student model.

Most recent works of [60] and [108] relax the strong assumption in the previous work that allow each teacher to have different architectures. [60] propose to train a student to imitate both from the teachers' logits and their final feature layers. On the other hand,

[108] split a student's classes into multiple subsets to be trained in corresponding to each teacher's specialty.

All of these methods study mainly on the unsupervised setting. Thus, none of them utilize any labeled data into the training even few some annotations may available. Moreover, they have been studied exclusively on images which no works explore yet on sequences.

### 3.3 Problem Definition

In this work, we are first to study the open *semi-supervised knowledge amalgamation* problem for sequence classification.

Let $X$ be an input sequence of length $m$ where $x_t$ denotes its value at timestep $t$. Given a training dataset $\mathcal{D}$, it consists of two subset: (1) a tiny subset of labeled data $\mathcal{D}_l$ which each instance has an associated $y_j \in \mathcal{Y}$ where $\mathcal{Y} = \{y_j\}_{j=1}^{c}$, and (2) a subset of unlabeled data $\mathcal{D}_u$, proportionately large with respect to $\mathcal{D}_l$. Thus, we have $\mathcal{D} = \mathcal{D}_l \cup \mathcal{D}_u$ where $\mathcal{D}_l = \{(X^l, y_j^l)\}_{l=1}^{n_l}$ and $\mathcal{D}_u = \{X^u\}_{u=1}^{n_u}$. All $n$ training instances are denoted by $\mathcal{X} = \{X^l\}_{l=1}^{n_l} \cup \{X^u\}_{u=1}^{n_u}$ which $n = n_l + n_u$. They can be depicted as:

$$\mathcal{X} = \underbrace{\left.\begin{pmatrix} x_1^1 & x_2^1 & \cdots & x_m^1 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^n & x_2^n & \cdots & x_m^n \end{pmatrix}\right\}}_{\text{Sequence of length } m} n \text{ instances}$$

Assume that we are given a set of $p$ pre-trained sequence classifiers (*teachers*), we denote them as $\mathcal{T} = \{T_k\}_{k=1}^{p}$. We note that each $T_k$ specializes in classifying a set of classes $\mathcal{Y}_k$ which is a subset of $\mathcal{Y}$, *i.e.* $\mathcal{Y} = \bigcup_{k=1}^{p} \mathcal{Y}_k$.

Our goal is to train a *student* model that, for an instance $X$, the model can predict accurately its associated class $y_j$ from the $c$ classes in $\mathcal{Y}$.

### 3.4 Challenges

There are several open challenges related to SKA problem. Here we summarize the three major challenges:

- *Very limited supervision*: While modern multi-class classifiers [30, 62] require access to a large amount of labeled training data, this problem is impeded significantly from having only few available labeled data. This could cause high risk of overfitting raising limitation on model generalization.

- *Disparate teachers*: An individual teacher is typically trained on different private training data to tackle a specific class set. Thus, its prediction is meaningful regarding only the classes existing in such specialized set which cannot infer any informative knowledge to the other classes out of scope that may be specialized by the other teachers. This means there is no relation between the predictions provided across multiple teachers creating a challenge in amalgamating such heterogeneous information from these *disparate* teachers.

- *Overconfident teachers*: The *scales* of raw predicted scores produced by each teacher may differ drastically since it is trained from completely different sets of classes. This situation exists prevalently when a teacher deals with the sheer *number* of classes as it has to produce clearly discriminative scores among lots of classes. However, such teacher may become a troublesome *overconfident* teacher if it in fact has no knowledge about the true class which could transfer misleading knowledge to a student. Therefore, this untrustworthy information should be effectively handled by the amalgamating process.

### 3.5 Proposed Method: Teacher Coordinator (TC)

We propose the **T**eacher **C**oordinator (TC), as shown in Figure 5, to address these challenges in the SKA problem. The main idea of TC to achieve estimating the probability over all target classes, combined from disparate teachers, is by rescaling predicted outputs of each teacher based on how trustworthy it is. For the sake of readability, we describe our method in terms of one instance.

The proposed TC is the two step training paradigm. We first train the *Teacher Trust*

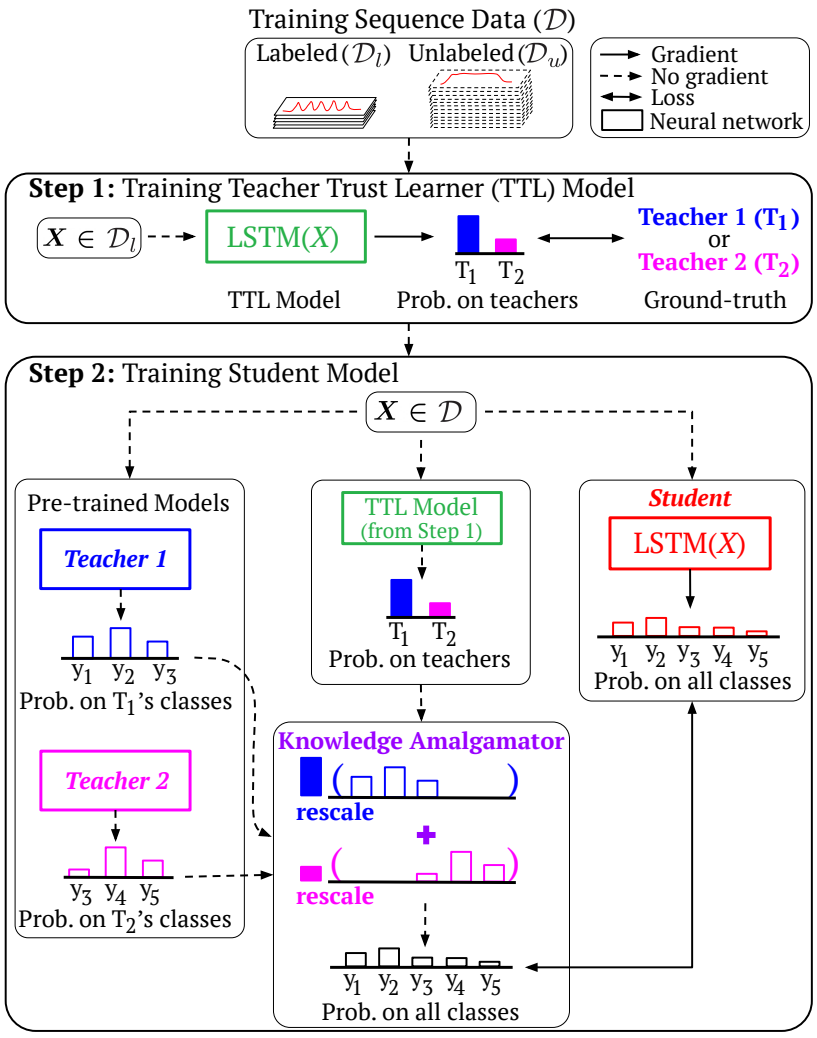



Figure 5: The proposed Teacher Coordinator.

*Learner* to estimate trustworthy teachers. It is then passed through the *Knowledge Amalgamator* that is used to normalize and fuse the predictions from each teacher to form the complete distribution over all classes which is served as a surrogate target for the student training.

### 3.5.1 Teacher Trust Learner

First, we train the *Teacher Trust Learner (TTL)* to estimate, for a given instance, which teacher to trust. Since multiple teachers could specialize on an instance's class simultaneously, the TTL naturally deals with a multi-label problem. That is, given an instance, TTL predicts one probability per teacher to estimate the likelihood such teacher is an expert on the class that this instance belongs to. We use the tiny available labeled dataset $\mathcal{D}_l$ to train TTL which is modeled as a Recurrent Neural Network with Long Short-Term Memory cells (LSTM) [43]. Given a sequence $X$, we denote the whole process of the LSTM network as one function:

$$h_a = \text{LSTM}_{\theta_a}(X) \tag{6}$$

where $h_a$ is the final hidden state and $\theta_a$ are all learnable parameters regarding the LSTM network. To estimate how likely each teacher is an expert on $X$'s true label $y_j$, we apply the sigmoid function on Equations 6 to map each vector element into the range [0, 1].

$$e = \sigma(W_a \cdot h_a + b_a) \tag{7}$$

where $W_a$, $b_a$ are learnable parameters and $\sigma$ is the sigmoid function. Here, each element of $e$ shows the probability of the corresponding teacher to be associated to the given instance. The vector $e$ is then normalized through a softmax function to extract relative scores distributed over all teachers:

$$P(y_j \in \mathcal{Y}_k|X) = \text{softmax}(e). \tag{8}$$

This probability distribution in then used as our target in the following training objective:

$$J(\theta_{TTL}) = -\sum_{k=1}^{p} \left( \mathbb{1}_{T_k} \log(e_k) + (1 - \mathbb{1}_{T_k}) \log(e_k) \right) \tag{9}$$

where $\mathbb{1}_{T_k}$ is 1 if $y_j \in \mathcal{Y}_k$ and 0 otherwise. $\theta_{TTL}$ refers to all parameters that are updated through the training including $\theta_a$, $W_a$, and $b_a$. Once the training is done, this TTL is forwarded to be used statically in Step 2.

### 3.5.2 Knowledge Amalgamator

At the second step, *Knowledge Amalgamator* uses the TTL's outputs to rescale the predicted information of disparate teachers to form the final distribution over all classes in $\mathcal{Y}$. The goal here is to estimate $P(y_j|X)$, where $y_j \in \mathcal{Y}$ for a given instance $X$ which could be achieved using the relationship between two conditional probabilities:

(1) The probability of trustworthy teachers estimated by the TTL, $P(y_j \in \mathcal{Y}_k|X)$, and

(2) The probability of $X$'s class being $y_j$ given a teacher model. More clearly, since each teacher, $T_k$, specializes on a corresponding set of classes $\mathcal{Y}_k$, the output of each teacher is:

$$P(y_j|y_j \in \mathcal{Y}_k, X). \tag{10}$$

These two conditional probabilities are used to estimate the $P(y_j|X)$ as follows:

$$\begin{aligned} P(y_j|y_j \in \mathcal{Y}_k, X) * P(y_j \in \mathcal{Y}_k|X) & \\ &= \frac{P(y_j, y_j \in \mathcal{Y}_k, X)}{P(y_j \in \mathcal{Y}_k, X)} * \frac{P(y_j \in \mathcal{Y}_k, X)}{P(X)} \\ &= \frac{P(y_j, y_j \in \mathcal{Y}_k, X)}{P(X)} \\ &= P(y_j, y_j \in \mathcal{Y}_k|X) \\ &= P(y_j|X) \end{aligned} \tag{11}$$

Finally, we model the **Student Network** by another LSTM network to generate $Q(y_j|X)$, the probability of $X$'s label being $y_j \in \mathcal{Y}$ which is trained to imitate $P(y_j|X)$, the output from the Knowledge Amalgamator. We again denote the whole process of the LSTM

network as one function:

$$h_S = \text{LSTM}_{\theta_S}(X) \tag{12}$$

where $h_S$ is the final features and $\theta_S$ are all learnable parameters in this LSTM network. These features are finally passed through the softmax function to produce the probability distribution over all classes as:

$$Q(y_j|X) = \frac{\exp(W_S \cdot h_S + b_S)}{\sum_j \exp(W_S \cdot h_S + b_S)} \tag{13}$$

where $y_j \in \mathcal{Y}$, $W_S$ and $b_S$ are trainable parameters.

The student network uses $P(y_j|X)$ as a surrogate target and is trained to iteratively update $\theta_S$, $W_S$ and $b_S$, grouped as $\theta_{SN}$, by optimizing the binary cross entropy:

$$J(\theta_{SN}) = -\sum_{j=1}^{c} P(y_j|X)\log(Q(y_j|X)). \tag{14}$$

### 3.6 Evaluation

We evaluate the proposed TC on four datasets, which are compared against the other eight methods.

**Datasets.** We conduct our experiments on four time series datasets including SyntheticControl (SYN) [3], MelbournePedestrian (PED) [14], Human Activity Recognition Using Smartphones (HAR) [6], and ElectricDevices (ELEC) [56]. The number of instances, timesteps, and classes for each dataset are shown in Table 2. Note that, for each dataset ($\mathcal{D}$), we assume there is a tiny subset of labeled data ($\mathcal{D}_l$) and a large subset of unlabeled data ($\mathcal{D}_u$).

| Dataset | Instances | Timesteps | Classes |
|---|---|---|---|
| SYN | 600 | 60 | 6 |
| PED | 3,633 | 24 | 10 |
| HAR | 10,299 | 561 | 6 |
| ELEC | 16,637 | 96 | 7 |

Table 2: Details of 4 sequential datasets used for evaluation.

**Compared Methods.** As the SKA problem is newly defined in this work, there is no state-of-the-art approaches that are directly designed for this problem. However, there are eight alternative methods that can apply to this problem which could be divided into three categories.

**Baselines:** Assume no amalgamating learning, there methods have no benefit from complementary knowledge among teachers.

- *Original Teachers*. Individual teacher is used independently which limits its prediction only on its specialized class set.
- *SupLSTM*. This is a standard LSTM, trained from scratch, supervised on the few available labeled data.
- *SelfTrain* [86]. This method iteratively learns pseudo-labels for unlabeled data then such predictions that pass a confident threshold are added into the training set.

**Unsupervised KA methods:** These methods combine knowledge from all teachers in $\mathcal{T}$ and train on all data in $\mathcal{D}$ without incorporating any labels.

- *KD* [42]. Knowledge distillation trains a student to imitate the average of teachers' logits or their concatenation if class sets are disjoint.
- *CFL* [60]. This trains a student to imitate not only the teachers' logits but also their final hidden features which are mapped to be trained via a common space.
- *UHC* [108]. This method splits the student's output into subsets that each of which is trained to imitate the prediction of the corresponding teacher.

**Supervised KA methods:** These methods combine knowledge from all teachers in $\mathcal{T}$ and utilize also the available labels in $\mathcal{D}_l$.

- *SupKD*. It trains a student on a few amount of labeled data to imitate the teachers' logits while to make correct prediction on such labels.

| | SYN | | | | PED | | | |
|---|---|---|---|---|---|---|---|---|
| | Ratio of labeled data | | | | Ratio of labeled data | | | |
| Methods | 2% | 4% | 6% | 8% | 2% | 4% | 6% | 8% |
| Teacher 1 | .66±.00 | .66±.00 | .66±.00 | .66±.00 | .55±.00 | .55±.00 | .55±.00 | .55±.00 |
| Teacher 2 | .64±.00 | .64±.00 | .64±.00 | .64±.00 | .49±.00 | .49±.00 | .49±.00 | .49±.00 |
| SupLSTM | .51±.11 | .58±.10 | .66±.12 | .71±.06 | .27±.05 | .49±.03 | .52±.06 | .57±.06 |
| SelfTrain | .58±.06 | .65±.04 | .75±.04 | .79±.05 | .48±.03 | .65±.02 | .65±.02 | .68±.06 |
| KD | .87±.01 | .87±.01 | .87±.01 | .87±.01 | .61±.03 | .61±.03 | .61±.03 | .61±.03 |
| CFL | .63±.01 | .63±.01 | .63±.01 | .63±.01 | .48±.02 | .48±.02 | .48±.02 | .48±.02 |
| UHC | .86±.01 | .86±.01 | .86±.01 | .86±.01 | .60±.03 | .60±.03 | .60±.03 | .60±.03 |
| SupKD | .55±.04 | .61±.14 | .67±.08 | .69±.01 | .51±.07 | .61±.02 | .64±.05 | .67±.01 |
| SupUHC | .47±.07 | .58±.12 | .66±.13 | .70±.04 | .46±.04 | .58±.02 | .65±.03 | .64±.01 |
| TC (Ours) | **.90**±.02 | **.91**±.04 | **.92**±.02 | **.93**±.01 | **.69**±.01 | **.70**±.02 | **.75**±.04 | **.76**±.03 |

| | HAR | | | | ELEC | | | |
|---|---|---|---|---|---|---|---|---|
| | Ratio of labeled data | | | | Ratio of labeled data | | | |
| Methods | 2% | 4% | 6% | 8% | 2% | 4% | 6% | 8% |
| Teacher 1 | .51±.00 | .51±.00 | .51±.00 | .51±.00 | .47±.00 | .47±.00 | .47±.00 | .47±.00 |
| Teacher 2 | .58±.00 | .58±.00 | .58±.00 | .58±.00 | .41±.00 | .41±.00 | .41±.00 | .41±.00 |
| SupLSTM | .42±.07 | .50±.12 | .61±.03 | .68±.07 | .52±.06 | .62±.05 | .69±.04 | .69±.01 |
| SelfTrain | .46±.06 | .54±.06 | .64±.05 | .65±.10 | .56±.03 | .62±.01 | .70±.03 | .69±.02 |
| KD | .60±.01 | .60±.01 | .60±.01 | .60±.01 | .65±.01 | .65±.01 | .65±.01 | .65±.01 |
| CFL | .30±.00 | .30±.00 | .30±.00 | .30±.00 | .58±.02 | .58±.02 | .58±.02 | .58±.02 |
| UHC | .66±.10 | .66±.10 | .66±.10 | .66±.10 | .62±.01 | .62±.01 | .62±.01 | .62±.01 |
| SupKD | .48±.09 | .58±.06 | .64±.04 | .64±.03 | .45±.04 | .64±.03 | .69±.04 | .68±.01 |
| SupUHC | .41±.06 | .46±.05 | .49±.05 | .65±.12 | .52±.05 | .61±.04 | .62±.05 | .63±.02 |
| TC (Ours) | **.75**±.01 | **.77**±.02 | **.78**±.01 | **.78**±.02 | **.66**±.01 | **.68**±.03 | **.71**±.02 | **.71**±.02 |

Table 3: Compared performance (Accuracy±SD) on varied tiny rates of available labeled data in the training data.

- *SupUHC*. This method extend UHC by adding a supervised objective into the loss function of the original UHC method to encourage correct prediction informed by the small set of labeled data.

**Experiments and Results.** We perform experiments to investigate three main properties regarding the SKA problem.

First, we observe how effective each method can utilize few annotations to solve SKA by setting available label proportions varying from 2%-8% of the training data. Table

| | Overlapping class sets | | | Exclusive class sets | | |
|---|---|---|---|---|---|---|
| Methods | SYN | PED | HAR | SYN | PED | HAR |
| Teacher 1 | .66±.00 | .55±.00 | .51±.00 | .50±.00 | .43±.00 | .38±.00 |
| Teacher 2 | .64±.00 | .49±.00 | .58±.00 | .47±.00 | .44±.00 | .45±.00 |
| SupLSTM | .51±.11 | .27±.05 | .42±.07 | .51±.11 | .27±.05 | .42±.07 |
| SelfTrain | .58±.06 | .48±.03 | .46±.06 | .58±.06 | .48±.03 | .46±.06 |
| KD | .87±.01 | .61±.03 | .60±.01 | .54±.04 | .44±.03 | .55±.02 |
| CFL | .63±.01 | .48±.02 | .30±.00 | .50±.03 | .30±.05 | .50±.08 |
| UHC | .86±.01 | .60±.03 | .66±.10 | .61±.02 | .43±.02 | .55±.01 |
| SupKD | .55±.04 | .51±.07 | .48±.09 | .48±.12 | .48±.02 | .50±.02 |
| SupUHC | .47±.07 | .46±.04 | .41±.06 | .53±.02 | .43±.09 | .22±.11 |
| TC (Ours) | **.90**±.02 | **.69**±.01 | **.75**±.01 | **.77**±.07 | **.64**±.04 | **.84**±.04 |

Table 4: Accuracy±SD when combining disparate teachers. Left: teachers are partially related, sharing 2 classes. Right: teachers are disjoint, sharing no classes.

| Methods | SYN | PED | HAR | ELEC |
|---|---|---|---|---|
| Teacher 1 | .33±.00 | .30±.00 | .28±.00 | .32±.00 |
| Teacher 2 | .64±.00 | .64±.00 | .58±.00 | .44±.00 |
| SupLSTM | .51±.11 | .27±.05 | .42±.07 | .52±.06 |
| SelfTrain | .58±.06 | .48±.03 | .46±.06 | .56±.03 |
| KD | .66±.01 | .66±.02 | .57±.02 | .37±.00 |
| CFL | .64±.01 | .63±.01 | .53±.04 | .36±.01 |
| UHC | .66±.03 | .64±.02 | .49±.10 | .36±.03 |
| SupKD | .49±.03 | .55±.04 | .31±.13 | .34±.06 |
| SupUHC | .51±.05 | .46±.06 | .35±.07 | .39±.05 |
| TC (Ours) | **.85**±.07 | **.70**±.08 | **.75**±.01 | **.64**±.01 |

Table 5: Learning from overconfident teachers. Teacher 2 has roughly twice as many classes as Teacher 1 for all tasks.

3 shows all results on these experiments. As expected, we found that when very small amount of labels available (at 2% or 4%) the unsupervised KA methods outperform baselines and the supervised KA. This shows that the teachers' knowledge is needed when too few annotations are available. However, once label ratios go up to 6% and 8%, the results do begin to switch which indicates the power of labels no matter how few are available which need to be carefully incorporate into the SKA problem. In all cases, we found that the proposed TC outperforms across the board significantly. It drives an accuracy up by 6% on average over the second best method. This clearly shows that TC successfully

incorporates both benefits from teachers' knowledge and the available annotations even very tiny amount are available.

Second, we investigate the performance of TC when it is challenged to learn from disparate teachers. We assume that teachers are more disparate when they have less overlap classes meaning that they have less information in common. Therefore, we set this experiment using two scenarios: (1) *Overlapping class sets* refer to cases when teachers has some shared classes. In this setting, we set both teachers to share exactly two classes. (2) *Exclusive class sets* refer to cases when the teachers have completely disjoint class sets. Our results in Table 4 shows that the student models in all other methods suffer heavily from fusing different knowledge from disparate teachers especially when they have completely disjoint. However, TC combine their heterogeneous sources of knowledge successfully by rescaling the output of each disparate teacher to learn for their joint probability. TC can improve accuracy by 14% on average.

Lastly, our proposed method is tested to overcome the impact of overconfident teachers. We observe that a teacher with more classes usually generates a large range of its logits and such large logits can produce overconfident predictions which may dominate the predictions of other teachers. Thus, in this experiment, we set the number of specialized classes in Teacher 2 to be roughly twice that of Teacher 1. As a result as shown in Table 5, we notice that since the Teacher 2 provides its predictions with high confidence, the student models in other KA methods are dominated by Teacher 2 and cannot improve the performance over such the overconfident teacher. On the other hand, our TC is able to extract the accurate predictions from the trustworthy teachers that can boost the accuracy up to 13% higher on average.

### 3.7 Conclusions

In this work, we introduce the challenging problem of semi-supervised knowledge amalgamation (SKA) for sequence classification. This new setting broadens the study of knowl-

edge amalgamation to more realistic case when only few annotations are available. We propose the novel solution, named the Teacher Coordinator (TC), to combine knowledge from multiple pre-trained teachers based on the teachers' trustworthiness. The key success of TC is by effectively rescaling the predicted output of disparate teachers. Our intensive experiments on four real datasets show that TC performs successfully even when the labeled data are available as little as 2% of the training data. It outperforms other alternative methods across a wide variety of tasks that could boost the accuracy over the second best method by 15% on average.

# 4 Amalgamating Label Dependencies

This task is published at AAAI 2023 [100].

## 4.1 Motivation

Recently, multi-label classification has received increased attention because a wide variety of real world applications [123, 55, 90, 31, 107] are naturally multi-label, as often multiple classes or *labels* can apply to a single instance simultaneously. The key success of modern multi-label classifiers is driven by its ability to capture dependencies among labels. However, in order to learn these dependencies these models need to observe many if not all possible label combinations. This means that training such models requires massive

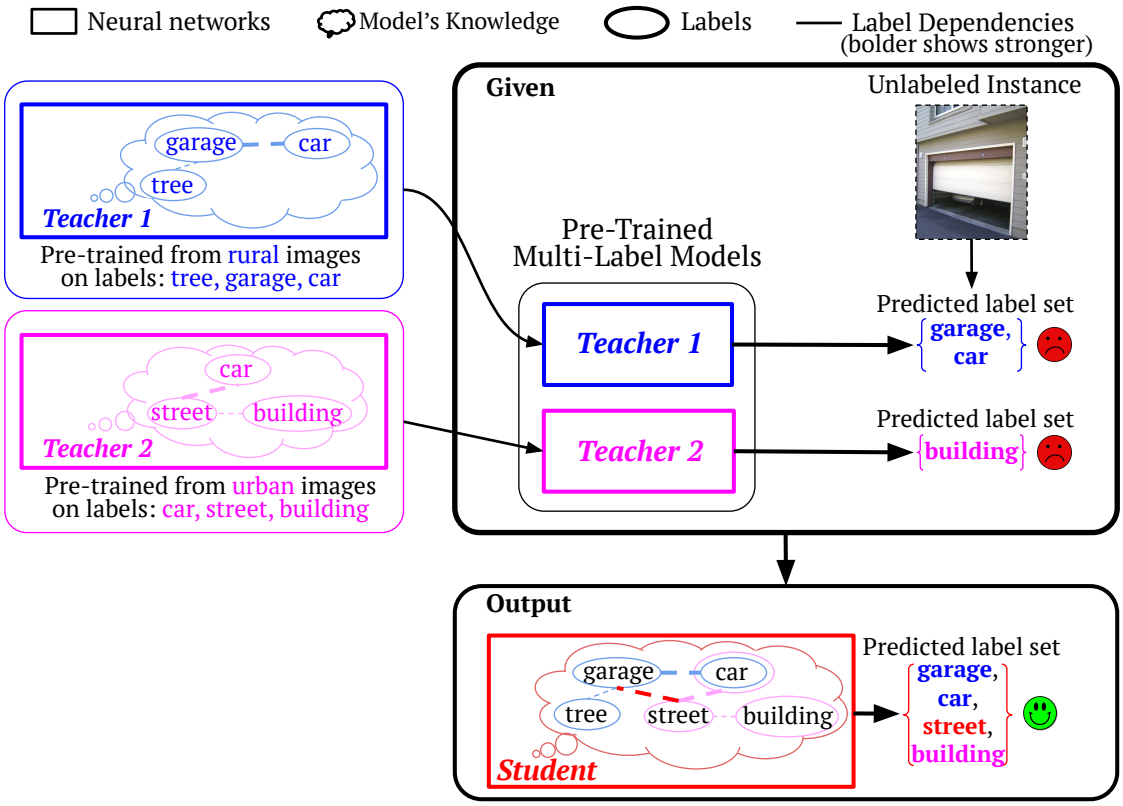



Figure 6: Given a set of pre-trained multi-label models (*teachers*) and unlabeled data, our goal is to train a *student* model to become an expert in capturing all dependencies among the union of teachers' labels.

labeled datasets. Acquiring such data is practically infeasible when the number of labels increases as their potential combinations grow exponentially.

Some pre-trained multi-label classifiers, trained by various groups [114, 48, 16], are thus made publicly available to support users that cannot obtain or work with the massive datasets required by these models. However, each of these released models are trained on different label sets and consequently capture different knowledge on label dependencies. Reusing each model is thus restricted to their original scope. Figure 6 shows an example of this; in the figure, reusing Teacher 1 is limited to a task related to the label set of {*ocean, bird, sky*} while reusing Teacher 2 is restricted to another label set of {*ocean, bird, tree*}.

To broaden the reuse potential of these pre-trained multi-label classifiers (teachers), as depicted in Figure 6, knowledge amalgamation proposes to train a student model to be knowledgeable on all labels in the union of the multiple teacher model's knowledge base. Specifically, we aim to do this using only the pre-trained teachers and unlabeled data, such that the student model can effectively infer all label dependencies from multiple subsets of label dependencies captured by each teacher.

### 4.2 Related works

**Multi-Label Classification (MLC).** MLC is a classification setting where multiple labels can correspond to the same instance simultaneously. Traditional approaches transform this problem into multiple binary classification tasks, one for each label [105, 37]. These methods fail to achieve the key task of MLC in exploiting dependencies between labels.

The best-known method for capturing label dependencies, *Classifier Chain* (CC) [23, 22, 85], has a long track record of successful use for challenging MLC tasks. CCs predict labels sequentially, conditioning each label prediction on those previously predicted. Classic CCs require a predefined order of labels for their training, which is rarely available in practice. Several recent works thus propose CCs that can be trained without a predefined label order, making them *order free* [71, 17, 70, 104]. These methods typically

use Recurrent Neural Networks (RNNs) to predict labels one by one while modeling the transition between the predicted labels. This is achieved by feeding predicted labels back into the network at each step. Order-free CCs are currently the state-of-the-art solution to MLC—they achieve strong performance in many impactful applications [19, 38, 121].

**Knowledge Amalgamation (KA).** KA [119] is a learning paradigm that, using only *unlabeled data*, combines the knowledge of multiple pre-trained models (*teachers*) into one *student*. The student then handles a broader task set than that of any of its teachers by covering the union of their labels. Ideally, KA could be used to combine the knowledge of *multi-label* classifiers to extend their knowledge-base without the expense of collecting more labeled data.

However, to date, no works study KA for multi-label classification yet. Most existing works [92, 60, 108, 101] study KA for the simpler *single-label* classification. The student is trained to predict one class per instance from the union of all the teachers' classes. With the context of these works, they do not consider informative dependencies between *labels*, which is essential in MLC. Some existing KA works [119, 93] study multi-task classification which can apply to the multi-label setting by treating each label as an independent task. However, by doing so, the student model will overlook the label dependencies needed to solve the MLC problem.

### 4.3 Problem Definition

We study the open problem of knowledge amalgamation for multi-label classification (KA-MLC). In this setting, we are given unlabeled data, denoted as $\mathcal{X} = \{\mathbf{x}^i\}_{i=1}^n$ where $\mathbf{x}^i \in \mathbb{R}^d$ represents an instance with $d$ features, and a set of $m$ powerful pre-trained classifier chain based models (*teachers*), $\mathcal{T} = \{\mathbf{T}^t\}_{t=1}^m$. Each teacher specializes in solving a particular multi-label task for a set of $\ell^t$ distinct labels, denoted by the label set $\mathcal{Y}^t$. Thus, the predicted outputs for each instance $\mathbf{x}^i$ from each teacher $\mathbf{T}^t$ are $\hat{\mathcal{Y}}^{t,i} = \{\hat{y}_j^{t,i}\}_{y_j \in \mathcal{Y}^t}$ where

$\hat{y}_j^{t,i} = 1$ (positive) if $\mathbf{T}^t$ predicts that the label $y_j$ associates with instance $\mathbf{x}^i$ or 0 (negative) otherwise.

Our goal is to train a *student* model that accurately classifies $\mathbf{x}^i$ to its associated labels in the union of specialized labels of all teachers, $\mathcal{Y} = \{y_j\}_{j=1}^{\ell}$ where $\ell$ is the number of distinct labels. We note that $\mathcal{Y} = \bigcup_{t=1}^{m} \mathcal{Y}^t$. The student's outputs for the given $\mathbf{x}^i$ are thus $\hat{\mathcal{Y}}^i = \{\hat{y}_j^i\}_{j=1}^{\ell}$ where $\hat{y}_j^i \in \{0, 1\}$. To improve readability, we describe the rest of the paper in terms of one instance $x^i$ and drop the superscript $i$ hereafter.

State-of-the-art multi-label classifiers tend not to merely predict the probability of each individual label conditioned only on the input $\mathbf{x}$. Instead, leading approaches also model the joint dependencies between labels [23, 22, 85], using either graph-based approaches [112, 53] or by iteratively predicting each label using information from previously predicted label [17, 70, 104]. The latter approach, referred to as *Order-Free Classifier Chains* [17], have become particularly popular in recent years. Thus, we describe our approach in terms of Order-free Classifier Chain-based teachers; however, with only slight modifications our method could equally be applied to other multi-label approaches that likewise model dependencies between labels.

### 4.4 Challenges

Amalgamating label dependencies from multiple teachers is challenging due to the three reasons as following.

- *No labeled data*. Traditional methods for MLC require access to a huge amount of training data with ground truth annotations. With only unlabeled training data, conventional supervised methods are not applicable, thus necessitating the development of a novel solution requiring no human annotations.
- *Teacher disagreement*. Each teacher may learn different knowledge as they are trained

on their own private data, respectively. In some cases, teachers may disagree about a label. For example, it may be unclear whether an instance in Figure 6 contains a *bird*. So Teacher 1 may predict positive while Teacher 2 predicts negative. To combine such contradictory predictions into one prediction, a good solution must determine which teacher should be trusted.

- *Partially overlapping label sets between teachers.* Depending on the training data, each teacher may specialize on a unique label set. This set is generally a subset of the labels to be learned by the student. Thus, each teacher may have incomplete knowledge with respect to the student's task, *e.g.*, *sky* and *sea* are not shared by the teachers in Figure 6. However, these disjoint labels may still be related. Knowledge must be leveraged across the teachers to train a comprehensive student that relates all labels to each other.

### 4.5 Proposed Method: Adaptive Knowledge Transfer (ANT)

We propose **A**daptive K**n**owledge **T**ransfer (ANT) to solve the KA-MLC problem. ANT trains a student model to integrate the label dependency knowledge of the teachers. With ANT, each teacher is encouraged to revise its predictions using knowledge transferred

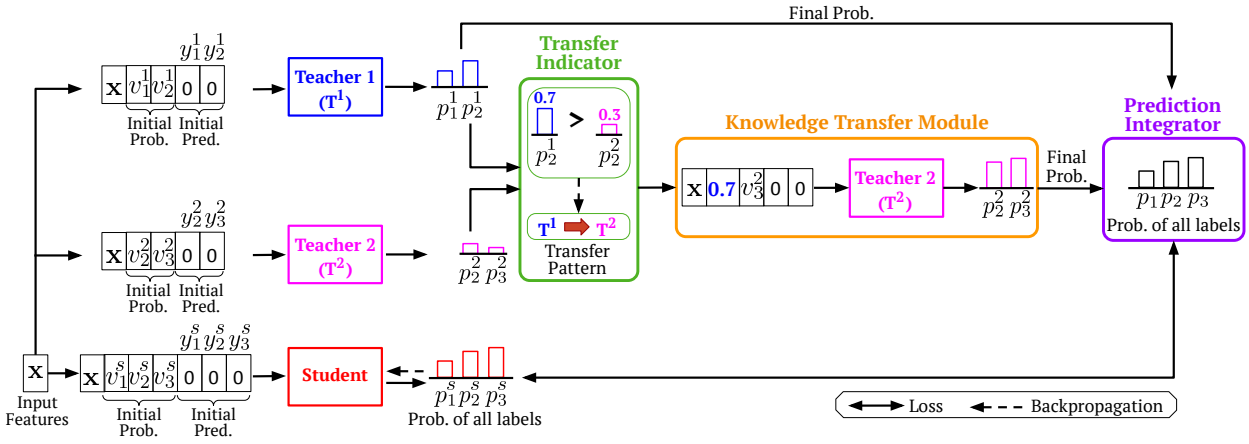



Figure 7: The proposed ANT.

from a more-competent teacher to adaptively boost the accuracy of its predictions by leveraging label dependencies.

The proposed ANT consists of three major components: (1) When teachers disagree on a class label, the *Transfer Indicator* decides which teacher should be trusted to transfer its prediction to which teacher; (2) the *Knowledge Transfer Module* revises a teacher's prediction, conditioned on the prediction of the more-competent teacher indicated by the Transfer Indicator; (3) the *Prediction Integrator* combines all teachers' final predictions into one integrated prediction that is used to train the student.

#### 4.5.1 Transfer Indicator (TI)

TI first extracts the set of teacher relations $\mathcal{R}$, containing teacher-pairs and the shared label that the teachers may potentially have beneficial knowledge to be transferred between them through such shared label: $\mathcal{R} = \{\mathbf{r}_o\}_{o=1}^{r}$. Each $\mathbf{r}_o$ consists of two teachers and the label $y_c$ that they specialize in common without regard for the teachers' order. For example, as shown in Figure 7, $\mathcal{R} = \{(\mathbf{T}^1, \mathbf{T}^2, y_2)\}$ since both $\mathbf{T}^1$ and $\mathbf{T}^2$ specialize on the label $y_2$.

Considering each instance $\mathbf{x}$ and $\mathbf{r}_o = (\mathbf{T}^m, \mathbf{T}^n, y_c)$, each teacher outputs the soft predictions or logits ($\mathcal{L}$) for its specialized labels that are passed through the sigmoid function ($\sigma$) to acquire their predicted probabilities ($\mathcal{P}$) as follows:

$$\mathcal{L}^m = \mathbf{T}^m(\mathbf{x}) \text{ and } \mathcal{P}^m = \sigma(\mathcal{L}^m) \tag{15}$$

$$\mathcal{L}^n = \mathbf{T}^n(\mathbf{x}) \text{ and } \mathcal{P}^n = \sigma(\mathcal{L}^n) \tag{16}$$

where $\mathcal{P}^t = \{p_j^t\}_{y_j \in \mathcal{Y}^t}$ and $p_j^t = P(y_j|\mathbf{x})$ which is the predicted probability of $y_j$ provided by $\mathbf{T}^t$. Then the hard prediction for each label is obtained by binarizing $p_j^t$ with a threshold of 0.5—positive if over 0.5 and negative otherwise.

In some cases, the features of the input $\mathbf{x}$ alone do not contain enough information in order to yield an agreed prediction between the teachers on the common label $y_c$, *i.e.*

one predicts positive while the other one predicts negative. Fundamentally, the multi-label teachers commonly predict low probabilities for this label since multi-label models are known as a standard method for *rejecting* instances when observing a *novel* class [41]. Regarding this principle, the teacher that provides positive prediction evidently shows stronger knowledge achieved by utilizing the dependencies between $y_c$ and the other labels it specializes on. Therefore, to integrate the knowledge between teachers adaptively for any instance, TI indicates the more-competent teacher (*transferor*) to transfer its positive prediction of $y_c$ toward the other teacher (*transferee*). For example, as depicted in Figure 7, $p_2^1$ yields positive prediction while $p_2^2$ yields negative prediction. Thus, TI indicates the transfer pattern to be $\mathbf{T}^1 \rightarrow \mathbf{T}^2$. This transfer pattern is then used in the Knowledge Transfer Module to encourage the transferee teacher to revise its predictions.

#### 4.5.2 Knowledge Transfer Module

For a given $\mathbf{r}_o = (\mathbf{T}^m, \mathbf{T}^n, y_c)$, assume TI indicates that $\mathbf{T}^m$ is the *transferor* and $\mathbf{T}^n$ is the *transferee*. The transferee $\mathbf{T}^n$ revises its predictions by conditioning on the prediction of the shared label $y_c$ informed by the transferor $\mathbf{T}^m$.

We show below that the prediction of the other labels specialized by $\mathbf{T}^n$ can benefit additional information provided by $\mathbf{T}^m$.

**Analysis of Information Gain:** Let $\mathcal{Y}^m$ and $\mathcal{Y}^n$ be the specialized label sets of the transferor $\mathbf{T}^m$ and the transferee $\mathbf{T}^n$, respectively. Assume that $y_c$ is their shared label and $y_c^*$ is its ground truth while $\hat{y}_c^m$ and $\hat{y}_c^n$ denote its predictions given by $\mathbf{T}^m$ and $\mathbf{T}^n$, respectively. We use $I(X;Y)$ to represent the mutual information between any random variables $X$ and $Y$.

As we assume that transferor has made a more accurate prediction for $y_c$, the mutual information shared between $\hat{y}_c^m$ and $y_c^*$ is higher than the mutual information shared between $\hat{y}_c^n$ and $y_c^*$, which is formalized in Assumption 1 as follow:

**Assumption 1 (A1)**:

$$I(y_c^*; \hat{y}_c^m) > I(y_c^*; \hat{y}_c^n) \tag{17}$$

$$\textit{i.e.} \exists \lambda \in (0,1), \;\; \lambda I(y_c^*; \hat{y}_c^m) = I(y_c^*; \hat{y}_c^n) \tag{18}$$

Additionally, we assume that $\hat{y}_c^m$ and $\hat{y}_c^n$ are not biased with respect to $y_j^*$, the ground truth for the label $y_j$ that is another label that $\mathbf{T}^n$ specializes in. Thus, if the transferee contains $\lambda$ (less) of the information between itself and $y_c$ than the transferor does, then it likewise contains $\lambda$ (less) information between itself and the information content of $y_j$ that is independent from $y_c$. This is stated formally in Assumption 2:

**Assumption 2 (A2)**: Let $y_j^*$ be the ground truth for the label $y_j$ that is specialized particularly by $\mathbf{T}^n$.

$$\lambda I(y_c^*; \hat{y}_c^m) = I(y_c^*; \hat{y}_c^n) \implies \lambda I(y_c^*; \hat{y}_c^m | y_j^*) = I(y_c^*; \hat{y}_c^n | y_j^*) \tag{19}$$

The following theorem provides justification for updating the predictions of the transferee using the more knowledgeable predictions from the transferor:

**Theorem 1**: Let A1 and A2 hold. Then, $I(\hat{y}_c^m; y_j^*) > I(\hat{y}_c^n; y_j^*)$.

*Proof.* By applying the chain rule of mutual information:

$$I(y_c^*; \hat{y}_c^m) = I(y_c^*; \hat{y}_c^m | y_j^*) + I(y_j^*; \hat{y}_c^m) \tag{20}$$

$$I(y_c^*; \hat{y}_c^n) = I(y_c^*; \hat{y}_c^n | y_j^*) + I(y_j^*; \hat{y}_c^n) \tag{21}$$

From A1 and A2, we have

$$\lambda(I(y_c^*; \hat{y}_c^m | y_j^*) + I(y_j^*; \hat{y}_c^m)) \overset{\text{A1}}{=} I(y_c^*; \hat{y}_c^n | y_j^*) + I(y_j^*; \hat{y}_c^n) \tag{22}$$

$$\lambda I(y_c^*; \hat{y}_c^m | y_j^*) + \lambda I(y_j^*; \hat{y}_c^m) = I(y_c^*; \hat{y}_c^n | y_j^*) + I(y_j^*; \hat{y}_c^n) \tag{23}$$

$$\lambda I(y_c^*; \hat{y}_c^m | y_j^*) + \lambda I(y_j^*; \hat{y}_c^m) \overset{\text{A2}}{=} \lambda I(y_c^*; \hat{y}_c^m | y_j^*) + I(y_j^*; \hat{y}_c^n) \tag{24}$$

$$\lambda I(y_j^*; \hat{y}_c^m) = I(y_j^*; \hat{y}_c^n) \tag{25}$$

Since $\lambda \in (0,1)$, $I(\hat{y}_c^m; y_j^*) > I(\hat{y}_c^n; y_j^*)$ □

Theorem 1 states that the information between the ground truth for $y_j$, which is a label that the transferee $\mathbf{T}^n$ particularly specializes in, and the prediction of the shared label from the transferor ($\hat{y}_c^m$) is greater than it is between the ground truth for $y_j$ and the transferee's prediction for the shared label ($\hat{y}_c^n$). Thus, we should instead use $\hat{y}_c^m$ to infer $\hat{y}_j^n$.

To achieve this, $\mathbf{T}^n$ sets the initial predicted probability for $y_c$ by using its predicted probability from $\mathbf{T}^m$. Thus, the predicted logits and probabilities of specialized labels in $\mathbf{T}^n$ in Equation 16 are revised to be:

$$\mathcal{L}^{n'} = \mathbf{T}^n(\mathbf{x}) \text{ and } \mathcal{P}^{n'} = \sigma(\mathcal{L}^{n'}) \tag{26}$$

To amalgamate many teachers, this transfer process proceeds recursively if the revised predictions in Equation 26 change the hard prediction for a label to contradict further with the other teachers.

#### 4.5.3 Prediction Integrator

We finally combine the probabilities from all teachers—some of which may have been revised according to Equations 15 and 26—to compute predictions for all labels in $\mathcal{Y}$. To obtain this final probability for each label $y_j$, ANT acquires the most confident prediction from all teachers that specialize on $y_j$. Let $\mathcal{B}_j$ denote a set of teachers that specialize on $y_j$ and $\mathcal{C}_j$ denote a set of candidate probabilities of $y_j$ provided by these teachers:

$$\mathcal{C}_j = \{p_j^t\}_{\mathbf{T}^t \in \mathcal{B}_j} \tag{27}$$

For the case that all $p_j^t \in \mathcal{C}_j$ are less than or equal to 0.5 indicating that all teachers agree on predicting negative for $y_j$, the smallest probability is used as the final probability for $y_j$. Otherwise, the highest is used when all indicate positive. When the teachers' predictions contradict, we again take the highest probability based on the same principle used in the

Transfer Indicator. Thus, the final integrated probability for each label is obtained as:

$$P(y_j|\mathbf{x}) = \begin{cases} min(\mathcal{C}_j) & \text{if } \forall p_j^t \in \mathcal{C}_j, p_j^t <= 0.5 \\ max(\mathcal{C}_j) & \text{otherwise.} \end{cases} \tag{28}$$

**Training a student model.** We use an RNN with Long Short-Term Memory cells (LSTM) to feed back the previously predicted labels into the model, allowing to learn each label conditioned on the other labels. The model is fed three input components at each time step including the input features ($\mathbf{x}$), and both the soft predictions or logits ($\mathcal{L}$) and the hard predictions ($\hat{\mathcal{Y}}$) from the previous time step. At the first time step, the initial hard predictions $\hat{\mathcal{Y}}_0$ are all set as negative while the initial soft predictions ($\mathcal{L}_0$) are set by passing the features $\mathbf{x}$ through a linear layer. Therefore, the initial input vector ($\mathbf{x}_0$) is:

$$\mathcal{L}_0 = W \cdot \mathbf{x} + b \text{ and } \hat{\mathcal{Y}}_0 = [0]^\ell \tag{29}$$

$$\mathbf{x}_0 = [\mathbf{x}, \mathcal{L}_0, \hat{\mathcal{Y}}_0] \tag{30}$$

where $W$ and $b$ are learnable parameters.

For the time step $k$, $\mathcal{L}_k$ is updated using the three input components together with the previous hidden state as:

$$\mathbf{x}_k = [\mathbf{x}, \mathcal{L}_{k-1}, \hat{\mathcal{Y}}_{k-1}] \tag{31}$$

$$\mathcal{L}_k = LSTM_\theta(\mathbf{x}_k, h_{k-1}) \tag{32}$$

where $LSTM$ denotes the entire process of an LSTM model, $\theta$ denotes all parameters for such the LSTM, and $h_{k-1}$ denotes its previous hidden state.

To obtain $\hat{\mathcal{Y}}_k$, all positive labels in $\hat{\mathcal{Y}}_{k-1}$ are removed from the candidate labels for prediction at the current step. Then the probabilities for all candidate labels are computed by passing the logits in Equation 32 through the sigmoid function. The label with the highest probability is predicted to be positive at this time step. Let $z_k$ denote such the label that gets positive prediction at time step $k$. Thus, the joint probability of all labels at

the last time step is:

$$Q(\mathcal{Y}|\mathbf{x}) = P(z_1|\mathbf{x}) \cdot \prod_{j=2}^{\ell} P(z_j|\mathbf{x}, z_1, \ldots, z_{j-1}) \tag{33}$$

The final logit for each label is obtained by carrying its logit from the time step that its hard prediction is selected to be positive. We denote these final logits for all labels in $\mathcal{Y}$ as:

$$\mathcal{L}^s = \{v_1, \ldots, v_\ell\} \tag{34}$$

This $\mathcal{L}^s$ is passed through the sigmoid function ($\sigma$) to obtain the predicted probabilities for each label as:

$$\mathcal{P}^s = \sigma(\mathcal{L}^s) \tag{35}$$

$$\text{where } \mathcal{P}^s = \{p_j^s\}_{j=1}^{\ell} \text{ and } p_j^s = Q(y_j|\mathbf{x}) \tag{36}$$

We note that $Q(y_j|\mathbf{x})$ denotes the predicted probability of the label $y_j$ learned by the student model. The hard prediction for each label ($\hat{y}_j^s$) is obtained by binarizing $p_j^s$ with a threshold of 0.5.

Finally, the student is trained to update $\theta$ iteratively by minimizing binary cross entropy between the predicted probability $Q(y_j|\mathbf{x})$ in Equation 36 and the integrated probability $P(y_j|\mathbf{x})$ in Equation 28 as follows:

$$\begin{aligned} J(\theta) = -\sum_{j=1}^{\ell} \Big( & P(y_j|\mathbf{x}) \log(Q(y_j|\mathbf{x})) \\ & + (1 - P(y_j|\mathbf{x})) \log(1 - Q(y_j|\mathbf{x})) \Big). \end{aligned} \tag{37}$$

### 4.6 Evaluation

**Datasets.** We conduct experiments on eight well-established benchmark datasets for evaluating multi-label classifiers. These datasets are from several applications in various domains as follows.

| Dataset | Domain | #Instances | Ave. #Labels | #Unique Labels | #Unique Label Sets |
|---|---|---|---|---|---|
| EMOTIONS | Media | 593 | 1.87 | 6 | 27 |
| SCENE | Media | 2,407 | 1.07 | 6 | 15 |
| YELP | Text | 10,806 | 1.64 | 8 | 118 |
| YEAST | Biology | 2,417 | 4.24 | 14 | 198 |
| BIRD | Media | 645 | 1.01 | 19 | 133 |
| TMC | Text | 28,596 | 2.16 | 22 | 1,341 |
| GENBASE | Biology | 662 | 1.25 | 27 | 32 |
| MEDICAL | Text | 978 | 1.25 | 45 | 94 |

Table 6: Details of 8 benchmark datasets used for evaluation.

- Media: *EMOTIONS* [103], *SCENE* [11] and *BIRD* [12].
- Text: *YELP* [89], *TMC* [97] and *MEDICAL* [78].
- Biology: *YEAST* [28] and *GENBASE* [26].

The number of instances, average labels per instance, unique labels and label sets per dataset are shown in Table 6.

**Compared Methods.** We compare ANT to five state-of-the-art alternatives:

- *Baseline (BL)*: The student is trained from the combined hard predictions from all teachers. We aggregate their predictions using majority voting. However, if the votes are even, we assume the instance is positive.
- *AKA* [93]: This method first trains an individual network for each label, acting as a student, from the teacher that is most confident in predicting such label. Then it trains one final student to combine all such networks, using them as teachers. At each stage, these teachers' features and the student's feature are trained to be as most similar.
- *FKA* [119]: Knowledge between the student and teachers is exchanged by projecting each layer of the student to the corresponding layer of each teacher (teacher-layer

| | **EMO Dataset** | | | | | **SCENE Dataset** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Methods | S-Loss↓ | H-Loss↓ | F1 Micro↑ | F1 Macro↑ | Rank↓ | S-Loss↓ | H-Loss↓ | F1 Micro↑ | F1 Macro↑ | Rank↓ |
| BL | .9500±.0273 | .3726±.0359 | .4351±.0621 | .4539±.0492 | 3.8 | .6615±.0273 | .1953±.0359 | .4592±.0621 | .4490±.0492 | 3.0 |
| AKA | .9786±.0273 | .4346±.0611 | .3184±.0848 | .2069±.1308 | 5.8 | .8490±.0273 | .2416±.0611 | .4038±.0848 | .4092±.1308 | 5.0 |
| FKA | .9714±.0404 | .4417±.0840 | .3812±.0757 | .2662±.0951 | 5.3 | 1.000±.0404 | .5680±.0840 | .2784±.0757 | .2339±.0951 | 6.0 |
| CFL | .9357±.0273 | .3464±.0416 | .4372±.0815 | .4193±.1194 | 3.0 | .6684±.0273 | .1933±.0416 | .4552±.0815 | .4450±.1194 | 3.8 |
| TC | **.9143**±.0234 | .3452±.0228 | .4366±.0542 | .4548±.0354 | 2.0 | .6424±.0234 | .1869±.0228 | .4645±.0542 | .4459±.0354 | 2.3 |
| ANT(Ours) | .9286±.0286 | **.3345**±.0211 | **.4545**±.0398 | **.4559**±.0177 | **1.3** | **.6077**±.0286 | **.1765**±.0211 | **.4840**±.0398 | **.4675**±.0177 | **1.0** |
| | **YELP Dataset** | | | | | **YEAST Dataset** | | | | |
| Methods | S-Loss↓ | H-Loss↓ | F1 Micro↑ | F1 Macro↑ | Rank↓ | S-Loss↓ | H-Loss↓ | F1 Micro↑ | F1 Macro↑ | Rank↓ |
| BL | **.9723**±.0070 | .3297±.0087 | .4151±.0121 | .3945±.0080 | 2.3 | .9983±.0035 | .4135±.1233 | .4733±.0614 | .3345±.0245 | 4.5 |
| AKA | .9796±.0064 | .3698±.0198 | .3925±.0186 | .3725±.0165 | 4.3 | .9966±.0040 | **.3409**±.0347 | .4749±.0167 | **.3630**±.0106 | 2.3 |
| FKA | 1.000±.0000 | .5807±.0722 | .3540±.0430 | .3031±.0793 | 6.0 | 1.000±.0000 | .5390±.0640 | .3884±.0361 | .3047±.0429 | 5.8 |
| CFL | .9769±.0133 | .3290±.0119 | .3778±.0175 | .3635±.0143 | 4.3 | **.9948**±.0066 | .3514±.0119 | .4792±.0081 | .3200±.0107 | 3.0 |
| TC | .9734±.0039 | .3228±.0050 | .3921±.0168 | .3779±.0109 | 3.0 | 1.000±.0000 | .3487±.0083 | .4917±.0050 | .3383±.0146 | 3.0 |
| ANT(Ours) | .9730±.0030 | **.3177**±.0063 | **.4217**±.0113 | **.4044**±.0060 | **1.3** | **.9948**±.0066 | .3553±.0112 | **.5073**±.0061 | .3535±.0091 | **2.0** |
| | **BIRD Dataset** | | | | | **TMC Dataset** | | | | |
| Methods | S-Loss↓ | H-Loss↓ | F1 Micro↑ | F1 Macro↑ | Rank↓ | S-Loss↓ | H-Loss↓ | F1 Micro↑ | F1 Macro↑ | Rank↓ |
| BL | .5855±.0543 | .0630±.0129 | .0000±.0000 | .0000±.0000 | 4.5 | .9997±.0003 | .1877±.0097 | .0910±.0225 | .0733±.0055 | 3.3 |
| AKA | 1.000±.0000 | .3850±.0621 | .0928±.0576 | .0682±.0276 | 3.5 | 1.000±.0000 | .4075±.0166 | .1967±.0023 | **.1874**±.0069 | 3.3 |
| FKA | 1.000±.0000 | .5613±.0339 | **.0937**±.0295 | **.0779**±.0159 | 3.3 | 1.000±.0000 | .5477±.1077 | **.2045**±.0068 | .1459±.0077 | 3.5 |
| CFL | **.5592**±.0584 | .0575±.0110 | .0000±.0000 | .0000±.0000 | 3.3 | .9997±.0006 | .1810±.0015 | .0761±.0025 | .0713±.0053 | 4.0 |
| TC | **.5592**±.0584 | .0599±.0093 | .0114±.0228 | .0038±.0075 | 3.0 | .9997±.0003 | .1820±.0036 | .0793±.0048 | .0725±.0056 | 3.8 |
| ANT(Ours) | **.5592**±.0584 | **.0561**±.0100 | .0132±.0263 | .0053±.0106 | **2.0** | **.9994**±.0005 | **.1784**±.0015 | .0825±.0033 | .0746±.0013 | **2.3** |
| | **GENBASE Dataset** | | | | | **MEDICAL Dataset** | | | | |
| Methods | S-Loss↓ | H-Loss↓ | F1 Micro↑ | F1 Macro↑ | Rank↓ | S-Loss↓ | H-Loss↓ | F1 Micro↑ | F1 Macro↑ | Rank↓ |
| BL | **.9295**±.0385 | .0833±.0077 | .0734±.0261 | .0370±.0000 | 2.5 | **.9957**±.0086 | .0519±.0029 | .0067±.0134 | .0078±.0156 | 3.5 |
| AKA | 1.000±.0000 | .3991±.0642 | **.1748**±.0299 | **.1853**±.0307 | 3.0 | 1.000±.0000 | .4013±.0300 | **.0655**±.0037 | **.0832**±.0098 | 2.5 |
| FKA | 1.000±.0000 | .6769±.1180 | .0808±.0136 | .0600±.0047 | 3.8 | 1.000±.0000 | .5403±.0166 | .0574±.0046 | .0373±.0063 | 3.3 |
| CFL | **.9295**±.0385 | .0845±.0097 | .0728±.0268 | .0370±.0000 | 3.5 | 1.000±.0000 | .0467±.0053 | .0114±.0146 | .0041±.0049 | 3.8 |
| TC | **.9295**±.0385 | .0836±.0091 | .0734±.0265 | .0370±.0000 | 3.0 | 1.000±.0000 | **.0432**±.0037 | .0046±.0092 | .0005±.0011 | 4.0 |
| ANT(Ours) | **.9295**±.0385 | **.0829**±.0083 | .0740±.0266 | .0370±.0000 | **2.0** | **.9957**±.0086 | .0458±.0018 | .0119±.0151 | .0089±.0117 | **2.3** |

Table 7: Compared performance (mean±std) on eight benchmark datasets. Subset Loss and Hamming Loss are abbreviated as S-Loss and H-Loss, respectively. Rank shows overall performance across all metrics. ↑/↓ indicates the larger/smaller the better.

filtering). Each layer is optimized by minimizing the loss between the outputs of the teacher that is modified and its original outputs without the layer replaced.

- *CFL* [60]: This method extends the student to not only imitate the teachers' logits but also the teachers' final representations. CFL maps the final representations of the student and the teachers into one common space where their similarities are minimized.

- *TC* [101]: TC trains the student to imitate the weighted average logits among the teachers. With only unlabeled data available, such weights are set equally. If a label

| | 3 Teachers | | | | | 4 Teachers | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Methods | S-Loss↓ | H-Loss↓ | F1 Micro↑ | F1 Macro↑ | Rank↓ | S-Loss↓ | H-Loss↓ | F1 Micro↑ | F1 Macro↑ | Rank↓ |
| BL | .9302±.0126 | .2987±.0126 | .4787±.0113 | .3752±.0115 | 3.5 | .9707±.0068 | .3645±.0053 | .3729±.0074 | .2793±.0106 | 3.8 |
| AKA | .9796±.0056 | .3696±.0200 | .3945±.0155 | .3819±.0061 | 4.3 | .9796±.0058 | .3794±.0156 | **.3847**±.0105 | **.3709**±.0126 | 3.0 |
| FKA | .9977±.0030 | .4951±.0622 | .3491±.0135 | .3085±.0111 | 6.0 | .9981±.0029 | .5055±.0387 | .3405±.0406 | .3062±.0429 | 5.0 |
| CFL | .9209±.0110 | .2829±.0031 | .4726±.0069 | .3654±.0100 | 3.5 | .9676±.0095 | .3487±.0071 | .3576±.0079 | .2655±.0043 | 3.8 |
| TC | .9198±.0058 | **.2805**±.0028 | .4812±.0041 | .3718±.0078 | 2.3 | **.9618**±.0060 | **.3410**±.0031 | .3614±.0051 | .2644±.0040 | 3.0 |
| ANT(Ours) | **.9178**±.0054 | .2857±.0043 | **.4879**±.0104 | **.3825**±.0104 | **1.5** | .9668±.0051 | .3553±.0052 | .3733±.0031 | .2808±.0039 | **2.5** |

| | 5 Teachers | | | | |
|---|---|---|---|---|---|
| Methods | S-Loss↓ | H-Loss↓ | F1 Micro↑ | F1 Macro↑ | Rank↓ |
| BL | .9769±.0045 | .3738±.0033 | .3732±.0088 | .2809±.0105 | 3.8 |
| AKA | .9896±.0041 | .3899±.0095 | **.3967**±.0109 | **.3837**±.0077 | 3.0 |
| FKA | .9965±.0026 | .4621±.0276 | .3403±.0351 | .2845±.0137 | 5.3 |
| CFL | .9641±.0100 | **.3376**±.0138 | .3495±.0085 | .2552±.0145 | 3.5 |
| TC | **.9564**±.0110 | .3415±.0055 | .3581±.0042 | .2669±.0087 | 3.0 |
| ANT(Ours) | .9703±.0051 | .3649±.0053 | .3821±.0016 | .2890±.0080 | **2.5** |

Table 8: Amalgamating many teachers observed on YELP.

| | 100% of labels shared between teachers | | | | | 75% of labels shared between teachers | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Methods | S-Loss↓ | H-Loss↓ | F1 Micro↑ | F1 Macro↑ | Rank↓ | S-Loss↓ | H-Loss↓ | F1 Micro↑ | F1 Macro↑ | Rank↓ |
| BL | .7830±.0215 | .2749±.0177 | .2977±.0302 | .3035±.0237 | 3.8 | .7761±.0285 | .2665±.0078 | .3121±.0170 | .3229±.0247 | 4.0 |
| AKA | .8924±.0509 | .2578±.0442 | **.3278**±.0665 | **.3412**±.0608 | 2.8 | .8577±.0509 | .2610±.0474 | **.3779**±.0428 | **.4112**±.0370 | 2.8 |
| FKA | .9948±.0066 | .4586±.0938 | .2480±.0529 | .1939±.0743 | 6.0 | .9948±.0104 | .4757±.1056 | .3158±.0575 | .2937±.1123 | 4.8 |
| CFL | .7761±.0302 | .2454±.0104 | .2742±.0194 | .2780±.0133 | 3.5 | .7760±.0104 | **.2434**±.0024 | .2972±.0156 | .2915±.0183 | 3.3 |
| TC | .7795±.0307 | **.2422**±.0074 | .2889±.0087 | .2857±.0126 | 3.0 | .7760±.0418 | .2474±.0087 | .2835±.0175 | .2845±.0148 | 4.0 |
| ANT(Ours) | **.7639**±.0321 | .2555±.0153 | .3012±.0244 | .3042±.0198 | **2.0** | **.7431**±.0204 | .2474±.0095 | .3273±.0225 | .3366±.0222 | **1.8** |
| | **50% of labels shared between teachers** | | | | | **25% of labels shared between teachers** | | | | |
| Methods | S-Loss↓ | H-Loss↓ | F1 Micro↑ | F1 Macro↑ | Rank↓ | S-Loss↓ | H-Loss↓ | F1 Micro↑ | F1 Macro↑ | Rank↓ |
| BL | .7795±.0208 | .2630±.0104 | .2916±.0125 | .3047±.0207 | 4.3 | .6389±.0589 | .1794±.0141 | .4730±.0342 | .4559±.0225 | 2.3 |
| AKA | .8837±.0622 | .2593±.0548 | **.3752**±.0421 | **.3818**±.0587 | 2.8 | .8490±.0521 | .2193±.0553 | .3344±.1254 | .2857±.1632 | 5.0 |
| FKA | 1.000±.0000 | .5440±.0672 | .2463±.0555 | .1907±.0455 | 6.0 | .9913±.0104 | .3927±.0623 | .2698±.0468 | .1840±.0427 | 6.0 |
| CFL | .7847±.0205 | .2506±.0076 | .2949±.0150 | .3032±.0237 | 3.8 | .6719±.0582 | .1843±.0143 | .4579±.0493 | .4470±.0320 | 4.0 |
| TC | .7743±.0230 | **.2419**±.0095 | .2987±.0097 | .3081±.0142 | 2.3 | .6372±.0398 | .1812±.0105 | .4687±.0297 | .4500±.0213 | 2.8 |
| ANT(Ours) | **.7656**±.0173 | .2535±.0047 | .3046±.0090 | .3087±.0209 | **2.0** | **.6181**±.0567 | **.1771**±.0201 | **.4914**±.0417 | **.4696**±.0242 | **1.0** |

| | 0% of labels shared between teachers | | | | |
|---|---|---|---|---|---|
| Methods | S-Loss↓ | H-Loss↓ | F1 Micro↑ | F1 Macro↑ | Rank↓ |
| BL | .4983±.0262 | .1221±.0123 | .6540±.0412 | .6597±.0460 | 3.0 |
| AKA | .8906±.0545 | .2737±.0241 | .3847±.0381 | .3976±.0569 | 5.0 |
| FKA | 1.000±.0000 | .5324±.0735 | .3013±.0250 | .2975±.0378 | 6.0 |
| CFL | .5174±.0370 | .1204±.0101 | .6579±.0276 | .6615±.0324 | 2.8 |
| TC | .5052±.0414 | .1227±.0133 | .6514±.0332 | .6622±.0422 | 3.3 |
| ANT(Ours) | **.4392**±.0313 | **.1056**±.0115 | **.6846**±.0322 | **.6901**±.0372 | **1.0** |

Table 9: Results of various number of labels shared between teachers observed on SCENE.

appears in one teacher, its logit is used directly.

**Metrics.** We use four standard multi-label metrics and the averaged rank to show the overall performance as follows:

- *Subset Loss (S-Loss)*: measures the label-set based performance by strictly assigning 0 if all elements in the predicted label-set match entirely the true label-set or 1 otherwise. The lower score indicates the better performance.
- *Hamming Loss (H-Loss)*: considers instead each label (label based) which can measure differences between the *almost correct* and *completely wrong* predictions. Again, the lower score is better.
- *F1 Micro*: is the harmonic mean between the global values of precision and recall. In the context of MLC, this metric measures label based performance much like H-Loss. However, a higher score indicates a better performance.
- *F1 Macro*: pays more attention on minority class labels by averaging the F1 scores calculated separately for each label $y_j$. A higher score is preferred.

Since each metric evaluates a different property of a model's performance, we also report each model's rank across all metrics—1 indicates the best performance.

**Experiments and Results.** We firstly demonstrate that ANT outperforms the five alternative methods on eight benchmark datasets by successfully leveraging label dependency knowledge captured differently between teachers. For this experiment, we train a student from two teachers trained on different data—from each original dataset—that cover different subsets of labels. For each dataset, teachers learn from an equal number of labels with half of their labels overlapping. The results on eight datasets are shown in Table 7. ANT consistently achieves the highest rank averaged across all metrics across the board while the second-best methods alternate between BL, AKA, and TC depending on different datasets. We notice that BL performs quite good comparing to the other methods. This is because we apply the same core rationale in the proposed ANT to trust more on the teacher that predicts positive when there is contradictory predictions between the teach-

ers. Moreover, as expected, ANT is clearly the strongest method particularly for Hamming Loss and Subset Loss, achieving the best performance for six out of eight datasets for both metrics. This demonstrates ANT's success at predicting both the individual labels and entire label sets, both of which are core tasks for multi-label learning.

Next, to investigate the case where teachers learn vastly different knowledge from each other, we follow other recent work [101] by varying the number of shared labels between teachers. Using the SCENE dataset, where most methods perform their best across all metrics, we vary the proportion of labels shared between two teachers from 100% (identical label sets) to 0% (completely distinct label sets). As expected, the results—shown in Table 9—indicate that as the teachers share fewer labels, the resulting students become more effective. This is because, with fewer shared labels, the teachers have less common labels that their predictions may contradict each other and provide contradictory knowledge for the student. All in all, ANT achieves the top average rank across the board on average. This shows that ANT can extract predictions from heterogeneous sources more reliably than state-of-the-art alternatives in all cases. ANT especially shows impressive performance in winning all other methods on the Subset Loss which is the only metric that can measure the effective method for MLC in terms of the label-set based performance. Furthermore, even for the simplest cases of 0% of labels shared between teachers which does not contain any disagreed prediction, ANT also achieves the best performance by integrating predictions based on the most confident predicted probability among teachers.

Finally, we explore the more-challenging case of amalgamating many teachers which naturally create several potential unobserved dependencies between labels that each label exists across teachers. We thus conduct experiments using three, four, and five teachers on the YELP dataset. We select YELP because it has many instances, which allows us to train more independent teachers. Furthermore, as reported in Table 7, all methods achieve good performance across all metrics for this dataset. The results on these settings

are shown in Table 8. Once again, ANT achieves the highest rank for all settings. This indicates that ANT not only learns the dependency knowledge from each particular teacher but also effectively infers unobserved dependencies between labels that exist across teachers. This is achieved by allowing the transferee teacher to revise its predictions based on the predictions of the more-competent teacher. The recursive learning aids ANT's success in inferring these dependencies even across many teachers where knowledge must be transferred between more teachers or different subsets of teachers.

## 4.7 Conclusions

We propose **A**daptive K**N**owledge **T**ransfer (ANT), the first solution for the open problem of knowledge amalgamation for multi-label classification (KA-MLC). The key idea is to train a student from teachers that transfer adaptively their label dependency knowledge to each other. This encourages each teacher to revise its prediction based on the knowledge from more competent teachers that can utilize other labels effectively to yield more accurate predictions. The biggest benefit arises when the features of an instance alone do not contain enough information to make confident predictions and thus there are disagreements between teachers. Moreover, this adaptive approach leads ANT to succeed in learning to infer dependencies between all labels across all teachers. Our comprehensive experimental study on eight real-world datasets demonstrates that ANT significantly outperforms the state-of-the-art alternatives by achieving the best averaged rank across many standard multi-label metrics for all datasets.

# 5 Multi-Tasking Knowledge Amalgamation

This task is in preparation for submission to CIKM 2023.

## 5.1 Motivation

Over the past decades, Multi-Task Learning (MTL) has gained attention in deep learning community that attempts to mimic human learning ability in using the knowledge learned from one task to support the learning for another task. For example, humans can use the skill learned from playing tennis to help learning to play squash. In principle, MTL aims to leverage the common knowledge contained in multiple related tasks to improve the generalization performance of all tasks simultaneously.

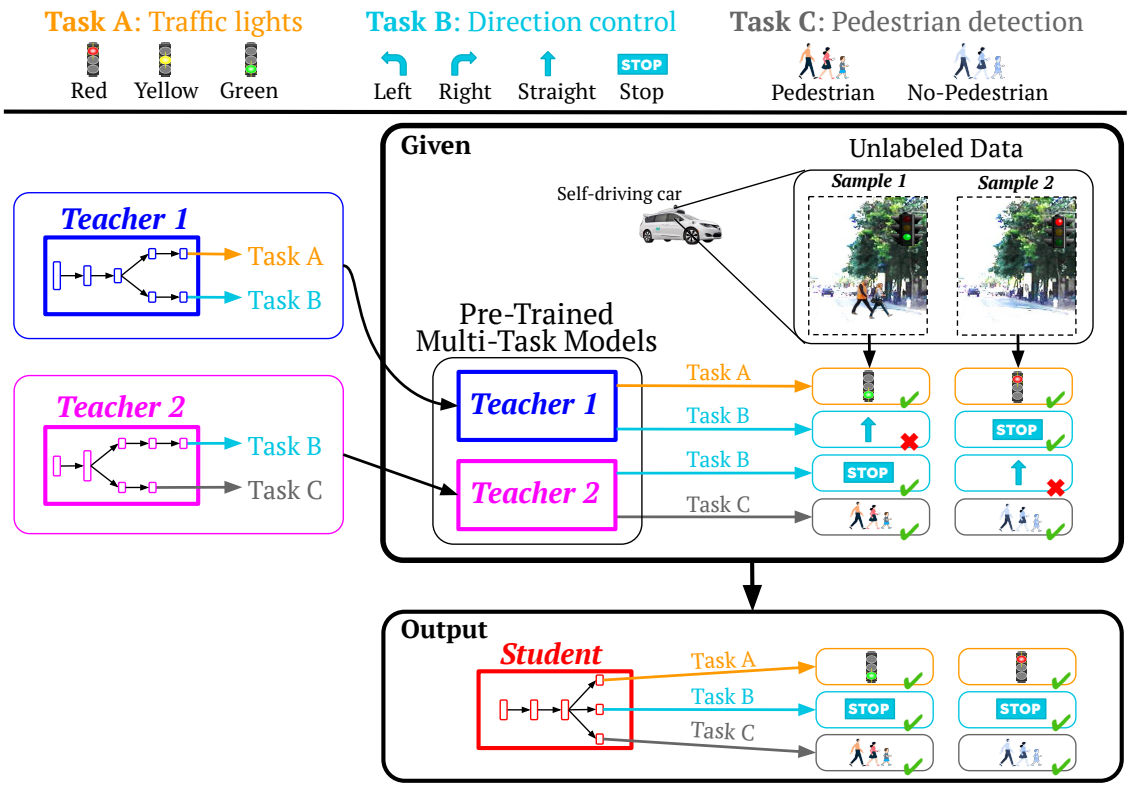



Figure 8: Learning a *student* model to handle multiple tasks by amalgamating multiple pre-trained multi-task models (teachers), which their specialized task sets related to each other.

Most existing MTL methods [57, 50, 67, 88] are developed for standard supervised learning that requires a huge amount of labeled training data. Unfortunately, acquiring such labeled data is practically hard to come by and even more severe as the number of tasks grows. To support the community, some institutes that own private datasets may release their pre-trained models without sharing the data themselve. However, each model may be trained on a different set of tasks leading them to contain different fundamental knowledge as they aim to generalize such knowledge on the different task sets.

To this end, as depicted in Figure 8, we study the problem of knowledge amalgamation for multi-task learning. More specifically, we aim to learn one unified multi-task model (student) from multiple pre-trained multi-task models (teachers) using purely unlabeled training data. The goal is to develop a student model that can effectively leverage fundamental knowledge among all tasks specialized across all teachers in order to improve the performance of all tasks at hand.

### 5.2 Related works

**Multi-Task Learning (MTL).** MTL is the learning paradigm where the training data from multiple tasks are used simultaneously. It aims to fuse the knowledge among the related tasks into a powerful shared representation that could be used to improve the generalized performance of all the tasks [15, 21]. The MTL works are typically studied using either *hard parameter sharing* or *soft parameter sharing* approaches [87].

Hard parameter sharing approach has been long studied since the early stage of developing the study of MTL [15]. Several works based on this approach [57, 50, 73, 8] output ultimately one model used across tasks which significantly reduce the computation issue, especially when dealing with a large number of tasks. Many existing works apply the hard parameter sharing approach to various variations in several domains. For example, in computer vision, [58] train all tasks to share the first five layers of AlexNet before

applying the task-specific fully-connected layers for each task. [57] propose MTAN to learn a task-specific layer with attention mechanism branching out of the shared representation between all tasks. Several other works also apply the hard parameter sharing for NLP such as [118, 4] introduce methods that jointly learn the shared parameters for sequence-tagging tasks across languages.

On the other hand, the other works based on the soft parameter sharing approach [67, 59, 88, 34] train each task with its own model with its own separate parameters. To encourage the share information across the tasks, they usually apply some regularization techniques to add a constraint on the parameters between the parallel layers from all models to be similar. For instance, [27] apply $l_2$ distance between the layers across the models; [117] use instead the trace norm; or [67] develop the cross-stitch units that combine the information for the other tasks by applying the linear combination of the previous layer from all models as the input to each layer. These methods however suffer heavily from computational or memory inefficient since their techniques require a huge amount of resources which grows proportionally with the number of tasks.

Most importantly, the above existing MTL works have been developed using standard supervised learning, They require a huge amount of labeled data especially when the number of tasks grows. However, our AmalMTH problem assumes no labels are available.

**Knowledge Amalgamation (KA).** Several existing KA works [92, 60, 108, 101] focus on single-task learning. That is the teachers and the student solve the same task. The first work proposed by [92] assume the architectures of the teachers and student are homogeneous, *i.e.* their structures have an aligned layer-to-layer. Thus, they propose a layer-to-layer training method to fuse knowledge from the teachers into the student model. Some later works [60, 101] extend the scope to handle cases when the teachers may have different structures but still develop the student model for the single-task learning.

Recently, some works [119, 93] study KA for multi-task classification. Unfortunately, they assume the architectures of the teachers and students contain identical number of layers for which they propose the layer-to-layer based approaches. For each corresponding layer between the teachers and the student, [119] modify each teacher by using the student's layer instead of its original layer. Then such student's layer is updated to ensure that the modified teacher still makes the same prediction with the original teacher. Similarly, [93] develop a layer-to-layer method by encouraging the corresponding layers of the teachers and the student to be similar through a transfer bridge. This work proposes the dual-stage training by firstly training one student model per each individual task. Then these students are finally combined into one student model to handle multiple tasks. This method could easily cause a computational issue, especially when dealing with many numbers of tasks.

Most importantly, in practice, the teacher models are pre-trained separately, leading them to often come with heterogeneous architectures. Thus, these layer-to-layer based approaches are not applicable to realistic cases.

### 5.3 Problem Definition

This task addresses the problem of Amalgamating Multi-Task Models with Heterogeneous Architectures (AmalMTH). We are given an unlabeled dataset containing $n$ instances with $d$ features, denoted as $\mathcal{X} = \{\mathbf{x}^i\}_{i=1}^n$ where $\mathbf{x}^i \in \mathbb{R}^d$, and also given a set of $m$ powerful pre-trained multi-task models (*teachers*), $\mathcal{M} = \{\mathbf{M}^j\}_{j=1}^m$. Each teacher $\mathbf{M}^j$ handles a particular task set of the $t_j$ distinct tasks, represented by $\mathcal{T}^j = \{\mathbf{T}_k^j\}_{k=1}^{t_j}$. We note that the teachers' task sets may or may not overlap with each other.

For simplicity, in this problem, we define each task as a binary classification task. Therefore, for each instance $\mathbf{x}^i$, the prediction from each teacher $\mathbf{M}^j$ on its specialized task $\mathbf{T}_k^j$ is $\hat{y}_k^{j,i}$ where $\hat{y}_k^{j,i} = 1$ if the teacher $\mathbf{M}^j$ predicts that the task $\mathbf{T}_k^j$ associates (posi-

tive) with instance $\mathbf{x}^i$ or 0 (negative) otherwise.

The ultimate goal is to train a *student* model to master all tasks in the union of the teachers' task sets, $\mathcal{T} = \bigcup_{j=1}^{m} \mathcal{T}^j$. For clarity, $\mathcal{T} = \{\mathbf{T}_k\}_{k=1}^{t}$ where $t$ is the number of distinct tasks in the union of all teachers' task sets. Thus, for each instance $\mathbf{x}^i$, the student outputs the prediction for all tasks as $\hat{\mathcal{Y}}^i = \{\hat{y}_k^i\}_{k=1}^{t}$ where $\hat{y}_k^i \in \{0, 1\}$. To improve readability, we describe the rest of the paper in terms of one instance $x^i$ and drop the superscript $i$ hereafter.

### 5.4 Challenges

There are three challenges associated with this new amalgamating task which are summarized as follows:

- *No labeled data*. Traditional MTL methods are developed under the standard supervised setting, which requires labeled data for training. With only the availability of unlabeled data in this problem, these existing models are not applicable. Therefore, developing a solution for MTL that does not need the labeled data is challenging.

- *Disparate knowledge captured across teachers*. Since each teacher is pre-trained separately, they are typically trained to handle a different set of tasks. Each of which contains a different subset of the tasks handled by the student. Consequently, the internal representations learned by each teacher may capture different information as they aim to generalize to the different task sets. Worst yet, when the teachers handle partial tasks in common, such different information captured between them may lead to their conflict predictions on their shared tasks. For example, both teachers in Figure 8 are trained to handle the direction control task but, for sample 1, Teacher 1 predicts to go straight while Teacher 2 predicts to stop. These conflicts definitely cause the troublesome in training the student. Thus, an ideal student should com-

bine the disparate knowledge from teachers effectively.

- *Combining knowledge from heterogeneous architecture teachers.* Learning from the teacher's internal layers is shown success in preserving the teacher's knowledge [119, 93]. However, the teachers may have different architectures. They may come with a different number of layers or different sizes/scales for each layer. This raises a challenge in training a student since there is no alignment between the architectures of teachers and student. Moreover, there is no ground truth on which layers of the teachers the student should learn from.

### 5.5 Proposed Method: Versatile Common Feature Consolidator (VENUS)

To overcome these challenges, we propose Versatile Common Feature Consolidator (VENUS) to solve the open problem of AmalMTH. VENUS improves the *common feature* shared across all tasks by fusing the generalizable knowledge captured differently in the final shared representation across all teachers. This learning is trained through the *Feature Consolidator* that is designed to allow the student to learn versatilely from teachers that may have different architectures.

In this problem, we are given the $m$ multi-task pre-trained models (teachers), $\mathcal{M} = \{\mathbf{M}^j\}_{j=1}^m$. Each teacher model consists of two main parts: the layers shared among all tasks and the task-specific layers.

Let $c$ be the number of the shared layers in the teacher $\mathbf{M}^j$ where $F_c^j$ is the final layer of these $c$ shared layers. We call the $F_c^j$ as the *final shared representation*. We denote the shared layers as $\{h_u^j\}_{u=1}^c$ and thus $F_c^j$ is outputted as:

$$F_c^j = h_c^j(\cdots(h_2^j(h_1^j(x)))). \tag{38}$$

For each task $\mathbf{T}_k^j \in \mathcal{T}^j$, assuming that there are $u_j$ task-specific layers branching out of $F_c^j$. We denote the logit obtained from these layers as $\ell_k^j$. Then the logit $\ell_k^j$ is passed

through the sigmoid function ($\sigma$) to acquire the predicted probability ($p_k^j$) as follow:

$$\ell_k^j = h_{u_j}^j(\cdots(h_{c+2}^j(h_{c+1}^j(F_c^j)))) \tag{39}$$

$$p_k^j = \sigma(\ell_k^j). \tag{40}$$

Our goal is to train a student model that can combine the shared knowledge across all tasks in the teachers' union set of tasks into the *common feature* that could generalize for

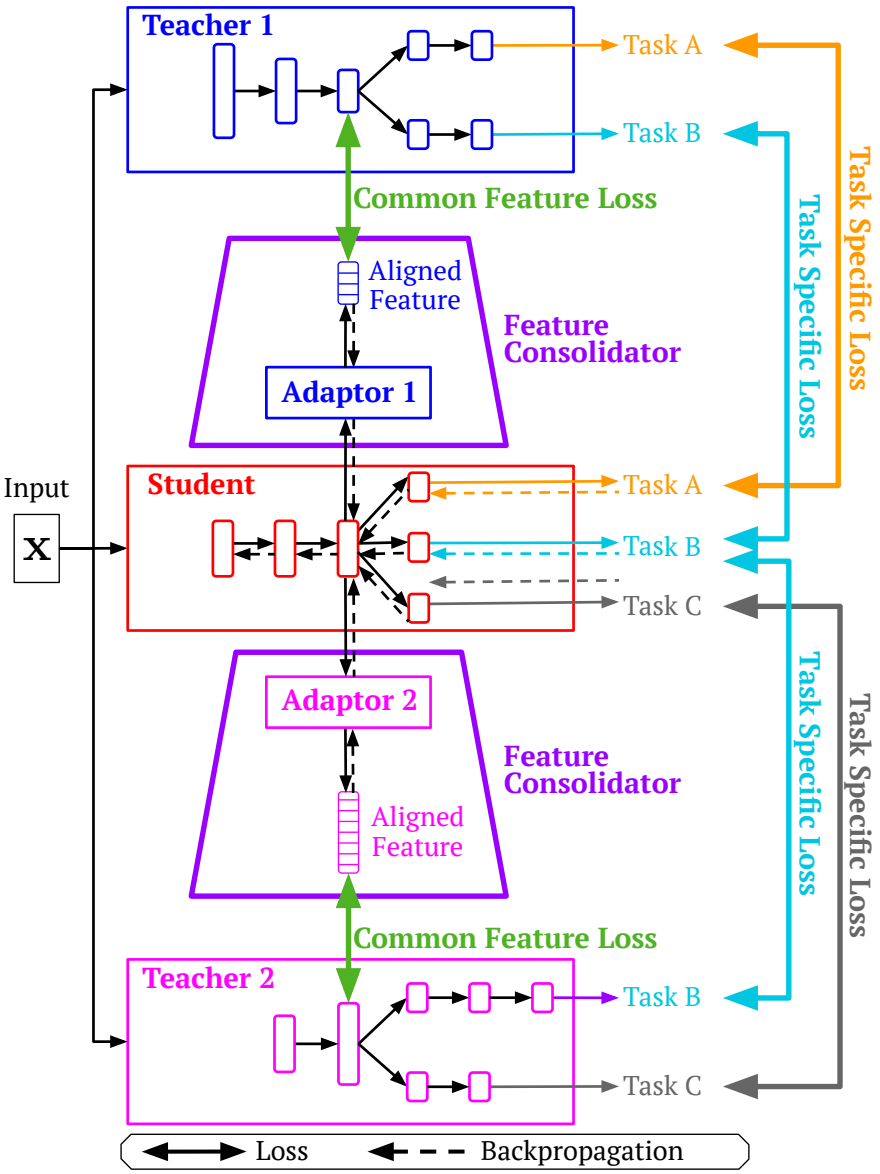



Figure 9: The proposed Versatile Common Feature Consolidator (VENUS).

better performance of all tasks. Our proposed student model also consists of two main parts, which are the shared layers and task-specific layers. We describe each part of the model as follows.

#### 5.5.1 Backbone Model.

In our method, we call the shared layers as the backbone model, which consists of the $s$ layers shared across all tasks in $\mathcal{T}$. We note that this backbone model can adopt any arbitrary architectures, *e.g.,* ResNet, DenseNet, VGG, or any customized architecture from scratch. The aim of this part is to unify the common feature ($F_s$) that can benefit all tasks simultaneously:

$$F_s = H_{\Theta_S}(x) \tag{41}$$

$$\text{where } H_{\Theta_S}(x) = h_s(\cdots(h_2(h_1(x)))), \tag{42}$$

$$\text{and } \Theta_S \text{ are learnable parameters.} \tag{43}$$

In order to extract the shared knowledge across all tasks, this common feature $F_s$ is trained to imitate the final shared representations captured in all teachers. That is the student model is trained to learn $F_s$ to be most similar to each $F_c^j$ from all teachers, *i.e.* $\Theta_S$ is updated according to the following loss function.

$$J(\Theta_S) = \frac{1}{m}\sum_{j=1}^{m}\left(F_s - F_c^j\right)^2. \tag{44}$$

However, the given teachers and the student may have heterogeneous architectures. That is their features may contain different scales or sizes, raising the computational incompatibility in Equation 44. Therefore, the proposed VENUS is designed to handle this issue through the Feature Consolidator, which is described next.

### 5.5.2 Feature Consolidator.

The feature consolidator, as shown in Figure 9, learns to align the common feature $F_s$ of the student versatilely to each individual teacher's final shared presentation ($F_c^j$). For this purpose, we develop a particular adapter for each teacher. An individual adaptor is a network that is designed to learn a mapping from the student's common feature to the particular teacher's shared representation. Thus, for each teacher $\mathrm{M}^j$, the adaptor is defined as the trainable network:

$$\hat{F}_c^j = \mathrm{ReLU}(W^j \cdot F_s + b^j) \tag{45}$$

where $W^j$ and $b^j$ are learnable parameters. ReLu is the rectified linear unit activation function.

These adaptors enable the student model to unify knowledge from the heterogeneous teachers by adjusting their different scales and biases through the learnable parameters. Specifically, the weight matrix $W^j$ is trained to transform the student's feature into the same size as the teacher's features while the $b^j$ is trained to capture their shifting. Moreover, the ReLU function supports the learning to capture the non-linearity transformation between their features. Once the training procedure is done; these adaptors can be removed at the inference time for unseen data.

Then the loss function in Equation 44 is modified to be:

$$J(\Theta_S) = \frac{1}{m}\sum_{j=1}^{m}\left(\hat{F}_c^j - F_c^j\right)^2. \tag{46}$$

### 5.5.3 Task-Specific Layers.

After obtaining the common feature shared across all tasks from Equation 45, the student model branches out task-specific layers for each task $\mathbf{T}_k \in \mathcal{T}$. These layers aim to learn the specific information for each task on top of the generalized knowledge so as to make the final prediction for the individual task $\mathbf{T}_k$. Let $\ell_k$ represent the logit for the task $\mathbf{T}_k$.

We learn $\ell_k$ as:

$$\ell_k = H_{\Theta_k}(F_s);\;\; H_{\Theta_k}(F_s) = h_{o_k}(\cdots(h_{s+2}(h_{s+1}(F_s)))) \tag{47}$$

where $o_k$ and $\Theta_k$ are respectively the number of layers and the learnable parameters specifically to the task $\mathbf{T}_k$. Then the predicted probability for $\mathbf{T}_k$, denoted by $q_k$, is calculated by passing the sigmoid function ($\sigma$) to its corresponding logit $\ell_k$. The final prediction is obtained by binarizing $q_k$ with a threshold of 0.5.

$$q_k = \sigma(\ell_k) \tag{48}$$

$$\hat{y}_k = \begin{cases} 1 & \text{if } q_k > 0.5 \\ 0 & \text{otherwise.} \end{cases} \tag{49}$$

Let $\mathcal{L}_k$ be the set of logits for the task $\mathbf{T}_k$ gathered from all teachers specializing on the task $\mathbf{T}_k$. The consensus predicted probability for the task $\mathbf{T}_k$, denoted as $p_k$, is thus obtained by passing the sigmoid function ($\sigma$) on the average logits in $\mathcal{L}_k$.

$$\forall \ell_a \in \mathcal{L}_k,\;\; p_k = \sigma\big(\frac{1}{|\mathcal{L}_k|}\sum_{a=1}^{|\mathcal{L}_k|}\ell_a\big). \tag{50}$$

For each task $\mathbf{T}_k$, its task-specific layers are trained to minimize the cross entropy loss between the predicted probability $q_k$ and the consensus predicted probabilities from the teachers specializing on the task $\mathbf{T}_k$. That is the parameters $\Theta_k$ are trained to minimize the task-specific loss for the task $\mathbf{T}_k$ as:

$$J(\Theta_k) = -p_k \log(q_k). \tag{51}$$

Finally, The student model is trained to combine the common knowledge across all teachers for all tasks and also imitate the teachers' consensus predictions simultaneously. Let $\omega$ denote all trainable parameters used for the whole training, which includes $\Theta_S$, $\Theta_k$, $W^j$, and $b^j$ for all tasks. These parameters are iteratively updated by minimizing the final

loss, combining both loss functions in Equation 46 and 51, defined as follows.

$$J(\omega) = \frac{1}{m}\sum_{j=1}^{m}\left(\hat{F}_c^j - F_c^j\right)^2 - \frac{1}{t}\sum_{k=1}^{t}\left(p_k \log(q_k)\right). \tag{52}$$

### 5.6 Evaluation

Our proposed method, VENUS, is evaluated against three alternative methods on two benchmark datasets.

**Datasets.** The two datasets used in our experiments are described as the following. We note that each dataset contains multiple labels, each of which is treated as an independent binary classification task.

• *PASCAL VOC 2007*: This dataset is the PASCAL Visual Object Classes Challenge 2007 [29] that comes with 9,963 images. Each image can relate to 20 object labels. We conduct 20 different tasks to predict the presence or absence of each label.

• *3D*: The 3D dataset contains four tasks that is extracted from the original 3d-shapes dataset [13]. The four tasks are to identify (1) whether the object's color is blue, (2) whether the floor's color is green, (3) whether the wall's color is purple, and (4) whether the wall's color is pink. The dataset contains 168,959 images in total.

**Compared Methods.** Since we are the first to define the AmalMTH in this work, the state-of-the-art approaches for knowledge amalgamation are not applicable to this setting. We, therefore, compare the proposed VENUS against three methods adapting based on some existing literature as follows:

• *MuST* [35]: This method follows the idea of pseudo labeling from [35] to train a student model that imitates the pseudo-predictions generated by the teacher models.

• *KD* [42]: The student model is trained using the Knowledge Distillation paradigm by

learning the student's soft targets (logits) to imitate the average of all teachers' logits.

• *CFL* [60]: CFL trains the student to imitate the teachers' logits and their final layers. It maps the final layers of the student and teacher into a common space and trains them to be similar.

**Experiments and Results.** All experiments are evaluated using both accuracy and AUC. We report the results of each task for each dataset. However, to show the overall performance for a dataset, we report *the average rank* across all tasks where 1 indicates the best performance. For a fair comparison, we use ResNet18 [40] as the backbone model for the student in all experiments.

**Effectiveness of VENUS in learning the common feature for all tasks across teachers.** We first investigate how effective our proposed method is when compared against the other methods across all datasets. To observe this, for each dataset, we train a student from the two teachers with heterogeneous architectures—DenseNet [45] for Teacher 1 and ResNet18 [40] for Teacher 2. Each of them is trained on the same number of tasks with 30% of their tasks shared. Thus, the final shared representation learned by each teacher contains different knowledge since they aim to benefit the generalization toward the different sets of tasks.

The results, as shown in Table 10, show that the proposed VENUS outperforms alternative methods significantly as it reaches the best average rank for both metrics in all datasets. We observe that MuST shows consistently the worst performance. This suggests that using purely the pseudo-predictions from the teachers cannot provide enough information for the student to learn high-quality common feature to be used across all tasks in all teachers.

In all settings, we see that CFL and VENUS outperform clearly over MuST, indicating that incorporating the learning from teachers' final shared layers could lead the student

**Dataset: PASCAL VOC 2007**

| Methods / Tasks | *Accuracy* | | | | *AUC* | | | |
|---|---|---|---|---|---|---|---|---|
| | MuST | KD | CFL | Ours: VENUS | MuST | KD | CFL | Ours: VENUS |
| Aeroplane | .5692±.0332 | .7642±.0163 | .7799±.0054 | **.8145**±.0109 | .5767±.0415 | .7913±.0085 | **.8042**±.0131 | .7938±.0137 |
| Bicycle | .5206±.0310 | .6127±.0220 | **.6174**±.0073 | .6079±.0099 | .5257±.0331 | .6074±.0161 | .6125±.0131 | **.6257**±.0021 |
| Boat | .5758±.0449 | .7333±.0344 | .7455±.0273 | **.7576**±.0292 | .5787±.0505 | .7451±.0288 | **.7506**±.0308 | .7465±.0232 |
| Bus | .5513±.0462 | .6688±.0134 | **.7073**±.0196 | .6923±.0192 | .5419±.0320 | .6603±.0100 | **.6994**±.0105 | .6908±.0230 |
| Car | .6242±.0241 | .6634±.0087 | .6577±.0166 | **.6757**±.0153 | .6126±.0145 | .6456±.0119 | .6453±.0101 | **.6569**±.0209 |
| Motorbike | .5604±.0306 | **.6996**±.0282 | .6813±.0291 | .6850±.0510 | .5699±.0319 | **.6895**±.0191 | .6689±.0143 | .6790±.0463 |
| Train | .6201±.0599 | **.6274**±.0307 | .6078±.0443 | .6054±.0237 | .6067±.0567 | .6274±.0181 | .6126±.0230 | **.6320**±.0100 |
| Bottle | .5413±.0546 | **.6320**±.0183 | **.6320**±.0302 | .6307±.0083 | .5488±.0613 | **.6535**±.0113 | .6440±.0334 | .6518±.0064 |
| Chair | .5063±.0055 | .5890±.0129 | **.5993**±.0078 | .5982±.0245 | .5076±.0075 | .6084±.0128 | .6102±.0050 | **.6163**±.0235 |
| Dining-table | .6083±.0449 | .6643±.0062 | .6702±.0273 | **.6893**±.0217 | .6204±.0440 | .6840±.0022 | .6926±.0157 | **.6939**±.0141 |
| Potted-plant | **.6142**±.0104 | .6037±.0060 | .5958±.0231 | .6076±.0378 | **.6354**±.0170 | .6137±.0050 | .6256±.0094 | .6301±.0070 |
| Sofa | .5654±.0752 | **.6577**±.0087 | .6335±.0179 | .6398±.0194 | .5691±.0818 | **.6686**±.0059 | .6554±.0026 | .6602±.0091 |
| TV | .5403±.0048 | **.6500**±.0341 | .6444±.0064 | .6222±.0292 | .5398±.0090 | .6416±.0172 | **.6437**±.0099 | .6315±.0210 |
| Bird | .5556±.0315 | .5711±.0154 | .5844±.0139 | **.6067**±.0611 | .5425±.0217 | .5717±.0046 | .5637±.0149 | **.5877**±.0286 |
| Cat | **.7078**±.0149 | .6648±.0065 | .6798±.0313 | .6910±.0195 | **.6936**±.0145 | .6743±.0052 | .6684±.0248 | .6912±.0148 |
| Cow | .5606±.0546 | .6162±.0487 | .5808±.0315 | **.6414**±.0175 | .5450±.0566 | .5635±.0227 | .5575±.0217 | **.6107**±.0106 |
| Dog | **.6224**±.0303 | .6142±.0205 | .5956±.0228 | .6131±.0020 | **.6248**±.0225 | .6082±.0135 | .5985±.0184 | .6022±.0023 |
| Horse | .5278±.0141 | .6193±.0247 | .6307±.0453 | **.6406**±.0102 | .5369±.0080 | .6393±.0057 | .6570±.0301 | **.6654**±.0083 |
| Sheep | .7179±.0555 | **.7500**±.0333 | .6859±.0294 | .6987±.0618 | **.7764**±.0445 | .7628±.0287 | .7362±.0573 | .7365±.0756 |
| Person | .5650±.0338 | .6234±.0129 | .6263±.0045 | **.6265**±.0067 | .5863±.0258 | .5951±.0123 | .5954±.0046 | **.5966**±.0084 |
| Ave. RANK | 3.35 | 2.25 | 2.40 | **1.95** | 3.40 | 2.40 | 2.50 | **1.70** |

**Dataset: 3D**

| Methods / Tasks | *Accuracy* | | | | *AUC* | | | |
|---|---|---|---|---|---|---|---|---|
| | MuST | KD | CFL | Ours: VENUS | MuST | KD | CFL | Ours: VENUS |
| Blue object | .9087±.0080 | **.9157**±.0055 | .9137±.0061 | .9147±.0026 | .9086±.0068 | **.9164**±.0053 | .9150±.0066 | .9158±.0031 |
| Green floor | .6005±.0661 | .8102±.0077 | .8027±.0099 | **.8268**±.0098 | .6006±.0662 | .8106±.0066 | .8024±.0095 | **.8276**±.0082 |
| Purple wall | .9630±.0011 | .9699±.0037 | **.9736**±.0006 | .9713±.0007 | .9632±.0015 | .9698±.0034 | **.9739**±.0009 | .9709±.0006 |
| Pink wall | .7204±.0530 | .9126±.0072 | .9244±.0174 | **.9357**±.0175 | .7201±.0527 | .9131±.0075 | .9243±.0167 | **.9361**±.0170 |
| Ave. RANK | 4.00 | 2.25 | 2.25 | **1.50** | 4.00 | 2.25 | 2.25 | **1.50** |

Table 10: Compared performance on the two datasets: PASCAL VOC 2007 and 3D.

to learn better common feature generalizable across all tasks. However, we notice that unlike VENUS that clearly performs better than KD, CFL does not show a clear superior performance over KD. That means although both CFL and VENUS utilize more information from the teachers' final shared representation, our proposed method comes with a more successful strategy in fusing such knowledge through the Feature Consolidator.

**Amalgamating disparate knowledge across teachers.** In practice, teachers typically contain disparate knowledge since they are pre-trained separately on different task sets and also different datasets. To evaluate how well the student model can combine their disparate knowledge, we follow the recent work [101] by observing the cases when the

**Accuracy**

| Methods / Tasks | *75% of tasks shared between teachers* | | | | *50% of tasks shared between teachers* | | | | *25% of tasks shared between teachers* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MuST | KD | CFL | Ours: VENUS | MuST | KD | CFL | Ours: VENUS | MuST | KD | CFL | Ours: VENUS |
| Aeroplane | **.868**±.038 | .808±.011 | .783±.016 | .789±.044 | **.918**±.005 | .786±.029 | .764±.016 | .818±.039 | .610±.076 | .802±.028 | .774±.009 | **.815**±.011 |
| Bicycle | .584±.068 | .613±.052 | **.629**±.013 | .619±.058 | .608±.012 | .606±.027 | **.633**±.022 | .630±.015 | .543±.048 | .600±.021 | .606±.019 | **.608**±.010 |
| Boat | **.882**±.009 | .755±.018 | .758±.010 | .764±.055 | .618±.036 | .694±.038 | .733±.014 | **.739**±.014 | .558±.014 | .718±.036 | .745±.027 | **.758**±.029 |
| Bus | **.878**±.006 | .667±.006 | .699±.013 | .699±.034 | .513±.022 | .607±.061 | .641±.034 | **.643**±.036 | .571±.032 | .692±.023 | **.716**±.027 | .692±.019 |
| Car | .635±.050 | .703±.010 | .712±.005 | **.725**±.012 | .591±.043 | .710±.016 | .680±.015 | **.718**±.015 | .619±.017 | .660±.023 | .658±.017 | **.676**±.015 |
| Motorbike | **.861**±.017 | .705±.013 | .722±.006 | .711±.021 | .559±.042 | .711±.014 | .669±.059 | **.718**±.027 | .544±.038 | .665±.027 | .672±.014 | **.685**±.051 |
| Train | .588±.029 | .676±.034 | .733±.004 | **.735**±.039 | .549±.036 | .630±.066 | .672±.077 | **.703**±.024 | **.640**±.027 | .615±.022 | .603±.037 | .605±.024 |
| Bottle | **.765**±.020 | .611±.019 | .623±.045 | .637±.029 | .505±.009 | .664±.004 | .631±.053 | **.681**±.030 | .561±.052 | **.636**±.008 | .633±.031 | .631±.008 |
| Chair | .591±.016 | .639±.010 | .663±.016 | **.665**±.019 | .592±.037 | .659±.044 | .659±.039 | **.666**±.018 | .503±.006 | .598±.006 | **.606**±.007 | .598±.024 |
| Dining-table | **.846**±.014 | .688±.011 | .701±.009 | .696±.013 | .630±.025 | .679±.050 | .670±.058 | **.704**±.037 | .557±.059 | .663±.009 | .669±.028 | **.689**±.022 |
| Potted-plant | **.731**±.026 | .606±.008 | .616±.002 | .631±.040 | **.740**±.010 | .593±.011 | .617±.047 | .593±.006 | **.617**±.008 | .600±.016 | .601±.019 | .608±.038 |
| Sofa | **.813**±.009 | .629±.021 | .660±.013 | .668±.036 | .573±.060 | .643±.011 | .652±.024 | **.667**±.013 | .553±.054 | **.640**±.003 | .638±.010 | **.640**±.019 |
| TV | .593±.086 | .601±.002 | .590±.036 | **.678**±.019 | .554±.048 | .628±.036 | .629±.055 | **.663**±.025 | .532±.017 | **.644**±.013 | .636±.015 | .622±.029 |
| Bird | .567±.033 | .662±.037 | **.667**±.018 | .633±.018 | .558±.062 | .638±.027 | **.680**±.023 | .673±.059 | .547±.031 | .593±.037 | .578±.023 | **.607**±.061 |
| Cat | **.777**±.012 | .631±.009 | .672±.009 | .663±.028 | **.809**±.006 | .631±.025 | .648±.037 | .637±.058 | **.695**±.009 | .661±.014 | .674±.031 | .691±.019 |
| Cow | .566±.038 | .621±.026 | .621±.015 | **.712**±.040 | .551±.023 | .687±.057 | .626±.009 | **.707**±.032 | .566±.053 | .626±.009 | .601±.053 | **.641**±.017 |
| Dog | **.769**±.018 | .606±.051 | .611±.016 | .626±.034 | **.810**±.013 | .573±.011 | .638±.020 | .606±.059 | .606±.029 | .605±.011 | .572±.023 | **.613**±.002 |
| Horse | **.845**±.019 | .631±.033 | .670±.027 | .698±.010 | **.850**±.006 | .659±.020 | .645±.006 | .663±.030 | .525±.018 | .619±.008 | .609±.042 | **.641**±.010 |
| Sheep | **.821**±.011 | .641±.089 | .705±.059 | .724±.011 | **.853**±.040 | .718±.087 | .686±.011 | .692±.039 | **.750**±.000 | .692±.088 | .686±.029 | .699±.062 |
| Person | .561±.046 | .592±.011 | .599±.017 | **.622**±.012 | .590±.017 | .568±.020 | **.592**±.010 | .573±.013 | .579±.019 | .620±.010 | .625±.005 | **.627**±.007 |
| Ave. Rank | 2.15 | 3.35 | 2.50 | **1.95** | 2.95 | 2.85 | 2.45 | **1.70** | 3.30 | 2.45 | 2.60 | **1.55** |

**AUC**

| Methods / Tasks | *75% of tasks shared between teachers* | | | | *50% of tasks shared between teachers* | | | | *25% of tasks shared between teachers* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MuST | KD | CFL | Ours: VENUS | MuST | KD | CFL | Ours: VENUS | MuST | KD | CFL | Ours: VENUS |
| Aeroplane | **.870**±.019 | .817±.005 | .806±.006 | .806±.021 | **.915**±.015 | .799±.021 | .786±.029 | .823±.031 | .626±.082 | **.804**±.023 | .789±.016 | .794±.014 |
| Bicycle | .584±.067 | .597±.025 | .614±.015 | **.637**±.051 | .610±.013 | .595±.007 | .618±.010 | **.639**±.013 | .549±.045 | .605±.025 | .604±.005 | **.626**±.002 |
| Boat | **.871**±.013 | .745±.024 | .741±.016 | .736±.030 | .615±.034 | .704±.025 | **.744**±.020 | .735±.013 | .555±.010 | .733±.021 | **.762**±.026 | .746±.023 |
| Bus | **.848**±.003 | .672±.021 | .693±.013 | .687±.033 | .513±.022 | .612±.045 | **.647**±.019 | .647±.024 | .560±.018 | .681±.018 | **.697**±.006 | .691±.023 |
| Car | .636±.050 | .681±.009 | **.695**±.009 | .692±.018 | .588±.040 | .688±.016 | .669±.010 | **.690**±.006 | .609±.010 | .645±.005 | .644±.007 | **.657**±.021 |
| Motorbike | **.860**±.004 | .693±.016 | .691±.015 | .697±.031 | .560±.041 | .687±.020 | .651±.021 | **.702**±.013 | .552±.042 | .662±.031 | .663±.005 | **.679**±.046 |
| Train | .586±.030 | .705±.060 | **.752**±.020 | .728±.033 | .550±.037 | .646±.087 | .682±.083 | **.728**±.031 | .632±.013 | .628±.015 | .617±.031 | **.632**±.010 |
| Bottle | **.776**±.018 | .608±.028 | .624±.050 | .651±.028 | .505±.009 | .658±.013 | .636±.043 | **.679**±.014 | .571±.059 | .648±.017 | .643±.033 | **.652**±.006 |
| Chair | .592±.015 | .665±.014 | .685±.018 | **.686**±.014 | .592±.036 | .674±.033 | .674±.040 | **.685**±.014 | .504±.008 | .614±.010 | .614±.010 | **.616**±.024 |
| Dining-table | **.861**±.019 | .690±.007 | .709±.008 | .700±.019 | .637±.022 | .686±.039 | .677±.063 | **.707**±.031 | .567±.065 | .681±.004 | .691±.017 | **.694**±.014 |
| Potted-plant | **.765**±.021 | .615±.009 | .615±.008 | .645±.034 | **.758**±.006 | .614±.019 | .615±.044 | .607±.005 | **.640**±.016 | .612±.014 | .631±.006 | .630±.007 |
| Sofa | **.830**±.008 | .659±.024 | .681±.021 | .692±.026 | .578±.068 | .666±.021 | .663±.010 | **.693**±.013 | .553±.054 | .659±.002 | .658±.006 | **.660**±.009 |
| TV | .592±.084 | .614±.008 | .604±.025 | **.685**±.012 | .553±.047 | .639±.032 | .625±.057 | **.657**±.023 | .533±.020 | .625±.018 | **.633**±.010 | .631±.021 |
| Bird | .566±.032 | .646±.014 | **.657**±.020 | .645±.013 | .556±.061 | .623±.039 | .675±.032 | **.682**±.027 | .534±.022 | .586±.023 | .561±.019 | **.588**±.029 |
| Cat | **.813**±.012 | .652±.007 | .675±.006 | .678±.030 | **.821**±.017 | .663±.006 | .666±.016 | .650±.053 | .686±.011 | .677±.026 | .664±.024 | **.691**±.015 |
| Cow | .563±.038 | .584±.025 | .623±.003 | **.676**±.007 | .548±.021 | .651±.051 | .624±.032 | **.669**±.009 | .548±.055 | .584±.019 | .556±.021 | **.611**±.011 |
| Dog | **.785**±.021 | .611±.042 | .616±.015 | .637±.010 | **.804**±.023 | .585±.012 | .627±.017 | .597±.049 | **.613**±.024 | .601±.025 | .583±.019 | .602±.002 |
| Horse | **.846**±.013 | .639±.046 | .681±.016 | .718±.005 | **.866**±.010 | .644±.014 | .662±.020 | .682±.025 | .536±.009 | .647±.013 | .644±.029 | **.665**±.008 |
| Sheep | **.840**±.004 | .692±.100 | .751±.030 | .772±.014 | **.879**±.037 | .751±.053 | .723±.007 | .743±.028 | **.782**±.038 | .707±.072 | .710±.045 | .737±.076 |
| Person | **.599**±.038 | .561±.008 | .561±.018 | .596±.016 | **.626**±.015 | .521±.029 | .551±.018 | .525±.015 | .594±.018 | .588±.015 | .593±.005 | **.597**±.008 |
| Ave. Rank | **2.05** | 3.25 | 2.55 | 2.10 | 2.90 | 2.80 | 2.50 | **1.80** | 3.25 | 2.70 | 2.65 | **1.40** |

Table 11: Compared performance when combining disparate teachers. The fewer the teachers share the tasks, the more they contain more disparate knowledge.

number of shared tasks is varied between the teachers. With that, the fewer the teachers share the tasks, the more their specialized task sets are diverse, *i.e.* the more they have disparate knowledge.

We use the PASCAL VOC 2007 dataset which contains 20 tasks, allowing us to vary

the proportion of the tasks shared between teachers. We conduct the experiments by reducing the proportion of the number of shared tasks between the teachers from 75%, 50%, to 25%. The fewer they share reflects the increment of disparate knowledge captured between them. In all settings, we use DenseNet [45] for Teacher 1 and ResNet18 [40] for Teacher 2.

As shown in Table 11, we first notice that as the teachers share fewer tasks, MuST performs worse. This indicates that when the teachers share more tasks, they tend to capture similar knowledge and thus they provide uniform pseudo-predictions, resulting in a good performance by MuST. On the other hand, when the teachers share fewer tasks, they capture more diverse knowledge and provide less uniformity in their pseudo-predictions. As a result, MuST performs worse and worse. In contrast, KD that learns from the teachers' average logits performs better when the teachers share fewer tasks. This is because learning from the average logits could incorporate more information about predicted confidence between the teachers.

More observations on the cases where teachers provide more disparate knowledge, especially at 25% shared tasks, show that utilizing the teachers' internal representation as done by CFL and the proposed VENUS shows significant improvement in the student's performance. It is because CFL and VENUS do not rely purely on the teachers' predictions but also incorporate their final shared representation which could generalize better to the unseen data. All in all, the proposed VENUS outperforms CFL consistently. This shows that allowing the student's common feature to learn knowledge adaptively from each teacher through the teacher-specific adaptor succeeds clearly in unifying the disparate knowledge between teachers.

**Consolidating knowledge from heterogeneous teachers.** Lastly, we explore more cases of learning from teachers with heterogeneous architectures. In this experiment, we pre-train the teachers using the three popular models including DenseNet [45], VGG [95],

Teacher 1: ***DenseNet***, Teacher 2: ***VGG***

| Methods / Tasks | *Accuracy* | | | | *AUC* | | | |
|---|---|---|---|---|---|---|---|---|
| | MuST | KD | CFL | Ours: VENUS | MuST | KD | CFL | Ours: VENUS |
| Blue object | .9369±.0112 | .9370±.0036 | .9218±.0071 | **.9397**±.0029 | .9374±.0116 | .9373±.0042 | .9215±.0093 | **.9393**±.0032 |
| Green floor | .6514±.0939 | **.8253**±.0154 | .8180±.0181 | .8158±.0282 | .6520±.0943 | **.8275**±.0152 | .8176±.0193 | .8173±.0280 |
| Purple wall | .9637±.0016 | .9539±.0128 | .9660±.0037 | **.9662**±.0041 | .9633±.0020 | .9546±.0110 | **.9663**±.0030 | .9661±.0042 |
| Pink wall | .5382±.0333 | .8926±.0202 | **.8996**±.0414 | .8991±.0205 | .5382±.0333 | .8931±.0199 | .8993±.0432 | **.9002**±.0209 |
| Ave. RANK | 3.50 | 2.50 | 2.25 | **1.75** | 3.25 | 2.75 | 2.25 | **1.75** |

Teacher 1: ***DenseNet***, Teacher 2: ***AlexNet***

| Methods / Tasks | *Accuracy* | | | | *AUC* | | | |
|---|---|---|---|---|---|---|---|---|
| | MuST | KD | CFL | Ours: VENUS | MuST | KD | CFL | Ours: VENUS |
| Blue object | .9189±.0061 | .9194±.0017 | .9141±.0118 | **.9332**±.0132 | .9197±.0069 | .9195±.0025 | .9154±.0131 | **.9339**±.0125 |
| Green floor | .5649±.0279 | .8070±.0207 | .8036±.0082 | **.8090**±.0173 | .5650±.0276 | .8079±.0204 | .8034±.0102 | **.8100**±.0161 |
| Purple wall | .9563±.0020 | .9605±.0017 | **.9628**±.0023 | .9620±.0086 | .9563±.0010 | .9611±.0025 | **.9634**±.0016 | .9622±.0082 |
| Pink wall | .5002±.0003 | **.9196**±.0143 | .8981±.0101 | .8980±.0180 | .5002±.0003 | **.9193**±.0151 | .8958±.0119 | .8970±.0177 |
| Ave. RANK | 3.75 | 2.00 | 2.50 | **1.75** | 3.50 | 2.25 | 2.75 | **1.50** |

Teacher 1: ***VGG***, Teacher 2: ***AlexNet***

| Methods / Tasks | *Accuracy* | | | | *AUC* | | | |
|---|---|---|---|---|---|---|---|---|
| | MuST | KD | CFL | Ours: VENUS | MuST | KD | CFL | Ours: VENUS |
| Blue object | **.9183**±.0096 | .9081±.0039 | .9097±.0041 | .9098±.0010 | **.9168**±.0084 | .9070±.0040 | .9103±.0053 | .9076±.0039 |
| Green floor | .5313±.0257 | **.8582**±.0141 | .8500±.0092 | .8327±.0177 | .5314±.0258 | **.8566**±.0132 | .8502±.0097 | .8306±.0176 |
| Purple wall | .9334±.0061 | .9209±.0183 | .9374±.0111 | **.9490**±.0037 | .9344±.0064 | .9219±.0177 | .9374±.0107 | **.9502**±.0028 |
| Pink wall | .5052±.0062 | .9011±.0219 | .8963±.0137 | **.9057**±.0029 | .5052±.0062 | .8993±.0209 | .8946±.0127 | **.9092**±.0060 |
| Ave. RANK | 3.00 | 2.75 | 2.50 | **1.75** | 3.00 | 2.75 | 2.25 | **2.00** |

Table 12: Compared performance on more cases of combining teachers with different architectures.

or AlexNet [52]. Then we train the student from the three different pairs of these pre-trained teachers including (1) Teacher 1 is DenseNet and Teacher 2 is VGG, (2) Teacher 1 is DenseNet and Teacher 2 is AlexNet, and (3) Teacher 1 is VGG and Teacher 2 is AlexNet. In each setting, the two teachers are trained using different data from the 3D dataset and they have shared tasks at approximately 50%.

Table 12 shows all results for this experiment. We find that VENUS consistently outperforms the other methods by achieving the best average rank across the board. This ascertains that our proposed method achieves in combining knowledge across heterogeneous teachers into a high-quality common feature that could generalize to all tasks effectively. Moreover, the results show that, for all settings, MuST performs the worst.

Once again, this indicates that learning only from the teachers' pseudo-predictions is not enough for the student to learn good common feature to be used across all tasks. That is learning from the teachers' final shared representations is necessary for the multi-task student model.

## 5.7 Conclusions

We introduce the new open problem of Amalgamating Multi-Task Models with Heterogeneous Architectures (AmalMTH) and propose the first solution, named Versatile Common Feature Consolidator (VENUS). Our proposed method trains the student which improves successfully the performance of all tasks existing across all teachers without using any labeled data. The key success of VENUS is by amalgamating the rich information encoded in the shared representations among the teachers. Moreover, VENUS introduces the Feature Consolidator, which is the cutting-edge module that allows the student model to learn from heterogeneous teachers. The extensive experiments demonstrate that VENUS significantly outperforms the alternative methods by achieving the top average rank across all settings.

# 6 Conclusions

## 6.1 Summary of Contributions

We study the problem of knowledge amalgamation (KA) that aims to combine complementary knowledge captured by heterogeneous pre-trained models (*teachers*). In this dissertation, we explore four different KA tasks regarding four different aspects of complementary knowledge inherent among the teachers.

First, we focus on the pre-trained models that learn representations for instances. These teachers are trained to capture different knowledge of relationships among instances. This phenomenon exists prevalently among the available pre-trained word embedding models in the NLP domain. Therefore, in this task, our study focuses on combining the pre-trained word embedding teachers into a meta-embedding student model. We propose Similarity-Preserving Meta-Embedding (SimME) to learn high-quality word representations that can preserve word relationships captured across the teacher models. SimME consistently outperforms state-of-the-art methods by 10% on average and with up to 20% across several core metrics in 4 popular text mining tasks.

Second, we broaden the KA study for the first time to the semi-supervised setting. In this work, we define the new challenging problem of semi-supervised knowledge amalgamation (SKA) for sequence classification. We propose Teacher Coordinator (TC) to combine complementary discriminative knowledge from the pre-trained teachers that handle different sets of classes. TC overcomes the challenging issue of overconfident teacher by effectively rescaling the predicted output of disparate teachers. The comprehensive experiments demonstrate that TC significantly outperforms eight state-of-the-art alternatives on four datasets by an average of 15% in accuracy.

Third, we define the new open problem of knowledge amalgamation for multi-label classification (KA-MLC). The goal is to combine the label dependency knowledge that are

captured differently across teachers. We propose Adaptive Knowledge Transfer (ANT) that facilitates the teachers to transfer their label dependency knowledge to each other. This way each teacher is encouraged to revise its prediction based on the knowledge from more competent teacher, allowing the student to learn the new dependency between the labels that exist across teachers. Our extensive experiments on eight real-world datasets show that ANT consistently outperforms the other methods on four standard metrics for multi-label classification across all datasets.

Lastly, we extend the KA problem for multi-task learning to a more realistic setting where the teachers and student may have heterogeneous architectures. In this setting, each multi-task teacher may encode different knowledge in their representation since they aim to generalize their representation to the different sets of tasks. Our proposed method, named Versatile Common Feature Consolidator (VENUS), consists of the key module of the Feature Consolidator that enables the student to learn knowledge versatilely from each teacher through the teacher-specific adaptor. VENUS shows impressive performance by achieving the top average rank across all experiments on two multi-task datasets.

## 6.2 Future Direction

In this dissertation, we broaden some existing knowledge amalgamation problems and also define some new problem settings for knowledge amalgamation. Here, we will discuss some ideas for future studies extending to the problems in this dissertation.

- In the first task, we focus our study on static language models that provide complementary knowledge on word embeddings. However, in NLP, this study can be extended to a more challenging task in combining some pre-trained language models such as BERT [24], ELMo [79], or the GPT series [83]. This study is very challenging since there is no clear information of what knowledge is encoded in each of them.

- Since we are the first to extend KA study for multi-class classification to the semi-supervised setting, there are still more potential techniques that could be applied to the problem. For example, we can apply reinforcement learning [98] or adapt Gumbel-Softmax Sampling [47] to learn a teacher trustworthy policy.
- Some other directions can be studied in the problem of KA for multi-label classification. One interesting direction is applying Graph Neural Networks (GNNs) to learn the dependency between the labels. For this setting, the models could be similar to those in [19].
- Several ideas can be explored in our last task of multi-tasking KA. For example, since there is no ground-truth available on which layers of the teachers the student should learn from, we can apply attention techniques [106] to allow the student to extract knowledge from the best informative layers of each teacher.

## References

[1] R. Adriana, B. Nicolas, K. S. Ebrahimi, C. Antoine, G. Carlo, and B. Yoshua. Fitnets: Hints for thin deep nets. In *Proceedings of ICLR*, 2015.

[2] A. Aghajanyan, A. Gupta, A. Shrivastava, X. Chen, L. Zettlemoyer, and S. Gupta. Muppet: Massive multi-task representations with pre-finetuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5799–5811. Association for Computational Linguistics, Nov. 2021.

[3] R. J. Alcock, Y. Manolopoulos, et al. Time-series similarity queries employing a feature-based approach. In *Hellenic conference on informatics*, pages 27–29, 1999.

[4] H. M. Alonso and B. Plank. When is multitask learning effective? semantic sequence prediction under varying data conditions. *arXiv preprint arXiv:1612.02251*, 2016.

[5] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, Q. Cheng, G. Chen, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning*, pages 173–182. PMLR, 2016.

[6] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Proceedings of ESANN*, volume 3, page 3, 2013.

[7] M. Baroni, S. Evert, and A. Lenci. Esslli 2008 workshop on distributional lexical semantics. hamburg, germany: Association for logic. *Language and Information*, 2008.

[8] H. Bilen and A. Vedaldi. Integrated perception with recurrent multi-task neural networks. *Proceedings of NeurIPS*, 29, 2016.

[9] D. Bollegala and C. Bao. Learning word meta-embeddings by autoencoding. In *Proceedings of COLING*, pages 1650–1661, 2018.

[10] D. Bollegala, K. Hayashi, and K.-I. Kawarabayashi. Think globally, embed locally: locally linear meta-embedding of words. In *Proceedings of IJCAI*, pages 3970–3976, 2018.

[11] M. R. Boutell, J. Luo, X. Shen, and C. M. Brown. Learning multi-label scene classification. *Pattern recognition*, 37(9):1757–1771, 2004.

[12] F. Briggs, H. Yonghong, R. Raich, et al. New methods for acoustic classification of multiple simultaneous bird species in a noisy environment. In *IEEE International Workshop on Machine Learning for Signal Processing*, pages 1–8, 2013.

[13] C. Burgess and H. Kim. 3d shapes dataset. https://github.com/deepmind/3dshapes-dataset/, 2018.

[14] E. Carter, P. Adam, D. Tsakis, S. Shaw, R. Watson, and P. Ryan. Enhancing pedestrian mobility in smart cities using big data. *Journal of Management Analytics*, pages 1–16, 2020.

[15] R. Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[16] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[17] S.-F. Chen, Y.-C. Chen, C.-K. Yeh, and Y.-C. F. Wang. Order-free rnn with visual attention for multi-label classification. In *Proceedings of AAAI*, volume 32, pages 6714–6721, 2018.

[18] Y. Chen, B. Perozzi, R. Al-Rfou, and S. Skiena. The expressive power of word embeddings. *arXiv preprint arXiv:1301.3226*, 2013.

[19] Z.-M. Chen, X.-S. Wei, P. Wang, and Y. Guo. Multi-label image recognition with graph convolutional networks. In *Proceedings of CVPR*, pages 5177–5186, 2019.

[20] J. Coates and D. Bollegala. Frustratingly easy meta-embedding–computing meta-embeddings by averaging source word embeddings. In *Proceedings of NAACL-HLT (Short Papers)*, pages 194–198, 2018.

[21] M. Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

[22] K. Dembczynski, W. Cheng, and E. Hüllermeier. Bayes optimal multilabel classification via probabilistic classifier chains. In *Proceedings of ICML*, pages 279–286, 2010.

[23] K. Dembszynski, W. Waegeman, W. Cheng, and E. Hüllermeier. On label dependence in multilabel classification. In *ICML Workshop on Learning from Multi-label data*, pages 5–12, 2010.

[24] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[25] T. G. Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15, 2000.

[26] S. Diplaris, G. Tsoumakas, P. A. Mitkas, and I. Vlahavas. Protein classification with multiple algorithms. In *Proceedings of Panhellenic Conference on Informatics*, pages 448–456. Springer, 2005.

[27] L. Duong, T. Cohn, S. Bird, and P. Cook. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of the 53rd annual meeting of the Association for Computational Linguistics and the 7th international joint conference on natural language processing (volume 2: short papers)*, pages 845–850, 2015.

[28] A. Elisseeff and J. Weston. A kernel method for multi-labelled classification. *Advances in neural information processing systems*, 14:681–687, 2001.

[29] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[30] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller. Transfer learning for time series classification. In *Proceedings of Big Data)*, pages 1367–1376, 2018.

[31] J. Fürnkranz, E. Hüllermeier, E. L. Mencía, and K. Brinker. Multilabel classification via calibrated label ranking. *Machine learning*, 73(2):133–153, 2008.

[32] M. Ganapathibhotla and B. Liu. Mining opinions in comparative sentences. In *Proceedings of COLING*, pages 241–248, 2008.

[33] B. Gao, J. Bian, and T.-Y. Liu. Wordrep: A benchmark for research on learning word representations. *arXiv preprint arXiv:1407.1640*, 2014.

[34] Y. Gao, J. Ma, M. Zhao, W. Liu, and A. L. Yuille. 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*, pages 3205–3214, 2019.

[35] G. Ghiasi, B. Zoph, E. D. Cubuk, Q. V. Le, and T.-Y. Lin. Multi-task self-training for learning general representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8856–8865, 2021.

[36] X. Glorot, A. Bordes, and Y. Bengio. Domain adaptation for large-scale sentiment classification: A deep learning approach. In *ICML*, 2011.

[37] S. Godbole and S. Sarawagi. Discriminative methods for multi-labeled classification. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 22–30. Springer, 2004.

[38] T. Hartvigsen, C. Sen, X. Kong, and E. Rundensteiner. Recurrent halting chain for early multi-label classification. In *Proceedings of ACM SIGKDD*, pages 1382–1392, 2020.

[39] H. Harutyunyan, H. Khachatrian, D. C. Kale, G. Ver Steeg, and A. Galstyan. Multitask learning and benchmarking with clinical time series data. *Scientific data*, 6(1):1–18, 2019.

[40] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of CVPR*, pages 770–778, 2016.

[41] K. Hendrickx, L. Perini, D. Van der Plas, W. Meert, and J. Davis. Machine learning with a reject option: A survey. *arXiv preprint arXiv:2107.11277*, 2021.

[42] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[43] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[44] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[45] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[46] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep networks with stochastic depth. In *Proceedings of ECCV*, pages 646–661, 2016.

[47] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[48] G. Jocher, A. Stoken, J. Borovec, NanoCode012, ChristopherSTAN, L. Changyu, Laughing, tkianai, yxNONG, A. Hogan, lorenzomammana, AlexWang1900, A. Chaurasia, L. Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Durgesh, F. Ingham, Frederik, Guilhen, A. Colmagro, H. Ye, Jacobsolawetz, J. Poznanski, J. Fang, J. Kim, K. Doan, and L. Y. . ultralytics/yolov5: v4.0 - nn.SiLU() activations, Weights & Biases logging, PyTorch Hub integration, Jan. 2021.

[49] J. Kittler, M. Hatef, R. P. Duin, and J. Matas. On combining classifiers. *IEEE TPAMI*, 20(3):226–239, 1998.

[50] I. Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of CVPR*, pages 6129–6138, 2017.

[51] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[52] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[53] C.-W. Lee, W. Fang, C.-K. Yeh, and Y.-C. F. Wang. Multi-label zero-shot learning with structured knowledge graphs. In *Proceedings of CVPR*, pages 1576–1585, 2018.

[54] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of ICCV*, pages 2980–2988, 2017.

[55] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[56] J. Lines and A. Bagnall. Ensembles of elastic distance measures for time series classification. In *Proceedings of SDM*, pages 524–532, 2014.

[57] S. Liu, E. Johns, and A. J. Davison. End-to-end multi-task learning with attention. In *Proceedings of CVPR*, pages 1871–1880, 2019.

[58] M. Long, Z. Cao, J. Wang, and P. S. Yu. Learning multiple tasks with multilinear relationship networks. *Advances in neural information processing systems*, 30, 2017.

[59] Y. Lu, A. Kumar, S. Zhai, Y. Cheng, T. Javidi, and R. Feris. 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*, pages 5334–5343, 2017.

[60] S. Luo, X. Wang, G. Fang, Y. Hu, D. Tao, and M. Song. Knowledge amalgamation from heterogeneous networks by common feature learning. In *Proceedings of IJCAI*, pages 3087–3093, 2019.

[61] T. Luong, R. Socher, and C. Manning. Better word representations with recursive neural networks for morphology. In *Proceedings of CoNLL*, pages 104–113, 2013.

[62] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. Shroff. Timenet: Pre-trained deep recurrent neural network for time series classification. In *Proceedings of ESANN*, 2017.

[63] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[64] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin. Advances in pre-training distributed word representations. In *Proceedings of LREC*, 2018.

[65] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, pages 3111–3119, 2013.

[66] G. A. Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[67] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

[68] R. Mormont, P. Geurts, and R. Marée. Multi-task pre-training of deep neural networks for digital pathology. *IEEE journal of biomedical and health informatics*, 25(2):412–421, 2020.

[69] A. Muromägi, K. Sirts, and S. Laur. Linear ensembles of word embedding models. In *Proceedings of NoDaLiDa*, pages 96–104, May 2017.

[70] J. Nam, Y.-B. Kim, E. L. Mencia, S. Park, R. Sarikaya, and J. Fürnkranz. Learning context-dependent label permutations for multi-label classification. In *Proceedings of ICML*, pages 4733–4742, 2019.

[71] J. Nam, E. L. Mencía, H. J. Kim, and J. Fürnkranz. Maximizing subset accuracy with recurrent neural networks in multi-label classification. In *Proceedings of NeurIPS*, pages 5413–5423, 2017.

[72] S. Nawaz, A. Calefati, M. Caraffini, N. Landro, and I. Gallo. Are these birds similar: Learning branched networks for fine-grained representations. In *2019 International Conference on Image and Vision Computing New Zealand (IVCNZ 2019)*, Dec 2019.

[73] V. Nekrasov, T. Dharmasiri, A. Spek, T. Drummond, C. Shen, and I. Reid. Real-time joint semantic segmentation and depth estimation using asymmetric annotations. In *Proceedings of ICRA*, pages 7101–7107, 2019.

[74] H. F. Nweke, Y. W. Teh, M. A. Al-Garadi, and U. R. Alo. Deep learning algorithms for human activity recognition using mobile and wearable sensor networks: State of the art and research challenges. *Expert Systems with Applications*, 105:233–261, 2018.

[75] B. Pang and L. Lee. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of ACL*, pages 115–124, 2005.

[76] Y.-S. Peng, K.-F. Tang, H.-T. Lin, and E. Chang. Refuel: Exploring sparse features in deep reinforcement learning for fast disease diagnosis. In *Advances in neural information processing systems*, pages 7322–7331, 2018.

[77] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of EMNLP*, pages 1532–1543, 2014.

[78] J. Pestian, C. Brew, P. Matykiewicz, D. J. Hovermale, N. Johnson, K. B. Cohen, and W. Duch. A shared task involving multi-label classification of clinical free text. In *Proceedings of Biological, translational, and clinical language processing*, pages 97–104, 2007.

[79] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of ACL*, pages 2227–2237, 2018.

[80] M. T. Pilehvar, D. Kartsaklis, V. Prokhorov, and N. Collier. Card-660: Cambridge rare word dataset-a reliable benchmark for infrequent word representation models. In *Proceedings of EMNLP*, pages 1391–1401, 2018.

[81] W. Ping, K. Peng, A. Gibiansky, S. O. Arik, A. Kannan, S. Narang, J. Raiman, and J. Miller. Deep voice 3: Scaling text-to-speech with convolutional sequence learning. In *International Conference on Learning Representations*, 2018.

[82] S. Pyysalo, F. Ginter, H. Moen, T. Salakoski, and S. Ananiadou. Distributional semantics resources for biomedical text processing. *Proceedings of LBM*, pages 39–44, 2013.

[83] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[84] N. Razavian, J. Marcus, and D. Sontag. Multi-task prediction of disease onsets from longitudinal laboratory tests. In *Proceedings of MLHC*, pages 73–100, 2016.

[85] J. Read, B. Pfahringer, G. Holmes, and E. Frank. Classifier chains for multi-label classification. *Machine learning*, 85(3):333–359, 2011.

[86] C. Rosenberg, M. Hebert, and H. Schneiderman. Semi-supervised self-training of object detection models. In *Proceedings of IEEE WACV/MOTION*, pages 29–36, 2005.

[87] S. Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[88] S. Ruder, J. Bingel, I. Augenstein, and A. Søgaard. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4822–4829, 2019.

[89] H. Sajnani, V. Saini, K. Kumar, E. Gabrielova, P. Choudary, and C. Lopes. Classifying yelp reviews into relevant categories, 2012.

[90] J. Shao, C.-C. Loy, K. Kang, and X. Wang. Slicing convolutional neural network for crowd video understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5620–5628, 2016.

[91] M. Sharma. Multi-lingual multi-task speech emotion recognition using wav2vec 2.0. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6907–6911. IEEE, 2022.

[92] C. Shen, X. Wang, J. Song, L. Sun, and M. Song. Amalgamating knowledge towards comprehensive classification. In *Proceedings of AAAI*, pages 3068–3075, 2019.

[93] C. Shen, M. Xue, X. Wang, J. Song, L. Sun, and M. Song. Customizing student networks from heterogeneous teachers via adaptive knowledge amalgamation. In *Proceedings of ICCV*, pages 3504–3513, 2019.

[94] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[95] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of ICLR*, page 1–14, 2015.

[96] S. Singh, D. Hoiem, and D. Forsyth. Swapout: Learning an ensemble of deep architectures. In *Proceedings of NeurIPS*, pages 28–36, 2016.

[97] A. N. Srivastava and B. Zane-Ulman. Discovering recurring anomalies in text reports regarding complex space systems. In *Proceedings of IEEE aerospace*, pages 3853–3862, 2005.

[98] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[99] M. Tan, R. Pang, and Q. V. Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of CVPR*, pages 10781–10790, 2020.

[100] J. Thadajarassiri, T. Hartvigsen, W. Gerych, X. Kong, and E. A. Rundensteiner. Knowledge amalgamation for multi-label classification via label dependency transfer. In *Proceedings of AAAI*.

[101] J. Thadajarassiri, T. Hartvigsen, X. Kong, and E. A. Rundensteiner. Semi-supervised knowledge amalgamation for sequence classification. In *Proceedings of AAAI*, volume 35, pages 9859–9867, 2021.

[102] J. Thadajarassiri, C. Sen, T. Hartvigsen, X. Kong, and E. Rundensteiner. Learning similarity-preserving meta-embedding for text mining. In *Proceedings of IEEE Big Data*, pages 808–817. IEEE, 2020.

[103] K. Trohidis, G. Tsoumakas, G. Kalliris, I. P. Vlahavas, et al. Multi-label classification of music into emotions. In *Proceedings of ISMIR*, volume 8, pages 325–330, 2008.

[104] C.-P. Tsai and H.-Y. Lee. Order-free learning alleviating exposure bias in multi-label classification. In *Proceedings of AAAI*, volume 34, pages 6038–6045, 2020.

[105] G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *Data Warehousing and Mining (IJDWM)*, 3(3):1–13, 2007.

[106] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[107] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel. Decision trees for hierarchical multi-label classification. *Machine learning*, 73(2):185, 2008.

[108] J. Vongkulbhisal, P. Vinayavekhin, and M. Visentini-Scarzanella. Unifying heterogeneous classifiers with distillation. In *Proceedings of CVPR*, pages 3175–3184, 2019.

[109] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus. Regularization of neural networks using dropconnect. In *Proceedings of ICML*, pages 1058–1066, 2013.

[110] H. Wang, H. Zhao, X. Li, and X. Tan. Progressive blockwise knowledge distillation for neural network acceleration. In *Proceedings of IJCAI*, pages 2769–2775, 2018.

[111] J. Wang, Y. Chen, S. Hao, X. Peng, and L. Hu. Deep learning for sensor-based activity recognition: A survey. *Pattern Recognition Letters*, 119:3–11, 2019.

[112] Y. Wang, D. He, F. Li, X. Long, Z. Zhou, J. Ma, and S. Wen. Multi-label classification with label graph superimposing. In *Proceedings of AAAI*, volume 34, pages 12265–12272, 2020.

[113] Y. Wang, S. Liu, N. Afzal, M. Rastegar-Mojarad, L. Wang, F. Shen, P. Kingsbury, and H. Liu. A comparison of word embeddings for the biomedical natural language processing. *Biomedical Informatics*, 87:12–20, 2018.

[114] B. Wu, W. Chen, Y. Fan, Y. Zhang, J. Hou, J. Liu, and T. Zhang. Tencent ml-images: A large-scale multi-label image database for visual representation learning. *IEEE Access*, 7, 2019.

[115] W. Xue and T. Li. Aspect based sentiment analysis with gated convolutional networks. In *Proceedings of ACL*, pages 2514–2523, 2018.

[116] X. Yang, Y. Chen, H. Yu, Y. Zhang, W. Lu, and R. Sun. Instance-wise dynamic sensor selection for human activity recognition. In *Proceedings of AAAI*, pages 1104–1111, 2020.

[117] Y. Yang and T. M. Hospedales. Trace norm regularised deep multi-task learning. *arXiv preprint arXiv:1606.04038*, 2016.

[118] Z. Yang, R. Salakhutdinov, and W. Cohen. Multi-task cross-lingual sequence tagging from scratch. *arXiv preprint arXiv:1603.06270*, 2016.

[119] J. Ye, X. Wang, Y. Ji, K. Ou, and M. Song. Amalgamating filtered knowledge: learning task-customized student from multi-task teachers. In *Proceedings of IJCAI*, pages 4128–4134, 2019.

[120] W. Yin and H. Schütze. Learning word meta-embeddings. In *Proceedings of ACL*, pages 1351–1360, 2016.

[121] R. You, Z. Guo, L. Cui, X. Long, Y. Bao, and S. Wen. Cross-modality attention with semantic graph embedding for multi-label classification. In *Proceedings of AAAI*, volume 34, pages 12709–12716, 2020.

[122] S. Zhai and Z. Zhang. Semisupervised autoencoder for sentiment analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[123] M.-L. Zhang and Z.-H. Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2013.

[124] Y. Zhang, Q. Chen, Z. Yang, H. Lin, and Z. Lu. Biowordvec, improving biomedical word embeddings with subword information and mesh. *Scientific data*, 6(1):1–9, 2019.