Amalgamating Multi-Task Models with Heterogeneous Architectures

Jidapa Thadajarassiri[1], **Walter Gerych**[2], **Xiangnan Kong**[2], **Elke Rundensteiner**[2]

[1]Srinakharinwirot University,
[2]Worcester Polytechnic Institute,
jidapath@g.swu.ac.th, {wgerych,xkong, rundenst}@wpi.edu

Abstract

Multi-task learning (MTL) is essential for real-world applications that handle multiple tasks simultaneously, such as self-driving cars. MTL methods improve the performance of all tasks by utilizing information across tasks to learn a robust shared representation. However, acquiring sufficient labeled data tends to be extremely expensive, especially when having to support many tasks. Recently, Knowledge Amalgamation (KA) has emerged as an effective strategy for addressing the lack of labels by instead learning directly from pre-trained models (*teachers*). KA learns one unified multi-task *student* that masters all tasks across all teachers. Existing KA for MTL works are limited to teachers with identical architectures, and thus propose layer-to-layer based approaches. Unfortunately, in practice, teachers may have heterogeneous architectures; their layers may not be aligned and their dimensionalities or scales may be incompatible. Amalgamating multi-task teachers with heterogeneous architectures remains an open problem. For this, we design Versatile Common Feature Consolidator (VENUS), the first solution to this problem. VENUS fuses knowledge from the shared representations of each teacher into one unified generalized representation for all tasks. Specifically, we design the Feature Consolidator network that leverages an array of teacher-specific trainable adaptors. These adaptors enable the student to learn from multiple teachers, even if they have incompatible learned representations. We demonstrate that VENUS outperforms five alternative methods on numerous benchmark datasets across a broad spectrum of experiments.

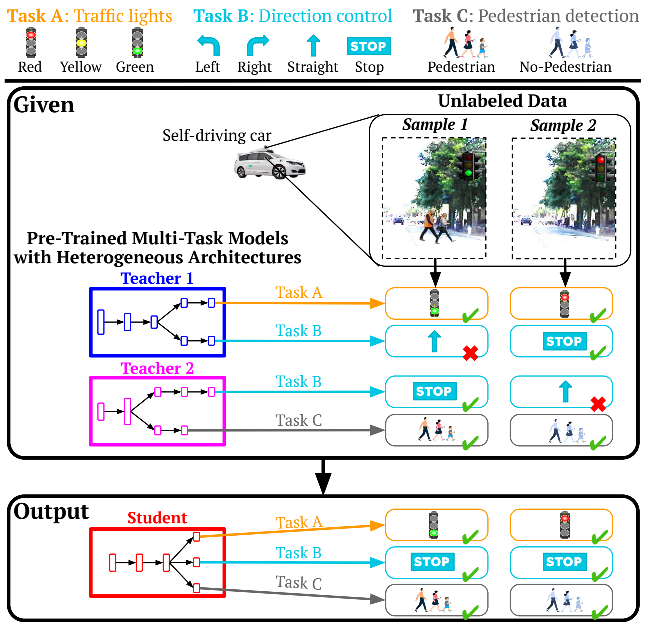


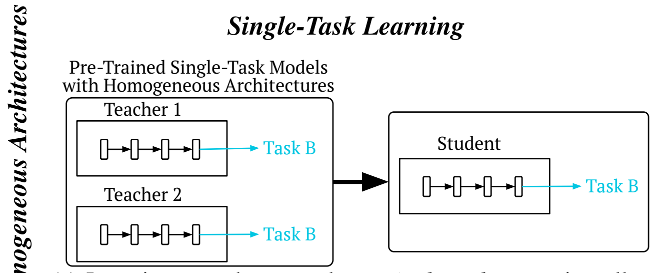
Figure 1: Amalgamating Multi-Task Models with Heterogeneous Architectures (AmalMTH). Given pre-trained multi-task models (*teachers*) and unlabeled data, the task is to train a *student* that well performs on the union of teachers' tasks.

Introduction

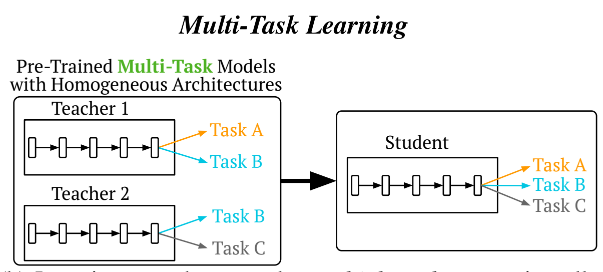
Multi-Task Learning (MTL) is the learning paradigm that aims to improve the performance of multiple tasks simultaneously (Ruder 2017). MTL models learn mutually beneficial shared representations between tasks, which tend to be more robust than the representations learned separately by single-task models (Caruana 1997). This robustness is required by real-world applications that solve multiple related tasks concurrently, *e.g.*, self-driving cars (Teichmann et al. 2018), disease detection (Zhou et al. 2011; Wan et al. 2012), and natural language understanding (Clark et al. 2019).

State-of-the-Art. MTL is an active area of research (Crawshaw 2020; Nekrasov et al. 2019; Bilen and Vedaldi 2016; Lu et al. 2017; Gao et al. 2019). However, the existing MTL works (Liu, Johns, and Davison 2019; Kokkinos 2017; Misra et al. 2016; Ruder et al. 2019) have been developed using *supervised* learning. As the number of tasks grows, training data and labeling requirements become large - making this too prohibitively expensive in practice.

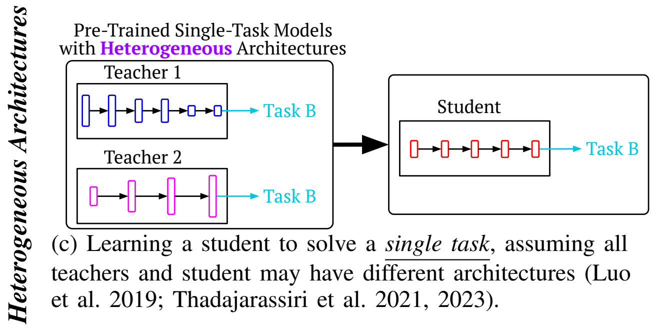
Fortunately, several organizations that utilize huge and at times private data sets and extensive compute power have released pre-trained multi-task models (Harutyunyan et al. 2019; Mormont, Geurts, and Marée 2020) for other practitioners to reuse. Since these released models are each pre-trained separately, they come with *different architectures* and tend to handle *different, though at times overlapping, sets of tasks*. However, the reuse of any individual model is limited to their pre-trained tasks; several applications, *e.g.*, self-driving cars, may need to solve a much broader task set


Homogeneous Architectures
Single-Task Learning
Pre-Trained Single-Task Models
with Homogeneous Architectures
Teacher 1
Task B
Teacher 2
Task B
Student
Task B

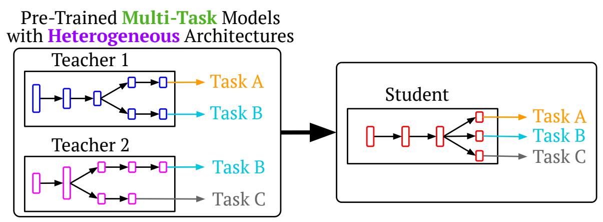
(a) Learning a student to solve a *single task*, assuming all teachers and student have identical architecture (Shen et al. 2019a).



(b) Learning a student to solve *multiple tasks*, assuming all teachers and student have identical architecture (Shen et al. 2019b; Ye et al. 2019).


Heterogeneous Architectures
Pre-Trained Single-Task Models
with Heterogeneous Architectures
Teacher 1
Task B
Teacher 2
Task B
Student
Task B

(c) Learning a student to solve a *single task*, assuming all teachers and student may have different architectures (Luo et al. 2019; Thadajarassiri et al. 2021, 2023).



(d) This paper: Learning a student to solve *multiple tasks*, assuming all teachers and student may have different architectures.

Figure 2: Comparison of related Knowledge Amalgamation (KA) problems.

covered across multiple pre-trained models. Utilizing many of these models concurrently is not ideal, due to the computational cost of using multiple models as well as the issue of potential conflicts between model predictions.

Recently, Knowledge Amalgamation (KA) (Shen et al. 2019a) has become a popular approach to combine the knowledge of multiple pre-trained models (*teachers*) into one unified compact *student* using only *unlabeled data*. The student's objective is to become a master of all tasks solved across all teachers. This unified student not only solves the potential scalability and conflict issue mentioned above but also mitigates costs in collecting the labeled data and in reusing the pre-trained teachers. Yet, an effective strategy for extracting and combining knowledge from disparate *multi-task teachers* needs to be developed. An example of how KA could be used for MTL is shown in Figure 1.

Unfortunately, as depicted in Figure 2, most existing KA works (Shen et al. 2019a; Luo et al. 2019; Thadajarassiri et al. 2021, 2023) focus on amalgamating knowledge for only a *single* task. There are few initial works that began to study KA for multiple tasks (Ye et al. 2019; Shen et al. 2019b), though they all make the unrealistic assumption that the teachers have identical architectures. This is too restrictive in practice, as models that specialize on different task sets are rarely identical.

Problem Definition. We propose to study the open problem of **Amal**gamating **M**ulti-**T**ask Models with **H**eterogeneous Architectures (AmalMTH) as illustrated in Figure 1. The goal is to train a multi-task *student* model using only *unlabeled* data and pre-trained multi-task models (*teachers*). The teachers may have different architectures and may each handle different sets of tasks. The student is trained to master the union of the teachers' task sets.

Challenges. Three challenges arise for AmalMTH:

• *No labeled data.* Traditional MTL methods are developed under the standard supervised setting, which requires labeled data for training. Without labels, these existing MTL methods are not applicable. Therefore, a solution that does not need labeled data must be developed.

• *Combining knowledge from heterogeneous architecture teachers.* Learning from the teacher's internal layers could preserve the teacher's knowledge (Ye et al. 2019; Shen et al. 2019b). However, teachers may have different architectures, with a different number and type of layers. This is challenging as there is no natural alignment between the architectures of the teachers and student, and it is unknown which layers the student may best learn from. Teachers may also exhibit different sizes and scales in each layer. Therefore, the student may be biased toward the teacher with larger scales when minimizing loss for imitating the teachers' layers.

• *Distinct knowledge captured across teachers.* Since each teacher is pre-trained separately, they typically handle different sets of tasks. Consequently, the internal representations learned by each teacher capture different information. Worst yet, when teachers share some but not all tasks in common, the different information captured among them may lead to conflicting predictions on their shared tasks. For example, both teachers in Figure 1 are trained to predict the direction control. However, for the input sample 1, Teacher 1 predicts to go straight while Teacher 2 predicts to stop. It is thus challenging for a student to combine such distinct knowledge into one integrated representation to be used across all tasks.

Proposed Method. In this work, we propose the first solution to solve the AmalMTH problem, named **Ve**rsatile Commo**n** Feat**u**re Con**s**olidator (VENUS). VENUS trains a multi-task student that combines the knowledge from multiple pre-trained multi-task teachers using only unlabeled training data. Since the teachers may provide conflicting predictions, using only their final predictive outputs will lead to contradicting signals when training the student model. We thus propose, for the first time, to train a student to also learn from the teachers' *shared* representations as the key information captured in most MTL models. This allows the multi-task student to combine the knowledge across all tasks handled by all teachers to improve generalized performance for all tasks. A major roadblock to combining the representations of heterogeneous teachers is the fact that they may have shared representations with disparate dimensionalities. VENUS overcomes this using a novel unique adaptor component for each teacher. Each adaptor facilitates the student to align its features to the given teacher's features by projecting the representations to a shared space. Therefore, it allows the student to learn from multiple teachers even when their architectures and representations are different.

Contributions. Our contributions include the following:

• We define the open problem of Amalgamating Multi-Task Models with Heterogeneous Architectures (AmalMTH): training a MTL student given heterogeneous MTL teachers and unlabeled training data.

• We design the novel **Ve**rsatile Commo**n** Feat**u**re Con**s**olidator (VENUS) strategy for AmalMTH. VENUS learns a generalized representation for all tasks by unifying the shared representations in all teachers using a Feature Consolidator and dimensionality-correcting adapters.

• We demonstrate that VENUS outperforms five alternative methods on several benchmark datasets by achieving on average the best accuracy across the board, and is consistent when the number of tasks shared by teachers varies.

Related Works

Multi-Task Learning (MTL). As described above, MTL aims to learn shared representations for related tasks to improve overall performance (Caruana 1997). Early work in MTL focused on hard parameter sharing (Caruana 1997), which learn a single model composed of numerous shared layers that ultimately split off into task-specific layers (Caruana 1997; Long et al. 2017; Liu, Johns, and Davison 2019; Yang, Salakhutdinov, and Cohen 2016; Alonso and Plank 2016). Other works utilize soft parameter sharing (Misra et al. 2016; Lu et al. 2017; Ruder et al. 2019; Gao et al. 2019), where separate model is trained for each task. To encourage sharing across tasks, they apply regularization techniques to constrain the parameters between the respective parallel layers from all models to be similar (Duong et al. 2015; Misra et al. 2016; Yang and Hospedales 2016). These methods suffer heavily from computational and/or memory inefficiency issues, requiring a huge amount of resources proportionally with the number of tasks.

Most importantly, existing MTL works have been developed using standard supervised learning. They thus require a huge amount of labeled data as the number of tasks grows. Since our target AmalMTH problem assumes no labels are available, these existing MTL methods are not applicable.

Knowledge Amalgamation (KA). KA (Shen et al. 2019a), a generalization of Knowledge Distillation (Hinton, Vinyals, and Dean 2015), follows the *teacher-student* training concept. While classic Knowledge Distillation learns a small student model to mimic the predictions of one single larger teacher model, KA combines knowledge from multiple teachers handling different tasks into a student model that learns the *union* of all teachers' tasks. As shown in Figure 2, many existing KA works (Shen et al. 2019a; Luo et al. 2019; Vongkulbhisal, Vinayavekhin, and Visentini-Scarzanella 2019; Thadajarassiri et al. 2021, 2023) study only *single-task learning.* Recent works have studied multi-task KA (Ye et al. 2019; Shen et al. 2019b), but they make the strong assumption that teachers share an identical architecture, and thus they propose dedicated layer-to-layer matching based approaches. However, as teacher models are pre-trained separately on disparate tasks, they tend to feature heterogeneous architectures. Thus, approaches are not applicable to many real-world cases.

Problem Formulation

This paper 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 $\mathbf{x}^i \in \mathbb{R}^d$. We are 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 $\mathcal{T}^j = \{T_k^j\}_{k=1}^{t_j}$. The teachers' task sets may or may not overlap with each other. For simplicity of exposition, we refer to each task as a binary classification task - though in principle any type of task is possible. Then, for each instance $\mathbf{x}^i$, the prediction from each teacher $\mathbf{M}^j$ on its specialized task 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 T_k^j associates (positive) with instance $\mathbf{x}^i$ or 0 (negative) otherwise.

Our 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, $\mathcal{T} = \{T_k\}_{k=1}^t$ 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 in $\mathcal{T}$ 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 $\mathbf{x}^i$ and henceforth drop the superscript i.

The Proposed Method: VENUS

We now describe our proposed **Ve**rsatile Commo**n** Feat**u**re Con**s**olidator (VENUS) method to solve the open AmalMTH problem. The two key principles of VENUS are *learning from robust representations* and merging knowledge from *diverse features of varying dimensionality*. The first principle is realized by our insight that the last shared layer among tasks for each teacher will be more information rich than the final representation of the model. We thus directly optimize our model to have a similar internal representation to the final shared representation across teachers. The second principle is required to merge these representations, as in general the final shared layer will be of different dimensionality across the various teachers. To this end we propose a *Feature Consolidator* that learns to project the teacher's representations along with the student's internal state into a shared information-preserving space. Together, these principles allow us to utilize richer representation from the teachers. We describe VENUS in more detail below.

Pre-Trained Teacher Models

We are given m pre-trained multi-task teachers, $\mathcal{M} = \{\mathbf{M}^j\}_{j=1}^m$. In general, MTL models can almost always be decomposed into two parts: the shared layers that learn a common feature for all tasks, and sequences of task-specific layers that learn particular task-specific features for each task (Caruana 1997; Ruder 2017). Let c^j be the number of the shared layers in the teacher $\mathbf{M}^j$ where $\mathbf{r}_j$ is the shared representation across all tasks in $\mathcal{T}^j$, which is the output of these c^j shared layers. We refer to $\mathbf{r}_j$ as the *final shared representation* of $\mathbf{M}^j$. We denote the sequence of c^j shared layers as $\{h_u^j\}_{u=1}^{c^j}$. Thus, $\mathbf{r}_j = h_{c^j}^j(\cdots(h_2^j(h_1^j(\mathbf{x}))))$

For each task $T_k^j \in \mathcal{T}^j$, we denote the number of task-specific layers branching out of $\mathbf{r}_j$ for this specific task as u_k^j. Then the task-specific layers for this particular task T_k^j are denoted by the sequence of layers: $\{h_u^j\}_{u=c^j+1}^{c^j+u_k^j}$. We use ℓ_k^j to represent the logit obtained from these task-specific layers for T_k^j, and predicted probability p_k^j is given by:

$$\ell_k^j = h_{c^j+u_k^j}^j(\cdots(h_{c^j+2}^j(h_{c^j+1}^j(\mathbf{r}_j)))) \tag{1}$$

$$p_k^j = \sigma(\ell_k^j). \tag{2}$$

The Proposed VENUS Framework

Our goal is to train one unified student model that effectively combines the shared knowledge across all tasks in the teachers' union set of tasks into the unified *common feature representation* that could generalize for better performance of all tasks simultaneously. Our proposed student model adopts an architecture composed of two main parts, namely, shared layers and task-specific layers.

Shared Layers of the Student Model In our method, we call the shared layers the *backbone model*, i.e., the c^s layers shared across all tasks in $\mathcal{T}$. We note that this backbone model can adopt any arbitrary architecture, *e.g.*, ResNet, DenseNet, VGG, or any other customized architecture. The aim of this component is to unify the common feature representation ($\mathbf{r}_s$) that benefits all tasks. We denote the sequence of these c^s shared layers as $\{h_v\}_{v=1}^{c^s}$. $\mathbf{r}_s$ is computed as $\mathbf{r}_s = H_\Theta(\mathbf{x})$, where $H_\Theta(\mathbf{x}) = h_{c^s}(\cdots(h_2(h_1(\mathbf{x}))))$ and Θ are learnable parameters.

To extract the shared knowledge across tasks, this common representation, $\mathbf{r}_s$, is trained to be similar to the final shared representation of each teacher. Thus, the loss L_C en-

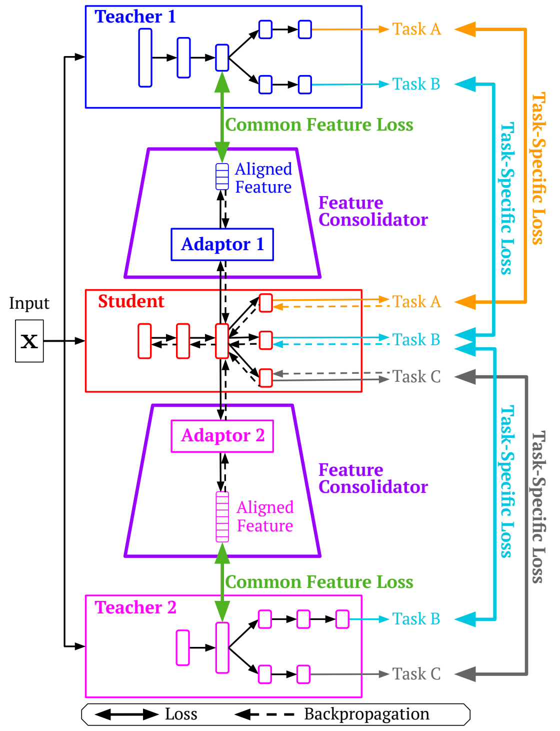


Figure 3: The architecture of our proposed method, named <u>V</u>ersatile Commo<u>n</u> Feat<u>u</u>re Con<u>s</u>olidator (VENUS).

courages the student to learn $\mathbf{r}_s$ to be similar to each $\mathbf{r}_j$:

$$L_C(\Theta) = \frac{1}{m}\sum_{j=1}^{m}\left(\mathbf{r}_s - \mathbf{r}_j\right)^2. \tag{3}$$

However, the teachers and the student may have heterogeneous architectures, meaning, $\mathbf{r}_s$ and $\mathbf{r}_j$ may be of different dimensionalities or with different supports. Thus, we cannot directly compute Equation 3. Therefore, we develop a solution to this matching challenge in the form of the Feature Consolidator strategy below.

Feature Consolidator (FC) As shown in Figure 3, FC learns to align the common feature representation $\mathbf{r}_s$ of the student with each teacher's final shared representation $\mathbf{r}_j$. For this purpose, we train adaptors for each teacher.

These adaptors enable the student to unify knowledge from heterogeneous teachers by learning to adjust their different sizes and scales through learnable parameters. For each teacher $\mathbf{M}^j$, the adaptor is a trainable network: $\hat{\mathbf{r}}_j = \text{ReLU}\left(W^j \cdot \mathbf{r}_s + b^j\right)$, where W^j and b^j are learnable parameters. Specifically, the weight matrix W^j is trained to transform the student's feature representation into the same size as the teacher's feature representation while parameter b^j is trained to adjust each value in this transformed representation to be most similar to the target representation of the teacher. Moreover, the ReLU function allows us to model non-linearity transformations with efficient computational costs. Using the output from the adaptor, $\hat{\mathbf{r}}_j$, the common feature loss function in Equation 3 is modified to:

$$L_C(\Theta) = \frac{1}{m}\sum_{j=1}^{m}\left(\hat{\mathbf{r}}_j - \mathbf{r}_j\right)^2. \tag{4}$$

These adaptors are only required when computing loss. After training they can be removed.

Task-Specific Layers of the Student Model The task-specific layers for each task $T_k \in \mathcal{T}$ in the student model are branched out of the common feature representation $\mathbf{r}_s$. These task-specific layers aim to learn the specific information for each task given the generalized knowledge $\mathbf{r}_s$ so to be able to make a final prediction for the individual task T_k. Let v_k be the number of the task-specific layers for the task T_k and ℓ_k be the logit for this task. We learn ℓ_k as:

$$\ell_k = H_{\Theta_k}(\mathbf{r}_s);\ H_{\Theta_k}(\mathbf{r}_s) = h_{c^s+v_k}(\cdots(h_{c^s+2}(h_{c^s+1}(\mathbf{r}_s)))) \tag{5}$$

where Θ_k are the learnable parameters specific for the task T_k. Then the predicted probability for the task T_k, denoted by q_k, is calculated by applying the sigmoid function (σ) to the corresponding logit ℓ_k. The final prediction is obtained by binarizing q_k with a threshold of 0.5. Thus, $\hat{y}_k = 1$ if $q_k > 0.5$ and otherwise $y_k = 0$. Let $\mathcal{L}_k$ be the set of logits for task T_k gathered from all teachers specializing on T_k. The consensus predicted probability for the task T_k, denoted as p_k, is obtained by applying the sigmoid function (σ) on the average of the logits in $\mathcal{L}_k$.

$$\forall \ell_a \in \mathcal{L}_k,\ p_k = \sigma\Big(\frac{1}{|\mathcal{L}_k|}\sum_{a=1}^{|\mathcal{L}_k|}\ell_a\Big). \tag{6}$$

For each task T_k, the 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 T_k. That is the parameters Θ_k are trained to minimize the task-specific loss for the task T_k as:

$$L_T(\Theta_k) = -p_k \log(q_k). \tag{7}$$

Procedure for Training the Student Model

The student is trained to combine the common knowledge across all teachers for all tasks and also imitate the teachers' consensus predictions simultaneously. Let ω denote all trainable parameters used for the overall training process, i.e., this includes Θ, Θ_k, W^j, and b^j for all tasks in $\mathcal{T}$ across all teachers in $\mathcal{M}$. These parameters are optimized by minimizing the final loss:

$$L(\omega) = \frac{1}{m}\sum_{j=1}^{m}\left(\hat{\mathbf{r}}_j - \mathbf{r}_j\right)^2 - \frac{1}{t}\sum_{k=1}^{t}\left(p_k \log(q_k)\right). \tag{8}$$

Experimental Study

Our method, datasets, and all experimental details are available at https://github.com/jida-thada/VENUS.

Datasets

We follow the recent KA works on multi-task learning (Ye et al. 2019; Shen et al. 2019b) by handling each class label in each dataset as an independent binary classification task.

• *PASCAL VOC 2007 (Everingham and Winn 2010)* has 9,963 images. Each image can have up to 20 object-type labels corresponding to 20 different predicting tasks.

• *3D* contains four tasks extracted from the 3d-shapes dataset (Burgess and Kim 2018). 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.

• *CIFAR-10* (Krizhevsky 2009) consists of 60,000 images. Each image is annotated with 10 class labels, leading to 10 binary classification tasks.

Compared Methods

We compare VENUS against two baselines and three KA methods from the literature that we adapt for AmalMTH:

Baseline Methods:

• *Teachers*: The pre-trained MTL teachers are used as is, each handles only a partial subset of the student's tasks.

• *Single-Task CFL* (Luo et al. 2019): As proposed in (Luo et al. 2019), each task has its own separate model trained by the CFL method for heterogeneous teachers. It trains the student to imitate the teachers' logits and also their last layers before the logits that are mapped into a common space.

Multi-Task KA Methods:

• *MuST* (Ghiasi et al. 2021): This method follows the idea of pseudo labeling from (Ghiasi et al. 2021) to train a student that imitates the pseudo-predictions generated by the teachers. For the shared tasks between teachers, it learns from the pseudo-predictions from all teachers with equal weights.

• *KD* (Hinton, Vinyals, and Dean 2015): The student is trained using the Knowledge Distillation paradigm by learning to imitate the average of all teachers' logits.

• *Multi-Task CFL* (Luo et al. 2019): We adapt the CFL method proposed for Single-Task KA to the multi-task setting. This solution learns from the teachers' average logits and here we also apply our proposed principle of learning from the final shared representations of all teachers.

Implementation Details

In each experiment, the dataset is randomly split into 70% for training the teachers, 20% for training the student, and 10% for testing. Since in our setup the data for each task would tend to suffer from a significant class imbalance as the majority of instances would belong to the negative class, we down-sample to obtain balanced datasets. In each experiment, the teachers may have a different number of shared tasks as described in the next section. The choice of the shared tasks is randomly assigned. The remaining tasks are randomly spit into the teachers' specialized task sets. The student is trained on unlabeled data to handle the union of the teachers' specialized tasks. Each experiment is replicated three times with different seeds. We report the mean and standard deviation of accuracy. Our model is written in PyTorch and optimized using Adam (Kingma and Ba 2015).

Methods	*Baseline Methods*			*Multi-Task KA Methods*			
Tasks	Teacher 1	Teacher 2	Single-Task CFL	MuST	KD	Multi-Task CFL	Ours: VENUS
Dataset: PASCAL VOC 2007							
Airplane	NA	.7552±.0335	.6852±.0248	**.9182**±.0054	.7862±.0288	.7642±.0163	.8176±.0393
Bicycle	.6061±.0167	NA	.6024±.0154	.6079±.0120	.6064±.0275	**.6333**±.0218	.6302±.0146
Boat	.7564±.0392	NA	.7009±.0369	.6182±.0364	.6939±.0379	.7334±.0139	**.7394**±.0138
Bus	.6901±.0396	NA	**.6534**±.0093	.5128±.0222	.6068±.0605	.6410±.0339	.6432±.0364
Car	NA	.6399±.0238	.6372±.0264	.5914±.0427	.7098±.0156	.6796±.0146	**.7175**±.0150
Scooter	.6794±.0414	NA	.6818±.0186	.5586±.0420	.7106±.0138	.6685±.0593	**.7180**±.0271
Train	NA	.6608±.0204	.6115±.0088	.5490±.0362	.6299±.0663	.6716±.0765	**.7034**±.0236
Bottle	NA	.6031±.0378	.6102±.0096	.5053±.0092	.6640±.0040	.6307±.0528	**.6813**±.0300
Chair	NA	.5918±.0394	.5821±.0205	.5919±.0367	.6593±.0437	.6587±.0386	**.6655**±.0183
Table	.6977±.0300	NA	.6380±.0200	.6297±.0253	.6786±.0496	.6702±.0578	**.7036**±.0371
Planter	.6231±.0338	.6014±.0335	.5831±.0097	**.7402**±.0104	.5932±.0114	.6168±.0470	.5932±.0060
Sofa	.6534±.0577	NA	.6324±.0298	.5735±.0601	.6434±.0112	.6523±.0244	**.6667**±.0135
TV	.6293±.0383	NA	.5955±.0199	.5542±.0481	.6278±.0360	.6292±.0547	**.6625**±.0254
Bird	NA	.5617±.0169	.5456±.0284	.5578±.0619	.6378±.0269	**.6800**±.0231	.6733±.0592
Cat	.6842±.0341	.6473±.0331	.6474±.0207	**.8090**±.0056	.6311±.0254	.6479±.0374	.6367±.0577
Cow	NA	.5626±.0518	.5603±.0134	.5505±.0232	.6869±.0574	.6263±.0088	**.7071**±.0315
Dog	.5756±.0084	.6071±.0349	.5708±.0156	**.8100**±.0133	.5734±.0105	.6375±.0201	.6061±.0594
Horse	.6675±.0802	NA	.6418±.0241	**.8496**±.0057	.6585±.0204	.6455±.0057	.6634±.0295
Sheep	.6917±.0703	.7453±.0459	.6180±.0061	**.8526**±.0400	.7179±.0867	.6859±.0111	.6923±.0385
Person	NA	.5851±.0115	.5811±.0431	.5902±.0172	.5682±.0201	**.5919**±.0104	.5728±.0128
Ave. RANK	NA	NA	4.05	3.45	3.05	2.55	**1.85**
Dataset: 3D							
Blue object	.7392±.0057	NA	.7450±.0089	.6428±.1193	.7402±.0159	.7139±.0073	**.7493**±.0014
Green floor	NA	.8247±.0012	.8358±.0065	.5542±.0442	.8338±.0036	.8362±.0010	**.8392**±.0034
Purple wall	.8801±.0024	.9434±.0180	.9726±.0024	.9703±.0025	.9723±.0050	.9761±.0027	**.9784**±.0024
Pink wall	NA	.9376±.0188	.9712±.0024	.6604±.1472	.9750±.0046	.9747±.0055	**.9787**±.0023
Ave. RANK	NA	NA	3.00	5.00	3.25	2.75	**1.00**
Dataset: CIFAR-10							
Airplane	NA	.7136±.0158	.7122±.0067	.5789±.0527	.8128±.0104	.7975±.0096	**.8170**±.0139
Automobile	.6933±.0117	NA	.7103±.0097	.5453±.0049	.8434±.0029	.8281±.0148	**.8509**±.0113
Bird	.6200±.0107	NA	.5930±.0021	.5422±.0351	.6970±.0054	.6936±.0224	**.7031**±.0093
Cat	NA	.6167±.0084	.6031±.0167	.5283±.0184	.7095±.0129	**.7314**±.0140	.7253±.0043
Deer	.6253±.0027	NA	.6014±.0276	.5564±.0342	.7286±.0169	.7153±.0102	**.7484**±.0101
Dog	.6631±.0056	NA	.6572±.0175	.5136±.0138	.7581±.0165	.7381±.0286	**.7633**±.0079
Frog	NA	.6719±.0097	.6603±.0117	.5050±.0047	.8056±.0138	.7967±.0217	**.8144**±.0167
Horse	.6511±.0155	.6203±.0089	.6336±.0180	**.8200**±.0080	.7844±.0179	.7883±.0036	.7747±.0281
Ship	.7505±.0118	.7345±.0046	.7505±.0107	**.8686**±.0048	.8503±.0122	.8547±.0043	.8645±.0086
Truck	NA	.6908±.0142	.6811±.0056	.5570±.0472	**.8092**±.0122	.7947±.0208	.8072±.0042
Ave. RANK	NA	NA	4.20	4.20	2.30	2.70	**1.60**

Table 1: Compared performance on the three benchmark datasets. Ave. RANK shows the overall performance across all tasks.

Experimental Results

We report the accuracy of all tasks. To show the overall performance for a dataset, we follow (Thadajarassiri et al. 2023) by reporting *the average rank* of all compared methods across all tasks, where 1 indicates the best performance. For a fair comparison, we use ResNet18 (He et al. 2016) as the backbone model for the student in all experiments.

Effectiveness of VENUS in learning a high quality common feature representation. We first investigate how effective our proposed method is 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 (Huang et al. 2017) for Teacher 1 and ResNet18 (He et al. 2016) for Teacher 2. Each of them is trained on approximately the same number of tasks with roughly 30% of their tasks shared.

The results, as demonstrated in Table 1, show that the proposed VENUS outperforms alternative methods significantly as it reaches the best average accuracy across all tasks for all datasets. First, we observe that the multi-task KA methods generally outperform the two baselines of using the teachers as is and the Single-Task KA. This means the multi-task KA methods succeed in utilizing information across each task in order to improve the performance of all tasks

Methods / Tasks	*Baseline Methods*			*Multi-Task KA Methods*			
	Teacher 1	Teacher 2	Single-Task CFL	MuST	KD	Multi-Task CFL	Ours: VENUS
Teacher 1: *DenseNet*, Teacher 2: *VGG*							
Blue object	.8913±.0286	.8337±.0097	.9379±.0021	.9369±.0112	.9370±.0036	.9218±.0071	**.9401**±.0022
Green floor	.8076±.0148	NA	.8088±.0008	.6514±.0939	**.8253**±.0154	.8180±.0181	.8205±.0204
Purple wall	.9037±.0577	.8516±.0208	**.9695**±.0015	.9637±.0016	.9539±.0128	.9660±.0037	.9685±.0006
Pink wall	NA	.8421±.0189	.8824±.0073	.5382±.0333	.8926±.0202	.8996±.0414	**.9045**±.0204
Ave. RANK	NA	NA	2.75	4.50	3.00	3.25	**1.50**
Teacher 1: *DenseNet*, Teacher 2: *AlexNet*							
Blue object	.8791±.0247	.8188±.0025	.9138±.0037	.9189±.0061	.9221±.0042	.9141±.0118	**.9332**±.0132
Green floor	.7971±.0133	NA	.8087±.0041	.5649±.0279	.8058±.0204	.8036±.0082	**.8090**±.0173
Purple wall	.8695±.0605	.8831±.0035	.9564±.0043	.9563±.0020	.9605±.0017	**.9628**±.0023	.9620±.0086
Pink wall	NA	.8272±.0012	.8810±.0019	.5002±.0003	**.9165**±.0188	.8981±.0101	.8980±.0180
Ave. RANK	NA	NA	3.75	4.50	2.25	2.75	**1.75**
Teacher 1: *VGG*, Teacher 2: *AlexNet*							
Blue object	.9061±.0007	.8263±.0144	.9086±.0015	**.9183**±.0096	.9081±.0039	.9097±.0041	.9098±.0010
Green floor	.8121±.0018	NA	.8364±.0037	.5313±.0257	**.8582**±.0141	.8500±.0092	.8327±.0177
Purple wall	.9375±.0017	.8690±.0246	.9254±.0027	.9334±.0061	.9209±.0183	.9374±.0111	**.9490**±.0037
Pink wall	NA	.8319±.0171	.8896±.0084	.5052±.0062	.9011±.0219	.8963±.0137	**.9057**±.0029
Ave. RANK	NA	NA	3.75	3.50	3.25	2.50	**2.00**

Table 2: Compared performance on more cases of combining teachers with heterogeneous architectures.

simultaneously. We notice that the Single-Task KA method barely shows an improvement over the performance of the baseline teachers. This Single-Task KA method may suffer from having not enough data to train each model separately.

We observe that MuST shows consistently the worst performance. This suggests that using only the pseudo-predictions from the teachers cannot provide enough information for the student to learn high-quality features to be used across all tasks. In all settings, we see that the Multi-Task CFL and VENUS clearly outperform MuST, indicating that incorporating the knowledge from teachers' final shared representations could lead the student to learn better common features that are generalizable across all tasks. However, we notice that unlike VENUS that clearly performs better than KD, Multi-Task CFL does not show a significantly superior performance over KD. This implies although both Multi-Task CFL and VENUS utilize more information from the teachers' final shared representations, our VENUS features a more successful strategy for fusing such knowledge achieved through the Feature Consolidator.

Further investigating heterogeneous teachers. We explore more cases of learning from teachers with heterogeneous architectures. We pre-train the teachers using three popular models with distinct architectures: DenseNet (Huang et al. 2017), VGG (Simonyan and Zisserman 2015), or AlexNet (Krizhevsky, Sutskever, and Hinton 2017). Then we train the student from different combinations of these pre-trained heterogenous teachers as shown in Table 2. In each setting, the two teachers are trained using different data from the 3D dataset and they have 50% of shared tasks.

In Table 2, we observe that VENUS consistently outperforms the other methods, achieving the best average rank. Thus, VENUS succeeds to combine knowledge across heterogeneous teachers into a high-quality common feature representation that generalizes effectively to all tasks. The results show that Multi-Task CFL and our VENUS clearly outperform MuST and KD. This indicates that learning from the teachers' final shared representations is effective for training the multi-task student model. VENUS consistently shows superior performance over Multi-Task CFL. This is likely achieved due to VENUS's strategy of fusing knowledge from teachers even with heterogeneous architectures.

Conclusion

We introduce the new problem of Amalgamating Multi-Task Models with Heterogeneous Architectures (AmalMTH) and propose the first solution, named Versatile Common Feature Consolidator (VENUS). Our method trains a multi-task student to improve the performance of all tasks across all teachers without using labeled data. VENUS amalgamates rich information encoded in the teachers' representations. VENUS introduces a Feature Consolidator that allows the student model to learn from teachers with heterogeneous architectures. Our experiments demonstrate that VENUS significantly outperforms all alternative methods by achieving the top average accuracy across all tasks in all settings.

Acknowledgements

This research was supported by NSF under IIS-1910880, CSSI-2103832, and NRT-HDR-1815866. We also thank all members of the DAISY research group at WPI.

References

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

Bilen, H.; and Vedaldi, A. 2016. Integrated perception with recurrent multi-task neural networks. In *Proceedings of NeurIPS*, volume 29.

Burgess, C.; and Kim, H. 2018. 3D Shapes Dataset. https://github.com/deepmind/3dshapes-dataset/. Accessed: 2023-01-13.

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

Clark, K.; Luong, M.-T.; Khandelwal, U.; Manning, C. D.; and Le, Q. 2019. BAM! Born-Again Multi-Task Networks for Natural Language Understanding. In *Proceedings of ACL*, 5931–5937.

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

Duong, L.; Cohn, T.; Bird, S.; and Cook, P. 2015. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. In *Proceedings of ACL*, 845–850.

Everingham, M.; and Winn, J. 2010. The PASCAL visual object classes challenge 2007 (VOC2007) development kit. *Int. J. Comput. Vis*, 88(2): 303–338.

Gao, Y.; Ma, J.; Zhao, M.; Liu, W.; and Yuille, A. L. 2019. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *Proceedings of CVPR*, 3205–3214.

Ghiasi, G.; Zoph, B.; Cubuk, E. D.; Le, Q. V.; and Lin, T.-Y. 2021. Multi-task self-training for learning general representations. In *Proceedings of ICCV*, 8856–8865.

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

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

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

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of CVPR*, 4700–4708.

Kingma, D. P.; and Ba, J. 2015. Adam: A method for stochastic optimization. In *Proceedings of ICLR*.

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

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. *Master's thesis, University of Tront*.

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

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

Long, M.; Cao, Z.; Wang, J.; and Yu, P. S. 2017. Learning multiple tasks with multilinear relationship networks. In *Proceedings of NeurIPS*, volume 30.

Lu, Y.; Kumar, A.; Zhai, S.; Cheng, Y.; Javidi, T.; and Feris, R. 2017. Fully-adaptive feature sharing in multi-task networks with applications in person attribute classification. In *Proceedings of CVPR*, 5334–5343.

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

Misra, I.; Shrivastava, A.; Gupta, A.; and Hebert, M. 2016. Cross-stitch networks for multi-task learning. In *Proceedings of CVPR*, 3994–4003.

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

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

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

Ruder, S.; Bingel, J.; Augenstein, I.; and Søgaard, A. 2019. Latent multi-task architecture learning. In *Proceedings of AAAI*, volume 33, 4822–4829.

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

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

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

Teichmann, M.; Weber, M.; Zoellner, M.; Cipolla, R.; and Urtasun, R. 2018. Multinet: Real-time joint semantic reasoning for autonomous driving. In *Proceedings of IEEE intelligent vehicles symposium (IV)*, 1013–1020.

Thadajarassiri, J.; Hartvigsen, T.; Gerych, W.; Kong, X.; and Rundensteiner, E. 2023. Knowledge Amalgamation for Multi-Label Classification via Label Dependency Transfer. In *Proceedings of AAAI*, volume 37, 9980–9988.

Thadajarassiri, J.; Hartvigsen, T.; Kong, X.; and Rundensteiner, E. 2021. Semi-Supervised Knowledge Amalgamation for Sequence Classification. In *Proceedings of AAAI*, volume 35, 9859–9867.

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

Wan, J.; Zhang, Z.; Yan, J.; Li, T.; Rao, B. D.; Fang, S.; Kim, S.; Risacher, S. L.; Saykin, A. J.; and Shen, L. 2012. Sparse Bayesian multi-task learning for predicting cognitive outcomes from neuroimaging measures in Alzheimer's disease. In *Proceedings of CVPR*, 940–947.

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

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

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

Zhou, J.; Yuan, L.; Liu, J.; and Ye, J. 2011. A multi-task learning formulation for predicting disease progression. In *Proceedings of ACM SIGKDD*, 814–822.