A Survey on Multi-Task Learning Methods

Anusha Dasari
Master of Science in
Analytics
Northeastern University
Boston, USA
dasari.an@northeastern.edu

Abstract—**The objective of this project is to explore and analyze popular loss mixing and batch mixing approaches in the Multi-Task Learning while also introducing a new loss mixing method for joint learning based on multi-armed bandits with Upper Confidence Bound.**

Keywords—Multi-Task Learning, AUTOSEM, DWA, UCB, loss mixing, batch mixing

I. Introduction

Multi-Task Learning (MTL) has been used successfully across all applications of machine learning, from computer vision, speech recognition, and natural language processing. There are two popular variants of a MTL problem, joint learning where all the related tasks are trained together, and alternate learning where an alternate sequence of batches from different tasks are used to train a deep neural network to improve the overall performance. In joint training, one of the key problems is to properly mix the loss functions of different tasks without prior knowledge about the relationship between the tasks. There could be a subset of tasks which could negatively impact the learning thus leading to the overall bad performance. The objective of this project is to explore and analyze some of the popular methods to mix the loss functions in joint learning namely uniform mixing method and Dynamic Mixing Average method(DWA[1]) while also introducing a new method that mixes the loss functions using multi-armed bandits with upper confidence bound (UCB). Further, the performance of the new mixing method is compared with DWA, Uniform Mixing approach, and also the popular alternate training method AUTOSEM[2].

II. Related work

Multi-Task Learning, known for improving generalization performance of a primary task or all tasks. These methods are very well-suited to the fields of computer vision where making robust predictions is crucial. However, without knowing the relationships between the tasks, obtaining the state-of-the-art performance is not possible. One simple but very inefficient approach to learn the relationships between the tasks is to perform grid search to find the best mixing parameters. The grid search is computationally expensive and will not often find the best parameters. One of the methods that have success in joint learning is to weigh multiple loss functions by considering the homoscedastic uncertainty of each task[3]. Another popular method is GradNorm[4] that automatically balances training in MTL networks by dynamically tuning gradient norms. But it is difficult to computationally expensive to apply GradNorm as it requires access to the network's internal gradient. Other related works including Ruder and Plank(2017)[5] explored data selection where only the samples that can improve the performance of the network are selected for training. In this project, the emphasis is to analyze simpler but efficient weighing methods like Dynamic Weight Average, and the newly proposed method which updates the weights of mixing parameters by Multi-armed bandits with UCB and compare their performances.

III. Multi-task learning weighing strategies

Following are the different approaches applied to train a Multi-Task network to solve an Multi-Task image classification problem. The Uniform Mixing, DWA, and Multi-Armed bandits with UCB are applied in the context of adjusting weights of losses in joint training. Whereas, AUTOSEM approach is used to learn the best mixing ratio of batches of a task for alternate training. The loss function of the network in joint training is defined as the weighted average of all the losses and given by the below equation.

$$L_{\text{tot}}(X, Y_{1:K}) = \sum_{i=1}^{K} \lambda_i \, l(x_i, y_i)$$

Where λ_i is the weight of loss for a task ***i*** in K total tasks.

A. Uniform Mixing of Loss Functions

This method is arguably the simplest approach to solving a MTL problem where the loss functions of all the tasks are weighted equally and the weights lambda are set to 1. By doing so, all the tasks are given the equal preference and the trained network's overall average performance gets better. This method may not be particularly useful when the objective is to improve the performance of a primary task with the help of auxiliary tasks.

B. Dynamic Weight Average

This approach is inspired by the GradNorm but it is a relatively simpler yet very effective adaptive weighting method. It requires the numerical task loss at (t - 1) and (t - 2) iterations to calculate the relative descending rate then these rates are used to calculate the softmax ratios for all the tasks. The following equation defines the DWA to be performed at each epoch:

$$\lambda(t) := \frac{K \, exp(w_k(t-1)/T)}{\Sigma_i \, exp(w_i(t-1)/T)}$$

$$w_k(t-1) = L_k(t-1) / L_k(t-2)$$

Here w_k captures the relative descent rate in the range $(0, +\infty)$, t is the index of an iteration L_k denotes the loss task of k. T is the temperature parameter that controls the softness of task weighting. At iterations $t = 1, 2$ the value of w_k is set to 1.

C. AUTOSEM

It is a two step process to improve the accuracy of a primary task by alternatively training the batches of auxiliary tasks with the batches of a primary task. In the first stage a multi-armed bandit controller is used for task selection where each arm represents a candidate auxiliary task. The usefulness of each auxiliary task is evaluated using the improvement in the performance of the primary task on the validation set. For the second stage, a subset of auxiliary tasks with the highest utilities are selected and the Gaussian process controller is used to estimate the mixing ratio of batches. This method has been proven to improve the performance of a primary task and the evaluation on the GLUE dataset surpassed the baseline performance.

For a MTL setup where there are N set of tasks and corresponding datasets $\{D_1,D_2,D_3,...,D_N\}$ and a multi-armed bandit with N arms is considered. The bandit controller selects the best task to optimize from all the tasks based on the expected future payoff. After each task selection, a deep neural network(whose parameters are shared by all the tasks) is optimized to minimize the loss of the selected task. Then the validation loss of the primary task is calculated, if the validation loss reduces from the previous iteration then the selected auxiliary task is considered to be positively contributing to the training. For each task, a beta distribution is maintained and after each step a reward of +1 for the primary task improvement other reward 0 is given. These rewards are used to update the beta distribution of all the tasks. After several iterations, a subset(of size s) of highest utility tasks are sampled using the beta distributions. The subset of selected tasks are used in the second stage of AUTOSEM to learn the batch mixing ratio.

In the second stage, a Gaussian Process(GP) model is learned on the mixing ratio x_i $(\eta_1 : \eta_2 : \eta_3 : \eta_s)$ and the y_i (performance of the network on the primary task's validation set). The covariance function or the kernel is defined by a function that measures the nearness of any two points in the Gaussian Process. For this particular model, a Matern Kernel is used. After the GP model is fit on the dataset, the best point that increases the utility is sampled. The best point in this context is a s dimensional vector that contains the batch mixing ratio to be used for the next step of training.The best point is again evaluated based on the performance of primary task's validation loss and then the GP is re-trained with the new datapoint. This method is performed iteratively until the GP converges to the mixing ratio that yields better results.

D. Multi-Armed Bandits With UCB for Adaptive Weighting

This is a new adaptive weighting method for joint learning that was experimented for this project which improved the accuracy of the primary task. In this method a multi-armed bandit with number arms is set to twice the number of tasks. For each task, the weightage of its loss is controlled by two arms - the first arm increases the weightage by +0.005 and the second arm decreases the weightage by -0.005. For each action, a Q value is maintained which gets updated by the average of the history of rewards it received. During the training process an action with the highest Q-value is selected, if there is more than one highest Q-value then the action will be selected randomly from the highest Q-valued actions. When an action is chosen, the new loss function characterized by the updated weights is used to optimize the neural network. Then the performance of the primary task on the validation set is used to generate a reward for the chosen action. If the validation loss of the primary task decreases from the previous iteration then a reward of +1 is added to the history of rewards for the action otherwise a reward of zero is given to the action. After multiple iterations, the Q-values of all the actions will be converged to the true Q-values.

Initially, all the Q-values will be set to zero and to balance the exploration and exploitation Upper Confidence Bound(UCB) method is used. The UCB method boosts the Q-value of underexplored actions by adding a term that is controlled by the number of times that particular action has been chosen in the past. The Q values + UCB are defined by the following equation:

$$A_t = \underset{a}{\operatorname{argmax}} \left(Q_t(a) + c\sqrt{\frac{\ln t}{N_t(a)}} \right)$$

Where A_t is the best action, $Q_t(a)$ is the Q-value of an action at timestep t, and c is the exploration parameter that increases the degree of exploration, $N_t(a)$ is the number of times an action a has been selected prior to timestep t. It can be seen from the equation that if an action is unexplored then its Q-value will be zero and it will be picked in the current timestep.

Using the above explained strategy, weightages of the loss functions are tuned. This is a relatively simpler method to achieve similar or sometimes even better performance to that of AUTOSEM.

IV. Experimental Setup

A. Dataset

The methods mentioned in the previous section are evaluated on an image dataset that consists of 409 images of game characters. Each image has 26 different one hot encoded labels and these labels are classified in 5 different classification tasks. Following are the descriptions of the tasks

- Task 1: Predict the gender of a character.
- Task 2: Predict the region of a character.
- Task 3: Predict the fighting style of a character.
- Task 4: Predict the alignment(ex., evil, neutral) of a character.
- Task 5: Predict the major color in a picture.

B. Network Architecture

The deep neural network used for training these tasks is a hard parameter sharing based network. The pretrained ResNet-50 is used for the initial layers and the weights of the network are frozen from updating the weights and the last layer so the ResNet-50 is modified to have 512 units along with an extra layer that is fully connected to the ResNet-50's final layer. For each task, a branch is created with a single linear layer and the softmax activation. These branches do not share parameters with each other. A learning rate of 0.001, batch size of 16 data points is used to update the weights using Adam optimizer.

C. *AUTOSEM*

For the first stage, all the auxiliary task beta distributions are initialized to Alpha = 1 and Beta = 1. To ensure that the primary task is sampled more often, higher priors Alpha=10 and Beta=2 are used. For the GP, GaussianProcessRegressor and Matern kernel from the sklearn library were used. While sampling the next best point(batch mixing ratio) 100 iterations were performed.

D. *Multi-Armed Bandits with UCB*

This method required the least number of parameters tbe manually selected. The exploration constant was set to 1 and the number of arms are set to 10 as there are 5 different tasks to be controlled.

V. Results

The Uniform Mixing of loss functions is applied to the above mentioned problem setup and the network was trained for 50 epochs. The network was able achieve the decent overall accuracy for all the tasks. Following plot shows the validation loss of all the tasks is reducing as the number of epochs increase

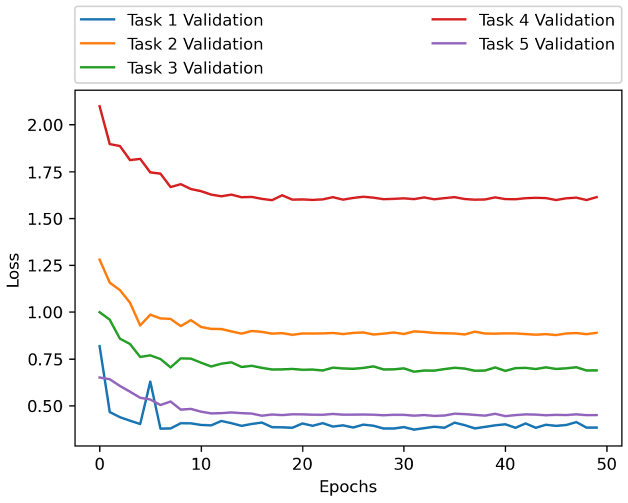

Fig. 1. Loss functions of 5 tasks with Uniform Mixing.

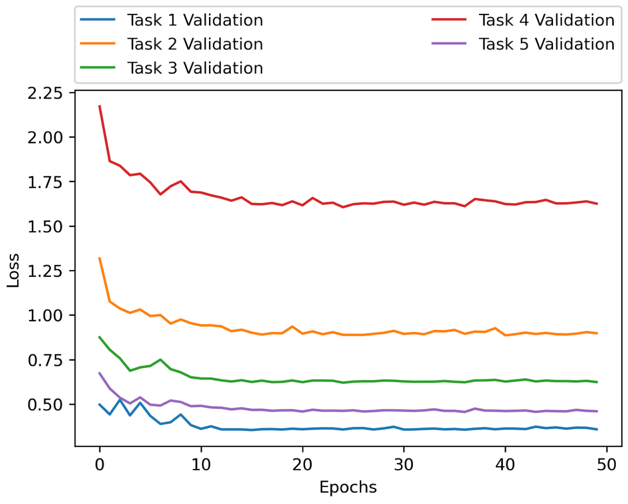

Fig. 2. Loss functions of 5 tasks with Dynamic Weight Average

Similarly, the Dynamic Weight Average method was used to update the weights of loss functions and the network was trained for 50 epochs. From the plot(Fig. 2.), the loss functions saturated after initial fluctuations

Also, the multi-armed bandits with UCB were trained on the same dataset with and the rewards were given for the overall reduction of the loss function from the previous iteration. The plot shows the saturation of validation losses for all the tasks.

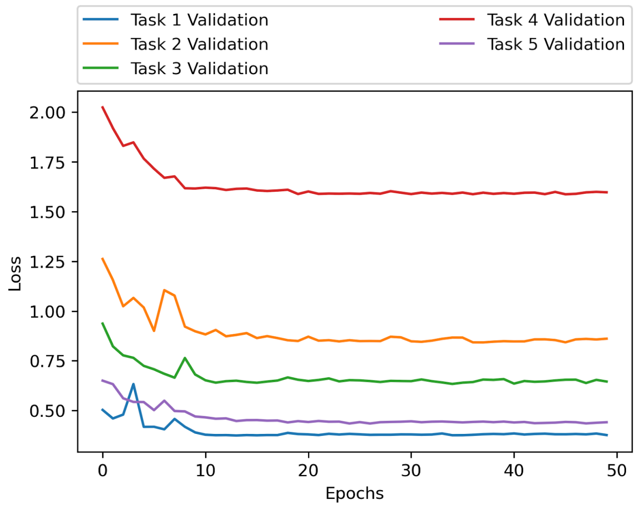

Fig. 3. Loss functions of 5 tasks with UCB Weight Adaption

Below table lists the accuracy of all the tasks when different loss mixing methods were used. It can be seen that UCB method achieved better overall accuracy when compared with the other 2 methods.

TABLE I. Overall tasks accuracy

Mixing	**ACCURACY(%)**				
	Task 1	***Task 2***	***Task 3***	***Task 4***	***Task 5***
Uniform	92.68	87.80	89.02	**82.93**	81.46
DWA	93.46	89.02	**93.90**	79.27	82.15
UCB	**93.62**	**91.46**	90.24	81.71	**82.77**

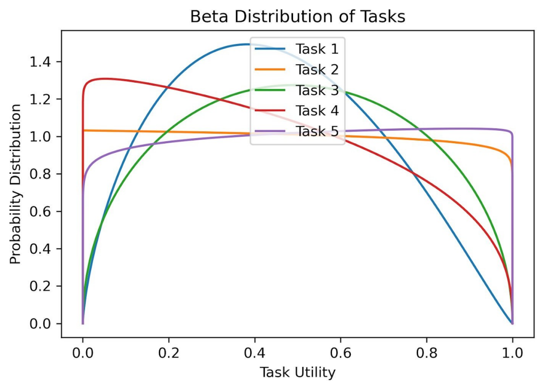

Fig. 4. Beta Distributions of 5 tasks.

From the alternate training methods, AUTOSEM was trained for 250 steps for both stage 1 and stage 2. In this case each step is considered as one run through the sequences of batches of all the tasks(subset of tasks for stage 2) and 1 epoch of validation set. The plot above(Fig. 4) shows the beta distribution of all the tasks after 250 steps with Task 1 as the primary task and remaining as the auxiliary tasks.

From the above plot(Fig. 4.), Task 1, Task 3 and Task 5 are selected for the stage 2. In the stage 2, the GP model found the best mixing ratio to be 8:3:3. After training for 250 steps, task 1 was able to achieve the accuracy of **80.49%**.

Also, the multi-armed bandits with UCB were trained on the same dataset with Task 1 as a primary task and all others as auxiliary tasks. The method was able to achieve the accuracy of **93.90%** in just 60 epochs with the same network architecture and hyperparameters.

V. Conclusion

The project explored three different existing approaches of mixing loss function in MTL problems while also introducing a new mixing method which performs better than the existing mixing methods. With the exact same hyperparameters and the network architecture the AUTOSEM approach was not performing very well on the selected dataset. If I had more time I'd have liked to experiment the UCB mixing method with GLUE dataset and compare its performance with AUTOSEM baselines. Another direction I'd like to explore is to come up with non-discrete updates to the loss weights instead of step updates(+0.005 and -0.005) which I believe can further improve the performance.

References

[1] Shikun Liu, Edward Johns, Andrew J. Davison, "End-to-End Multi-Task Learning with Attention," arXiv preprint – arXiv:1803.10704

[2] Han Guo, Ramakanth Pasunuru, and Mohit Bansal, "AUTOSEM: Automatic Task Selection and Mixing in Multi-Task Learning," arXiv preprint – arXiv:1904.04153

[3] Alex Kendall, Yarin Gal, Roberto Cipolla, "Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics," arXiv preprint – arXiv:1705.07115

[4] I. S. Jacobs and C. P. Bean, "GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks," arXiv preprint – arXiv:1711.02257

[5] Sebastian Ruder, Barbara Plank, "Learning to select data for transfer learning with Bayesian Optimization," arXiv preprint – arXiv:1707.05246