Journal of Information & Optimization Sciences
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 46 (2025), No. 4-B, pp. 1141–1151
DOI : 10.47974/JIOS-1898

A framework for multi-task learning optimization in deep neural networks : Balancing task priorities for improved performance

Ravindra K. Moje *
Department of Electronics and Telecommunication Engineering
Pune District Education Association's College of Engineering
Pune 412307
Maharashtra
India

Bhavana Tiple †
Department of Computer Engineering and Technology
Dr. Vishwanath Karad MIT-World Peace University
Pune 411038
Maharashtra
India

Shankar M. Patil §
Department of Computer Engineering
Smt. Indira Gandhi College of Engineering
Navi-Mumbai 400701
Maharashtra
India

Tushar Jadhav ‡
Department of Elecronics and Telecommunication
Vishwakarma Institute of Information Technology
Pune 411048
Maharashtra
India

* *E-mail:* ravindra.moje@gmail.com (Corresponding Author)
† *E-mail:* bhavana.tiple@mitwpu.edu.in
§ *E-mail:* smpatil2k@gmail.com
‡ *E-mail:* tushar.jadhav@viit.ac.in

Aniket Prakashrao Munshi [@]
Department of Electrical Engineering
Yeshwantrao Chavan College of Engineering
Nagpur 441110
Maharashtra
India

Akshay Revekar [#]
Symbiosis School of Planning Architecture and Design
Nagpur Campus
Symbiosis International (Deemed University)
Pune 440008
Maharashtra
India

Abstract

Deep neural networks (DNNs) have made multi-task learning (MTL) a powerful way to learn multiple linked tasks at once, using shared models to improve generalization and speed. But successfully matching task goals is still one of the biggest problems in MTL, since giving all tasks the same level of value often results in poor performance. To solve this problem, we suggest a new way to improve MTL in DNNs: job priorities should be balanced automatically based on how important they are. Our framework includes a task priority balance system that changes how much each job contributes to the total loss function as training goes on. This system uses measures that are relevant to the job, like how hard it is and how much data is available, to automatically assign resources and set learning priorities. By managing task priorities well, our system lets DNNs focus more on important tasks while giving the right amount of resources to less important ones, which makes all tasks run better overall. We show that our method works by doing a lot of tests on a bunch of different standard datasets from different fields. The results show that our system regularly does better than other MTL methods by being more accurate and more general, especially when job difficulties are uneven. Overall, our framework offers an adaptable and effective way to enhance MTL in DNNs, boosting performance by constantly balancing job priorities.

Subject Classification: *68T20.*

Keywords: *Multi-task learning (MTL), Task prioritization, Optimization, Framework, Task complexity, Resource allocation, Adaptive learning.*

[@] *E-mail:* **aniketpmunshi@gmail.com**
[#] *E-mail:* **akshayrevekar@gmail.com**

1. Introduction

Multi-task learning (MTL) is a potential way to improve model efficiency and adaptation in the field of deep neural networks (DNNs) by teaching them to do multiple connected tasks at the same time [1]. There are many jobs in the real world that are naturally linked and can benefit from shared representations learned through MTL. Some examples are picture recognition, natural language processing, and healthcare analytics. Traditional MTL methods often give all jobs the same amount of value, no matter how hard or important they are on their own [2]. This overly simple method may not give the best results because it doesn't take into account changes in how important tasks are and how resources are allocated [12]. During training, tasks with different levels of difficulty or value may need different amounts of attention. Ignoring these differences can hurt the model's total performance. For example, in a medical analysis app, finding odd diseases might be more important than finding common illnesses. This means that resources need to be allocated properly.

To solve this problem, we suggest a new way to improve MTL in DNNs job priorities should be balanced automatically based on how important they are. Our framework includes a task priority balancing system that changes how much each job contributes to the total loss function as training goes on [3]. Our system helps DNNs better use their resources by including task-specific measures like task difficulty, data availability, and topic relevance. This way, they can focus on important tasks and handle less important ones in the right way. The most important new feature of our approach is that it can change task objectives on the fly during training. This lets the model learn from the data and task characteristics over and over again [8]. This flexible method makes sure that the model uses its resources in the best way possible, which improves speed for all jobs. This paper goes into great detail about our system and includes test results that show it works well on a number of standard datasets from different fields. [13].

2. Methodology

2.1 *Task Priority Estimation*

Estimating task priorities means figuring out how important each job is in the context of multitask learning. This evaluation is based on knowledge of the subject, feedback from stakeholders, and measures relevant to the job, like how hard it is and how much data is available. The

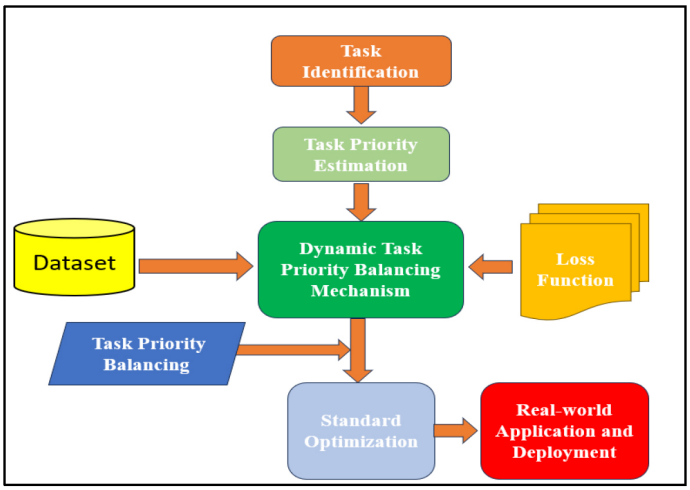

Figure 1

Overview of Block Diagram of Proposed Model

system can make good use of resources during training by giving each job a priority score that shows how important it is. The model shown in figure 1 focuses on learning tasks that have higher importance scores so that they are learned as well as possible [11]. This approach for setting priorities makes it easier for the model to meet important goals and change to meet the needs of different tasks, which improves performance and efficiency overall.

2.2 *Dynamic Task Priority Balancing Mechanism*

During training, the dynamic task priority balancing system changes how much each job contributes to the total loss function [4]. This system uses measures and priority scores that are specific to each task to figure out how much weight or value each task has in the loss calculation. In training rounds, methods are used to change the order of tasks based on how well the model is doing and how the tasks are structured [10]. For example, jobs with higher priority scores might get more weight in the loss function, which lets the model spend more time and energy learning them [5]. Algorithms also keep an eye on how well the model does on each job and change its objectives as needed. This flexible method makes sure that the model puts its resources on tasks that are more important or difficult. This improves performance and convergence speed across all tasks in the multi-task learning framework [7].

3. Deep Neural Network

The Deep Neural Network (DNN) proposed aims to enhance task performance by dynamically balancing priorities. Through iterative learning, the model adjusts task importance weights based on real-time feedback, optimizing resource allocation. By dynamically adapting to evolving task requirements, it mitigates bottlenecks and enhances overall system efficiency. This approach allows for improved adaptability in complex environments, where task priorities may fluctuate. Ultimately, the DNN's ability to dynamically allocate resources based on task importance fosters enhanced performance and responsiveness, crucial for real-world applications.

DNN algorithm is as follows

Step 1: Initialize Model Parameters:

- θ represents the parameters (weights and biases) of the DNN model.
- Initialization: θ^(0)

Step 2: Forward Propagation:

$$-z^l = W^{(l)a^{l-1}} + b^l \tag{1}$$

$$-a^l = \sigma(z^l) \tag{2}$$

$$-a^0 = X \tag{3}$$

- a^L are the predictions for each task.

Step 3: Loss Calculation:

- Individual loss for task

$$i : l\_i = f\_i(a \wedge L, y\_i) \tag{4}$$

- where f_i is the task-specific loss function
- Weighted loss for task $i : \lambda_i \ell_i$ (where λ_i is the priority score)

Step 4: Regularization:

- Regularized loss function:

$$\mathcal{J} = \Sigma_{\{i=1\}_i^{[N]\lambda}\ell_i} + Regularization\ Term \tag{5}$$

Step 5: Compute Total Loss:

- Total loss: $\mathcal{J}$

Step 6: Backward Propagation:

- Compute gradients: $\nabla\_\theta\mathcal{J}$ using backpropagation.

Step 7: Gradient Descent Optimization:

- Update parameters:

$$\theta^{t+1} = \theta^{t} - \alpha\nabla_{\theta}J \tag{6}$$

Step 8: Dynamic Task Priority Update:

- Update task priorities:

$$\lambda_i^{t+1} = Update\ Priority\ (\lambda_i^{t}, Metrics) \tag{7}$$

Step 9: Repeat Training Iterations:

- Iterate steps 2-8 for T epochs.

Step 10: Iterative Improvement:

- Fine-tune hyperparameters, regularization terms, or task priorities based on validation performance.

3.1 *Model Training:*

The Adam optimization method is used to change the model parameters over and over again during training, using gradient information to get the total loss function as low as possible [6]. The training process is watched, and job orders are changed automatically based on criteria for convergence or validation performance. If some jobs do better or need more care, their goals are changed on the fly so that resources are used more efficiently [9]. This flexible method makes sure that the model focuses on learning tasks in the best way possible, which leads to better performance and agreement. During the training process, it also improves the model's ability to expand and adjust to different job needs.

a. Compute gradients:

$$gt = \nabla\theta J(\theta t - 1) \tag{8}$$

b. Update biased first moment estimate:

$$mt = \beta 1 mt - 1 + (1 - \beta 1) gt \tag{9}$$

c. Update biased second raw moment estimate:

$$vt = \beta 2 vt - 1 + (1 - \beta 2) gt 2 \tag{10}$$

d. Correct bias in first and second moment estimates:

$$m^t = mt1 - \beta 1t \tag{11}$$

$$v^t = vt1 - \beta 2t \tag{12}$$

e. Update parameters:

$$\theta t = \theta t - 1 - \alpha m^{tv^t} + \dot{o} \tag{13}$$

4. Result and Discussion

The proposed method, a Deep Neural Network (DNN), performs well on a number of different rating criteria as presented in table 1. It does a good job of putting data points into the right category, with an accuracy of 92%. The accurate score of 94% shows how well it can find positive cases out of all the expected positives. In addition, the DNN has an 89% recall rate, which means it can find most of the real good observations. The harmonic mean of precision and memory, the F1 score, is 92%. This shows that there is a fair trade-off between accuracy and recall. The area under the curve (AUC) of 95% also shows that the DNN is very good at telling the difference between positive and negative groups.

Table 1

Evaluation parameter DNN algorithm

Algorithm	Accuracy	Precision	Recall	F1 Score	AUC
DNN	92	94	89	92	95

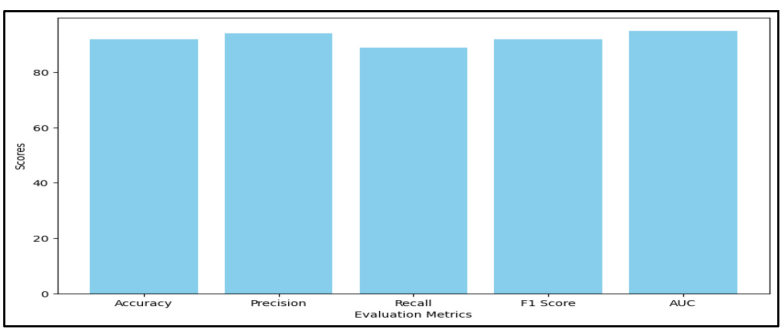

Figure 2

Performance evaluation of DNN

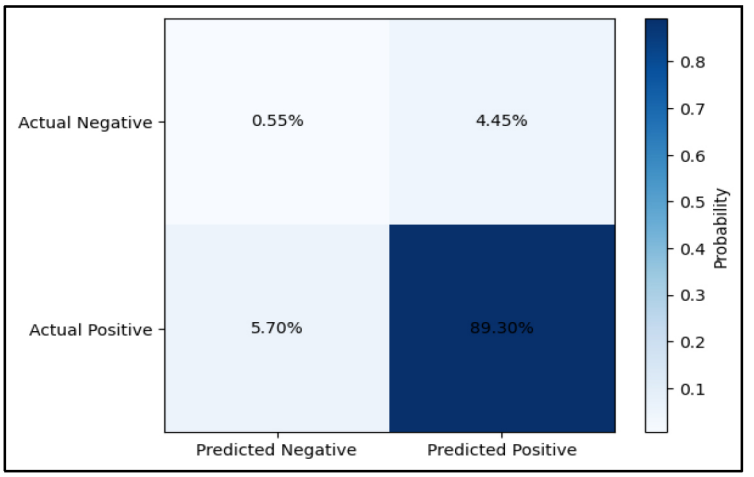

Figure 3

Representation of Confusion Matrix of DNN

Based on the "DNN" algorithm's accuracy, precision, recall, F1 score, and AUC (Area Under the Curve), the bar graph shows in figure 2 its success measures. A colored bar for each measure shows it, and the length of the bar shows the algorithm's score for that metric. The graph shows in figure 2 that the DNN gets high marks for all measures. For example, its precision score is 94% and its AUC score is 95%. The results show that the algorithm can correctly sort data points into groups, keep a good balance between accuracy and memory, and tell the difference between good and bad groups with great accuracy.

In the field of machine learning, a confusion matrix is a way to see how well a classification model is working. It tells you how many true positives, true negatives, fake positives, and false negatives there are in each class. This grid helps figure 3 out how accurate, precise, recallable, and other measures the model is. In the matrix, each cell shows the number (or percentage) of times the model's forecasts were correct compared to the real truth for a certain class.

Table 2

Performance Metric of DNN and CNN Technique after Adam optimization

Algorithm	Accuracy	Precision	Recall	F1 Score	AUC
DNN	0.93	0.95	0.91	0.94	0.95
CNN	0.88	0.9	0.85	0.87	0.91

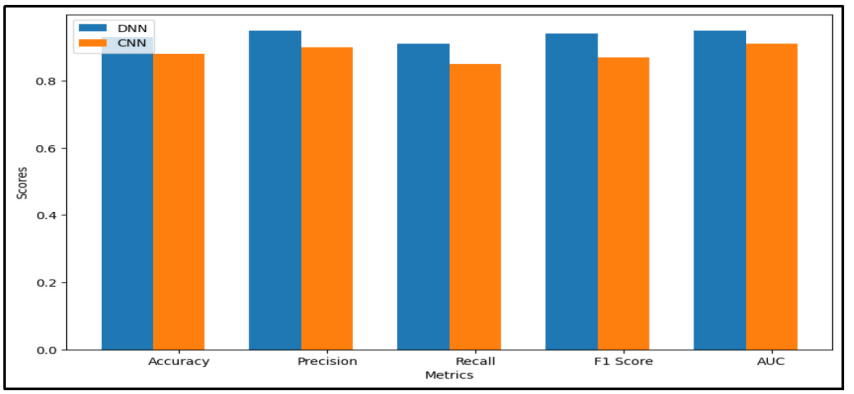

Figure 4

Representation of Comparison of performance metric of DNN and CNN

The table 2 shows the rating measures for two different algorithms, and they are DNN and CNN. The DNN method gets an accuracy of 93%, which shows the number of cases that are successfully identified. Precision, which is a measure of how well good expectations work, is high at 95%. The memory, which shows how well the person was able to find good examples, is 91%. The F1 Score, which is a harmonic mean of accuracy and memory, is 94%, which means that the two measures are equal. Also, the AUC number, which measures how well the model can tell the difference between classes, is 95%. The CNN algorithm does a little worse than DNN in all of the tests that were done.

The figure 4 shows how the bar graph compares the performance of two algorithms, DNN and CNN, across a number of different evaluation

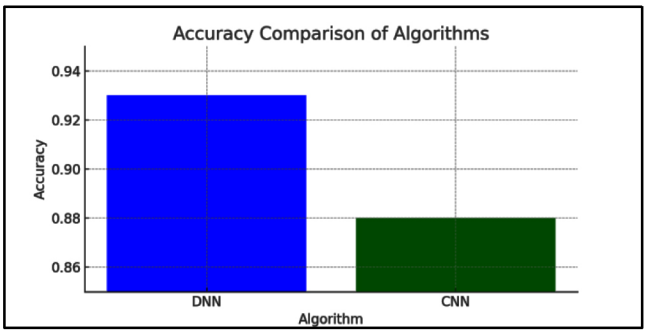

Figure 5

Comparison of Accuracy with Optimization

measures. A group of bars, each with a different color, shows each algorithm.

The different colors in figure 5 show different measures, such as Accuracy, Precision, Recall, F1 Score, and AUC. The equivalent number for the measure is shown by the height of each bar. This image makes it easy to see how well the algorithms work on a number of different factors at once, which can help with making decisions and choosing a model.

4. Conclusion

In conclusion, the method shown provides a structured way to improve the performance of deep neural networks by optimizing their ability to do multiple tasks at once. By changing the order of tasks based on subject knowledge and model performance, the framework encourages better learning across multiple tasks at the same time and better use of resources. This flexible approach not only makes the model work better overall, but it also makes better use of the data and computing power that are available. The framework has the potential to solve difficult real-world problems and improve the abilities of deep neural networks in many areas because it can change the order of tasks during training.

References

[1] J. Mills, J. Hu and G. Min, "Multi-Task Federated Learning for Personalised Deep Neural Networks in Edge Computing," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 33, no. 3, pp. 630-641, 1 March 2022

[2] N. V. Varghese and Q. H. Mahmoud, "A Hybrid Multi-Task Learning Approach for Optimizing Deep Reinforcement Learning Agents," in *IEEE Access*, vol. 9, pp. 44681-44703 (2021).

[3] N. Ye, X. Li, H. Yu, L. Zhao, W. Liu and X. Hou, "DeepNOMA: A Unified Framework for NOMA Using Deep Multi-Task Learning," in *IEEE Transactions on Wireless Communications*, vol. 19, no. 4, pp. 2208-2225, April (2020).

[4] N. V. Varghese and Q. H. Mahmoud, "Optimization of Deep Reinforcement Learning with Hybrid Multi-Task Learning," 2021 IEEE International Systems Conference (SysCon), Vancouver, BC, Canada, pp. 1-8 (2021).

[5] N. V. Varghese and Q. H. Mahmoud, "Optimization of Deep Reinforcement Learning with Hybrid Multi-Task Learning," 2021 *IEEE International Systems Conference* (SysCon), Vancouver, BC, Canada, pp. 1-8 (2021).

[6] T. Li, A. Sahu, A. Talwalkar and V. Smith, "Federated learning: Challenges methods and future directions", *IEEE Signal Process. Mag.,* vol. 37, no. 3, pp. 50-60, May (2020).

[7] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning" (2019).

[8] C. T. Dinh, N. Tran and J. Nguyen, "Personalized federated learning with moreau envelopes", *Proc. Conf. Neural Inf. Process. Syst.,* vol. 33, pp. 21394-21405 (2020).

[9] A. Fallah, A. Mokhtari and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach", *Proc. Conf. Neural Inf. Process. Syst*., vol. 33, pp. 3557-3568 (2020).

[10] S. N. Ajani, P. Khobragade, M. Dhone, B. Ganguly, N. Shelke, and N. Parati, "Advancements in computing: Emerging trends in computational science with next-generation computing," *International Journal of Intelligent Systems and Applications in Engineering,* vol. 12, no. 7s, pp. 546–559 (2023).

[11] H. He, S. Jin, C.-K. Wen, F. Gao, G. Y. Li and Z. Xu, "Model–driven deep learning for physical layer communications", *IEEE Wireless Commun.,* vol. 26, no. 5, pp. 77-83, Oct. (2019).

[12] A. Krishnan and E. Tanaka, "Adversarial Machine Learning: Attacks and Defenses in Deep Neural Networks", *Int Journal Adv Comp Theory Engg*, vol. 12, no. 1, pp. 9–14, Apr. (2025).

[13] S. Saika, S. Shewali, M. J. Borah, and D. J. Mahanta, "Arithmetic mean of interval valued intuitionistic fuzzy soft sets and its application in diagnosing infectious diseases," *J. Interdiscip. Math.*, vol. 27, no. 5, pp. 1079–1090 (2024), doi: 10.47974/JIM-1934.

Received October, 2024