Analyzing the Effects of Combining Gradient Conflict Mitigation Methods in Multi-Task Learning

Richard Alison[1*], Welly Jonathan[2], and Derwin Suhartono[3]
[1−3]Computer Science Department, School of Computer Science, Bina Nusantara University
Jakarta, Indonesia 11480
Email: [1]richard.alison@binus.ac.id, [2]welly.jonathan@binus.ac.id, [3]dsuhartono@binus.edu

***Abstract*—Multi-task machine learning approaches involve training a single model on multiple tasks at once to increase performance and efficiency over multiple single-task models trained individually on each task. When such a multi-task model is trained to perform multiple unrelated tasks, performance can degrade significantly since unrelated tasks often have gradients that vary widely in direction. These conflicting gradients may destructively interfere with each other, causing weights learned during the training of some tasks to become unlearned during the training of others. The research selects three existing methods to mitigate this problem: Project Conflicting Gradients (PCGrad), Modulation Module, and Language-Specific Subnetworks (LaSS). It explores how the application of different combinations of these methods affects the performance of a convolutional neural network on a multi-task image classification problem. The image classification problem used as a benchmark utilizes a dataset of 4,503 leaf images to create two separate tasks: the classification of plants and the detection of disease from leaf images. Experiment results on this problem show performance benefits over singular mitigation methods, with a combination of PCGrad and LaSS obtaining a task-averaged F1 score of 0.84686. This combination outperforms individual mitigation approaches by 0.01870, 0.02682, and 0.02434 for PCGrad, Modulation Module, and LaSS, respectively in terms of F1 score.**

***Index Terms*—Gradient Conflict Mitigation Methods, Multi-Task Learning, Project Conflicting Gradients (PCGrad), Modulation Module, Language-Specific Subnetworks (LaSS)**

I. Introduction

IT is possible for humans to utilize knowledge gained while learning to perform one task to learn to perform other related tasks. For example, experienced musicians are likely to learn new instruments faster than those without prior music experience. This inherent human ability has inspired the development of a machine learning approach called Multi-Task Learning (MTL) [1]. MTL approaches train one singular machine learning model to solve a multitude of tasks, leveraging information in related tasks to achieve better performance relative to multiple classic Single-Task Learning (STL) models trained individually on each task. MTL-based approaches are also utilized to improve overall computation efficiency by jointly performing training only once on the entire set of tasks [2].

Machine learning models incorporating MTL have outperformed STL models in supervised classification [3–5], visual scene understanding [6], image retrieval [7, 8], parameter auto-tuning [9], forecasting [10, 11], data upscaling [12], sentiment analysis [13], and speech recognition [14], among other tasks. While MTL is designed specifically to leverage information contained in related tasks [1], previous research has also shown performance benefits during training of unrelated tasks [15]. However, during such applications, a phenomenon known as destructive gradient interference may arise [16]. This phenomenon, caused by conflicting directions in the optimization gradient of multiple tasks, may lead to significantly degraded performance relative to single-task learning approaches. Existing research has produced multiple approaches that attempt to mitigate this phenomenon [3, 7, 13, 16, 17].

Previous research has attempted to quantify the exact cause of the destructive effects of gradient interference with regard to model performance. It is explained that significant performance degradation may arise when three different conditions forming the so-called tragic triad occur [16]. The first condition, conflicting gradients, arises when the optimization gradient for different tasks differs significantly in direction. The second condition, dominating gradients, arises when

Received: Aug. 27, 2022; received in revised form: Nov. 18, 2022; accepted: Nov. 18, 2022; available online: April 29, 2024.
*Corresponding Author

the optimization gradient for different tasks differs significantly in magnitude. When the second condition occurs along with the first condition, the task with the larger magnitude will dominate the average gradient. The third condition, high curvature, compounds the negative effects of the previous two conditions by creating an environment. It is easy to overestimate the performance improvement gained by the dominating task and vice versa.

Along with this analysis, the previous presents a general approach, called Project Conflicting Gradients (PCGrad), for minimizing the performance degradation caused by the aforementioned conditions [16]. This approach involves projecting the optimization gradient of a given task onto the normal plane of another task whenever conflicting gradients are found. This approach does not alter the gradients of non-conflicting tasks, meaning it makes no assumptions regarding the form of the model itself, allowing network architectures with both shared and unshared parameters to also benefit from this approach. It also presents experiment results that highlight the success of their proposed approach on various tasks, including supervised and reinforced MTL [16].

Moreover, another previous research has likened the destructive gradient interference problem to the interference of waves in physics [7]. Their approach to mitigating this, called the Modulation Module, adds layers containing task-specific weights in between other layers in neural network-based models. It effectively creates a hybrid model consisting of elements from both MTL and STL. These layers act as a channel-wise weighted scaling filter for the weights of the immediate previous layer. This scaling process uses task-specific weights that are unchanged during the training of other tasks. This approach notably requires relatively few additional parameters.

Multilingual machine translation models have also been shown to suffer from similar problems. It attributes performance degradation during joint training of multiple language pairs to parameter interference [17]. It presents Language-Specific Subnetworks (LaSS), an approach that attempts to alleviate this interference without increasing model capacity by finding subnetwork patterns that are implicitly learned during joint training of multiple language pairs. These patterns are searched using heuristics derived from neural network weight pruning approaches. Further fine-tuning and joint training are performed, not distorting weights outside the subnetwork relevant to the current language pair.

There have also been attempts to avoid the gradient interference problem entirely by designing neural network architectures specific to the task at hand. In one instance, previous research has used two attention-like modules to explicitly learn the relationships between tasks and output classes in a multi-task classification problem [3]. One module learns the relationship between tasks, while another learns the relationship between the classes of every task. The outputs of these two modules are concatenated and fed into task-specific branches consisting of four fully connected neural network layers. The resulting model manages to perform better than models built using single-task approaches.

Based on the explanation mentioned earlier, the research aims to explore the effects of applying combinations of some existing mitigation approaches to the performance of a neural network. The researchers hypothesize that combining some approaches will achieve better performance than a single approach. The researchers pick three previously developed mitigation methods and perform experiments on all eight possible mathematical combinations of the methods mentioned earlier on a supervised MTL classification dataset. The performance figures obtained from these experiments are then used to analyze and validate the hypothesis.

II. Research Method

Three previously developed conflict mitigation approaches are described. It consists of PCGrad [16], Modulation Module [7], and LaSS [17]. The method used to combine these previous approaches and test the validity of the hypothesis is also outlined.

A. Methods Used

PCGrad gradient update algorithm is proposed to deconflict interfering weight update gradients by directly altering the gradients themselves [16]. The algorithm iterates over the gradient g_t (the original gradient of the current task t) of each task and checks whether it conflicts with the gradient of other tasks g_0 (the original gradient of another task o, which is the task toward which g_t is projected). If such a conflict is found, g_t is replaced with its projection towards the normal plane of g_0.

It conceptually removes the conflicting parts from the gradient of each task, as illustrated in Fig. 1. The altered gradients are then passed to a gradient-based optimizer and handled normally. Algorithm 1 details this process in a programmatic way for ease of understanding.

As described in previous research [7], a Modulation Module is a layer that is inserted between convolutional layers in neural networks to perform channel-wise vector scaling of the output of the immediate

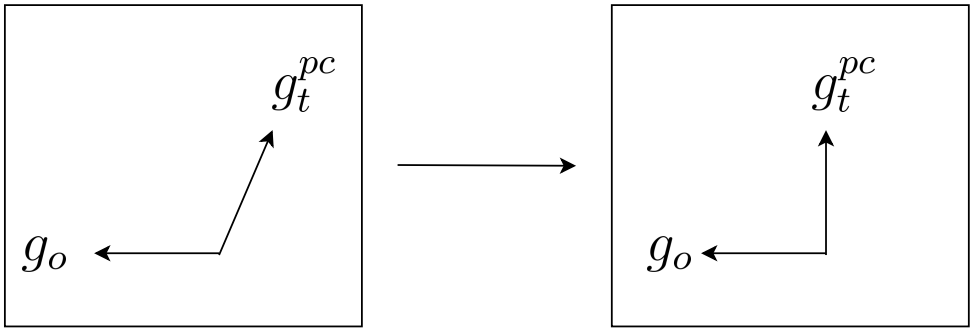

Fig. 1. How Project Conflicting Gradients (PCGrad) affecting conflicting gradients. It shows g_0 as the original gradient of another task o, which is the task toward which g_t is projected and g_t^{pc} as the gradient of task t in g^pc (variable used to store task gradients after projection).

Algorithm 1 PCGrad gradient update algorithm

Let $T \leftarrow$ tasks
Let $g \leftarrow$ gradients
$g^{pc} \leftarrow g$
for $t \in T$ **do**
  for $o \in T$ in random order **do**
    if $g_t^{pc} \cdot g_o < 0$ **then**
      $g_t^{pc} \leftarrow g_t^{pc} - \frac{g_t^{pc} \cdot g_o}{||g_o||^2} g_o$
    end if
  end for
end for
$g \leftarrow g^{pc}$
Pass g to optimizer

previous convolutional layer using trainable per-task independent weights.

Figure 2 shows an overview of the computation in a scaling layer during the feed-forward phase of a neural network. The research only implements vector scaling instead of full matrix scaling in accordance with the assertion of previous research [7] that full matrix scaling leads only to marginal improvements while increasing complexity.

Let O (output of the layer before the scaling layer) be the output of the previous layer of the shape $C \times N \times M$. It shows C (depth of O), N (height of O), and M (width of O). Every scaling layer contains a weight matrix W (weight matrix contained in the scaling layer) of the shape $T \times C$, where T is the total number of tasks. Let W_t (part of the weight matrix W that is used for scaling the current task t) be the per-task weights of the scaling layer for the current task t.

Then, following equation, the researchers calculate the scaled output O' and pass it along to the next layer. It has $O'_{(i,j,k)}$ as element of the scaled output matrix at index i, j, k, $O_{(i,j,k)}$ as element of the original output matrix at index i, j, k, and $W_{(t,i)}$ as element of the weight matrix W, corresponding to the current task t at index i.

$$O'_{i,j,k} = O_{i,j,k} \times W_{t,i}.$$

Intuitively, the per-task nature of the weights learned in these scaling layers allows the neural network to learn how to adjust its outputs better to suit the current task. In other words, it modulates those outputs with regard to the current task. The addition of these layers in between the convolutional layers of a neural network helps to alleviate the problem of gradient interference by allowing information that will otherwise be lost due to gradient interference to be retained inside the task-specific weights.

This concept is essentially a simplified version of creating separated and unshared branches specific to each task and merging their outputs back together into the main shared branch. In fact, it is possible to see that attempting to insert these scaling layers in between the non-convolutional layers of a neural network will scale the weights of each neuron in each layer in a task-specific manner, leading to the creation of a separate unshared branch that merges back together into the next shared layer.

Next, LaSS, originally designed for multilingual translation, is the approach used to explicitly find subnetwork masks for every task, which are then utilized to improve the performance of jointly-trained models [17]. A subnetwork mask is a task-specific binary matrix $M_t \in 0, 1$, where $M_t^w = 1$ means that the weight w corresponding to that mask is relevant in the current subnetwork. It shows M_t as binary matrix M for the current task and M_t^w as element of matrix M_t that corresponds to the weight w. For brevity, the researchers describe the general outline of this approach in simplified terms as follows:

1) Start with a neural network θ_0 (weights of neural

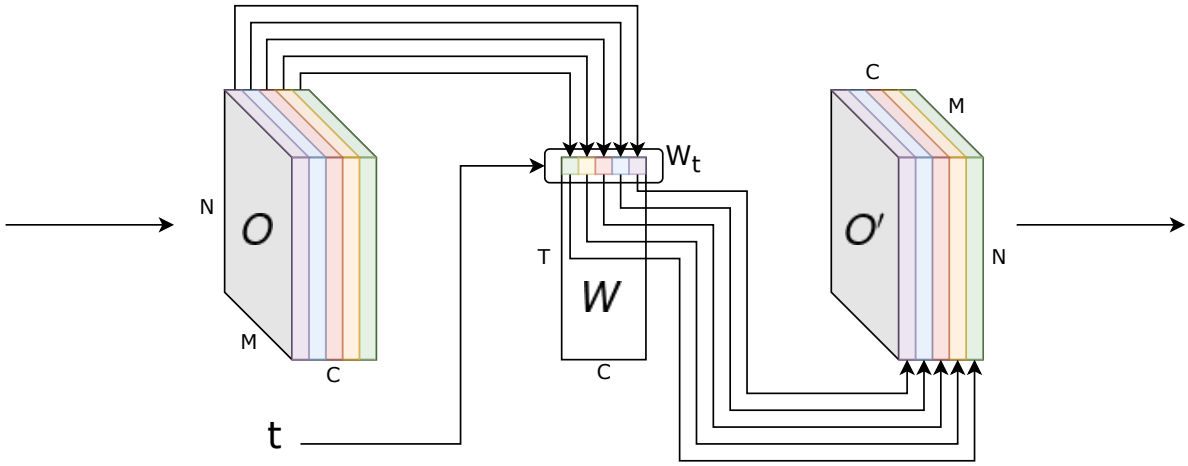

Fig. 2. Overview of computation flow in a scaling layer. It consists of O (output of the layer before the scaling layer), C (depth of O), N (height of O), M (width of O), W (weight matrix contained in the scaling layer), W_t (part of the weight matrix W that is used for scaling the current task t), t (the current task), and O' (the scaled output after going through the Modulation Module).

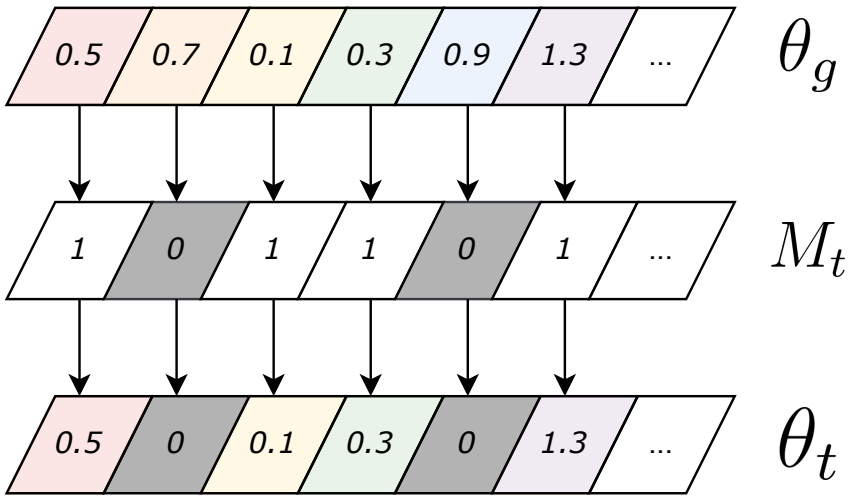

Fig. 3. Computing θ_t from θ_g and M_t.

network after joint training) jointly trained on all tasks.

2) For each task t, fine-tune the network using data specific to t to obtain the task-specific fine-tuned network θ_t (weights of neural network after fine tuning for task t). It is explained that this step amplifies the magnitude of important weights to the current task t while diminishing unimportant ones [17].
3) Prune the lowest α percent of the weights in θ_t (weights of neural network after fine tuning for task t).
4) Build a subnetwork mask M_t for θ_t by setting the mask value to 0 if the corresponding weight was pruned in the previous step or 1 otherwise.

The obtained subnetwork mask is then used during further training and inference to ignore unimportant weights to the current task t by replacing the global weight matrix θ_g (global weight matrix that will be replaced by θ_t during inference of task t) with θ_t, where $\theta_t = \theta_g \times M_t$. This process can also be utilized for, e.g., zeroing out gradients during back-propagation of gradients. Figure 3 illustrates this process.

The utilization of LaSS alters the standard joint training process of a neural network by splitting it into two distinct parts. The first part, referring to the pre-mask phase, is performed to obtain θ_0 prior to finding the subnetwork masks. The second part, the post-mask

TABLE I
EVERY POSSIBLE COMBINATION OF THE THREE APPROACHES.

Approach	PCGrad	Modulation Module	LaSS
A	No	No	No
B	No	No	**Yes**
C	No	**Yes**	No
D	No	**Yes**	**Yes**
E	**Yes**	No	No
F	**Yes**	No	**Yes**
G	**Yes**	**Yes**	No
H	**Yes**	**Yes**	**Yes**

Fig. 4. Sample images taken from the dataset. The blemishes on the left leaf are a telltale sign that the leaf is diseased.

phase, is performed after subnetwork masks based on the previously obtained θ_0 weights are found. This part utilizes the task-specific subnetwork mask M_t to mask weights deemed unimportant to the current task while preventing important weights to other tasks from being affected during back-propagation. It aligns with the potential uses of subnetwork masks described in the previous paragraph.

B. Validation Design

To empirically validate the hypothesis, the researchers design an experiment to benchmark the performance of seven different approaches. Each utilizes a subset of the three mitigation methods described in the Methodssection. Along with these approaches, a baseline approach without any mitigation methods is also tested for comparison. Table I describes these eight approaches. The approach of H deconflicts gradients using PCGrad, incorporating scaling layers in between convolutional layers and utilizing subnetworks during joint training to illustrate the meaning of Table I.

The experiment utilizes the dataset of available leaf images in previous research [18]. The dataset consists of 4,503 leaf images from 12 different plants. Of the 4,503 leaves featured in the dataset, 2,278 are healthy, while 2,225 are not. A sample of two images taken from the dataset is shown in Fig. 4. From this dataset, the researchers create an MTL image classification problem consisting of two tasks: classifying which of the 12 given plant species the leaf belongs to and classifying whether the current leaf is diseased or not.

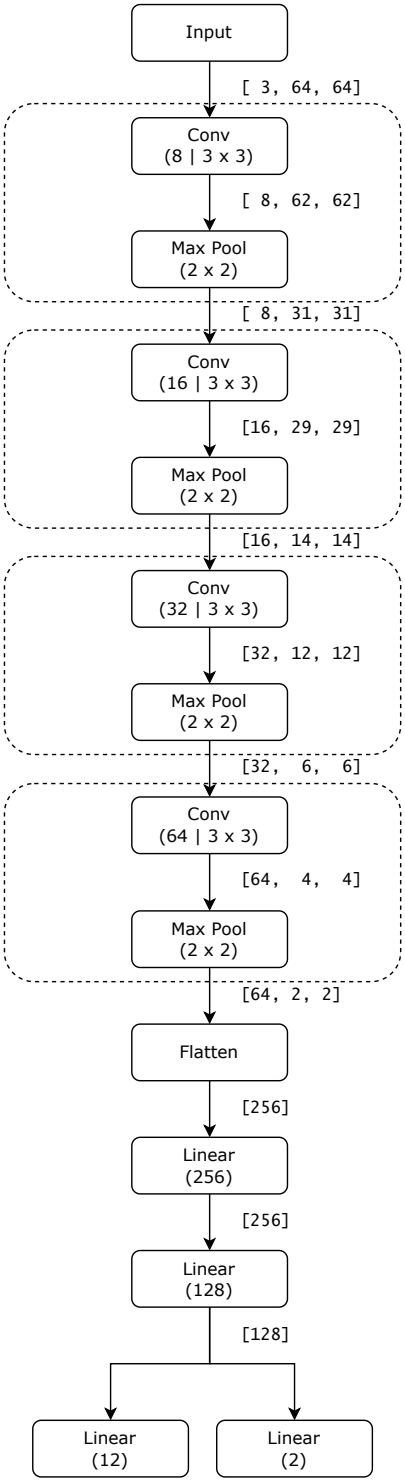

Fig. 5. Baseline model architecture.

The baseline approach solves this MTL image classification problem using the convolutional neural network model, illustrated in Fig. 5. For approaches in-

TABLE II
PERFORMANCE STATISTICS OF THE TESTED APPROACHES.

Approach	Accuracy in Task 1	F1 in Task 1	Accuracy in Task 2	F1 in Task 2	Average Accurracy	Average F1
A	0.70169	0.69532	0.85162	0.85671	0.77666	0.77602
B	0.77722	0.78359	0.85917	0.86145	0.81820	0.82252
C	0.77187	0.76522	**0.86673**	**0.87486**	0.81930	0.82004
D	0.77810	0.77360	0.86029	0.87123	0.81920	0.82241
E	0.78522	0.79088	0.86272	0.86545	0.82397	0.82816
F	**0.82075**	**0.82243**	0.86361	0.87129	**0.84218**	**0.84686**
G	0.79587	0.79742	0.85429	0.85840	0.82508	0.82791
H	0.78410	0.79721	0.85340	0.85789	0.81875	0.82755

corporating methods, the researchers apply each given method on top of the baseline model. In E to H approaches, PCGrad is applied by wrapping the base optimizer in an open-source PyTorch PCGrad implementation [19]. In approaches C–D and G–H, Modulation Module layers are inserted directly after every convolutional layer. For approaches incorporating LaSS, the researchers consider the first half of training to be the pre-mask phase and use the weights obtained during that phase as θ_0 to find the subnetwork masks used in the later half of training during the post-mask phase. Following the implementation of the original authors, the researchers only apply LaSS masking to weights contained in fully connected layers.

III. RESULTS AND DISCUSSION

A. Experiment Settings

The images in the dataset are first resized to a dimension of 64×64 pixels, utilizing zero padding to keep their original ratios. These images are then grouped into batches with a batch size of 32. Joint training is run on the aforementioned batches for a total of 100 epochs using the PyTorch Adam optimizer with a learning rate of 5×10^{-4}. Where applicable, the α value used during the pruning process to obtain subnetwork masks is set to 0.5. The researchers utilize 5-fold cross-validation and average the accuracy values and weight-averaged F1 scores obtained from the test partition of every fold for analysis.

B. Experiment Results

Looking at Table II, it is possible to infer that the baseline approach of A suffers from significant gradient interference. As seen in Fig. 6, a very significant gap exists between the performance of Task 1 and Task 2 relative to other approaches. It matches the analysis presented by previous research [16], which states that the gradients of one task may become dominated by the gradients of another task if a severe difference exists in direction and magnitude between them. For illustrative purposes, a human observer will take into account the shape of the leaf to solve Task 1. Meanwhile, in Task 2, they will visually look for blemishes in the color of the leaf. These two tasks are not necessarily related. Therefore, their gradients may differ significantly in direction and magnitude.

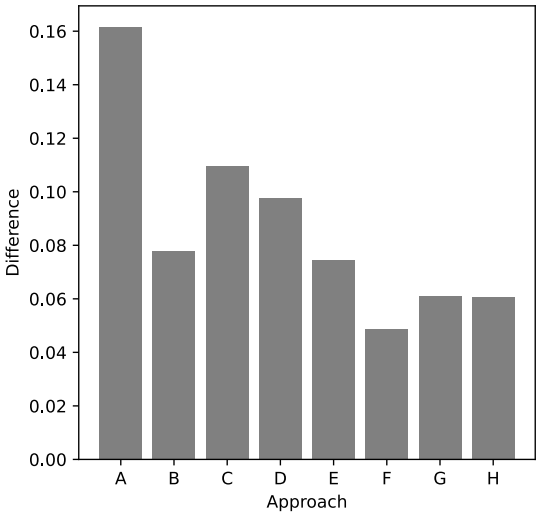

Fig. 6. Difference between the F1 score of Task 1 and Task 2 for each model.

Applying any gradient interference mitigation method successfully reduces the aforementioned gap, as seen in the performance of B, C, and E approaches in Table II. It is also possible to see in Fig. 7 that the approach of F, which combines the methods used in approaches B and E, significantly outperforms all other methods on Task 1. However, Fig. 8 shows that this approach performs slightly worse on Task 2 than approach B. It may be caused by PCGrad changing the direction of the training gradients slightly away from Task 2 during its gradient projection phase, improving the overall task-averaged performance of the model significantly at the cost of the aforementioned

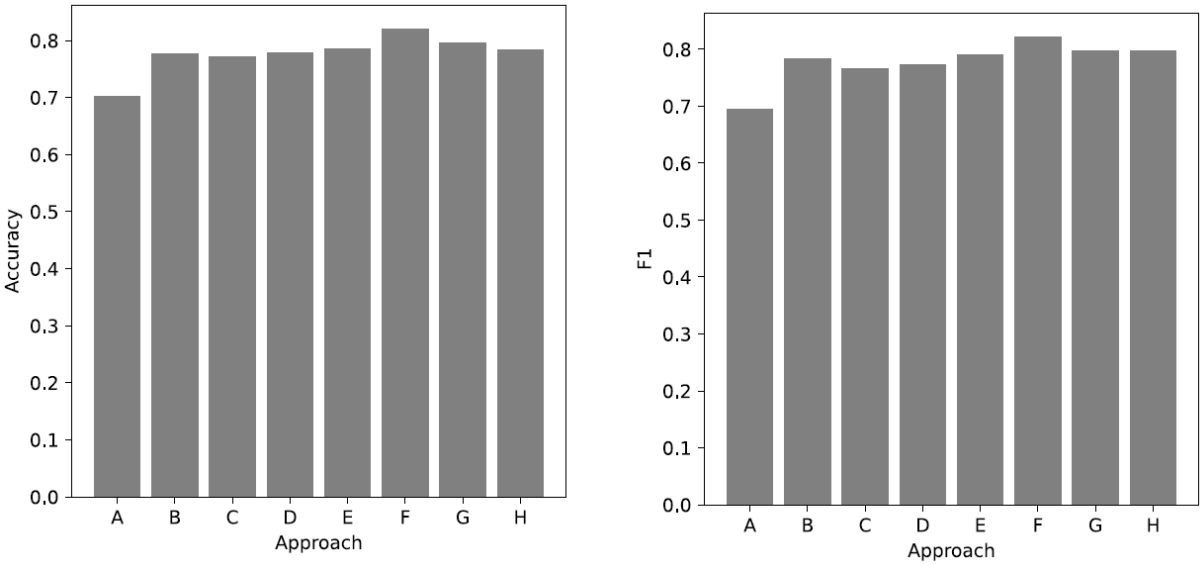

Fig. 7. Performance of each approach in Task 1.

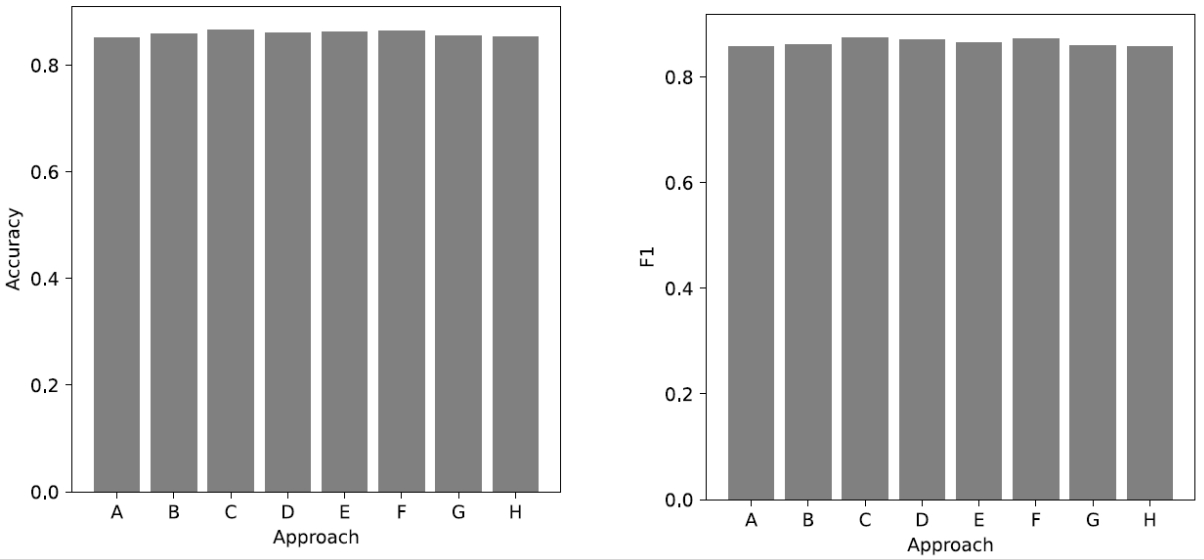

Fig. 8. Performance of each approach in Task 2.

slight performance degradation on one task. Overall, the approach of F performs the best in task-averaged accuracy and F1 score out of all the combinations tested in this experiment, as can be seen in Table II and Fig. 9. The result is in line with the hypothesis that a combination of mitigation methods outperforms its individual constituent methods.

However, the D, G, and H approaches fail to provide significant improvements in performance compared to the subsets of the methods they incorporate. For example, the approach of H performs worse than F on both tasks, while the approach of G does not perform as well as E does on average despite having less of a performance gap between Task 1 and Task 2. The performance in approaches of E, F, G, and H suggests the existence of a destructive incompatibility issue

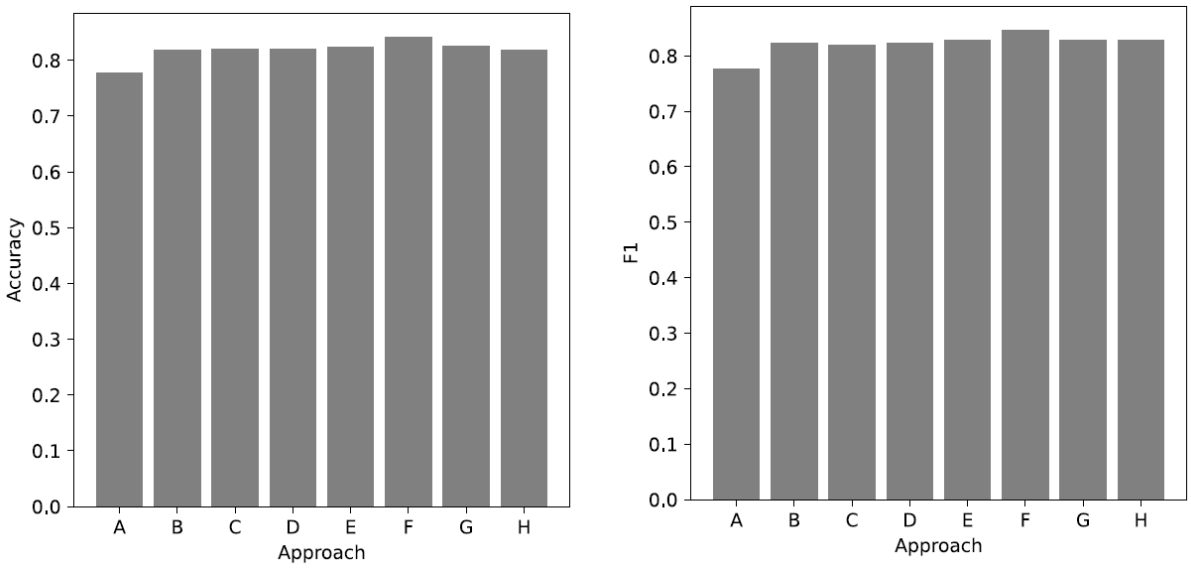

Fig. 9. Performance of each approach, averaged across both tasks.

between PCGrad and Modulation Module. PCGrad alters gradients in the shared layers of the model, and it is possible that the weights learned in these layers may not interact well with the weights learned using unaltered gradients inside the layers of the Modulation Module.

With regard to approaches of D and H, it is possible that the lower per-task capacity of the model, a natural outcome of the usage of LaSS' per-task subnetwork masks, may have negated the additional capacity gained by incorporating Modulation Module layers into the model. Along with this, the utilization of subnetwork masks may have also caused the weights learned in the Modulation Module layers during the pre-mask phase of training to become unlearned as the neural network attempts to adjust its weights to accommodate the zeroed-out neurons during the post-mask phase. It may be possible to solve this problem by tuning the α value used during the computation of subnetwork masks so as not to zero out too many neurons to promote constructive interaction between these two mitigation methods.

Specifically, a hybrid approach combining PCGrad and LaSS manages to achieve a task-averaged F1 score of 0.84686 on an MTL image classification problem, outperforming individual mitigation approaches by 0.01870, 0.02682, and 0.02434 for PCGrad, Modulation Module, and LaSS respectively. Comparatively, a baseline convolutional neural network with an identical architecture but without any gradient conflict mitigation methods only achieves an F1 score of 0.77602, which is 0.07084 below the best hybrid approach. Furthermore, the research also notes the existence and provides possible explanations for destructive interactions between combinations of mitigation methods. Such interaction is the addition of Modulation Module layers to a model incorporating PCGrad and LaSS, which results in a degradation in the F1 score of the model by 0.01931.

IV. Conclusion

Using MTL-based neural networks to solve unrelated tasks may lead to performance degradation of one or even all tasks. Fortunately, previously developed mitigation approaches exist and effectively alleviate this problem. The research benchmarks different combinations of three previously developed mitigation approaches. It empirically verifies the possibility of gaining significant performance improvements by combining a subset of these approaches in accordance with the hypothesis that motivates the research.

The results open an avenue for future research to analyze how different mitigation methods interact. Such research may further lead to the discovery of better hybrid gradient conflict mitigation approaches. It can suggest the possibility of achieving increased performance across many MTL models by preventing performance degradation caused by unrelated tasks.

Author Contribution

Writing—original draft, R. A.; Methodology, R. A., W. J., and D. S.; Formal analysis, R. A., and D. S.;

Analysis result review, D. S. All authors have read and agreed to the published version of the manuscript.

REFERENCES

[1] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 12, pp. 5586–5609, 2021.

[2] R. Caruana, "Multitask learning," *Machine Learning*, vol. 28, pp. 41–75, 1997.

[3] D. S. Chauhan, S. R. Dhanush, A. Ekbal, and P. Bhattacharyya, "All-in-one: A deep attentive multi-task learning framework for humour, sarcasm, offensive, motivation, and sentiment on memes," in *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, Suzhou, China, 2020, pp. 281–290.

[4] V. Davoodnia and A. Etemad, "Identity and posture recognition in smart beds with deep multitask learning," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. Bari, Italy: IEEE, Oct. 6–9, 2019, pp. 3054–3059.

[5] M. Pisov, G. Makarchuk, V. Kostjuchenko, A. Dalechina, A. Golanov, and M. Belyaev, "Brain tumor image retrieval via multitask learning," 2018. [Online]. Available: https://arxiv.org/abs/1810.09369

[6] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Salt Lake City, Utah, June 19–21, 2018, pp. 7482–7491.

[7] X. Zhao, H. Li, X. Shen, X. Liang, and Y. Wu, "A modulation module for multi-task learning with applications in image retrieval," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany, Sept. 8–14, 2018, pp. 401–416.

[8] S. Sridhar and S. Sanagavarapu, "Fake news detection and analysis using multitask learning with BiLSTM CapsNet model," in *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. Noida, India: IEEE, Jan. 28–29, 2021, pp. 905–911.

[9] Y. Liu, W. M. Sid-Lakhdar, O. Marques, X. Zhu, C. Meng, J. W. Demmel, and X. S. Li, "GPTune: Multitask learning for autotuning exascale applications," in *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, Virtual Event Republic of Korea, Feb. 2021, pp. 234–246.

[10] K. Zhang, L. Wu, Z. Zhu, and J. Deng, "A multitask learning model for traffic flow and speed forecasting," *IEEE Access*, vol. 8, pp. 80 707–80 715, 2020.

[11] J. Li, X. Shao, and R. Sun, "A DBN-based deep neural network model with multitask learning for online air quality prediction," *Journal of Control Science and Engineering*, vol. 2019, pp. 1–9, 2019.

[12] Y. Li, J. Song, W. Lu, P. Monkam, and Y. Ao, "Multitask learning for super-resolution of seismic velocity model," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, no. 9, pp. 8022–8033, 2020.

[13] J. Zhang, K. Yan, and Y. Mo, "Multi-task learning for sentiment analysis with hard-sharing and task recognition mechanisms," *Information*, vol. 12, no. 5, pp. 1–13, 2021.

[14] F. Tao and C. Busso, "End-to-end audiovisual speech recognition system with multitask learning," *IEEE Transactions on Multimedia*, vol. 23, pp. 1–11, 2020.

[15] K. H. Thung and C. Y. Wee, "A brief review on multi-task learning," *Multimedia Tools and Applications*, vol. 77, no. 22, pp. 29 705–29 725, 2018.

[16] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn, "Gradient surgery for multi-task learning," *Advances in Neural Information Processing Systems*, vol. 33, pp. 5824–5836, 2020.

[17] Z. Lin, L. Wu, M. Wang, and L. Li, "Learning language specific sub-network for multilingual machine translation," 2021. [Online]. Available: https://arxiv.org/abs/2105.09259

[18] S. S. Chouhan, U. P. Singh, A. Kaul, and S. Jain, "A data repository of leaf images: Practice towards plant conservation with plant pathology," in *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*. Mathura, India: IEEE, Nov. 21–22, 2019, pp. 700–707.

[19] W. C. Tseng, "WeiChengTseng/Pytorch-PCGrad," 2020. [Online]. Available: https://github.com/WeiChengTseng/Pytorch-PCGrad.git