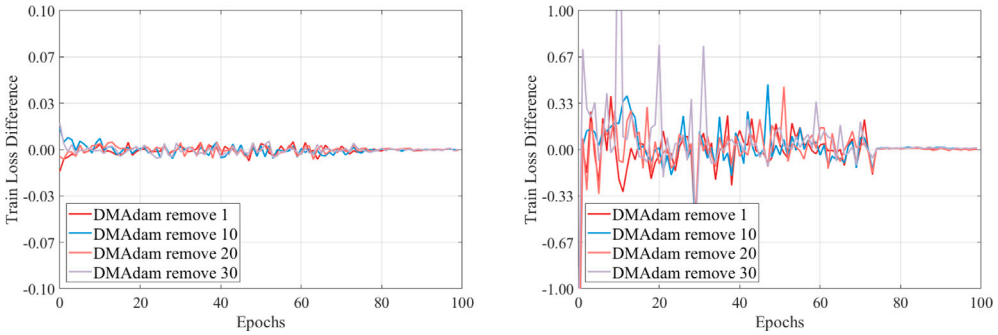


Fig. 8. Uniform stability error for DMAdam.

Proof. See Appendix B.3. □

In fact, the conditions in Theorem 3 are the same as those in Theorem 1. Therefore, to satisfy these conditions, it is sufficient to choose $\eta_k = \frac{\eta_0}{\sqrt{k}}$ and $\beta_k = 1 - \frac{1}{\sqrt{k}}$, where η_0 is a positive constant.

4.3. Uniform stability

Uniform stability provides a quantitative measure of the sensitivity of an algorithm to small changes in the dataset. Consider two samples S and S' of size n differing by one example, and let $\boldsymbol{x} = P(S)$ denote the output of a (potentially randomized) algorithm P (e.g., DMAdam) on data S [33].

Definition 1 ([34]). Algorithm P is ϵ-uniformly stable if

$$\epsilon_{stab} = \sup_{S,S'} \sup_{\xi \in \mathbb{P}} \mathbb{E}_P \left[\ell\left(P(S); \xi\right) - \ell\left(P\left(S'\right); \xi\right) \right] \leq \epsilon, \tag{17}$$

where ϵ_{stab} denotes the upper bound on the expected difference in the loss ℓ between outputs of P on samples S and S' that differ by only one example.

Fig. 8 illustrates the performance of VGG-13 trained on the CIFAR-10 dataset, with the curves highlighting differences between datasets with deleted samples and the complete dataset. Deleting a small number of samples (e.g., 1, 10, 20, or 30) from the training set has minimal impact on the training loss, indicating that DMAdam is insensitive to small data variations and can maintain stable model outputs. On the test set, the loss difference initially oscillates but gradually diminishes as training progresses, eventually approaching zero, indicating that the algorithm exhibits uniform stability.

5. Experiment

This section evaluates the performance of DMAdam through a series of numerical experiments. A comparative analysis is conducted with several algorithms, including SGD [5], Adam [11], NAdam [12], AMSGrad [13], MSVAG [14], AdamW [15], RAdam [16], AdaBelief [17], AdamP [18], Adan [19], and Lion [35]. Table 2 provides the hyperparameter settings employed in the DMAdam algorithm.

We obtain the hyperparameters in Table 2 from grid search results and optimize hyperparameters based on performance evaluations across various models and datasets. The hyperparameter η_k typically does not have a determining effect on experimental results, and using the default value is sufficient (see Fig. 17). However, fine-tuning η_k may improve experimental performance. In contrast, α_k is the most important hyperparameter and is usually preferred for tuning. Refer to Table 2 for α_k adjustment suggestions. To simplify the hyperparameter tuning process for researchers, η_k can be viewed as similar to β_1 in Adam, while α_k corresponds to the learning rate. The value of β_k is fixed at 0.999 and ϵ is set at 10^{-8}.

Table 2
Hyperparameter settings for the DMAdam.

Model	Dataset	η_k	α_k	β_k	ϵ	Weight decay
VGG-13	CIFAR-10	0.9	0.001	0.999	10^{-8}	0.0005
	CIFAR-100	2.1	0.001	0.999	10^{-8}	0.0005
ResNet-34	CIFAR-10	1.4	0.001	0.999	10^{-8}	0.0005
	CIFAR-100	1.2	0.001	0.999	10^{-8}	0.0005
LSTM-1	PTB	0.8	0.1	0.999	10^{-8}	0.0000012
LSTM-2		1.9	0.1	0.999	10^{-8}	0.0000012
LSTM-3		2.5	0.1	0.999	10^{-8}	0.0000012
YOLOv5 (s)	VOC (2007)	0.037	0.001	0.999	10^{-8}	0.0005
MobileNet-V3-L	Tiny ImageNet	1.5	0.005	0.999	10^{-8}	0.0001

To ensure the fairness and accuracy of the comparison, we use the parameters adopted for the comparison algorithms in their original papers. When the comparison algorithms fail to perform well on our task, we adjust their learning rates to optimize performance. Further details can be consulted in Appendix C.

5.1. Image classification experiment results

• Performance on CIFAR-10 Dataset

CIFAR-10, a widely used dataset for image classification, consists of 60,000 color images (32 × 32 pixels) distributed across 10 classes. In this study, we evaluate the performance of DMAdam against other optimization algorithms using VGG [36] and ResNet [37] models. As shown in Figs. 9 and 10, DMAdam achieves fast convergence and high accuracy during the training and testing of VGG-13 and ResNet-34. DMAdam outperforms traditional algorithms such as SGD and Adam, as well as recent improvements like RAdam, AdamP, and AdaBelief. After learning rate decay, DMAdam effectively mitigates oscillations and demonstrates high accuracy and stability in deep neural networks.

• Performance on CIFAR-100 Dataset

CIFAR-100 is an extension of CIFAR-10, consisting of 100 classes and providing a more challenging benchmark for optimization algorithms. As shown in Figs. 11 and 12, DMAdam performs similarly on the CIFAR-100 dataset as it does on CIFAR-10 when applied to VGG-13 and ResNet-34 models. DMAdam achieves higher accuracy compared to traditional algorithms like SGD and Adam, as well as improved methods such as AdamW, RAdam, and AdamP. Additionally, after 150 epochs of learning rate decay, DMAdam effectively maintains stability.

• Performance on Tiny ImageNet

Tiny ImageNet, a derivative of the ImageNet dataset, is widely used for training and evaluating deep learning models. It contains 200 categories, each comprising 500 training images, 50 validation images, and 50 unannotated test images at a resolution of 64 × 64 pixels. Since the test set lacks annotations, the validation set is commonly employed for evaluation. Despite its smaller scale, Tiny ImageNet

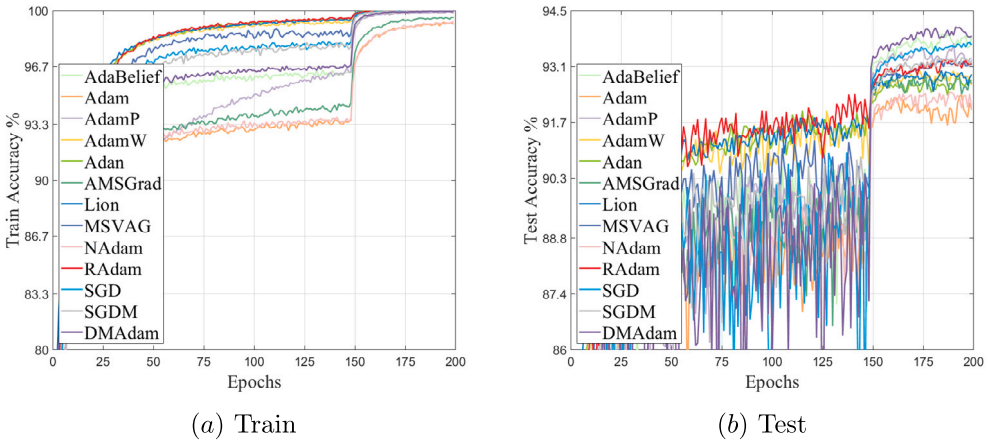


Fig. 9. VGG-13 performance on CIFAR-10.

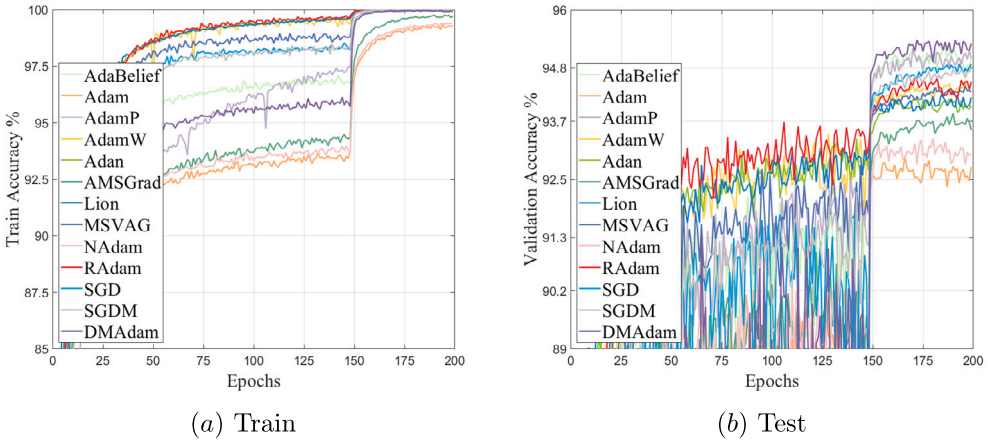


Fig. 10. ResNet-34 performance on CIFAR-10.

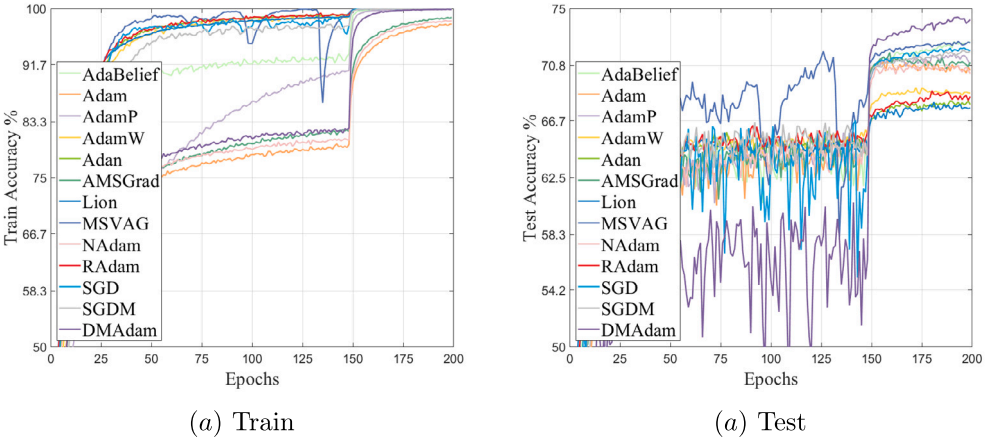
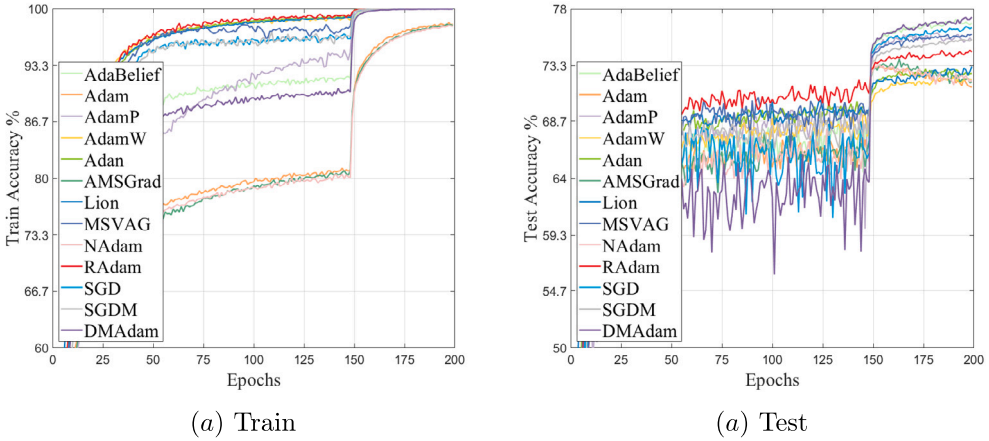


Fig. 11. VGG-13 performance on CIFAR-100.

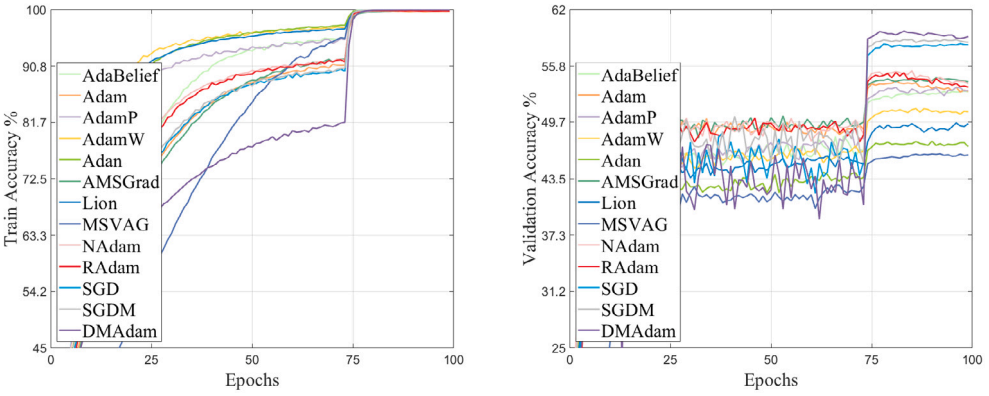
retains the complexity of multi-class classification tasks, making it a popular benchmark for assessing algorithm performance. Its structured format and category diversity further enhance its utility in model evaluation.

As shown in Fig. 13, the MobileNet-V3-Large model is trained on the Tiny ImageNet dataset. DMAdam begins with slower performance compared to Adam and AdamW in the early training stages but steadily improves, maintaining a stable trajectory and mitigating the risk of



(a) Train (a) Test

Fig. 12. ResNet-34 performance on CIFAR-100.



(a) Train (b) Validation

Fig. 13. MobileNet-V3-L performance on Tiny ImageNet.

early overfitting. On the validation set, DMAdam exhibits oscillations similar to SGD. Following the learning rate decay at the 75th epoch, its smooth validation curve highlights its robustness, establishing it as a reliable algorithm for training complex models on challenging datasets such as Tiny ImageNet.

5.2. Natural language processing experiment results

Natural language processing (NLP) tasks involving the analysis and generation of human language present challenges in modeling complex sequential data. To evaluate the effectiveness of various optimization algorithms, we train the LSTM model on the Penn Treebank (PTB) dataset, a widely recognized benchmark in NLP research. For additional experiments using Transformer-based models, please refer to Appendix C.

Fig. 14 presents the results of experiments conducted using LSTM models with varying numbers of layers. Compared to other optimization algorithms, DMAdam exhibits a more significant perplexity decline and achieves lower final perplexity for LSTM-1, LSTM-2, and LSTM-3. DMAdam exhibits an oscillatory descent in the early stages, with perplexity further decreasing after two learning rate decays. These results indicate that DMAdam is well-suited for natural language processing tasks, effectively managing the complexities associated with deeper models.

5.3. Object detection experiment results

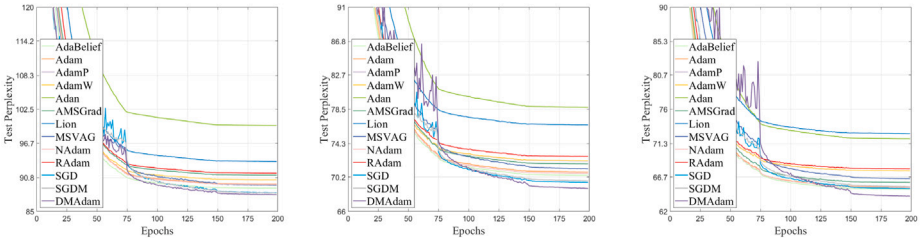
Object detection aims to identify and locate objects within an image by providing class labels and bounding boxes. For this study, we utilize the VOC (2007) dataset, a widely recognized benchmark featuring 20 object categories and detailed annotations. To evaluate performance, we employ YOLOv5,[2] a state-of-the-art real-time object detection model known for its speed and accuracy.

Fig. 15 shows the confusion matrix comparing the prediction results of Adam and DMAdam in the object detection task. Diagonal values represent the probability of correct classification for each category, while off-diagonal values indicate the likelihood of misclassification between categories. As shown in Table 3, DMAdam achieves higher mean average precision (mAP), demonstrating improved performance in object detection tasks. Additionally, Fig. 16 illustrates the classification accuracy at an Intersection Over Union (IoU) threshold of 0.5.

5.4. Sensitivity analysis of η_k

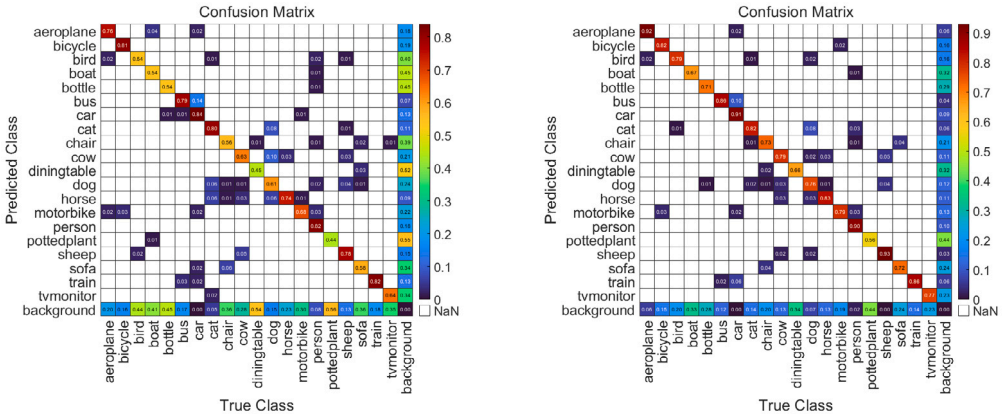
We evaluate the performance of Algorithm 1 under different η_k values on the CIFAR-100 and PTB datasets. To ensure fairness, the

[2] https://github.com/ultralytics/yolov5.git



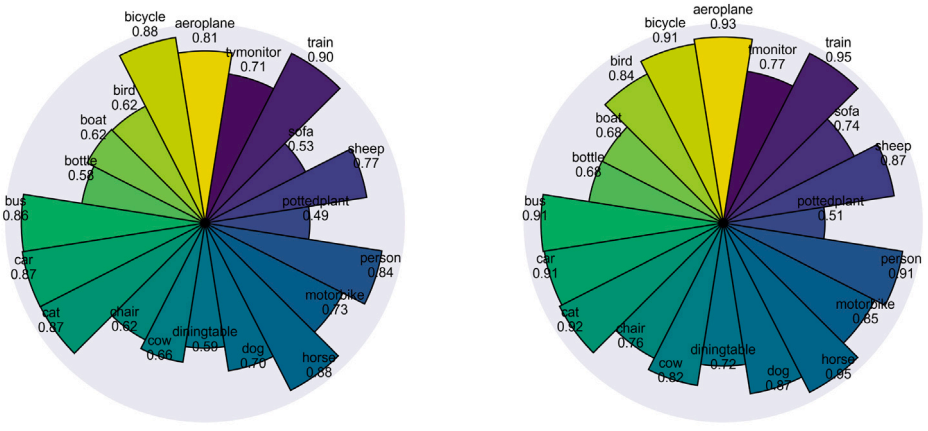
(a) Single-layer (b) Two-layer (c) Three-layer

Fig. 14. Perplexity results for LSTM on the Penn Treebank dataset.



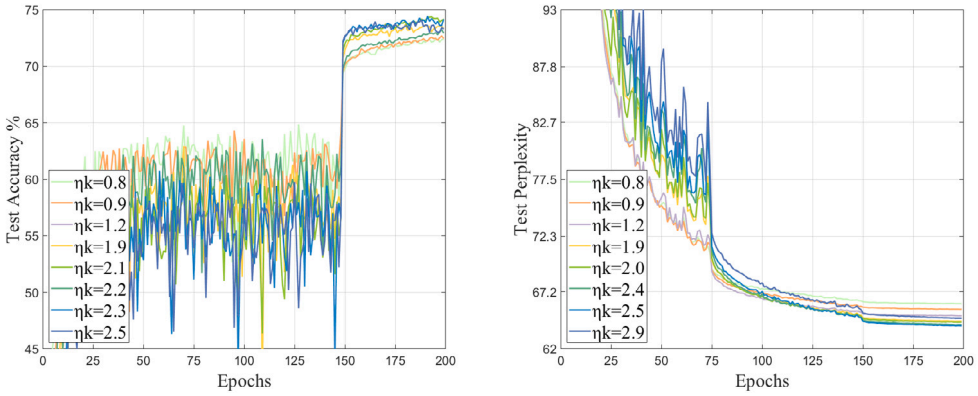
(a) Adam (b) DMAdam

Fig. 15. YOLOv5 (s) confusion matrix results on the VOC dataset.



(a) Adam (b) DMAdam

Fig. 16. Scores by category (mAP@0.5) using YOLOv5 (s) based on VOC.



(a) VGG-13 (b) LSTM-3

Fig. 17. DMAdam performance under different η_k values.

Table 3
YOLOv5 (s) mAP on VOC (higher scores are better).

Optimizer	SGD	Adam	NAdam	AMSGrad	MSVAG	AdamW
mAP	0.518	0.478	0.467	0.471	0.501	0.484
Optimizer	RAdam	AdaBelief	Lion	AdamP	Adan	DMAdam
mAP	0.511	0.506	0.211	0.456	0.518	0.566

experiments are conducted with the same hyperparameter settings (α_k, β_k, ϵ) over 200 epochs, and the results are illustrated in Fig. 17. We observe that the impact of different η_k values on Algorithm 1 is slight. This indicates that using the default η_k is practical for most cases, but fine-tuning may lead to better results.

6. Conclusion

We have presented a novel algorithm that integrates the dual averaging strategy with the Adam method. We have performed a convergence analysis of DMAdam in two cases. In the ergodic case, we achieve the $\mathcal{O}\left(\ln K/\sqrt{K}\right)$ convergence rate in the expectation sense. In the non-ergodic case, we prove the convergence of the gradient sequence for the last iteration. Furthermore, numerical experiments indicate that DMAdam outperforms other optimization algorithms for various well-known deep neural network models, including VGG, ResNet, LSTM, and Yolov5. Future work could focus on exploring adaptive hyperparameter tuning strategies and investigating their theoretical properties in different optimization settings.

CRediT authorship contribution statement

Wenhan Jiang: Software, Methodology, Investigation, Formal analysis. **Jinlan Liu:** Writing – review & editing, Writing – original draft. **Naimin Zhang:** Writing – review & editing, Writing – original draft. **Dongpo Xu:** Writing – review & editing, Writing – original draft, Methodology, Funding acquisition, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors would like to thank the three anonymous reviewers for their insightful comments and suggestions, which have led to important improvements. The authors thank Dr. Kasper Karlsson for polishing the manuscript.

Appendix A. Technical lemmas

Lemma 1. *Let $(a_k)_{k\geq 1}$ and $(b_k)_{k\geq 1}$ be non-increasing and non-negative sequences of real values, respectively. Then for any $0 \leq \beta < 1$, we have*

$$\sum_{k=1}^{K} a_k \sum_{j=1}^{k} \beta^{k-j} b_{j-1} \leq \sum_{k=1}^{K} \sum_{j=1}^{k} \beta^{k-j} a_j b_{j-1}, \tag{18}$$

$$\sum_{k=1}^{K} a_k \sum_{j=1}^{k} \beta^{k-j} b_j \leq \sum_{k=1}^{K} \sum_{j=1}^{k} a_j \beta^{k-j} b_j. \tag{19}$$

Proof. (1) For the inequality in Eq. (18),

$$\begin{aligned}
&\sum_{k=1}^{K} a_k \sum_{j=1}^{k} \beta^{k-j} b_{j-1} = \sum_{k=1}^{K} \left(a_k \sum_{j=1}^{k} \beta^{k-j} b_{j-1} \right) \\
&= a_1 \beta^0 b_0 + a_2 \left(\beta^1 b_0 + \beta^0 b_1 \right) + \cdots + a_K \left(\beta^{K-1} b_0 + \beta^{K-2} b_1 + \cdots + \beta^0 b_{K-1} \right) \\
&= \left(a_1 \beta^0 + a_2 \beta^1 + \cdots + a_K \beta^{K-1} \right) b_0 + \left(a_2 \beta^0 + \cdots + a_K \beta^{K-2} \right) b_1 + \cdots \\
&\quad + a_K \beta^0 b_{K-1} \\
&\overset{(a)}{\leq} \left(a_1 \beta^0 + a_1 \beta^1 + \cdots + a_1 \beta^{K-1} \right) b_0 + \left(a_2 \beta^0 + \cdots + a_2 \beta^{K-2} \right) b_1 + \cdots \\
&\quad + a_K \beta^0 b_{K-1} \\
&= \beta^0 a_1 b_0 + \left(\beta^1 a_1 b_0 + \beta^0 a_2 b_1 \right) + \cdots
\end{aligned}$$