When Creative Machines Learn from Each Other

Haven Kim[1] **Yusong Wu**[2] **Taylor Berg-Kirkpatrick**[1] **Julian McAuley**[1]
[1]University of California San Diego
[2] Mila - Quebec AI Institute, Université de Montréal

Abstract

We explore the potential for creative neural network models to learn from each other. To enable this, we train two policy models using the exactly same architectures, configurations, and data, but with different random seeds. Then, we obtain a judge model that automatically rates the performance. We take samples from each policy model, rate them, and optimize each model by maximizing the probability of the better sample and minimizing the probability of the worse sample, using a popular model alignment technique, Direct Preference Optimization (DPO), in an online manner. The results show that our approach effectively improves the performance of models in three distinct, open-ended creative tasks: symbolic music generation, lyric generation, and lyric translation. However, it shows minimal benefit for a closed-ended task, Georgian-to-English machine translation. We will release the Python implementation upon the paper's acceptance.

1 Introduction

Learning from each other has been established to be a crucial component in human creativity development, according to psychological and pedagogical literature [25, 2]. Humans share ideas to enhance the aesthetic quality of their creative work. This process differs fundamentally from solving mathematical problems or translating languages [8], which typically have determinate solutions where learning from the answer key might be quicker than learning from others [3].

Building on insights from these literatures, we explore the potential for creative neural network models to learn from one another. To enable this, we train two policy models using the exactly same configurations, architecture, and data, but with different random seeds. We take samples from each policy model, where the samples are rated by a neural or non-neural judge model in an online manner. Each model is optimized to maximize the probability of the better sample and minimize the probability of the worse sample, using a popular model alignment technique, Direct Preference Optimization (DPO) [22].

Our co-training framework may superficially resemble two-player game models, representatively Adversarial Artificial Curiosity (AC) [26], Predictability Minimization (PM) [27] and Generative Adversarial Networks (GANs) [7] where two different networks are simultaneously trained to achieve better performance than its counterpart. However, a key limitation of these frameworks is the difficulty in balancing two models, often leading to training instability and mode collapse. In contrast, our co-training mechanism is designed to be inherently self-balancing as co-trained models have been pre-trained using the same architecture, methods, and data.

We evaluate our method across diverse set of tasks with varying degress of creative demand. The results show that the proposed framework effectively improves the performance of models in three distinct, open-ended creative tasks: symbolic music generation, lyric generation, and lyric translation. Moving further, we demonstrate that the method's benefits compared to existing methods are confined to creative tasks, showing minimal benefit on a closed-ended task, conventional machine translation.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: AI for Music.

2 Methods and Experiments

2.1 MultiOnlineDPO

We enable neural networks to learn from each other, with a simple method we call **MultiOnlineDPO**. In MultiOnlineDPO, we train two policy models with the same architectures, configurations, and data, but with different random seeds. Therefore, the two policy models are expected to have different weights due to random initialization, but their overall performance is expected to be comparable. Then, we obtain a judge model which automatically rates the performance of policy models. A judge model can be a neural model, a non-neural model, or a combination of both. During the DPO stage, we take samples from each policy models and the samples are rated by the pre-defined judge model. Using the DPO loss [22], both models are optimized to maximize the probability of the winner sample and minimize the probability of the loser sample.

2.2 Tasks and Implementation Details

To assess the effectiveness of MultiOnlineDPO, we apply this method to four different tasks:.

In **Symbolic Music Generation**, the objective of the policy model is to generate the next symbolic music sequence given a preceding sequence. Each policy model adopts a LLaMA architecture [28] with a pre-trained tokenizer (`Natooz/Maestro-REMI-bpe20k`) [6]. During the DPO phase, each model generates 128 tokens conditioned on the preceding 128 tokens which are optimized to best follow the prompt tokens. In **Lyric Generation**, the policy model aims to produce subsequent lyric lines in English, adhering to specified syllable constraints, given an initial line of lyrics. We employ an encoder–decoder architecture [23] for training the policy model, incorporating syllable count tokens [9], where <SYLn> specifies that the corresponding lyric line should contain n syllables. For example, the input "<SYL3> One more time <SYL5> <SYL6> " instructs the model to generate two lyric lines following "One more time", which consists of three syllables, with respective syllable counts of 5 and 6. In **Lyric Translation**, the policy model aims to translate Korean lyrics into English lyrics while maintaining syllable count constraints. Note that the task is inherently open-ended because lyric translation often avoids literal translation in order to meet syllable count constraints and preserve singability [19, 15]. Following the approach of [14], we train two policy models based on the MarianMT architecture [12]. Similar to the lyric generation task, the syllable counts of translated lyrics are controlled using syllable count tokens <SYLn> [9]. Following the previous approach [14], training is conducted in three stages. In the first stage, the model is trained on conventional Korean-to-English machine translation data. In the second stage, it is trained on non-singable lyric translation data, in which the target texts consist of English commercial lyrics and the source texts are their Korean machine translation. In the final stage, it is trained on singable lyric translation data where Korean and English singable lyrics for the same songs are paired section-by-section. In **Conventional Machine Translation**, a policy model translates Georgian to English with a pre-trained Georgian encoder (`Davit6174/georgian-distilbert-mlm`) and pre-trained English decoder (`bert-base-uncased`) [5].

Except for the final stage of lyric translation, where the model is trained for one epoch without a validation stage, all policy models are trained on the training set with validation performed every epoch. Training stops when the validation loss fails to improve for three consecutive epochs. Detailed training configurations for each policy model are provided in Appendix A.

During the DPO stage, the learning rate is fixed at 1×10^{-6} for all tasks. Each training sample goes through a single DPO cycle, consisting of sampling, rating with the judge model, and optimization, using the DPO loss with a batch size of 32.

2.3 Design of Judge Model

When designing the judge models, we adhere to one key requirement: the Best of 2 approach, where two policy models generate outputs independently and the judge model selects the better output for each item in the evaluation set, must achieve higher scores across all objective metrics than any of each policy model. This criterion ensures that the judge model can distinguish and select higher-quality outputs, at least to some extent. In Appendix B, we compare the detailed objecctive evaluation results of the Best-of-2 approach with those of the two policy models to demonstrate that our judge models meet this criterion.

In **Symbolic Music Generation**, the judge model is a neural network that estimates the probability that two sequences are consecutive, where each sequence consists of 128 tokens. For its training, we employ a BERT-based architecture [5] on pairs of sequences drawn from the training and validation sets, labeling them as 1 if they are truly consecutive and 0 if randomly sampled from different tracks. In **Lyric Generation** and **Lyric Translation**, we consider both syllable-count fidelity and semantic similarity. The judge model evaluates performance using the formula $SCD \times (Sim_s - 1)$, where SCD denotes the Syllable Count Distance [15], and Sim_s represents sentence similarity. Specifically, Sim_s is defined as the cosine similarity between embeddings of the ground truth and the generated lyrics, computed using the same pre-trained SentenceBERT model (`all-MiniLM-L6-v2`) [24]. In **Conventional Machine Translation**, the judge model assesses the performance via sentence similarity, using the same pre-trained SentenceBERT model as in the previous tasks.

2.4 Dataset

For **Symbolic Music Generation**, the policy models are trained and validated on the MAESTRO [11] and GiantMIDI-Piano [16] datasets using a 9:1 split. Evaluation is performed on Aria-MIDI [1]. For each song, the first 128 tokens are used as prompt tokens, and the subsequent 128 tokens serve as ground-truth tokens. For **Lyric Generation**, we divide LyCon [13], a dataset of artificially generated lyrics for 7,863 song, into training, validation, and evaluation sets with an 8:1:1 ratio. For **Lyric Translation**, following previous work [14], we use three stages of data: Korean-to-English machine translation data (500k sentence pairs [21]), non-singable lyric translation data (English commercial songs and their Korean machine translation), and singable lyric translation data [14]. Machine translation and non-singable lyric translation datasets are split into 9:1 ratios for training and validation. The singable lyric translation dataset is split into a 9:1 ratio for training and evaluation. For **Conventional Machine Translation**, we obtain the OpenSubtitles dataset [17] and split Georgian-English pairs into 8:1:1 for training, validation, and evaluation.

2.5 Evaluation Metrics

In **Symbolic Music Generation**, we compare the distributions of ground-truth and generated music (both consisting of 128 tokens) with respect to pitch and beat. Specifically, we compute the distributions of pitch and beat (quantized to 1/16 beat) and evaluate their differences using KL divergence. We refer to these as Pitch Divergence and Beat Divergence, respectively. In **Lyric Generation** and **Lyric Translation**, we compare the ground truth and generated lyrics in terms of syllable count and semantic similarity. For syllable count evaluation, we use the Syllable Count Distance (SCD) metric [15] to measures differences in syllable counts. Semantic similarity is measured using BERTScore [1] [29]. In **Conventional Machine Translation**, we evaluate translational quality using n-gram–based metrics, specifically BLEU ($n = 1, 2$) [20], along with one neural metric, BERTScore [29], by comparing the translations to the ground truth.

2.6 Models for Comparison

We evaluate our method, **MultiOnlineDPO**, by comparing it against alternative models to assess its effectiveness:

Baseline refers to a policy model that has not undergone the DPO stage. Specifically, upon training two policy models with different random seeds, we define the primary baseline as the model that attains a higher win rate when tested on the evaluation set—according to the judge model—while the secondary baseline refers to the other model. Their performances are expected to be comparable, given the symmetry in training methods, datasets, and architectures; the win rates of the primary baseline models are 50.97%, 52.48%, 54.01%, and 52.15% in symbolic music generation, lyric generation, lyric translation, and conventional machine translation, respectively.

SelfOnlineDPO is similar to our method, but both samples are generated by a primary baseline model. It is worth noting that this approach is essentially equivalent to the previously proposed method, known as Online DPO [4, 10]. Each cycle, comprising sampling, rating, and optimization, is performed with a batch size of 32, consistent with our method.

[1] We use `FacebookAI/roberta-large` [18] embeddings to compute BERTScore, and report the F1 variant throughout this paper.

Table 1: Objective Evaluation Results

	Pitch Divergence (↓)	**Beat Divergence** (↓)
Baseline (Primary)	13.04	9.65
Baseline (Secondary)	13.05	9.69
SelfOnlineDPO	12.78	9.27
MultiOnlineDPO	**12.71**	**9.09**

(a) Symbolic Music Generation

	SCD (↓)	**BERTScore** (↑)
Baseline (Primary)	10.97	85.70
Baseline (Secondary)	11.37	85.70
SelfOnlineDPO	10.84	85.74
MultiOnlineDPO	**9.53**	**85.84**

(b) Lyric Generation

	SCD (↓)	**BERTScore** (↑)
Baseline (Primary)	36.77	81.61
Baseline (Secondary)	40.83	**81.89**
SelfOnlineDPO	51.26	81.71
MultiOnlineDPO	**32.80**	81.74

(c) Lyric Translation

	BLEU-1 (↑)	**BLEU-2** (↑)	**BERTScore** (↑)
Baseline (Primary)	23.66	15.37	85.29
Baseline (Secondary)	23.54	15.10	85.06
SelfOnlineDPO	**25.65**	**16.77**	85.41
MultiOnlineDPO	25.30	16.45	**85.49**

(d) Conventional Machine Translation

In the following section, we report the performance of the models with the lowest validation loss. For the SelfOnlineDPO and MultiOnlineDPO models, validation is performed every 100 cycles. For MultiOnlineDPO, we report the performance of the primary model after the DPO stage.

3 Results and Analysis

We provide evaluation results in Table 1. Overall, SelfOnlineDPO proves effective in improving performance. In particular, it yields consistent gains across two creative tasks (symbolic music generation and lyric generation) and one closed-ended task (machine translation). However, in lyric translation, convergence was unstable: minimal improvements in BERTScore came at the cost of significantly higher syllable count distance compared to the baseline models. In contrast, MultiOnlineDPO achieves improvements over SelfOnlineDPO across all creative tasks, including symbolic music generation, lyric generation, and lyric translation. Notably, unlike SelfOnlineDPO, MultiOnlineDPO simultaneously reduces syllable count distance while achieving higher semantic similarity than SelfOnlineDPO. However, in conventional machine translation, MultiOnlineDPO does not provide a clear advantage. Although it improves BERTScore relative to SelfOnlineDPO, BLEU scores actually decline. We hypothesize that this is due to the limited number of valid outputs in machine translation. SelfOnlineDPO alone appears sufficient to produce a diverse enough dataset. In MultiOnlineDPO, many instances involved the primary and secondary baseline models producing identical outputs, leaving the system with little to learn. As a result, the benefit of incorporating multiple models is minimal in this setting.

In contrast, three applications in open-ended tasks show clear advantages for MultiOnlineDPO where the primary and secondary baseline models almost always generate distinct outputs even under identical inference conditions. This diversity enables each model to access higher-quality and varied samples than the SelfOnlineDPO approach, which must generate data from a single model. The limited success of SelfOnlineDPO in lyric translation further supports this view: MultiOnlineDPO models appear better able to access the higher-quality data needed for stable convergence, while SelfOnlineDPO appears to produce lower-quality samples when relying solely on its own generations.

4 Conclusion

In this paper, we introduced MultiOnlineDPO, an approach in which alignment samples are drawn from two independently trained models with identical architectures, configurations, and data but different random seeds, rather than from a single model, essentially enabling the models to learn from one another. We show that this method is particularly effective for open-ended, creative tasks—those requiring divergent rather than convergent thinking in educational psychology terms [8]—as demonstrated in applications such as symbolic music generation, lyric generation, and lyric translation. We hypothesize that the diversity introduced by different random initializations of the two models contributes to the improved performance in such tasks. In contrast, because SelfOnlineDPO models can already generate sufficiently diverse data for closed-ended tasks using a single model, MultiOnlineDPO did not yield significant improvements in those cases.

References

[1] Louis Bradshaw and Simon Colton. Aria-midi: A dataset of piano midi files for symbolic music modeling, 2025.

[2] Paula Catarino, Paulo Vasco, José Lopes, Helena Silva, and Eva Morais. Cooperative learning on promoting creative thinking and mathematical creativity in higher education. *REICE. Revista Iberoamericana Sobre Calidad, Eficacia y Cambio En Educacion*, 17(3):5–22, 2019.

[3] Ouhao Chen, Endah Retnowati, BoBo Kai Yin Chan, and Slava Kalyuga. The effect of worked examples on learning solution steps and knowledge transfer. *Educational Psychology*, 43(8):914–928, 2023.

[4] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[6] Nathan Fradet, Nicolas Gutowski, Fabien Chhel, and Jean-Pierre Briot. Byte pair encoding for symbolic music. *arXiv preprint arXiv:2301.11975*, 2023.

[7] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[8] Joy Paul Guilford. The structure of intellect. *Psychological bulletin*, 53(4):267, 1956.

[9] Fenfei Guo, Chen Zhang, Zhirui Zhang, Qixin He, Kejun Zhang, Jun Xie, and Jordan Boyd-Graber. Automatic song translation for tonal languages. *arXiv preprint arXiv:2203.13420*, 2022.

[10] Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Rame, Thomas Mesnard, Yao Zhao, Bilal Piot, Johan Ferret, and Mathieu Blondel. Direct language model alignment from online ai feedback, 2024.

[11] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*, 2019.

[12] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, et al. Marian: Fast neural machine translation in c++. *arXiv preprint arXiv:1804.00344*, 2018.

[13] Haven Kim and Kahyun Choi. Lycon: Lyrics reconstruction from the bag-of-words using large language models. *arXiv preprint arXiv:2408.14750*, 2024.

[14] Haven Kim, Jongmin Jung, Dasaem Jeong, and Juhan Nam. K-pop lyric translation: Dataset, analysis, and neural-modelling. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 9974–9987, May 2024.

[15] Haven Kim, Kento Watanabe, Masataka Goto, and Juhan Nam. A computational evaluation framework for singable lyric translation. *arXiv preprint arXiv:2308.13715*, 2023.

[16] Qiuqiang Kong, Bochen Li, Jitong Chen, and Yuxuan Wang. Giantmidi-piano: A large-scale midi dataset for classical piano music. *arXiv preprint arXiv:2010.07061*, 2020.

[17] Pierre Lison and Jörg Tiedemann. OpenSubtitles2016: Extracting large parallel corpora from movie and TV subtitles. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 923–929, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA).

[18] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[19] Peter Low. Singable translations of songs. *Perspectives: Studies in Translatology*, 11(2):87–103, 2003.

[20] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[21] Chanjun Park, Midan Shim, Sugyeong Eo, Seolhwa Lee, Jaehyung Seo, Hyeonseok Moon, and Heuiseok Lim. Empirical analysis of parallel corpora and in-depth analysis using liwc. *Applied Sciences*, 12(11):5545, 2022.

[22] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

[23] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[24] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[25] Karin Scager, Johannes Boonstra, Ton Peeters, Jonne Vulperhorst, and Fred Wiegant. Collaborative learning in higher education: Evoking positive interdependence. *CBE—Life Sciences Education*, 15(4):ar69, 2016.

[26] Jürgen Schmidhuber. *Making the world differentiable: on using self supervised fully recurrent neural networks for dynamic reinforcement learning and planning in non-stationary environments*, volume 126. Inst. für Informatik, 1990.

[27] Jürgen Schmidhuber. Learning factorial codes by predictability minimization. *Neural computation*, 4(6):863–879, 1992.

[28] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[29] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.

A Detailed Baseline Model Configurations

A.1 Symbolic Music Generation

Model Architecture

- **Tokenizer**: Pre-trained BPE tokenizer (`Natooz/Maestro-REMI-bpe20k`) [6] with vocabulary size of 20,000
- **Hidden Size**: 768
- **Intermediate Size**: 3,072 (4$\times$ hidden size for FFN)
- **Number of Layers**: 12
- **Attention Heads**: 12
- **Key-Value Heads**: 12 (no grouped-query attention)
- **Activation Function**: SiLU (SwiGLU)
- **Position Embeddings**: Rotary Position Embedding (RoPE) with $\theta = 10,000$
- **Maximum Position**: 2,048
- **Normalization**: RMSNorm with $\epsilon = 10^{-6}$
- **Attention Dropout**: 0.1

Training Configuration

- **Batch Size**: 32
- **Learning Rate**: 10^{-4}
- **Weight Decay**: 0.01
- **Optimizer**: AdamW
- **Warmup Steps**: 500
- **Maximum Epochs**: 100
- **Mixed Precision**: FP16
- **Gradient Checkpointing**: Enabled

A.2 Lyric Generation

Training Configuration

- **Batch Size**: 32
- **Learning Rate**: 5×10^{-5}
- **Weight Decay**: 0.01
- **Optimizer**: AdamW
- **Scheduler**: Linear schedule with 10% warmup

A.3 Lyric Translation

Model Architecture

- **Encoder Layers**: 6
- **Decoder Layers**: 6
- **Hidden Size**: 512
- **FFN Dimension**: 2,048
- **Attention Heads**: 8 per layer
- **Maximum Position**: 1,024

Training Configuration

- **Batch Size**: 32
- **Learning Rate**: 1×10^{-4}
- **Weight Decay**: 0.01
- **Optimizer**: AdamW
- **Warmup Steps**: 500

A.4 Conventional Machine Translation

Model Architecture

- **Encoder**: Georgian DistilBERT (`Davit6174/georgian-distilbert-mlm`)
  - Hidden Size: 768
  - Layers: 6
  - Attention Heads: 12
  - FFN Size: 3,072
  - Dropout: 0.1
- **Decoder**: BERT-based decoder (`bert-base-uncased`) [5]
  - Hidden Size: 768
  - Layers: 6
  - Attention Heads: 12
  - FFN Size: 3,072
  - Dropout: 0.1
- **Maximum Sequence Length**: 128

Training Configuration

- **Batch Size**: 16
- **Learning Rate**: 5×10^{-5}
- **Optimizer**: AdamW
- **Scheduler**: Linear schedule with warmup
- **Warmup Steps**: 1,000
- **Maximum Epochs**: 100

Table 2: Comparing the Best-of-2 approach with the primary and secondary baseline models

(a) Symbolic Music Generation

	Pitch Divergence	Beat Divergence
Primary	13.04	9.65
Secondary	13.05	9.69
Best of 2	7.91	5.13

(b) Lyric Generation

	SCD	BERTScore
Primary	10.97	85.70
Secondary	11.37	85.70
Best of 2	9.22	85.88

(c) Lyric Translation

	SCD	BERTScore
Primary	36.77	81.61
Secondary	40.83	81.89
Best of 2	22.09	82.02

(d) Conventional Machine Translation

	BLEU-1	BLEU-2	BERTScore
Primary	23.66	15.37	85.29
Secondary	23.54	15.10	85.06
Best of 2	25.75	17.14	85.63

B Best of 2 Evaluation

When designing judge models, a key requirement is that the Best-of-2 approach, where outputs from the primary and secondary baseline models are generated independently, and the judge selects the better one for each evaluation item, must outperform both individual models across all objective metrics. This ensures the judge is effectively distinguishing and selecting higher-quality outputs. Table 2 reports the evaluation results of the Best-of-2 approach compared with the primary and secondary models across all tasks.