Halu-NLP at SemEval-2024 Task 6: MetaCheckGPT - A Multi-task Hallucination Detection Using LLM Uncertainty and Meta-models

Rahul Mehta*
IIIT Hyderabad, India
rahul.mehta@research.iiit.ac.in

Andrew Hoblitzell*
Purdue University, USA
ahoblitz@purdue.edu

Jack O'Keefe
Northwestern University, USA
jackokeefe2024@u.northwestern.edu

Hyeju Jang
Indiana University Indianapolis, USA
hyejuj@iu.edu

Vasudeva Varma
IIIT Hyderabad, India
vv@iiit.ac.in

Abstract

Hallucinations in large language models (LLMs) have recently become a significant problem. A recent effort in this direction is a shared task at Semeval 2024 Task 6, **SHROOM**, a *Shared-task on Hallucinations and Related Observable Overgeneration Mistakes (Mickus et al., 2024)*. This paper describes our winning solution ranked *1st* and *2nd* in the 2 sub-tasks of model agnostic and model aware tracks respectively. We propose a meta-regressor framework of LLMs for model evaluation and integration that achieves the highest scores on the leader board. We also experiment with various transformer based models and black box methods like ChatGPT, Vectara, and others. In addition, we perform an error analysis comparing GPT4 against our best model which shows the limitations of the former.

1 Introduction

The recent rapid deployment of large language models (LLMs) has led to a hallucination proliferation which poses a barrier to the reliability and trustworthiness of LLMs (Lin et al., 2022). One of the widely agreed upon definition of hallucinations (Maynez et al., 2020; Xiao and Wang, 2021) is output text containing information not relevant to the input or a desired output. Hallucinations should not be thought of as an occasional nuisance, but rather as a systemic issue inherent to these models and their web-sourced training data which can be rife with bias and misinformation. This can directly cause user discontent when these systems are implemented in production or real-world scenarios.

These type of hallucinations have been widely studied in the context of text related tasks like machine translation (Dale et al., 2022; Guerreiro et al., 2023a,b), summarization (Huang et al., 2023; van der Poel et al., 2022) and dialogue generation (Shuster et al., 2021a). Gaps in hallucination detection methods in LLM outputs persist across many such tasks.

Despite some progress in hallucination detection, existing methods may rely upon comparisons to reference texts, overly simplified statistical measures, reliance upon individual models, or annotated datasets which can limit their extensibility. Our approach leverages the uncertainty signals present in a diverse basket of LLMs to detect hallucinations more robustly.

In this paper, we present a meta-regressor framework for LLM model selection, evaluation, and integration.[1] The overall approach is depicted in Figure 1. For the first step, each LLM-generated sentence is compared against stochastically generated responses with no external database as with SelfCheckGPT (Manakul et al., 2023). A meta-model that leverages input from a diverse panel of expert evaluators evaluates and integrates the output of multiple iterations of the process.

Our framework focuses on creating a meta-model for identifying hallucinations, with the idea that the meta-model's prediction power is linked to the performance of the underlying base models. This model achieves the highest scores in the SemEval-2024 Task 6 competition across three subtasks: Machine translation, Paraphrase generation, and Definition modeling.

2 Related Work

In this section, we describe prior work on hallucination detection methods. We will examine two potential streams for hallucination detection: model-aware detection and black-box detection. Model-aware techniques have access to the model's internals, such as weights and logits while black-box methods do not has access to such model internals.

*Equal contribution

[1]The code of MetaCheckGPT is available at https://github.com/rahcode7/semeval-shroom

Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024), pages 342–348

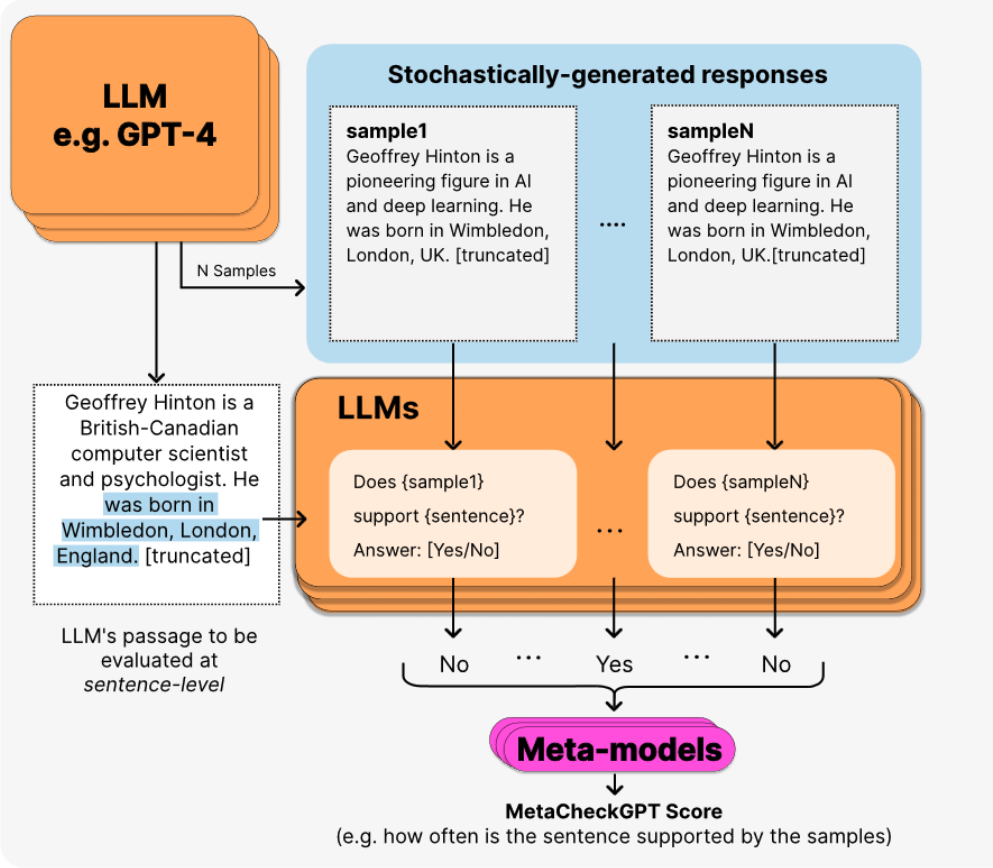

Figure 1: MetaCheckGPT: Generated sentences are compared against stochastically generated responses.

2.1 Model aware Detection

These methods require access to model weights and their logits (van der Poel et al., 2022). For machine translation task, Guerreiro et al. (2023b) showcased that sequence log-probability performs quite well compared to reference based methods. For article generation task, (Varshney et al., 2023) uncertainty estimation techniques(Azaria and Mitchell, 2023) (Tian et al., 2023) were used to detect hallucination in ChatGPT. Other methods to detect hallucinations include Retrieval Augmented Generation (Shuster et al., 2021b) and Chain of Verification based techniques (e.g., (Lei et al., 2023)).

2.2 Black box Detection

With the prevalence of closed source models, there has been recent work on black-box hallucination detection methods which doesn't require the model inputs, only the generated text. For example, a recently proposed system SelfCheckGPT (Manakul et al., 2023) utilizes a sampling-based technique based on the idea that sampled responses for hallucinated sentences will contradict each other.

This model achieves the highest scores across two sub-tasks: Machine translation, Paraphrase generation, and Definition modeling. We perform extensive studies of LLMs like ChatGPT, Mistral, and others to showcase their failure points.

3 Task Description and Datasets

In the SHROOM Task-6, the organizers propose a binary classification task to predict a machine generated sentence is a hallucination or not.

The organizers considered 3 types of text generation tasks - Definition Modelling, Machine Translation and Paraphrase Generation.

3.1 Task Tracks

The shared task was further divided into 2 tracks: **model agnostic** and **model aware**. Figure 2 describes sample examples of hallucinations containing source, reference and output text for each task type.

3.1.1 Model Agnostic Track

In this track, only the inputs, references and outputs were provided The details of the model that produced the text was masked from the participants.

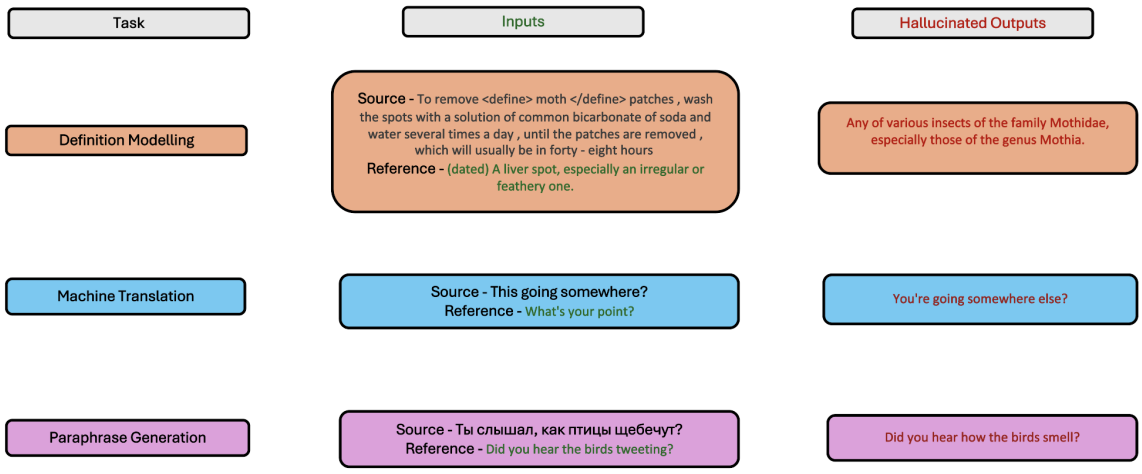

Figure 2: Hallucination examples for each task type

For data preparation, the SHROOM organizers followed the structure described in (Bevilacqua et al., 2020).

	Model Agnostic Track		
Task	Train	Dev	Test
Definition Modeling	10000	187	562
Machine Translation	10000	187	563
Paraphrase Generation	10000	125	375
Total	30000	499	1500

Table 1: Sample counts for the Model Agnostic Track

3.1.2 Model Aware Track

In this track, along with the inputs, references and outputs, the model name and its checkpoints were also provided from which the outputs were generated.

	Model Aware Track		
Task	Train	Dev	Test
Definition Modeling	10000	188	562
Machine Translation	10000	188	563
Paraphrase Generation	10000	125	375
Total	30000	501	1500

Table 2: Sample Statistics for the Model Aware Track

It is worthwhile to note that the organizers chose to share the training which was not labeled and only the development set was labeled.

4 Our Methodology

Algorithm 1 Meta-Model Training/Evaluation

Input: Base models M, Meta-models N, Threshold x
Output: Top performing meta-model
for each base model m in M **do**
    $score_m \leftarrow$ Evaluate m (MAE)
end for
$FilteredMs \leftarrow Models$.filter(MAE $< x$)
for each meta-model n in N **do**
    Train n with $FilteredMs$
    $metaScore_n \leftarrow$ Spearman MAE
end for
$TopMeta \leftarrow$ Meta-model in N with lowest Spearman MAE

Our approach is centered around building a meta-model for hallucination detection, with the hypothesis that the quality of prediction from underlying base models is highly correlated with the meta-model's predictive power. Given a set of base models $M = \{m_1, m_2, ..., m_n\}$ and actual labels $L = \{l_1, l_2, ..., l_n\}$ in the dataset, the Spearman correlation between predicted hallucination scores H and actual labels is given by:

$$\rho_s(H, L) = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)}$$

where d_i is the difference between the ranks of corresponding elements in H and L.

Our overall process was to identify the meta-model that minimized this mean absolute error

(MAE) function ϵ, where

$$\epsilon = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)$$

because Spearman correlation was one of the secondary metrics for Task 6 evaluation. Here, Y_i represents the actual Spearman correlation values for hallucination and $\hat{Y}_i$ represents the predicted values. Our overall process is captured in Algorithm 1.

Algorithms 2, 3, and 4 detail how some of our different meta-models were trained. These algorithms follow a unified framework, initiating with the setup of training data and labels, with the ultimate aim of fine-tuning a meta-regressor model. A meta-search cross-validation approach was used to conduct a hyperparameter space for each model's architecture. The process involves iterating over the defined hyperparameter space for each algorithm, fitting the meta-regressor with the training data, and concluding with the identification and preparation of the highest-performing model for deployment. The training process for selecting meta-models is included in the Appendix. RMSE, MAE, and R-squared were used as additional proxies in meta-model evaluation.

Because this problem was assessed with binary classification accuracy, data was classified based on the Spearman correlation coefficient according to:

$$\text{Class} = \begin{cases} \text{'Hallucination'}, & \rho_s > 0.5 \\ \text{'Not Hallucination'}, & \text{otherwise} \end{cases}$$

to convert our regression problem into a binary classification task, simplifying the analysis and interpretation of results. Once converted to a classification problem, the primary metric used for evaluation was accuracy. Precision, Recall, F1 Score, and a confusion matrix were used for secondary evaluation.

5 Experiments & Results

5.1 Experimental set-up

Training was conducted both on cloud using APIs as well as locally on V100/A100 GPUs for faster processing times.

We conducted our initial experiments with simpler base models including DeBERTa (He et al., 2021), DistilBERT (Sanh et al., 2020), RoBERTa (Liu et al., 2019), LLaMA 2 (Touvron et al., 2023), and Mixtral of Experts (Jiang et al., 2024) among others. Preliminary results indicated an accuracy of 0.5 to 0.6, prompting us to continue our search for more performant base models.

Additional analysis indicated our base models ChatGPT(Achiam et al., 2023), SelfCheckGPT(Manakul et al., 2023) and Vectara(Hughes, 2023) showed promising results in initial tests, with accuracy in the range of 0.6 to 0.7. Prompt engineering, self-consistency checks and uncertainty based modeling techniques were used to maximize performance in base models. The training process for more performant meta-models, including random forest and elementary neural ensemble models, can be found in the Appendix.

5.2 Results

Classification performance obtained on the training data, which includes an accuracy of 0.8317, precision of 0.7447, recall of 0.875, and an F1 score of 0.8046.

	Positive	Negative
Positive	TP: 49	FN: 5
Negative	FP: 12	TN: 35

Cross-validation and regularization techniques were applied to increase confidence that the performance observed on the training data would be maintained on test data.

Track	Accuracy	Rho	Rank
Aware	80.6	0.71	1/46
Agnostic	84.7	0.77	2/49

Table 3: Final Modeling results on the test set

5.3 Discussion

Our results, as summarized in Table 3, demonstrate the effectiveness of meta-regressor models in detecting hallucinations across various text generation tasks. One of the key strengths of the approach is that a diverse set of base models is able to better capture a wide range of features indicative of hallucinations than a single model or knowledge base alone may be able to. High performance metrics underline the promise of combining base models/knowledge bases through meta-learning.

Our approach is not without its limitations. The black-box nature of some base models (e.g. GPT4), limits understanding of the internal mechanisms

driving the generation and detection of hallucinations. More detailed limitations of the system and directions for future work are examined in the following section.

5.4 Limitations

There are several limitations to the current work. For example, all language models have inherent limitations such as bias and lack of world grounding. Unfortunately, more recent models such as GPT have also started to function as black box systems. The corpus for training data for base language models is predominantly English. The system also would not readily integrate into a production system without additional effort. The system could also benefit from the ability to learn from feedback. All of the base language models may also suffer from potential safety issues like false confidence and over-reliance, etc.

6 Conclusion

Our meta-model strategy represents a step forward in addressing the challenges of mitigating hallucinations and the importance of a nuanced approach to model selection, evaluation, and integration. The work also acknowledges the need for additional research into more transparent, interpretable, and multilingual models, as well as the integration of external knowledge sources and feedback mechanisms to refine and improve hallucination detection methods. In the future, some areas we would like to work on include utilizing additional multilingual datasets, expand the scope of our work to more set of text generation task, and focus more on white box hallucination detection systems.

While the current system was tested on some machine translation tasks, we believe it could benefit from more work on multilingual datasets. The current system could improve by integrating with external knowledge bases via retrieval augmented generation. The system could also be made more usable by distilling its knowledge into a portable fine-tuned model widely available to others. Another area for potential improvement includes integration of human or agent feedback through reinforcement learning.

Acknowledgements

We would like to thank the Machine Learning Collective (MLC), a group which promotes collaboration and mentorship opportunities being accessible and free for all, for helping connect us as researchers. We would also especially like to thank Timothee Mickus for being very responsive to questions on the listserv. We would also like to thank Elaine Zosa, Raúl Vázquez, Jörg Tiedemann, Vincent Segonne, Alessandro Raganato, and Marianna Apidianaki for organizing the Shared-task on Hallucinations and Related Observable Overgeneration Mistakes, we felt delighted to have a forum to work with others who have a shared interest in language model capabilities. Thanks to Steven Bethard, Ryan Cotterell, Rui Yanfrom, and many others for their work on the template which we adapted from.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, and et al. 2023. Gpt-4 technical report. Technical report, OpenAI.

Amos Azaria and Tom Mitchell. 2023. The internal state of an llm knows when its lying. *arXiv preprint arXiv:2304.13734*.

Michele Bevilacqua, Marco Maru, and Roberto Navigli. 2020. Generationary or "how we went beyond word sense inventories and learned to gloss". In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7207–7221, Online. Association for Computational Linguistics.

David Dale, Elena Voita, Loïc Barrault, and Marta R. Costa-jussà. 2022. Detecting and mitigating hallucinations in machine translation: Model internal workings alone do well, sentence similarity even better.

Nuno M. Guerreiro, Duarte Alves, Jonas Waldendorf, Barry Haddow, Alexandra Birch, Pierre Colombo, and André F. T. Martins. 2023a. Hallucinations in large multilingual translation models.

Nuno M. Guerreiro, Elena Voita, and André F. T. Martins. 2023b. Looking for a needle in a haystack: A comprehensive study of hallucinations in neural machine translation.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention.

Yichong Huang, Xiachong Feng, Xiaocheng Feng, and Bing Qin. 2023. The factual inconsistency problem in abstractive text summarization: A survey.

Simon Hughes. 2023. Cut the bull... detecting hallucinations in large language models.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, and et al. Marie-Anne Lachaux. 2024. Mixtral of experts.

Deren Lei, Yaxi Li, Mengya Hu, Mingyu Wang, Vincent Yun, Emily Ching, and Eslam Kamal. 2023. Chain of natural language inference for reducing large language model ungrounded hallucinations. *arXiv*, cs.CL(arXiv:2310.03951).

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Truthfulqa: Measuring how models mimic human falsehoods.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. 2023. Selfcheckgpt: Zero-resource black-box hallucination detection for generative large language models.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online. Association for Computational Linguistics.

Timothee Mickus, Elaine Zosa, Raúl Vázquez, Teemu Vahtola, Jörg Tiedemann, Vincent Segonne, Alessandro Raganato, and Marianna Apidianaki. 2024. Semeval-2024 shared task 6: Shroom, a shared-task on hallucinations and related observable overgeneration mistakes. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, pages 1980–1994, Mexico City, Mexico. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021a. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021b. Retrieval augmentation reduces hallucination in conversation. *arXiv*, cs.CL(arXiv:2104.07567).

Katherine Tian, Eric Mitchell, Allan Zhou, Archit Sharma, Rafael Rafailov, Huaxiu Yao, Chelsea Finn, and Christopher D. Manning. 2023. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models fine-tuned with human feedback. https://doi.org/10.48550/arXiv.2305.14975. ArXiv:2305.14975v2 [cs.CL].

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, and et al. 2023. Llama 2: Open foundation and fine-tuned chat models.

Liam van der Poel, Ryan Cotterell, and Clara Meister. 2022. Mutual information alleviates hallucinations in abstractive summarization.

Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. 2023. A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation.

Yijun Xiao and William Yang Wang. 2021. On hallucination and predictive uncertainty in conditional language generation.

A Appendix: MR Training Processes

Algorithm 2 MR1 Training: Algorithm 2 outlines the process of training a meta-regressor model with hyperparameters for random forest.

Require: X_{train}, y_{train} ▷ Training data and labels
Ensure: $model_{best}$ ▷ Optimally tuned model

$MR \leftarrow MetaRegressor()$
$H \leftarrow \{n\_estimators \in \{\alpha_1, \ldots, \alpha_N\},$
  $max\_depth \in \{\beta_1, \ldots, \beta_M\},$
  $min\_samples\_split \in \{\gamma_1, \ldots, \gamma_L\},$
  $min\_samples\_leaf \in \{\delta_1, \ldots, \delta_K\},$
  $max\_features \in \{$'auto', 'sqrt'$\},$
  $bootstrap \in \{$True, False$\}\}$
$MetaCV = MetaSearchCV(MR, H, cv)$
$MetaCV.fit(X_{train}, y_{train})$
$params_{best} = MetaCV.best_params$
$model_{best} = MetaRegressor(params_{best})$
$model_{best}.fit(X_{train}, y_{train})$

Algorithm 3 MR2 Training: Algorithm 3 outlines the process of training a meta-regressor model with hyperparameters for gradient boosted trees.

Require: X_{train}, y_{train} ▷ Training data and labels
Ensure: $model_{best}$ ▷ Optimally tuned model

$MR \leftarrow MetaRegressor()$
$H \leftarrow \{n\_estimators \in \eta_1, \ldots, \eta_n,$
  $learning\_rate \in \theta_1, \ldots, \theta_n,$
  $max\_depth \in \iota_1, \ldots, \iota_n,$
  $min\_child\_weight \in \kappa_1, \ldots, \kappa_n,$
  $gamma \in \lambda_1, \ldots, \lambda_n,$
  $subsample \in \mu_1, \ldots, \mu_n,$
  $colsample\_bytree \in \nu_1, \ldots, \nu_n,$
  $reg\_alpha \in \xi_1, \ldots, \xi_n,$
  $reg\_lambda \in \zeta_1, \ldots, \zeta_n\}$
$MetaCV = MetaSearchCV(MR, H, cv)$
$MetaCV.fit(X_{train}, y_{train})$
$params_{best} = MetaCV.best_params$
$model_{best} = MetaRegressor(params_{best})$
$model_{best}.fit(X_{train}, y_{train})$

Algorithm 4 MR3 Training: Algorithm 4 the training procedure for a meta-regressor model designed for an elementary neural ensemble model.

Require: X_{train}, y_{train} ▷ Training data and labels
Ensure: $model_{best}$ ▷ Optimally tuned model

$MR \leftarrow MetaRegressor()$
$H \leftarrow \{num\_layers \in \eta_1, \ldots, \eta_n,$
  For each layer i **in** $1, \ldots, num\_layers$:
    $units\_i \in \delta_1, \ldots, \delta_n,$
    $activation\_i \in \zeta_1, \ldots, \zeta_n,$
    $l2\_reg \in \iota_1, \ldots, \iota_n,$
  $dropout \in \gamma_1, \ldots, \gamma_n$
  $learning\_rate \in \theta_1, \ldots, \theta_n, \}$
$MetaCV = MetaSearchCV(MR, H, cv)$
$MetaCV.fit(X_{train}, y_{train})$
$params_{best} = MetaCV.best_params$
$model_{best} = MetaRegressor(params_{best})$
$model_{best}.fit(X_{train}, y_{train})$