# Green Runner

## A Tool for Efficient Deep Learning Component Selection

Kannan, Jai; Barnett, Scott; Cruz, Luis; Simmons, Anj; Selvi, Taylan

**DOI**
10.1145/3644815.3644942

**Publication date**
2024

**Published in**
Proceedings - 2024 IEEE/ACM 3rd International Conference on AI Engineering - Software Engineering for AI, CAIN 2024

# Green Runner: A tool for efficient deep learning component selection

Jai Kannan, Scott Barnett
jai.kannan@deakin.edu.au
scott.barnett@deakin.edu.au
Applied Artificial Intelligence Institute
Geelong, Australia

Luís Cruz
l.cruz@tudelft.nl
Delft University of Technology
Delft, Netherlands

Anj Simmons, Taylan Selvi
a.simmons@deakin.edu.au
taylan.selvi@deakin.edu.au
Applied Artificial Intelligence Institute
Geelong, Australia

## ABSTRACT

For software that relies on machine-learned functionality, model selection is key to finding the right model for the task with desired performance characteristics. Evaluating a model requires developers to i) select from many models (e.g. the Hugging face model repository), ii) select evaluation metrics and training strategy, and iii) tailor trade-offs based on the problem domain. However, current evaluation approaches are either ad-hoc resulting in sub-optimal model selection or brute force leading to wasted compute. In this work, we present GreenRunner, a novel tool to automatically select and evaluate models based on the application scenario provided in natural language. We leverage the reasoning capabilities of large language models to propose a training strategy and extract desired trade-offs from a problem description. GreenRunner features a resource-efficient experimentation engine that integrates constraints and trade-offs based on the problem into the model selection process. Our preliminary evaluation demonstrates that GreenRunner is both efficient and accurate compared to ad-hoc evaluations and brute force. This work presents an important step toward energy-efficient tools to help reduce the environmental impact caused by the growing demand for software with machine-learned functionality. Our tool is available at Figshare GreenRunner.

## CCS CONCEPTS

• **Software and its engineering**; • **Computing methodologies** → **Machine learning**;

## KEYWORDS

Green-AI, Large Language Model, Component Selection

**ACM Reference Format:**
Jai Kannan, Scott Barnett, Luís Cruz, and Anj Simmons, Taylan Selvi . 2024. Green Runner: A tool for efficient deep learning component selection. In *Conference on AI Engineering Software Engineering for AI (CAIN 2024), April 14–15, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3644815.3644942


*CAIN 2024, April 14–15, 2024, Lisbon, Portugal*

ACM ISBN 979-8-4007-0591-5/24/04…$15.00
https://doi.org/10.1145/3644815.3644942

## 1 INTRODUCTION

While deep learning (DL) significantly enhances software, it also brings various challenges that need careful attention. A central concern is that of the environmental impact of deep learning models [10, 17, 20, 27]. The environmental impact of training DL models has primarly been the focus [16, 22]. However, environmental impact that occurs when reusing deep learning models through additional training and evaluation of models is less studied. Selecting an appropriate deep learning component (a pretrained model or an API web service) requires a) evaluating against a large sample dataset, b) comparing multiple alternatives, and c) discovering an optimal reuse strategy (i.e. fine tuning, transfer learning, ensembling etc.) All this evaluation wastes compute resources. Finding energy efficient strategies to compare and select DL components is important to mitigate the environmental impact of deep learning.

Efficient methods for choosing deep learning (DL) components typically involve: 1) using benchmark dataset metrics [2, 15], 2) applying proxy metrics to estimate performance [11, 26], or 3) assessing transferability [1, 3, 12, 21]. These methods focus on approximating a component's performance but often overlook important factors like memory needs, computational demands, and hardware compatibility, leading to choices that may not be suitable for specific application contexts. This paper addresses the gap in current selection methods by considering the trade-offs software engineers face in different application scenarios.

For example, selecting DL components for an agricultural drone, which requires a balance between accuracy and energy efficiency for longer flight times, differs greatly from an autonomous vehicle, where less emphasis is on energy efficiency and more on low-latency detection for safety [14]. Thus, the deployment scenario, system safety criticality, and hardware requirements significantly influence the selection criteria for DL components, underscoring the need for a context-specific approach.

Software engineers have to select more than a DL component (i.e. a model on Huggingface) but also the configurations for evaluation. A selection of tasks, models, metrics, reuse process and tradeoffs to consider are depicted in Figure 1, which presents only the computer-vision models on Huggingface[1]. For image object detection, a developer must select from 1245 models[2], select the right combination of metrics and training process and iteratively evaluate the models and then compute the tradeoffs. This highlights the need for a resource-efficient and cost-effective DL component selection

[1] https://huggingface.co/
[2] https://huggingface.co/models?pipeline_tag=object-detection&sort=trending

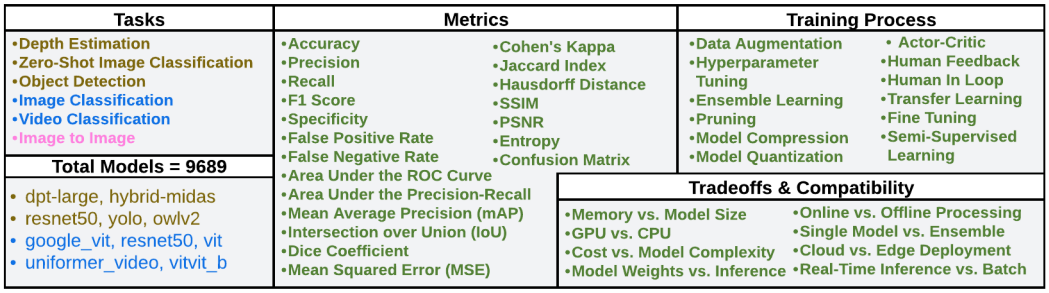


**Figure 1: The problem space where developers have to pick $X$ tasks containing $N$ models, evaluate them with $E$ metrics and $T$ tradeoffs which needs to be evaluated iteratively, resulting in wasted resources and increased costs.**

strategy that considers 1) operational criteria, 2) environmental impact, and 3) comparisons between competing components.

To assist software engineers in selecting deep learning (DL) components, we introduce GreenRunner. This tool streamlines the evaluation process by incorporating an energy-efficient experimentation engine based on a multi-armed bandit framework and utilizing the reasoning capabilities of a large language model (LLM). The LLM suggests application-specific configurations, including the model, metrics, reuse strategies, and trade-offs. Engineers input a description of their application and its operational context, and GreenRunner generates the necessary evaluation configuration.

Our approach is grounded in the hypothesis that an LLM, trained on scientific publications, blog articles, and open-source machine learning repositories [5, 7, 23], can produce near-optimal configurations. We posit that the computational cost of training and operating a single LLM is offset by the reduced compute requirements for future applications developed using DL components.

We demonstrate the effectiveness of our approach with a preliminary evaluation using the ObjectNet dataset and 71 object detection models, showcasing GreenRunner's potential in optimizing DL component selection.

**Key Contributions arising from this work:**

- A preliminary empirical evaluation of an LLM's world model for informing component selection (focusing on DL component selection).
- An approach for optimising the DL component selection process. Our approach uses a 1) dynamically created multi-objective function derived from a natural language description of the problem, and 2) an efficient evaluation process based on multi-armed bandits.
- A tool implementation of the approach, GreenRunner, available for download and [3]is hosted at:https://green-runner-web-eehhqvzsqa-km.a.run.app/

## 2 MOTIVATING EXAMPLE

To motivate the need for GreenRunner, consider Tom, a software engineer at an agricultural drone manufacturing company. The WeedWhack drones are to detect and then spray weeds without affecting the main crop. Tom is tasked with implementing the weed detection component. Tom has been loosely following the advances in object detection but is primarily a software engineer, not a machine learning expert. To complete the task Tom needs to select an appropriate DL component by 1) finding a set of candidate components (i.e. online models or web services), 2) select evaluation metrics, reuse strategy, and hyperparameters, and 3) evaluate and compare the results. These tasks have to be completed while considering the constraints of the problem domain namely a) resource efficient models (cover the whole field without recharging), b) accurate detection (only target weeds not crops), and c) adapt to a multitude of weed types.

Tom starts identifying DL components for WeedWhack by selecting object detection models from Huggingface and online services (e.g. AWS Rekognition API[4]). Tom selects a handful of components to evaluate against the dataset of weeds and starts evaluating them all against the benchmark dataset. Each component needs to be adapted to the weed dataset to find the ideal model which wastes time, increases compute costs, and is energy inefficient. Tom is also unaware of the range of strategies suitable for the problem domain (i.e. pruning, quantization, fine-tuning, and/or transfer learning) and ends up making a sub-optimal choice. Once the drone has been deployed Tom finds the model he selected sprays too much of the crop and needs to evaluate a new set of models. This wastes development time and increases the cost of implementation.

What Tom needs is a system that effectively recommends the most suitable DL component and strategy for WeedWhack, facilitating a quick yet thorough comparison and selection process.

## 3 GREENRUNNER

To design GreenRunner we took inspiration from the tool proposed by Cummaudo et al. [6] , we describe the core components of GreenRunner, shown in figure Figure 2. The core components are 1) a GPT Based Reasoning Module to suggest metric choice & weights for a target use case, which forms a reward function, and 2) a resource-efficient experimentation Engine that evaluates pre-trained models on a target dataset against using a reward function to produce a set of top-ranked components for the target use case.

### 3.1 GPT Based Reasoning Module

The integration of GreenRunner with an LLM enables users to provide a concise plain text prompt that describes the specific use case for the application which is the first step in identifying an

[3]https://figshare.com/s/248381647619ba223334 (anonymised for blind review)

[4]https://docs.aws.amazon.com/rekognition/latest/dg/API_Reference.html

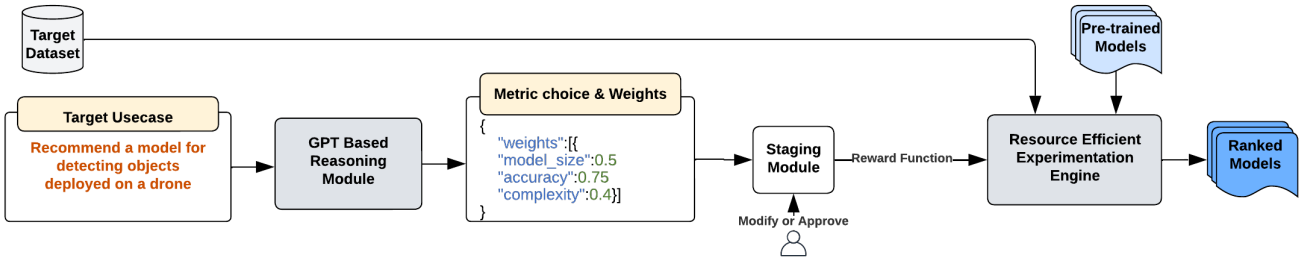


**Figure 2: Overview of GreenRunner describing the internal processes and outputs from each process.**

optimal DL component for a particular use case. For this paper, the LLM used by the reasoning module was GPT-4. Analysing the description, the LLM generates a set of metrics and weights, which are used as part of a reward function for performing the selection process. These weights are distributed across the metrics suggested by the LLM such as: i) Model accuracy, ii) Model size, and iii) Model complexity.

Additionally, the LLM offers comprehensive justifications for the selected weights, optimizing them in accordance with the use case at hand. These metric weights serve as configuration parameters for resource-efficient evaluation algorithms, which effectively navigate the extensive model repository and select the most appropriate component. Prior to conducting the experiment, the staging module allows users to thoroughly examine and refine the metric weights, ensuring alignment with the intended use case.

### 3.2 Resource Efficient Experimentation Engine

GreenRunner utilizes multi-armed bandit (MAB) algorithms [19] to streamline the selection of deep learning components for specific applications, reducing the need for extensive evaluations. These algorithms adeptly balance the exploration of diverse options with the exploitation of known advantageous actions.

In GreenRunner, deep learning models from a repository function as "arms". The system is designed to identify top-performing models with a limited number of evaluations. It employs a custom reward function that considers metrics such as accuracy, size, complexity generated by the reasoning module subsection 3.1, thereby adapting the selection to the unique requirements of each use case.

GreenRunner offers three MAB selection strategies: Epsilon Greedy, Upper Confidence Bound, and Thompson Sampling. The evaluation of these models is based on a user-defined budget,and strategy where a larger budget enhances the likelihood of finding the optimal model, whereas a smaller budget limits evaluations, potentially leading to the selection of less-than-ideal models. As more data is processed, the MAB algorithm continuously refines its understanding of each model's performance, effectively pinpointing the most suitable models for the specific dataset.

## 4 USAGE EXAMPLE

To address Tom's challenge, we present an innovative tool designed to optimize model selection while minimizing resource consumption. Tom inputs a simple query like *"Recommend a model for drone-based object detection"* and provides the target dataset. This triggers our GPT-Based reasoning module, which proposes a set of metrics and their importance as shown in Figure 3 ①. The module also justifies these choices in ②, linking them to key use case considerations such as model size, complexity, and performance, detailed in section 2.

Tom can then adjust these metric weights prior to running the experiment ③, which will affect the outcome of the resource-efficient experiment engine. After running the experiment, Tom reviews the results on the analysis screen in ④, which displays the top models in ⑤, the number of evaluations made per model in ⑥, and the computational savings in ⑦ compared to a brute force approach. From here, Tom can select the best model to fine-tune and deploy on his drone.

## 5 PRELIMINARY EVALUATION

To evaluate our approach we proposed two research questions: ***1. Does GreenRunner find the most suitable model for inferred trade-offs?*** and ***2. How does GreenRunner compare to other transferability metrics for model selection?*** This evaluates if our approach balances tradeoffs and how it compares to state-of-the-art. We compared GreenRunner to three baselines: 1) benchmark results (results of the model trained on a benchmark dataset), 2) brute force (compare all models on all data points), and 3) a transferability metric from the literature [12].

**Dataset:** In this research, DL components refer to deep learning models selected from a repository. We initially considered the complete collection of image classification models available on PyTorch Hub[5], amounting to 80 models. However, due to API issues that prevented the download of 9 models, our experiment proceeded with the remaining 71 models. PyTorch Hub was chosen for its prevalence as an open-source framework in the machine learning community and for the ease of conducting standardized comparisons, as it hosts models pre-trained on the benchmark ImageNet dataset [8].

[5]https://pytorch.org/vision/stable/models.html

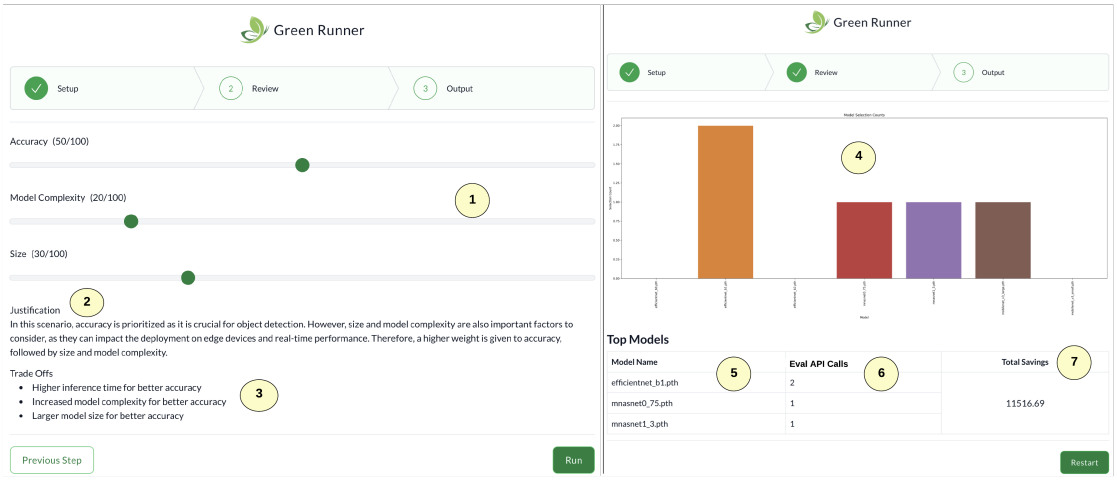


**Figure 3: GreenRunner's user interface displaying the configuration of an experiment and the analysis report.**

**Table 1: Comparative analysis of i) benchmark result, ii) Brute Force against GreenRunner**

| Method | Metric | Top Models | Avg. Performance on Target | Avg. Model Size | Avg. Model Complexity |
|---|---|---|---|---|---|
| Benchmark result | Accuracy | maxvit_t | 0.29 | 124.5 MB | 19670 MMAC |
| | Accuracy, Size, Complexity | mobilenet_v3 | 0.17 | 22 MB | 229 MMAC |
| Brute Force | Accuracy | regnet_y_128gf | 0.45 | 2581 MB | 127750 MMAC |
| | Accuracy, Size, Complexity | convex_net_small | 0.31 | 114 MB | 4470 MMAC |
| GreenRunner | Accuracy | regnet_y_32gf | 0.32 | 581 MB | 32380 MMAC |
| | Accuracy, Size, Complexity | swin_v2_s | 0.30 | 199 MB | 5790 MMAC |

In our experiments, we used the ObjectNet dataset [2] as our target for model evaluation. ObjectNet, known for challenging benchmark models, is tailored for testing vision models in realistic settings. For our initial assessment, we randomly selected 200 images from 113 classes corresponding to ImageNet categories to evaluate the models.

**GreenRunner configuration:** To calculate the metrics and weights for the use case we used a GPT Based Reasoning module (based on GPT-4) using the following prompt *Recommend a model for detecting objects deployed on a drone.* As the response is non-deterministic, we executed the prompt 100 times resulting in average weights for the use case. For the purpose of the experiment we only chose the accuracy, model size and complexity metrics which are: accuracy weight of 0.63, size weight of 0.25, and complexity weight of 0.21 respectively.

**Method:** To answer: *1. Does GreenRunner find the most suitable model for inferred trade-offs?*, We performed a comparative analysis using two metric combinations against the benchmark and brute force approaches. The first set of metrics evaluated accuracy alone, while the second also considered model size and complexity, based on suggested weights for the use case. We compared the selected models in terms of on-target accuracy, size (in MB), and complexity (measured in Million Multiply-Accumulate Operations, or MMACs). Given that our approach can recommend varying models per run, we averaged results over 200 iterations, and presented in Table 1.

To answer: *2. How does GreenRunner compare to other transferability metrics for model selection?*, we used the TransRate metric proposed in [12] and compared the model selected using TransRate to the model selected by GreenRunner

**Results:** ***1. Does GreenRunner find the most suitable model for inferred trade-offs?*** In our analysis in Table 1, we concentrate on the top model ranked by each approach. The benchmark method often chooses models for their high scores (about 0.80) on benchmark datasets, but these models frequently underperform on target datasets, with accuracy dropping to 0.29. When adjusted for size and complexity, further declines to 0.17, although this results in a smaller model. However, this method risks overlooking models that may offer superior overall performance.

The brute force method, assessing every model against the full dataset, identifies the most accurate model with an accuracy of 0.45 accuracy on the target dataset. However, its resource-intensive nature is a drawback, rendering the best model (2581MB in size and 127750 MMACs in complexity) impractical for resource-constrained applications like drone deployment.

For GreenRunner, we implemented Thomson Sampling as the selection strategy. Focusing solely on accuracy, our method outperformed the benchmark by selecting a model with 0.32 accuracy on the target dataset, though it did not reach the brute force method's accuracy.

**Table 2: Comparative analysis of TransRate [12] against GreenRunner**

| Method | Metric | Top 3 Models | Avg. Accuracy on Target | Avg. Model Size | Avg. Model Complexity |
|---|---|---|---|---|---|
| TransRate [12] | N/A | vgg11 | 0.23 | 531 MB | 7300 MMAC |
| | | vgg_16bn | 0.17 | 553 MB | 15530 MMAC |
| | | shufflenet_v2 | 0.14 | 30 MB | 593 MMAC |
| GreenRunner | Accuracy, Size, Complexity | swin_s | 0.32 | 199 MB | 5790 MMAC |
| | | regnet_y_800mf | 0.21 | 26 MB | 851 MMAC |
| | | efficientnet_b0 | 0.20 | 22 MB | 401 MMAC |

However, when considering a combination of accuracy, size, and complexity, GreenRunnerselected a model with 0.30 accuracy, close to the brute force method's 0.31. The model's reduced size and lower computational needs offset its slight dip in accuracy. This efficiency in model selection aligns GreenRunnerwith practical requirements, streamlining the process to identify the most appropriate model for specific use cases.

***How does GreenRunner compare to other transferability metric for model selection?*** In this experiment, we compare GreenRunner with TransRate, as outlined in [12], focusing on performance, size, and complexity metrics crucial for drone use cases. TransRate evaluates models based on transferability scores, assessing how a model's learned features align with a specific target task. This method, while adept in gauging feature adaptability, predominantly measures the relative transferability and doesn't consider the tradeoffs for the usecase.

Conversely, GreenRunner employs a broader approach, emphasizing not only transferability but also model size and computational complexity. This assessment is critical in scenarios like drone-based object detection, where resource efficiency is a key consideration. Considering that TransRate's scores are relative, we compare the top three models selected by each method. As demonstrated in Table 2, GreenRunner effectively identifies models that strike a balance between accuracy, size, and complexity, surpassing TransRate in meeting the operational requirements of drone applications.

## 6 RELATED WORK

To approach the model selection problem recent research has proposed several online and offline approaches [4, 9, 13, 18, 25]. To provide an approach to solve this problem one approach is to dissect models into distinctive building blocks and reassemble them to produce a customised network [24], however reassembling models is complex, and the proposed approach requires significant computational resources and expertise to implement. Research has also proposed platforms which assess the adaptation to a domain task for models in a model zoo [4]. These platforms require specialised knowledge and need empirical evaluation requiring computational resources.

Proposed approaches have also calculated the class similarity between the dataset the models are trained on vs the target dataset for a downstream task [18]. The assumption being a high class similarity between datasets produces similar performance between the models. Research has also produced lightweight tools to estimate transferability such as [12, 25]. These approaches utilise a small amount of data to be passed through the models to assess the potential performance of a target task. However, these approaches require at least one pass of the data through the models to estimate the potential performance of the models on the dataset where resources are utilised to compute the performance on the dataset. The way these approaches differ to our approach is that GreenRunner is able to select an appropriate amount of evaluation data on a per model basis and considers trade-offs rather than just optimising for accuracy.

## 7 CONCLUSION AND FUTURE WORK

GreenRunner addresses a practical challenge that developers encounter when integrating machine learning functionality into software. It efficiently selects machine learning models for specific use cases, surpassing traditional benchmark and brute force methods in terms of both optimality and efficiency. Benchmark methods often fall short, as they rely on source data accuracy, which might not reflect the target data performance. brute force methods, while accurate, consume significant resources in evaluating all models. GreenRunner balances high accuracy with manageable model sizes and complexity by leveraging metrics and weights tailored to each use case. This expedites the model selection process and significantly reduces computational resources, thus minimizing resource consumption and environmental impact and contributing to sustainable DL software development.

Our current work focuses on only one part of the DL component selection i.e. choosing appropriate pre-trained models from repositories for a downstream task. Looking ahead, we aim to expand our scope to include model fine-tuning and the selection of machine learning API web services. These areas align with our framework and reward function but will necessitate distinct multi-armed bandit (MAB) strategies and user interfaces. Further developments will explore enabling the GPT-based Reasoning Module to recommend a broader range of metrics beyond accuracy, size, and complexity. This expansion is planned to be seamlessly integrated into our existing system but will call for adjustments to the reward function and strategies for addressing newly important metrics as suggested by the LLM.

## ACKNOWLEDGEMENT

The authors extend their profound appreciation to Irini Logothetis and Shangeetha Sivasothy for their remarkable contributions and dedicated collaboration that significantly enhanced this work.

# REFERENCES


[1] Yajie Bao, Yang Li, Shao-Lun Huang, Lin Zhang, Lizhong Zheng, Amir Zamir, and Leonidas Guibas. 2019. An information-theoretic approach to transferability in task transfer learning. In *2019 IEEE international conference on image processing (ICIP)*. IEEE, 2309–2313.

[2] Andrei Barbu et al. 2019. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems* 32 (2019).

[3] Daniel Bolya, Rohit Mittapalli, and Judy Hoffman. 2021. Scalable diverse model selection for accessible transfer learning. *Advances in Neural Information Processing Systems* 34 (2021), 19301–19312.

[4] Emanuele Bugliarello, Fangyu Liu, Jonas Pfeiffer, Siva Reddy, Desmond Elliott, Edoardo Maria Ponti, and Ivan Vulić. 2022. IGLUE: A benchmark for transfer learning across modalities, tasks, and languages. In *International Conference on Machine Learning*. PMLR, 2370–2392.

[5] Yutian Chen, Xingyou Song, Chansoo Lee, Zi Wang, Richard Zhang, David Dohan, Kazuya Kawakami, Greg Kochanski, Arnaud Doucet, Marc'aurelio Ranzato, et al. 2022. Towards learning universal hyperparameter optimizers with transformers. *Advances in Neural Information Processing Systems* 35 (2022), 32053–32068.

[6] Alex Cummaudo, Scott Barnett, Rajesh Vasa, and John Grundy. 2020. Threshy: Supporting safe usage of intelligent web services. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1645–1649.

[7] Chris Cummins, Volker Seeker, Dejan Grubisic, Mostafa Elhoushi, Youwei Liang, Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Kim Hazelwood, Gabriel Synnaeve, et al. 2023. Large language models for compiler optimization. *arXiv preprint arXiv:2309.07062* (2023).

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 248–255.

[9] Nan Ding, Xi Chen, Tomer Levinboim, Soravit Changpinyo, and Radu Soricut. 2022. PACTran: PAC-Bayesian Metrics for Estimating the Transferability of Pretrained Models to Classification Tasks. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIV*. Springer, 252–268.

[10] Jesse Dodge, Taylor Prewitt, Remi Tachet des Combes, Erika Odmark, Roy Schwartz, Emma Strubell, Alexandra Sasha Luccioni, Noah A Smith, Nicole DeCario, and Will Buchanan. 2022. Measuring the carbon intensity of AI in cloud instances. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*. 1877–1894.

[11] Kshitij Dwivedi and Gemma Roig. 2019. Representation similarity analysis for efficient task taxonomy & transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12387–12396.

[12] Long-Kai Huang, Junzhou Huang, Yu Rong, Qiang Yang, and Ying Wei. 2022. Frustratingly easy transferability estimation. In *International Conference on Machine Learning*. PMLR, 9201–9225.

[13] Long-Kai Huang, Ying Wei, Yu Rong, Qiang Yang, and Junzhou Huang. 2021. Frustratingly Easy Transferability Estimation. (2021). arXiv:2106.09362 http://arxiv.org/abs/2106.09362

[14] Puneet Kohli and Anjali Chadha. 2020. Enabling Pedestrian Safety Using Computer Vision Techniques: A Case Study of the 2018 Uber Inc. Self-driving Car Crash. In *Advances in Information and Communication. FICC 2019*. 261–279. https://doi.org/10.1007/978-3-030-12388-8_19

[15] Simon Kornblith, Jonathon Shlens, and Quoc V Le. 2019. Do better imagenet models transfer better?. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2661–2671.

[16] Anne-Laure Ligozat, Julien Lefèvre, Aurélie Bugeau, and Jacques Combaz. 2021. Unraveling the hidden environmental impacts of AI solutions for environment. *arXiv preprint arXiv:2110.11822* (2021).

[17] Tao Luo, Weng-Fai Wong, Rick Siow Mong Goh, Anh Tuan Do, Zhixian Chen, Haizhou Li, Wenyu Jiang, and Weiyun Yau. 2023. Achieving Green AI with Energy-Efficient Deep Learning Using Neuromorphic Computing. *Commun. ACM* 66, 7 (2023), 52–57.

[18] Michal Pándy, Andrea Agostinelli, Jasper Uijlings, Vittorio Ferrari, and Thomas Mensink. 2022. Transferability estimation using bhattacharyya class separability. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 9172–9182.

[19] Aleksandrs Slivkins. 2019. Introduction to multi-armed bandits. *Foundations and Trends® in Machine Learning* 12, 1-2 (2019), 1–286.

[20] Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. *arXiv preprint arXiv:1906.02243* (2019).

[21] Anh T Tran, Cuong V Nguyen, and Tal Hassner. 2019. Transferability and hardness of supervised classification tasks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1395–1405.

[22] Carole-Jean Wu, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, Fiona Aga, Jinshi Huang, Charles Bai, et al. 2022. Sustainable ai: Environmental implications, challenges and opportunities. *Proceedings of Machine Learning and Systems* 4 (2022), 795–813.

[23] Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V Le, Denny Zhou, and Xinyun Chen. 2023. Large language models as optimizers. *arXiv preprint arXiv:2309.03409* (2023).

[24] Xingyi Yang, Daquan Zhou, Songhua Liu, Jingwen Ye, and Xinchao Wang. 2022. Deep model reassembly. *Advances in neural information processing systems* 35 (2022), 25739–25753.

[25] Kaichao You, Yong Liu, Jianmin Wang, and Mingsheng Long. 2021. Logme: Practical assessment of pre-trained models for transfer learning. In *International Conference on Machine Learning*. PMLR, 12133–12143.

[26] Yi-Kai Zhang, Ting-Ji Huang, Yao-Xiang Ding, De-Chuan Zhan, and Han-Jia Ye. 2023. Model Spider: Learning to Rank Pre-Trained Models Efficiently. *arXiv preprint arXiv:2306.03900* (2023).

[27] You Zhou, Xiujing Lin, Xiang Zhang, Maolin Wang, Gangwei Jiang, Huakang Lu, Yupeng Wu, Kai Zhang, Zhe Yang, Kehang Wang, et al. 2023. On the Opportunities of Green Computing: A Survey. *arXiv preprint arXiv:2311.00447* (2023).