Masters Thesis

Bridging the Gap Between Research and Practice: An Analysis of Machine Learning and Deep Learning Model Adoption on Kaggle

Department of Statistics
Ludwig-Maximilians-Universität München

Davit Martirosyan

Munich, November, 2024

Supervised by Matthias Feurer & Matthias Aßenmacher

Contents

Abstract

The alignment between machine learning (ML) and deep learning (DL) models proposed in academic research and those employed in practical applications remains an open question. This study investigates the models utilized by practitioners on Kaggle to determine whether they align with those advocated in scholarly papers for equivalent tasks. We developed a sophisticated data collection pipeline using The Onion Router (TOR) to securely scrape and compile a comprehensive dataset of models used across various Kaggle competitions. Leveraging the ChatGPT-4o API, we analyzed and cleaned this dataset to extract insights into the models favored by practitioners. While we amassed extensive results detailing the practical implementations on Kaggle, attempts to perform a parallel analysis of academic research were hindered by the lack of structured datasets and the complexity of extracting model information from research papers. Our efforts to extract model names from abstracts of papers from ACL, ICLR, and NeurIPS were fruitless, as the abstracts often lacked explicit model names, were highly specific, or referred to models by unique names coined by the authors. These challenges highlight the difficulties in comparing practical implementations with academic proposals and underscore the need for more accessible and structured dissemination of research findings to bridge the gap between theoretical advancements and real-world applications in ML and DL.

1 Introduction

The rapid evolution in ML and DL has led to a proliferation of models and techniques emerging from academic research. Conferences such as the Association for Computational Linguistics (ACL), the International Conference on Learning Representations (ICLR), and the Conference on Neural Information Processing Systems (NeurIPS) are at the forefront of introducing innovative algorithms that push the boundaries of what is possible in areas like natural language processing (NLP), computer vision, and reinforcement learning. These developments hold significant potential for impacting practical applications, offering new tools and methods for tackling complex real-world problems.

Parallel to these academic advancements, platforms like Kaggle have become hubs for practitioners to apply ML and DL techniques to solve practical challenges. Kaggle hosts competitions that attract data scientists and engineers who implement models to achieve the best performance on specific tasks. This environment provides a unique opportunity to observe the models and methods that practitioners prefer when faced with real-world data and constraints.

Despite the ongoing innovations in academia, there is a perception of a disconnect between the models proposed in research papers and those employed by practitioners. Factors such as model complexity, computational resource requirements, interpretability, and the practicality of implementation may influence the adoption of new academic models in industry and practice. Understanding whether practitioners are utilizing the latest academic models or relying on more established techniques is crucial. It can inform researchers about the real-world impact of their work, guide educators in aligning curricula with industry needs, and help bridge the gap between theory and practice.

This study aims to investigate the alignment between the ML and DL models proposed in academic research and those used by practitioners on Kaggle for similar tasks. Specifically, we seek to answer the following research questions: To what extent do practitioners on Kaggle adopt the latest models introduced in academic conferences such as ACL, ICLR, and NeurIPS? What are the most commonly used ML and DL models among winning solutions in Kaggle competitions? What factors influence the selection of models by practitioners in real-world applications?

To explore these questions, we focused on collecting and analyzing data from Kaggle winning competition solutions to identify the models most commonly used by participants. We developed a sophisticated data collection pipeline utilizing TOR network to securely scrape and compile a comprehensive dataset of models implemented across various competitions. For data analysis and cleaning, we leveraged the ChatGPT-4o API, which provided advanced NLP capabilities to extract meaningful insights from the collected data. The use of ChatGPT allowed us to handle unstructured text data effectively, identifying model names, architectures, and other relevant information from Kaggle notebooks and discussion forums.

While our initial objective included performing a parallel analysis of academic research papers to directly compare the models proposed in literature with those used in practice, we encountered many obstacles. Processing entire research papers through the ChatGPT-4o API proved unfeasible due to token limits, accuracy loss, and inference time. Attempts to extract model names from the abstracts of papers published in leading conferences were also unsuccessful. The abstracts often lacked explicit mentions of model names, were highly specific to niche contributions, or introduced new models with unique names coined by the authors, making it difficult to categorize and compare them systematically.

These obstacles highlighted a key issue: the unstructured and diverse nature of academic publications poses significant challenges for large-scale automated analysis. Unlike Kaggle, where model implementations are often directly referenced and discussed openly, academic papers may not present model information in a standardized or easily extractable manner. This lack of structured data inhibits the ability to perform comprehensive comparisons between academic proposals and practical implementations.

Despite these problems, our analysis of Kaggle data yielded extensive results. We identified trends in model usage among practitioners, noting preferences for certain architectures, frameworks, and techniques. These findings provide valuable insights into the models that are most effective or accessible in practical settings. They also underscore the importance of factors such as ease of use, computational efficiency, and community support in model adoption.

Our contributions in this study are multifaceted. We developed a novel data collection and analysis pipeline that can serve as a framework for future research in this area. We provided empirical evidence on the models preferred by practitioners in real-world competitions, highlighting the gap between academic research and industry practice. Furthermore, we offered reflections on the challenges of analyzing academic literature using automated tools, emphasizing the need for more accessible and structured dissemination of research findings.

In this thesis, we aim to contribute to the ongoing dialogue about aligning ML and DL research with industry needs. By revealing the models favored in practice and the barriers to analyzing academic proposals, we aim to inform researchers, educators, and practitioners alike.

We begin with a review of relevant literature, followed by a detailed description of our methodology. We then present our findings from the Kaggle data analysis and discuss the implications of our results. Finally, we conclude with reflections on the limitations of our study and suggestions for future research directions.

By exploring the alignment between academic models and those used in practice, this research endeavors to bridge the gap between theory and application. It seeks to enhance understanding of how academic advancements can be more effectively translated into tools that address real-world challenges, ultimately fostering a more cohesive relationship between researchers and practitioners in the field of ML and DL.

2 Related Work

The intersection of academic research and industry practice in ML has been the subject of various studies aiming to understand the translation of theoretical advancements into practical applications. Previous research has explored the adoption of cutting-edge models in industry, highlighting factors that influence practitioners' choices.

For instance, Sculley et al. (2015) discussed the challenges of deploying ML systems in production environments, emphasizing issues like system complexity and maintainability. They argued that while novel models may offer performance improvements, the added complexity can hinder their adoption in real-world applications.

In the field of NLP, there is a notable divergence between academic research and industry application. Academic studies often prioritize optimizing model accuracy through complex architectures, while industry practitioners emphasize computational efficiency and interpretability to meet practical deployment constraints. This disparity is highlighted in the work by Ruder et al. (2022), who discuss the "square one bias" in NLP research. They observe that the prototypical NLP experiment focuses on training standard architectures on labeled English data, optimizing for accuracy without accounting for other dimensions such as fairness, interpretability, or computational efficiency. This one-dimensional focus means that research often explores only a fraction of the NLP research space, leading to a gap between research and practical application.

ML benchmarking environments provide a systematic way for practitioners to assess the trade-offs between accuracy and computational efficiency. Bischl et al. (2019) introduced the OpenML Benchmarking Suites, enabling researchers and practitioners to evaluate models like gradient boosting machines and random forests across various datasets and computational settings. Such frameworks help practitioners identify which models perform reliably within practical constraints, indirectly encouraging the selection of efficient and interpretable models even if this involves a slight accuracy trade-off.

Regarding tools and frameworks, the emergence of large language models such as OpenAI's ChatGPT (OpenAI, 2024) and Meta AI's LLaMA (Touvron et al., 2023) series has influenced both research and practice. These models have demonstrated remarkable capabilities in natural language understanding and generation, leading to their adoption in various applications (Brown et al., 2020). The accessibility and versatility of these models make them valuable resources for tasks like data analysis, code generation, and content creation.

In our methodology, the decision to utilize the ChatGPT-4o API over models like LLaMA was influenced by considerations of performance, ease of integration, and the availability of support resources. While LLaMA models offer open-source alternatives, the infrastructure and expertise required to deploy them effectively may pose challenges. The use of ChatGPT provided a balance between advanced capabilities and practical implementation within our project's scope. Our work contributes to the existing literature by providing empirical insights into the models used by practitioners on Kaggle and reflecting on the challenges of aligning academic research with practical applications. By highlighting the obstacles in analyzing academic literature and the trends observed in practitioner model selection, we aim to inform future efforts to bridge the gap between ML and DL research and industry practice.

3 Data

This section describes the data sources, collection methods, and preprocessing steps taken to build a dataset of ML models used in Kaggle competitions and to attempt a comparable dataset from academic sources.

3.1 Data Collection and Preprocessing

To investigate the models utilized by practitioners on Kaggle and compare them with those proposed in academic research, we undertook a comprehensive data collection process. This section details the methodologies employed in acquiring and preparing data from both Kaggle competitions and academic conference abstracts.

3.2 Data Acquisition from Kaggle

We developed a robust data collection pipeline that utilizes TOR network (Tor Project, 2024) to securely and anonymously scrape data from Kaggle's winning solutions platform. By leveraging TOR's IP rotation capabilities, the pipeline mitigates potential issues related to network blocks and throttling. Initial attempts using Selenium (Selenium Project, 2024) with a Chrome browser proved ineffective due to frequent IP blocks, necessitating the integration of Selenium with TOR to take advantage of its IP rotation and enhanced anonymity features. Despite these measures, many of TOR's default IP addresses were still detected and subsequently blocked. To address this, we implemented a retry mechanism that automatically relaunches the browser until a functional IP address is obtained, repeating the process whenever an IP block occurs.
The data scraping focused on extracting relevant information from Kaggle competitions and their corresponding winning solutions. We targeted competitions across various domains to capture a diverse range of tasks and models. The scraping process resulted in a dataset comprising 2,618 observations, each representing a unique competition and its winning solution.

3.2.1 Kaggle Dataset Structure

The initial Kaggle dataset included the following columns:

- **Competition Link**: The URL to the Kaggle competition page.

- **Competition Title**: The official title of the competition.
- **Competition Short Description**: A brief summary of the competition's objective.
- **Competition Extended Description**: A more detailed explanation of the competition.
- **Competition Tags**: Keywords and tags associated with the competition, indicating the type of task (e.g., classification, regression, computer vision).
- **Winning Solution Links**: URLs to the winning solution notebooks or repositories.
- **Solution Description**: Descriptions provided by the competition winners detailing their approach and models used.
- **Date**: Solution submission date, for time series analysis purposes.

Among these columns, the most critical for our analysis were the Competition Tags and the Solution Description. The Competition Tags enabled us to categorize each competition by task type, while the Solution Description provided insights into the models and techniques employed by the winning practitioners.

3.2.2 Preprocessing and Cleaning of Kaggle Data

The raw Kaggle data required extensive preprocessing to ensure accuracy and consistency in our analysis. Several challenges were identified during the cleaning process:

1. **Inconsistencies in Model Naming**: Model names in the solution descriptions were not standardized. For example, the Light Gradient Boosting Machine was variably referred to as "LGBM," "LightGBM," "lightgm," "lgb," etc. This variability necessitated a normalization process to standardize model names for accurate aggregation and analysis.
2. **Typographical Errors and Informal Language**: As the solution descriptions were manually inputted by users, they contained typographical errors, abbreviations, and informal language, which could hinder automated text processing.
3. **Incomplete or Vague Descriptions**: Some solution descriptions lacked sufficient detail about the models used, requiring us to infer or exclude such entries from certain analyses.
4. **Irrelevant Information**: Descriptions sometimes included extraneous information unrelated to the models, such as personal anecdotes or acknowledgments, which needed to be filtered out.

To address these challenges, we implemented a multi-step cleaning processing (specific prompts can be found in the Appendix:

1. **Automated Text Normalization**: We employed text preprocessing techniques to convert all text to lowercase, remove punctuation, and standardize commonly used terms and abbreviations. Regular expressions were used to identify and replace variant spellings of model names.

2. **Leveraging ChatGPT-4o API for Data Cleaning**: We utilized the ChatGPT-4o API to assist in cleaning and normalizing the solution descriptions. The API's natural language understanding capabilities allowed us to:

   - **Extract Model Names**: Parse the solution descriptions to identify mentions of specific models and architectures.
   - **Standardize Terminology**: Convert colloquial expressions and abbreviations into standardized terms.
   - **Correct Typographical Errors**: Identify and correct common misspellings of model names and technical terms.

3. **Manual Verification**: Despite the automated processes, we conducted manual reviews of a subset of the data to ensure the accuracy of the cleaning procedures. This step helped validate the effectiveness of the automated cleaning and allowed us to make adjustments as necessary.

4. **Categorization of Competitions**: Using the Competition Tags, we mapped each competition to specific task categories (e.g., image classification, time series forecasting, NLP). This categorization facilitated the analysis of model usage trends within and across different task types.

3.3 Data Acquisition from Academic Conferences

In an effort to compare the models used in practice with those proposed in academic research, we attempted to collect data from leading ML conferences. We wrote a script to scrape and randomly select 10 abstracts each from the Association for ACL, ICLR, and NeurIPS. This resulted in a collection of 30 abstracts representing recent research contributions in the field.

3.3.1 Academic Abstracts Dataset Structure

The collected abstracts were stored along with metadata, including:

- **Paper Title**: The title of the research paper.
- **Authors**: The list of authors contributing to the paper.
- **Conference Source**: The conference where the paper was presented (ACL, ICLR, or NeurIPS).

- **Abstract Text**: The text of the abstract summarizing the research work.

3.3.2 Challenges in Processing Academic Abstracts

Processing the academic abstracts presented significant challenges:

1. **Lack of Explicit Model Names**: Many abstracts did not explicitly mention the names of the models proposed or used. Instead, they focused on the problem addressed or the methodology without naming specific models.

2. **Highly Specific or Novel Models**: When model names were mentioned, they often referred to new models introduced by the authors, with unique names not widely recognized or used in practice.

3. **Complex Language and Terminology**: The abstracts contained dense academic language and domain-specific terminology, making it difficult to extract model information using automated tools.

4. **Limited Information in Abstracts**: Abstracts provide a concise summary and may omit detailed information about the models and algorithms used, which are typically found in the full paper.

3.3.3 Attempted Processing with ChatGPT and Manual Inspection

We employed the ChatGPT-4o API to parse the abstracts and extract model names and related information. The API was tasked with identifying any mentions of models or architectures and standardizing them for comparison. However, the API often failed to extract model names due to their absence or obscurity in the abstracts, and, in some cases, the API misinterpreted technical terms or methodologies as model names, leading to inaccurate data.

3.3.4 Example Abstracts

The prompts for model name extraction from the abstracts can be found in the Appendix. The following three examples of abstracts along with ChatGPT's outputs will depict the problems described above:

1. **Abstract**: Lexical complexity is a highly subjective notion, yet this factor is often neglected in lexical simplification and readability systems which use a ”one-size-fits-all” approach. In this paper, we investigate which aspects contribute to the notion of lexical complexity in various groups of readers, focusing on native and non-native speakers of English, and how the notion of complexity changes depending on the proficiency level of a non-native reader. To facilitate reproducibility of our approach and foster further research into these aspects, we release a dataset of complex words annotated by readers with different backgrounds. (Gooding et al., 2021)

1.1. **Output**: []

2. **Abstract**: Learning effective protein representations is critical in a variety of tasks in biology such as predicting protein function or structure. Existing approaches usually pretrain protein language models on a large number of unlabeled amino acid sequences and then finetune the models with some labeled data in downstream tasks. Despite the effectiveness of sequence-based approaches, the power of pretraining on known protein structures, which are available in smaller numbers only, has not been explored for protein property prediction, though protein structures are known to be determinants of protein function. In this paper, we propose to pretrain protein representations according to their 3D structures. We first present a simple yet effective encoder to learn the geometric features of a protein. We pretrain the protein graph encoder by leveraging multiview contrastive learning and different self-prediction tasks. Experimental results on both function prediction and fold classification tasks show that our proposed pretraining methods outperform or are on par with the state-of-the-art sequence-based methods, while using much less pretraining data. Our implementation is available at https://github.com/DeepGraphLearning/GearNet. (Zhang et al., 2023)

2.1. **Output**: ['GearNet']

3. **Abstract**: The task of point cloud segmentation, comprising semantic, instance, and panoptic segmentation, has been mainly tackled by designing task-specific network architectures, which often lack the flexibility to generalize across tasks, thus resulting in a fragmented research landscape. In this paper, we introduce ProtoSEG, a prototype-based model that unifies semantic, instance, and panoptic segmentation tasks. Our approach treats these three homogeneous tasks as a classification problem with different levels of granularity. By leveraging a Transformer architecture, we extract point embeddings to optimize prototype-class distances and dynamically learn class prototypes to accommodate the end tasks. Our prototypical design enjoys simplicity and transparency, powerful representational learning, and ad-hoc explainability. Empirical results demonstrate that ProtoSEG outperforms concurrent well-known specialized architectures on 3D point cloud benchmarks, achieving 72.3%, 76.4% and 74.2% mIoU for semantic segmentation on S3DIS, ScanNet V2 and SemanticKITTI, 66.8% mCov and 51.2% mAP for instance segmentation on S3DIS and ScanNet V2, 62.4% PQ for panoptic segmentation on SemanticKITTI, validating the strength of our concept and the effectiveness of our algorithm. The code and models are available at https://github.com/zyqin19/PROTOSEG. (Qin et al., 2023)

3.1. **Output**: ['ProtoSEG', 'Transformer']

The aforementioned examples are arranged in order of increasing usefulness, with the first being the least useful and the last being relatively more useful. In the case of the first abstract,

ChatGPT's output was an empty list. As previously discussed, one of the primary challenges in extracting model names from abstracts is the absence of explicit mentions of model names within the abstract itself. In this instance, both ChatGPT and manual inspection confirmed that there are no model names mentioned in the paper's abstract, resulting in an empty output.
The second abstract output presents another example that appears more useful, as it includes a model name "GearNet." However, this is not actually useful, as it is simply the name of the method proposed by the paper itself, rather than a widely recognized model.
Finally, the last case exemplifies the most useful type of output among these examples, though it still offers limited utility from a data standpoint. The first element of the output list is "ProtoSEG," which, as can be observed, is once again the name of the method implemented by the authors of the paper. The second element, "Transformer," is slightly more informative, as it provides insight into the general DL architecture employed. However, this is still not substantially valuable because, first, it lacks the granularity of the Kaggle data, and second, such outputs are rare among the sample of thirty abstract outputs analyzed.

3.3.5 Summary of Data Limitations from Academic Sources

The challenges encountered highlighted significant limitations in using abstracts for model extraction. The inability to reliably extract model names from abstracts meant that we could not compile a meaningful dataset of models proposed in academic research for comparison with Kaggle data. Also, the unstructured and variable nature of abstracts made automated processing difficult without access to full papers or additional context.

Despite extensive efforts, the challenges in processing academic abstracts prevented us from constructing a comparable dataset of models proposed in academic research. As a result, our analysis focuses primarily on the Kaggle data, providing insights into the models favored by practitioners. The difficulties encountered underscore the need for more structured and accessible reporting of model information in academic publications to facilitate such comparative studies.

3.4 Final Dataset for Analysis

After the comprehensive cleaning and feature engineering process, the final dataset included the following additional columns:

- **Extracted Model Names**: Specific models mentioned in the solution descriptions, standardized for consistency.
- **Generalized Model Names**: Broader categories under which the extracted models fall, facilitating higher-level analysis.
- **Filtered Generalized Model Names**: A refined list focusing on the most relevant and frequently used models, enhancing the clarity of the analysis.

- **Clean Competition Tags**: Standardized tags that accurately represent the competition tasks, enabling consistent categorization.
- **Broader Hierarchical Group**: Overarching categories representing major areas in ML and DL, aiding in trend identification across different domains.

By augmenting the dataset with additional columns and hierarchical groupings, we enhanced our ability to analyze and interpret the data. These enhancements allowed us to conduct a more comprehensive examination of the models used by practitioners on Kaggle. The final dataset's structure, with standardized and generalized model names and task categories, provided a solid foundation for identifying broader trends and insights in model adoption across different ML and DL tasks.

The meticulous data collection and preprocessing efforts resulted in a robust dataset that forms the foundation of our analysis. While we faced challenges on the academic side, the enriched Kaggle dataset enables us to contribute valuable knowledge to the discourse on the alignment between academic research and practical applications in ML and DL.

4 Results & Analysis

This section presents ChatGPT-4o API's accuracy, an in-depth analysis of model usage trends, the top models within specific competition tags and the patterns in model selection across tasks, providing insights into practitioner preferences.

In order to assess the effectiveness of the ChatGPT-4o API in accurately extracting model names from solution descriptions, we conducted an evaluation using a subset of the Kaggle dataset. This section outlines the methodology of the evaluation, presents the results, and discusses the implications of the findings.

4.1 Evaluation Methodology

To evaluate the performance of the ChatGPT-4o API in extracting model names, we created a test set by randomly selecting 139 rows from the Kaggle dataset. This subset was reserved exclusively for testing purposes and was not used during any of the preprocessing or cleaning stages to ensure an unbiased assessment.

4.1.1 Manual Extraction of Model Names (Ground Truth)

For each of the 139 solution descriptions in the test set, we manually extracted the model names mentioned. This manual extraction served as the ground truth against which the ChatGPT-4o API's performance would be measured. The manual extraction process involved: Reading each solution description thoroughly, identifying all mentions of ML and DL models and recording the exact model names and normalizing them according to the standards established during preprocessing.

4.1.2 Automated Extraction Using the ChatGPT-4o API

We then utilized the ChatGPT-4o API to automatically extract model names from the same set of solution descriptions. The API was prompted to: Analyze each solution description, identify and list all ML and DL models mentioned and output the extracted model names in a standardized format.

4.1.3 Comparison and Evaluation Metrics

To compare the ChatGPT-4o API's extracted model names with the manually obtained ground truth, we employed the F1 score and Jaccard index evaluation metrics. These metrics were chosen for their suitability in evaluating the accuracy of extraction tasks, particularly when dealing with sets of items (in this case, model names). The F1 score is a harmonic mean of precision and recall, providing a single metric that balances both false positives and false negatives. It is particularly useful when the dataset is imbalanced or when both precision (proportion of correctly identified items among all items) and recall (proportion of correctly identified items among all relevant items) are important. The F1 score is calculated as follows:

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{1}$$

where:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}, \tag{2}$$

and

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}. \tag{3}$$

The F1 score ranges from 0 to 1, where 1 indicates perfect precision and recall, and 0 indicates the worst performance.

The Jaccard Index, also known as the Intersection over Union, measures the similarity between two sets. It is calculated as the size of the intersection divided by the size of the union of the two sets:

$$Jaccard\ Index = \frac{|A \cap B|}{|A \cup B|}, \tag{4}$$

where:

- A is the set of model names extracted by the ChatGPT-4o API, and
- B is the set of model names from the ground truth.

The Jaccard index ranges from 0 to 1, where 1 indicates perfect performance, and 0 indicates the worst performance.

4.1.4 Final Accuracy

After conducting the comparison using the aforementioned metrics, we obtained the following results:

- **F1 Score**: 0.85
- **Jaccard Index**: 0.78

The evaluation demonstrates that the ChatGPT-4o API is a reliable tool for extracting model names from unstructured text in solution descriptions. The high F1 score suggests that the API effectively captures relevant information with minimal errors, making it suitable for large-scale data processing tasks where manual extraction would be impractical.

4.1.5 Limitations and Considerations

While the results are promising, it is important to consider potential limitations:

- **Contextual Understanding**: The API may sometimes misinterpret context, especially if the solution descriptions use unconventional terminology or abbreviations not encountered during its training.
- **Variability in Descriptions**: Since solution descriptions can vary widely in style and detail, there may be instances where the API's performance could differ from the test results.
- **Dependence on Input Quality**: The quality of the input text affects the API's performance. Poorly written or ambiguous descriptions may lead to lower extraction accuracy.

The following table depicts some examples of actual model names (model names that should have been extracted from the given text) and model names extracted by the ChatGPT-4o API:

True Model Names	**Extracted Model Names**
nn model	lstm
UNet++	UNet++, Long Short-term Memory
	RotNet, VGG
deberta-large, rembert, roberta	deberta-large, rembert
Cascade Mask RCNN, Convnext v2 large	Cascade Mask RCNN, Convnext v2 large

Table 1: Comparison of true and extracted model names

Table 1 presents five examples comparing the true model names with those extracted by the ChatGPT-4o API. In the first example, there is a complete mismatch between the true and extracted model names. The second example demonstrates a case where the API correctly extracted the model name but also included an additional incorrect element. In the third example, the API was unable to analyze the context accurately. While the solution description mentioned certain model names, these were merely referenced and not actually employed in the solution. Consequently, the true model name was an empty list, whereas the extracted model names constituted an incorrect extrapolation. The fourth example illustrates a scenario where the API partially extracted the correct model names but omitted "RoBERTa." The final example represents a correct extraction by the API.

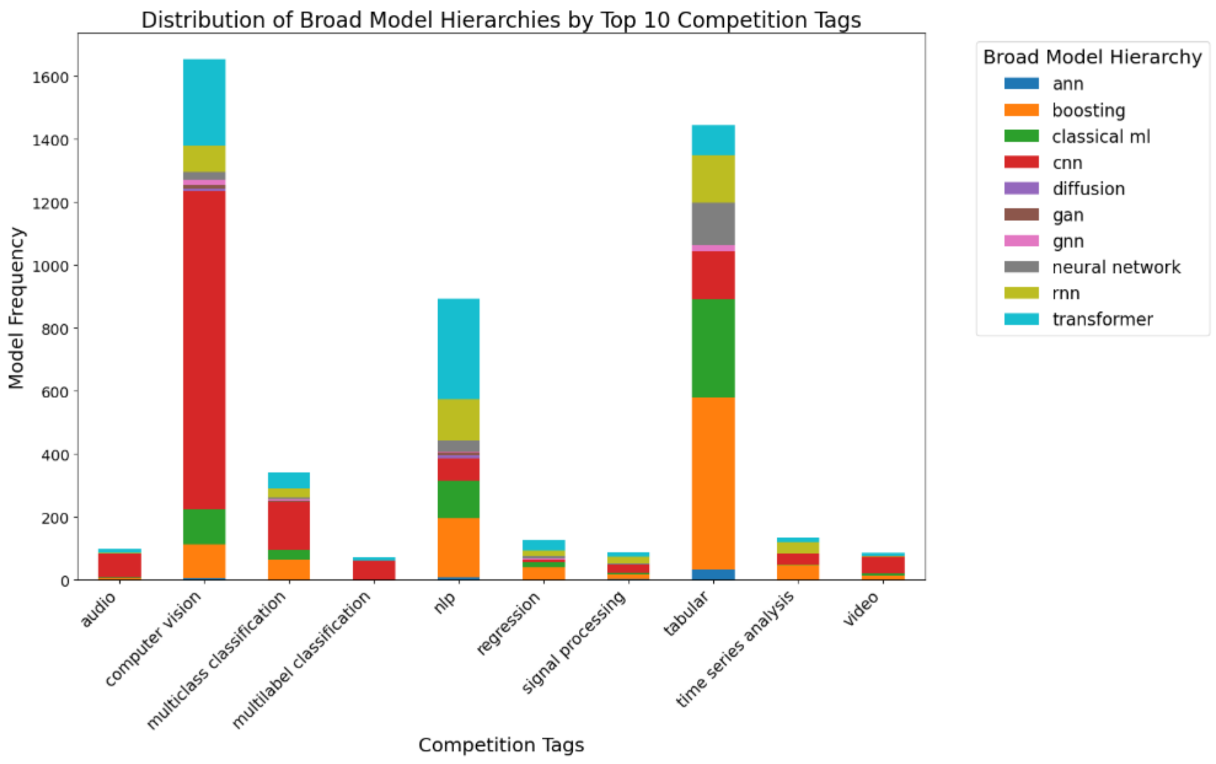

Figure 1: Frequency plot of high level model usage among top 10 competition tags

As illustrated in Figure 1, computer vision tasks overwhelmingly favor convolutional neural networks (CNNs), which constitute the majority of model types used. This preference reflects CNNs' efficacy in handling spatial data and their established role as the standard approach for image processing tasks. Transformers, though less prevalent than CNNs, also appear in computer vision tasks. This trend highlights the emerging utility of transformer-based architectures, such as Vision Transformers (ViTs), which have recently gained traction for various image classification and segmentation applications.

For NLP tasks, the predominant model hierarchy is transformer-based architectures. This is consistent with recent advances in the field, where models like BERT, GPT, and T5 have set new benchmarks across a wide range of language understanding tasks. In addition to transformers, traditional neural networks and classical ML methods are also present within the NLP category, though their usage is markedly lower. This suggests that while other model types remain relevant, transformers are the preferred choice due to their superior performance on sequence-to-sequence and contextual understanding tasks.

In competitions involving tabular data, boosting models emerge as a favored approach, underscoring their effectiveness for structured data. Models such as XGBoost, LightGBM, and CatBoost are widely utilized due to their capability to handle heterogeneous data types and their strong performance in predictive tasks. The use of various model types in the tabular category, including classical ML and neural networks, indicates a diverse approach to structured data problems. This diversity suggests that practitioners select models based on specific task

requirements, leveraging boosting models for their robustness and interpretability.
Hence, generally, the presence of traditional models, such as classical ML algorithms, across various categories indicates that despite the rise of DL, simpler models still hold value in certain domains, particularly where interpretability and resource constraints are prioritized.

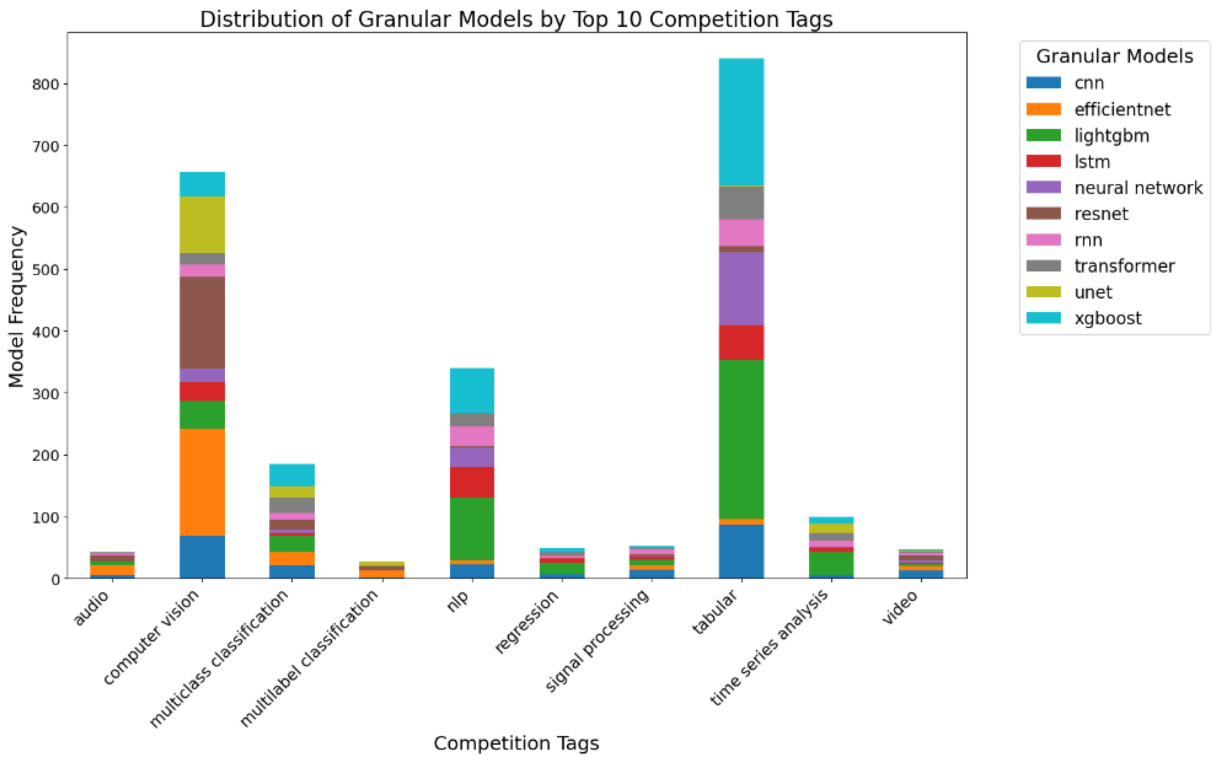

Figure 2: Frequency plot of granular model usage among top 10 competition tags

Figure 2 provides a more detailed breakdown of model types compared to Figure 1, specifying finer-grained model architectures. In the tabular data competition tag, LightGBM and XGBoost are prominently utilized, underscoring their popularity in structured data tasks. For computer vision competitions, the distribution is dominated by neural network architectures such as EfficientNet, ResNet, and U-Net, reflecting their widespread application in image processing and segmentation tasks.

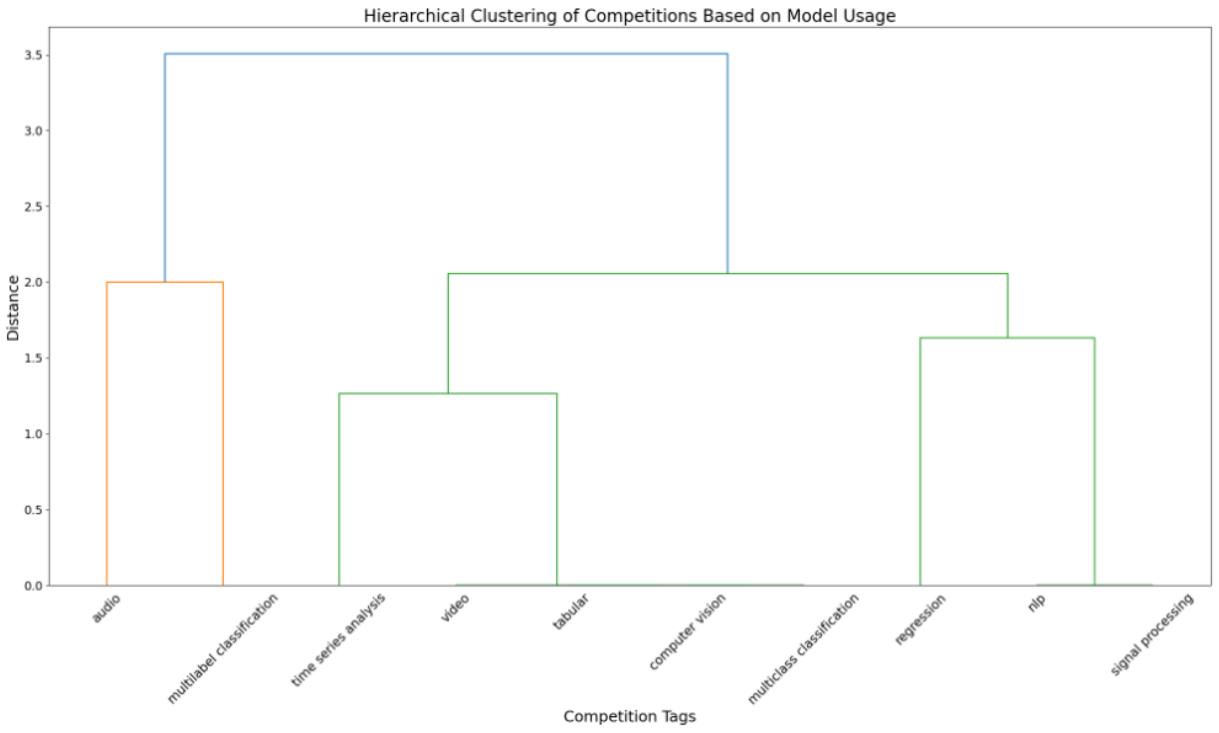

Figure 3: Dendogram of competition tags and model usage

Figure 3 illustrates the hierarchical clustering of competition tags, based on model usage frequency, where the vertical axis represents the distance (or dissimilarity) between clusters. This dendrogram provides insights into how closely related the model preferences are across various competition tags, which helps identify domains with similar model usage patterns.

The first major division occurs between the audio and multilabel classification tags and the rest of the tags, indicating a significant dissimilarity in model usage. This suggests that the models favored in audio and multilabel classification tasks are quite distinct from those used in other types of competitions. A second grouping is evident between video, tabular, and time series analysis, which form a closer cluster. This indicates that these competition tags share similar model types, possibly favoring models optimized for temporal or sequential data (e.g., LSTM, GRU, or traditional boosting methods for tabular data).

The computer vision and multiclass classification tags are highly similar, suggesting a strong overlap in the types of models used for these tasks. Given the prevalence of CNNs and DL architectures in both domains, this close relationship aligns with the reliance on similar neural network architectures. Another close clustering is seen between regression and NLP tags. This may reflect a shared usage of transformer-based or neural network models that are effective in learning complex patterns, relevant to both regression tasks and language-based tasks.

The hierarchical clustering also reveals that signal processing and regression are the furthest apart within their subgroup, suggesting distinct model usage patterns in signal processing tasks compared to regression-based competitions. This could imply that signal processing relies more on specialized models that can handle frequency or waveform data.

This hierarchical clustering helps to contextualize the selection of model hierarchies across different ML tasks. Understanding these relationships can guide practitioners in model selection, emphasizing the specialization required for each domain. The clustering further reveals how certain competition tags share overlapping model architectures, which can inform future research on the transferability of model types across tasks.

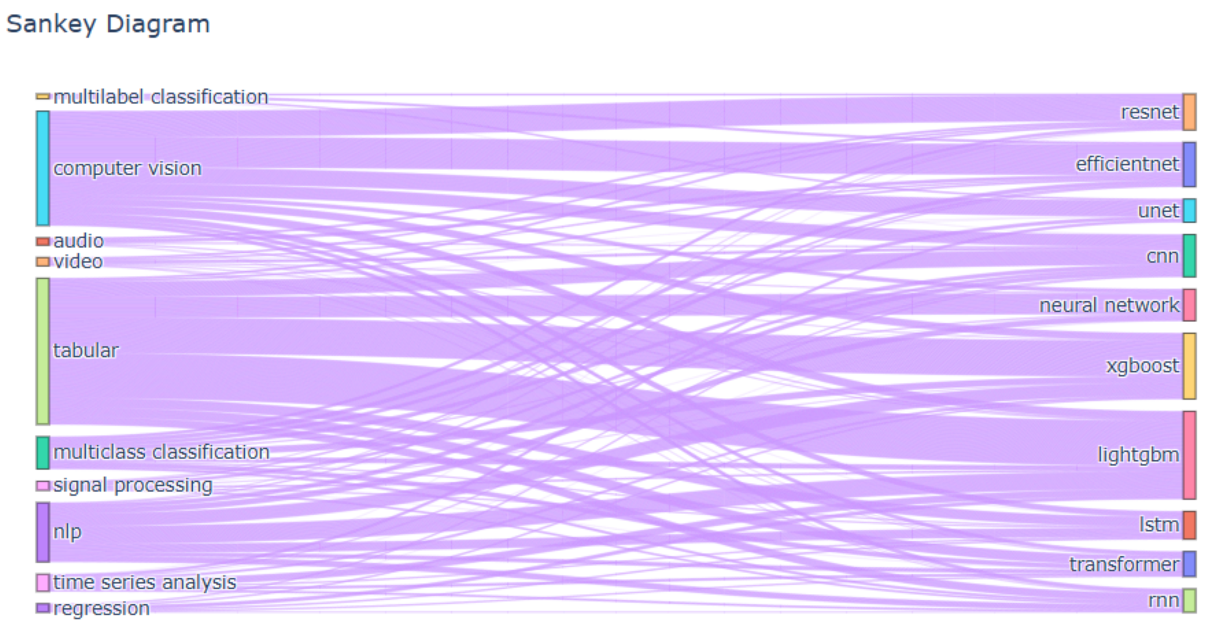

Figure 4: Sankey diagram

The Sankey diagram in Figure 4 visualizes the connections between various competition tags on the left and the specific model types on the right. Each flow line represents the relationship between a competition tag and a model type, with the thickness of the lines indicating the frequency or strength of these connections.
The tabular competition tag exhibits strong connections to LightGBM, XGBoost, and neural networks. This indicates a clear preference for boosting algorithms and neural network models for tabular data, which are known for their adaptability to structured datasets and effectiveness in handling complex features. For NLP competitions, transformers are the most frequently associated model type. The computer vision tag shows significant flows toward CNN, EfficientNet, ResNet, and U-Net architectures. These models are widely recognized for their effectiveness in image-related tasks. Specifically, CNNs are fundamental to image processing, while EfficientNet, ResNet, and U-Net are specialized architectures for classification, feature extraction, and segmentation, respectively. The diversity in model types within computer vision tasks suggests that practitioners select models based on the specific requirements, such as classification, detection, or segmentation. Neural networks (as a general category) and transformers show connections to multiple competition tags, including NLP, regression, multiclass classification, and signal processing. This versatility highlights the adaptability of these architectures across

different domains, which likely benefits from their ability to model complex, non-linear relationships. RNNs and LSTMs are predominantly associated with time series analysis and signal processing tasks, indicating that recurrent neural networks remain relevant for sequential data due to their inherent design for handling time-dependent relationships.

Certain competition tags have unique model preferences. For example, audio and video tasks appear to have connections to a broad range of models but maintain distinct associations with CNN and transformer models. This likely reflects the diversity of data types in audio and video tasks, which can include sequential, spatial, and contextual elements. The signal processing tag also shows connections with CNN and transformer models, suggesting that these models are adapted for tasks that require high-dimensional feature extraction, commonly found in signal processing applications.

Overall, the diagram emphasizes the alignment between competition domains and model architectures. For instance, boosting models like LightGBM and XGBoost are almost exclusively linked to tabular data tasks, while transformers are central to NLP and increasingly relevant in computer vision and signal processing. The interconnectedness of tags like multiclass classification and computer vision with multiple models underlines the variety of approaches practitioners use within these fields, adapting model architectures to meet diverse objectives. This diagram provides a comprehensive view of model selection patterns across different ML tasks. It highlights the role of specialized models within specific domains, while also showing the flexibility of more general architectures (e.g., neural networks, transformers) across multiple types of competitions. Such visualizations can inform both practitioners and researchers about prevailing trends in model usage, enabling a better understanding of how task requirements influence model choice.

4.2 Time series Analysis

In this section we will display the time series plots for the top three most frequently used models in the top three most frequently encountered competition tags. Additional visualizations will be provided in the Appendix.

4.2.1 Computer Vision

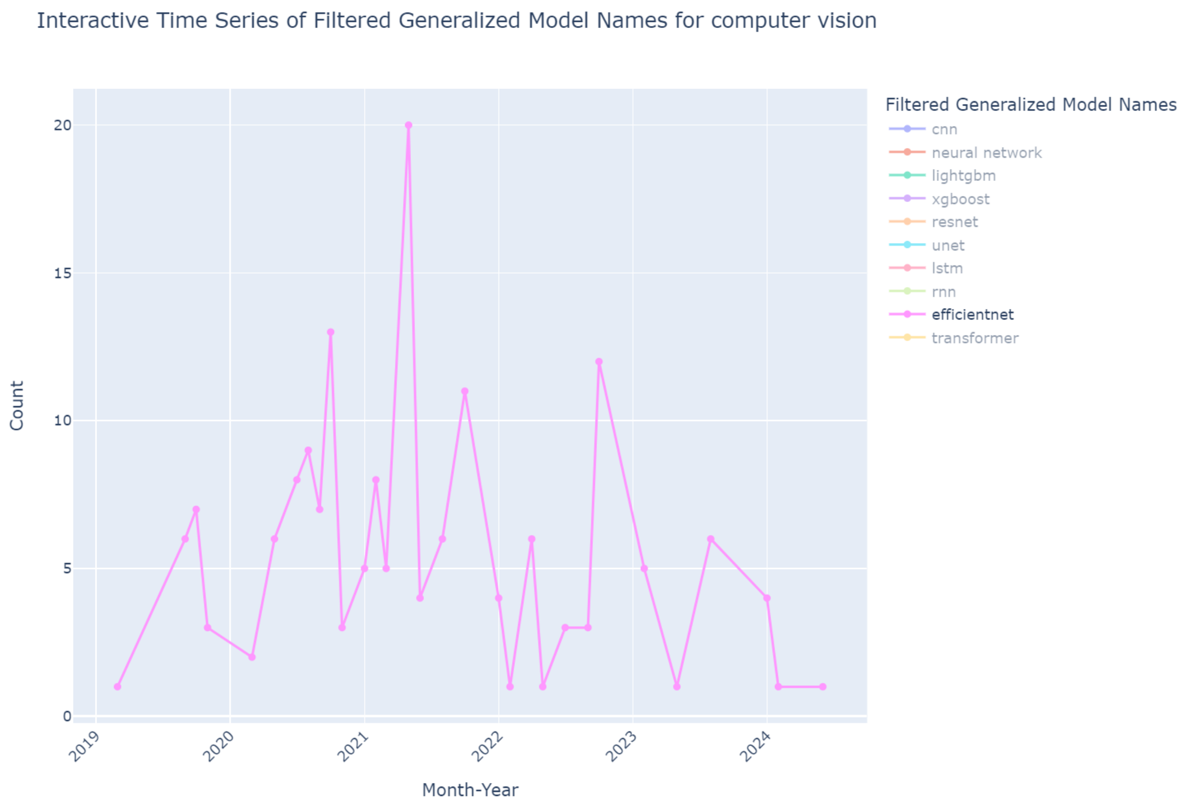

Figure 5: Time series plot for the EfficientNet architecture stratified on the computer vision competition tag

The plot in Figure 5 shows the monthly counts of EfficientNet usage within computer vision competitions from 2019 to 2024. The usage of EfficientNet shows a general upward trend from late 2019 through to a peak in 2021, indicating a period of rapid adoption within computer vision competitions. This rise correlates with the period following EfficientNet's introduction, as it quickly gained popularity due to its state-of-the-art performance and parameter efficiency for image classification tasks. The highest recorded frequency occurs in early 2021, with a noticeable count of over 20 uses in competitions. This peak suggests that EfficientNet was particularly favored in this period, likely due to increased recognition of its efficacy in improving model accuracy while reducing computational cost. The overall trend after 2021 appears to decline slightly, indicating that while EfficientNet remains relevant, other model architectures (such as Vision Transformers) might be influencing the choice of model in computer vision tasks. From mid-2023 onward, there is a notable decrease in the frequency of EfficientNet usage, with the counts dropping to very low levels by early 2024. This suggests a shift in the preferred architectures for computer vision tasks, possibly due to the emergence of newer models with competitive performance, and also reflects the natural progression in ML competitions, where novel architectures often replace older ones as benchmarks evolve and new research validates alternative approaches.

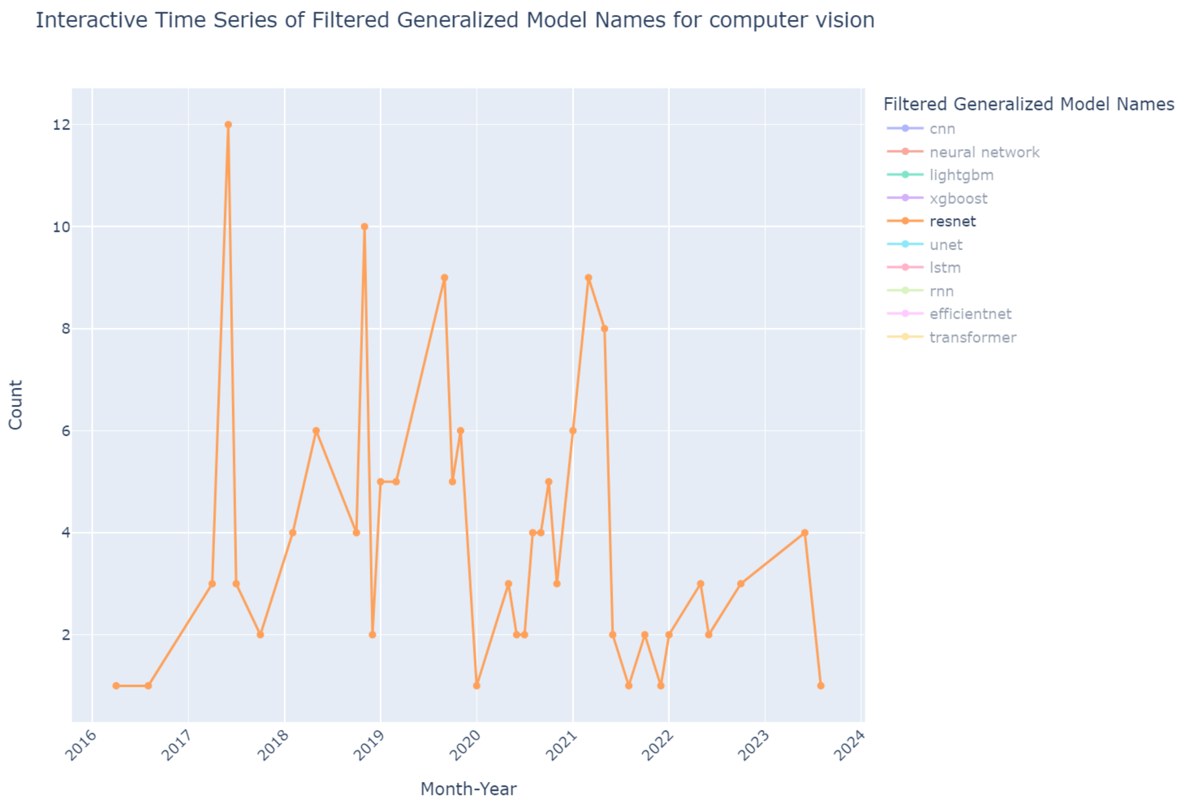

Figure 6: Time series plot for the ResNet architecture stratified on the computer vision competition tag

In Figure 6 is depicted the time series plot for ResNet within computer vision competitions from 2016 to 2024. ResNet shows an early adoption trend, with initial peaks observed in 2017, suggesting that it quickly became popular after its introduction. The count reaches a high of around 12 during this period, underscoring ResNet's immediate impact on computer vision competitions due to its innovative residual learning framework, which effectively addresses vanishing gradient issues in deep networks. This early rise in usage reflects ResNet's status as a foundational model for image classification and feature extraction, making it a favored choice for a variety of computer vision tasks. From 2017 to 2021, the usage of ResNet demonstrates fluctuating peaks and troughs. The frequency of ResNet's application appears to vary with certain periods of increased interest, particularly around 2018 and 2019. This fluctuation may indicate a competitive landscape in model selection within computer vision tasks, where practitioners alternate between different architectures based on the task requirements or new advancements in model design. Another prominent peak is observed in early 2021, suggesting that ResNet remains relevant in computer vision competitions over an extended period. After 2021, there is a noticeable decline in ResNet usage, with fewer peaks and lower overall counts. The reduction in frequency likely reflects the emergence of newer architectures, such as Vision Transformers, which have challenged traditional CNN-based approaches by offering improved parameter efficiency and performance.

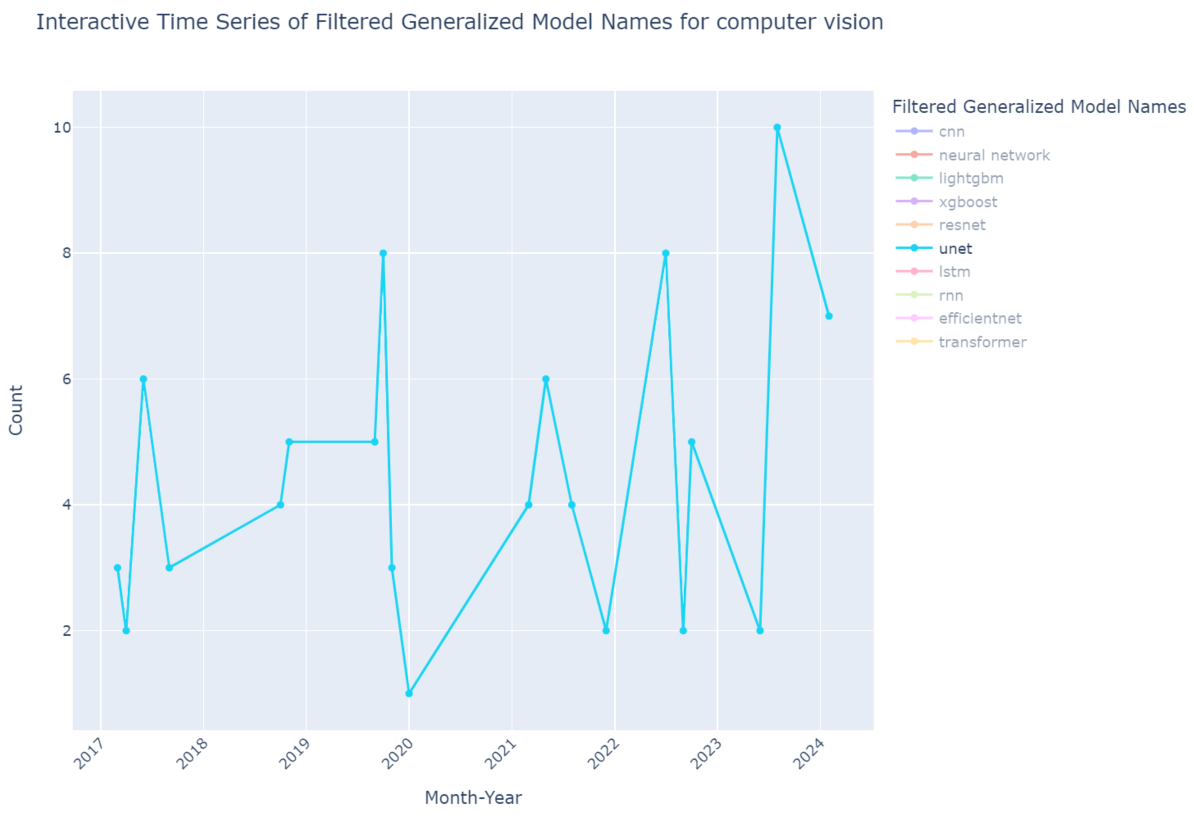

Figure 7: Time series plot for the U-net architecture stratified on the computer vision competition tag

The time series plot in Figure 7 depicts the monthly counts of U-Net usage within computer vision competitions from 2017 to 2024. U-Net exhibits a consistent level of adoption starting in 2017, with usage counts fluctuating during the early years. This steady usage reflects U-Net's reputation as a go-to architecture for image segmentation tasks, particularly in fields such as medical imaging where precise delineation is crucial. The plot shows a peak around 2018, where U-Net's popularity reaches a high of approximately 8 counts. From 2020 to 2023, the usage pattern of U-Net becomes more variable, with noticeable peaks and troughs. For instance, there is a sharp increase in early 2023, reaching another high of about 10 counts, indicating renewed interest. The time series shows a peak in early 2024, after which U-Net's usage begins to decrease again. This recent decline may indicate a shift towards alternative models for segmentation tasks, such as Transformer-based models (e.g., SegFormer) or more advanced CNN-based architectures.

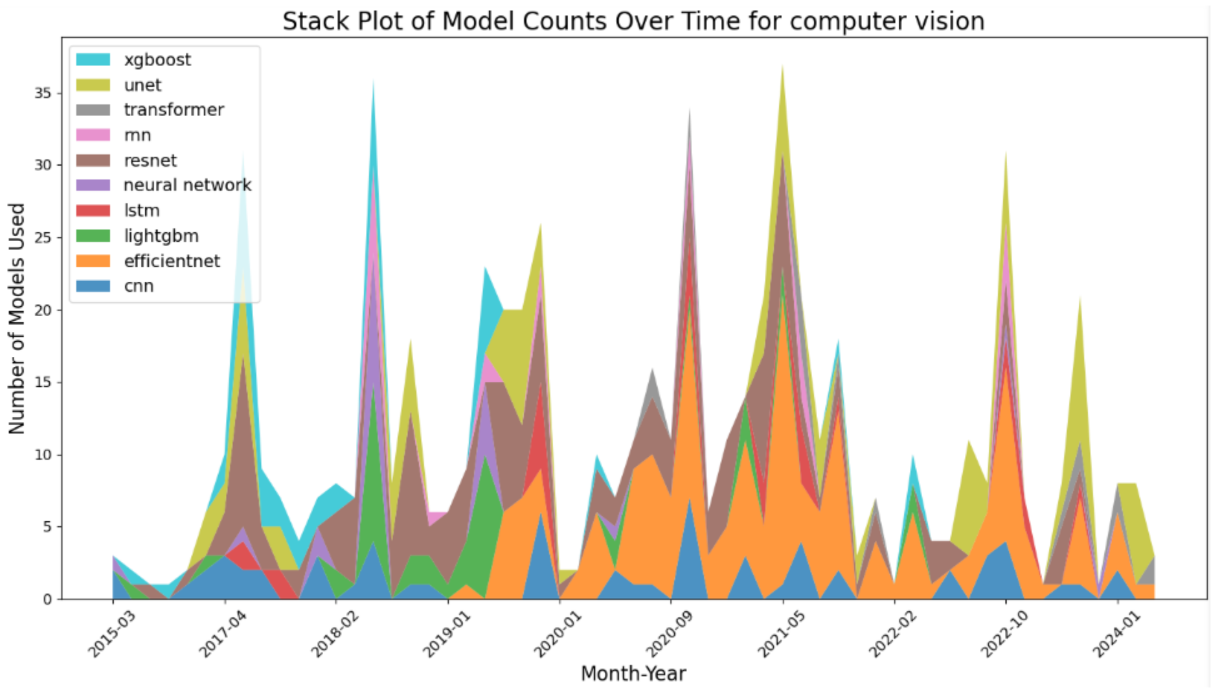

Figure 8: Stackplot for the computer vision competition tag

The stackplot in Figure 8 shows an alternative depiction of all model usages over time for the computer vision tag. Throughout the timeline, U-net, ResNet and EfficientNet models consistently appear, highlighting the central role of convolutional architectures in computer vision tasks. The substantial area covered by these models reflects their persistent popularity and versatility in handling image data. Starting around 2020, transformer models begin to appear more prominently, indicating a shift towards transformer-based architectures in computer vision. This trend aligns with the increasing adoption of Vision Transformers (ViTs) for classification and detection tasks in image processing. The presence of multiple model types—ranging from LSTM and RNN (for sequence-based tasks) to LightGBM and XGBoost—demonstrates the diverse approaches taken by practitioners in computer vision competitions. This suggests that while CNN-based models dominate, other architectures are periodically leveraged for specific tasks that may involve temporal or tabular data alongside image data.

4.2.2 Tabular

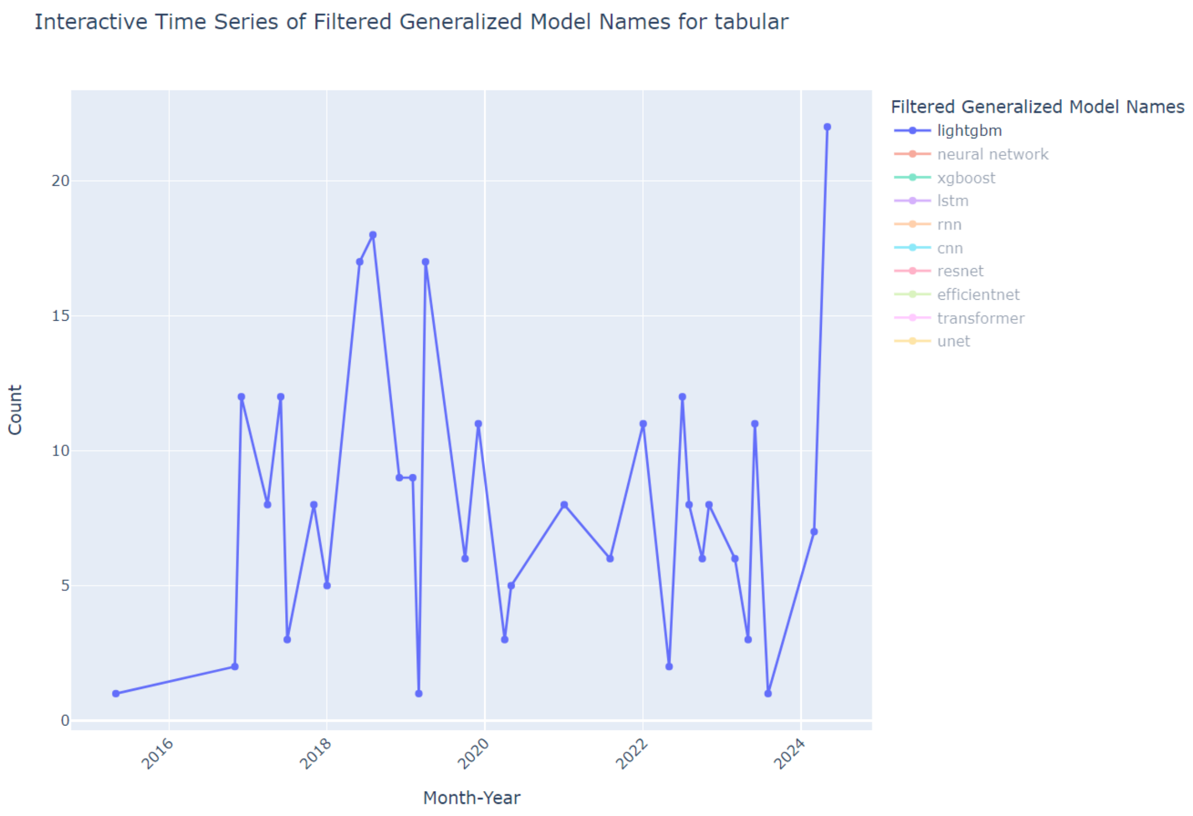

Figure 9: Time series plot for LightGBM stratified on the tabular competition tag

The plot in Figure 9 shows the monthly counts of LightGBM usage within tabular data competitions from 2016 to 2024. LightGBM begins to show notable usage in 2017, with an upward trend and periodic peaks over time. The model reaches its first significant peak around 2018, reflecting its adoption as a preferred model for tabular data tasks due to its high efficiency and accuracy in handling structured datasets. This early adoption underscores LightGBM's effectiveness as a gradient boosting model, optimized for speed and performance, making it well-suited for tabular competitions that prioritize predictive accuracy on structured data. Despite fluctuations, LightGBM maintains a consistent presence, with counts rarely dropping to zero, indicating that it remains a staple choice for tabular data tasks. The plot shows a sharp increase in LightGBM usage in early 2024, reaching a high of over 20 counts. This resurgence may reflect a renewed focus on tabular competitions where gradient boosting models are particularly advantageous, possibly driven by competitions emphasizing interpretability and efficiency. While other models may periodically rise in popularity, LightGBM continues to be a dominant choice for practitioners in tabular data competitions.

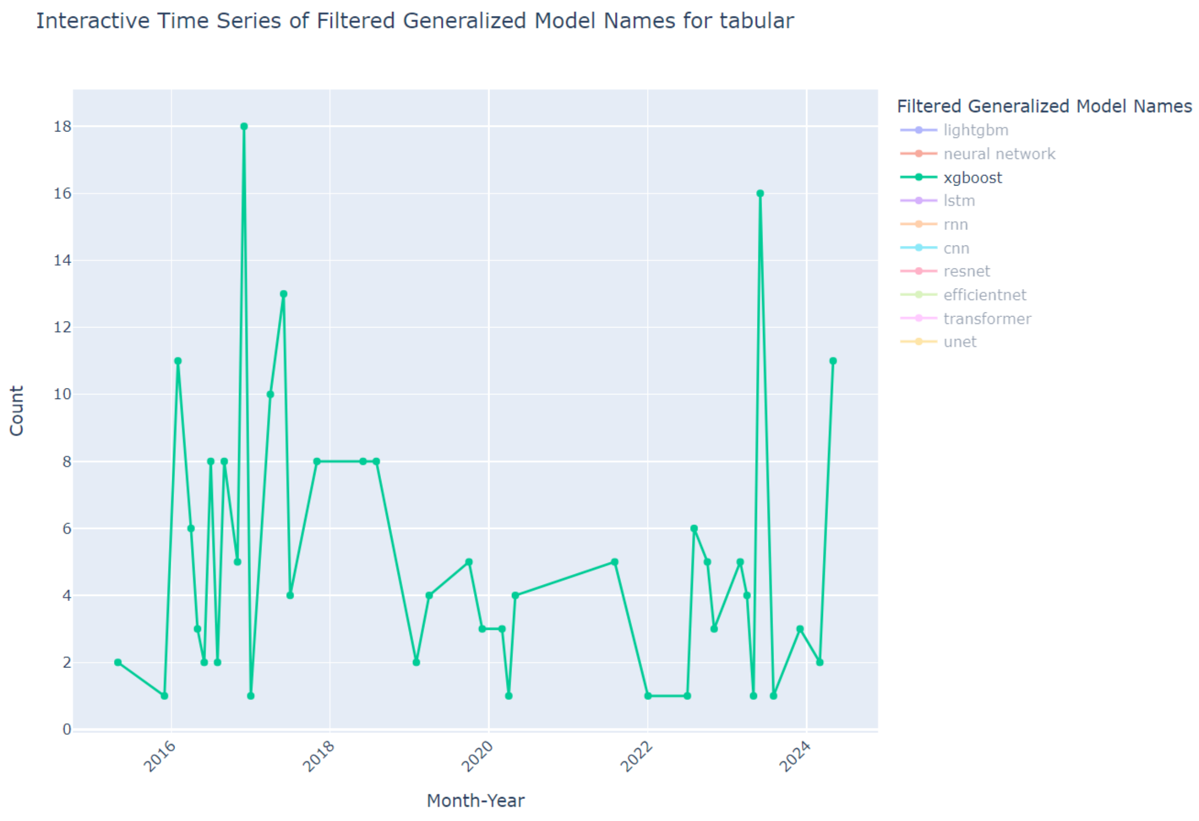

Figure 10: Time series plot for XGBoost stratified on the tabular competition tag

The plot in Figure 10 shows the monthly counts of XGBoost usage within tabular data competitions from 2016 to 2024. XGBoost shows a strong presence from the beginning of the timeline, with substantial usage spikes in 2016 and 2017, reaching counts as high as 18. This early popularity reflects XGBoost's effectiveness and its position as one of the pioneering gradient boosting models widely adopted for structured data tasks. After the initial peaks, the usage of XGBoost remains steady with fluctuations over the years, particularly noticeable from 2017 to 2020. During this period, counts vary but generally remain above a baseline of 4 to 6, suggesting XGBoost's enduring relevance despite competition from newer models. The recent peaks underscore XGBoost's staying power in the field, as it continues to be chosen frequently, particularly in tabular competitions where model interpretability and execution speed are valued. Ultimately, while there are periods of reduced usage, XGBoost consistently appears throughout the timeline, indicating that it has become a reliable mainstay for tabular data tasks. Its frequent resurgence and steady baseline counts suggest that practitioners continue to favor XGBoost for competitions focused on structured data, leveraging its scalability and optimization capabilities.

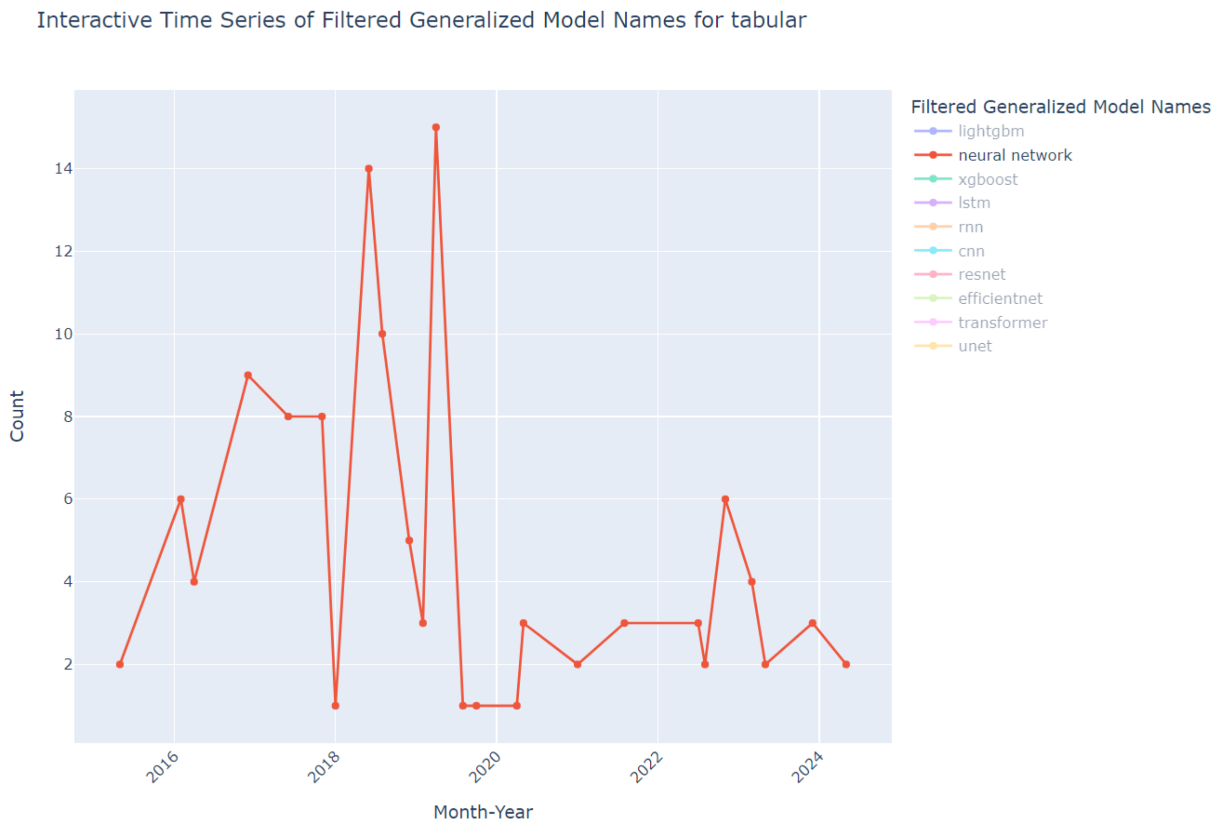

Figure 11: Time series plot for neural networks stratified on the tabular competition tag

The time series plot in Figure 11 depicts the monthly counts of neural network usage within tabular data competitions from 2016 to 2024. Neural networks demonstrate an early increase in usage starting in 2016, with peaks around 2017 and 2018, reaching counts as high as 15. This reflects the growing experimentation with neural networks for tabular data during this period, indicating an initial surge in interest, possibly driven by advances in neural network architectures and the broader adoption of DL approaches across various ML tasks. After the peaks in 2017 and 2018, there is a noticeable decline and more sporadic usage from 2019 onwards, with counts often dropping to low levels by 2020. This downward trend suggests that, while neural networks remain relevant, their application in tabular data competitions may have been overshadowed by more specialized models like gradient boosting frameworks (e.g., LightGBM, XGBoost), which are often more efficient for structured data. The long-term trend suggests that neural networks have not established themselves as a dominant model in tabular competitions. Their occasional resurgence indicates that they may be applied for specific tabular tasks, particularly where non-linear relationships are crucial. However, more efficient models such as gradient boosting machines have likely become the standard choice for these tasks.

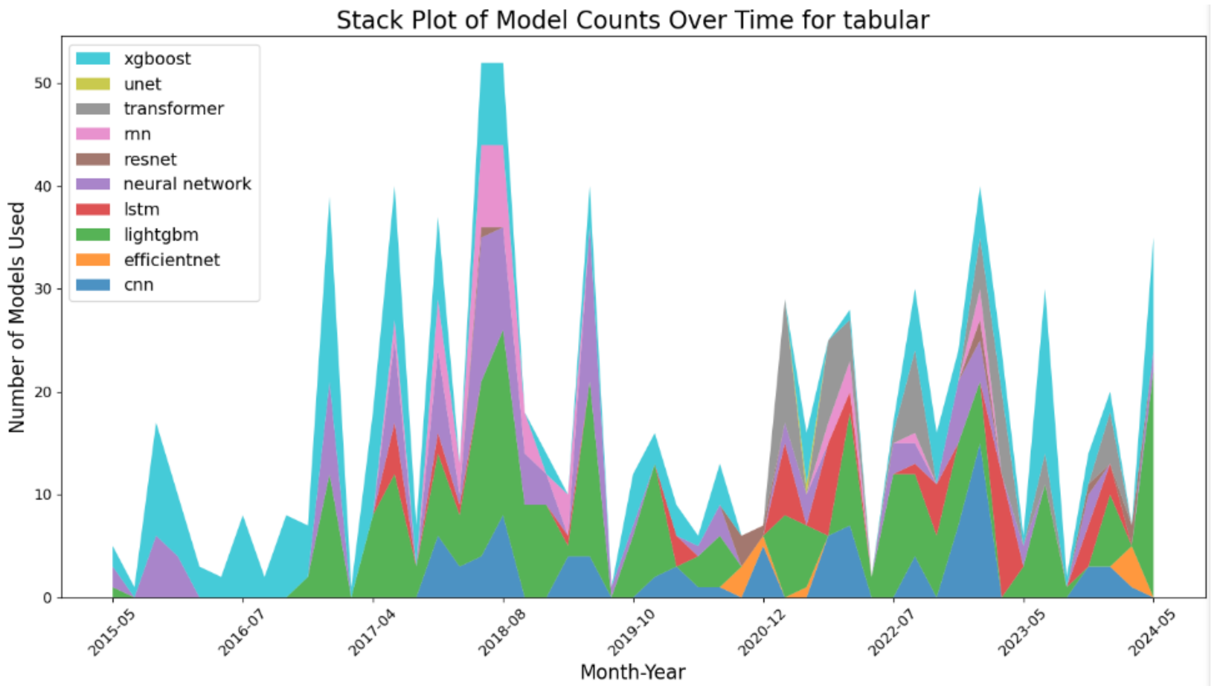

Figure 12: Stackplot for the tabular competition tag

The stackplot in Figure 12 shows an alternative depiction of all model usages over time for the tabular tag. XGBoost is the dominant model throughout the timeline, as indicated by the large area they occupy in the plot. This trend reflects the strong preference for gradient boosting frameworks in tabular competitions. The prominence of these models underscores their effectiveness in handling the unique challenges of tabular data, such as heterogeneous feature types and the need for interpretability. Neural networks appear intermittently but in smaller areas, suggesting their exploratory use for certain tasks. However, their lesser prominence indicates that they are generally not the primary choice for tabular data, which often favors models optimized for structured data. Peaks in overall model counts occur periodically, particularly around 2017, 2020, and 2023, suggesting bursts of increased activity and experimentation with multiple model types. These spikes may correspond to competition cycles or specific tasks that prompt competitors to try a range of models. Throughout the years, XGBoost and LightGBM show consistent peaks, reflecting their enduring relevance. These models dominate the stack plot, especially during key peaks, underscoring their status as go-to solutions for tabular data competitions. The plot shows a marked increase in LightGBM usage over time, hinting at its increasing preference in recent years due to its speed and efficiency enhancements over XGBoost.

4.2.3 NLP

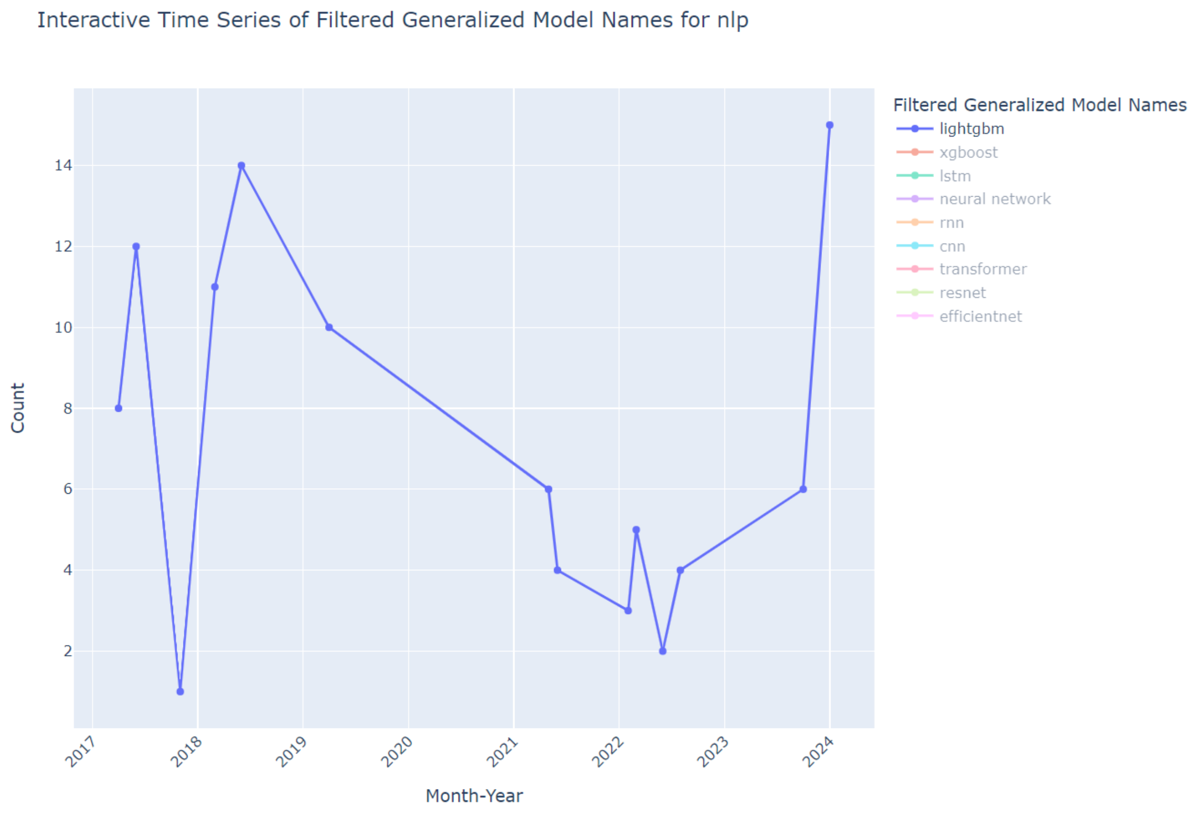

Figure 13: Time series plot for LightGBM stratified on the NLP competition tag

The plot in Figure 13 illustrates the monthly counts of LightGBM usage within NLP competitions from 2017 to 2024. LightGBM shows significant usage in the early years, with a peak around 2018, reaching a count of over 14. The high early counts suggest that LightGBM was initially explored in NLP tasks, possibly for handling features like word embeddings or structured metadata that lend themselves to gradient boosting approaches. After the peak in 2018, there is a gradual decline in LightGBM usage, with counts steadily dropping through 2021. This decline corresponds with the rise of specialized NLP models, such as transformers, which are more effective at handling sequence-based and contextual language data. Between 2021 and 2023, LightGBM usage remains low, showing occasional small peaks but not regaining the earlier levels. This suggests that while LightGBM may still be utilized for certain NLP tasks, its role is limited as the competition landscape shifts towards architectures better suited for NLP. The plot reveals a sudden resurgence in LightGBM usage in 2024, reaching a count as high as the earlier peak. This increase might indicate renewed interest or the appearance of NLP tasks where gradient boosting frameworks are beneficial, such as those involving structured data or hybrid models where tabular features complement text features. The overall trend underscores that while LightGBM may have limited use in pure NLP contexts, it still retains utility for hybrid tasks where its strengths can complement language models.

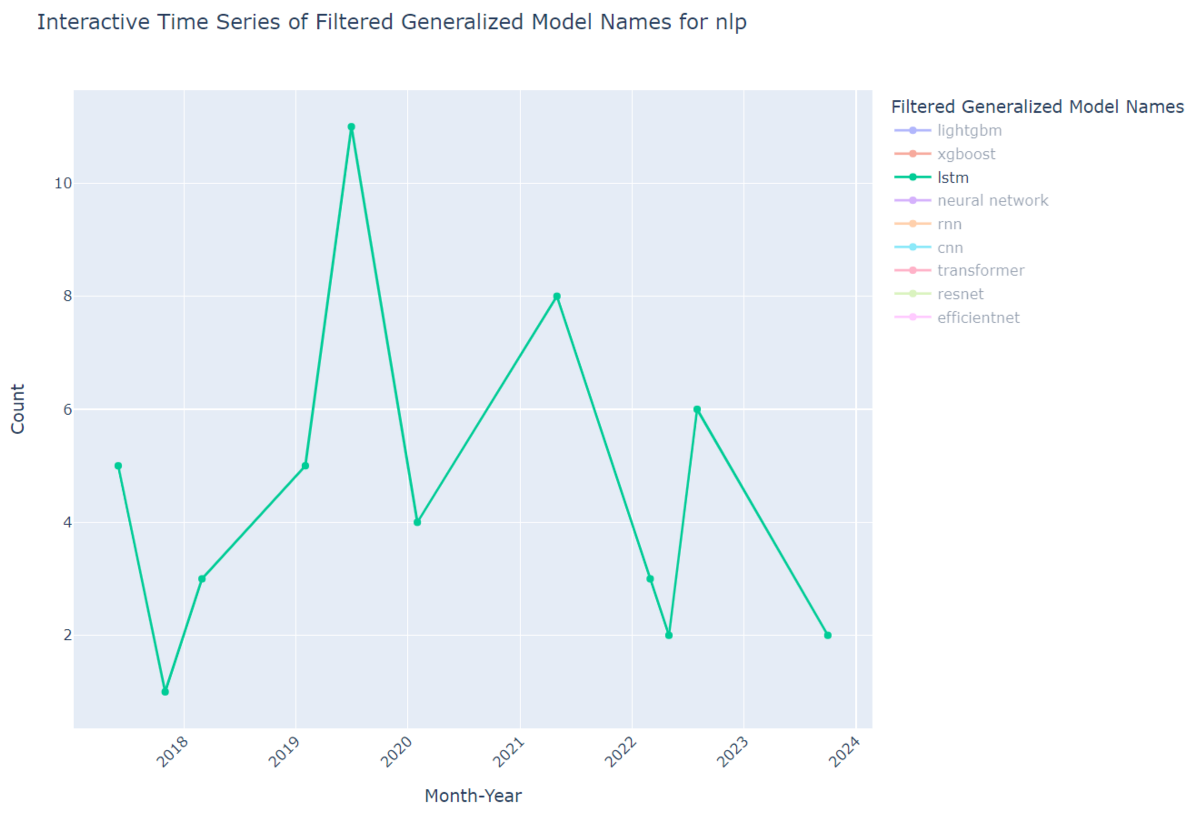

Figure 14: Time series plot for LSTM stratified on the NLP competition tag

The plot in Figure 14 shows the monthly counts of LSTM usage within NLP competitions from 2017 to 2024. LSTM usage begins around 2017 and shows a gradual rise, reaching a peak in 2020 with a count of over 10. This early trend reflects the widespread adoption of LSTMs in NLP tasks, given their ability to handle sequential data and maintain long-term dependencies, which are critical for language processing. The consistent growth leading up to 2020 highlights LSTMs' position as a standard model for various NLP tasks, including language modeling, text classification, and sentiment analysis. After the peak in 2020, the plot shows a noticeable decline, with counts decreasing steadily through 2021 and reaching a low in 2022. This downward trend aligns with the rise of transformer-based models, which have largely replaced LSTMs in many NLP tasks. From 2022 onward, LSTM usage remains low, with only minor increases. This pattern suggests that, while still present, LSTMs are no longer a preferred choice for most NLP competitions, as transformers have become the dominant architecture for language tasks. The plot demonstrates that LSTMs, once foundational for NLP tasks, have been gradually supplanted by newer architectures. The recurring but diminishing peaks imply that LSTMs retain value for certain applications, although they are generally no longer the go-to model for mainstream NLP competitions.

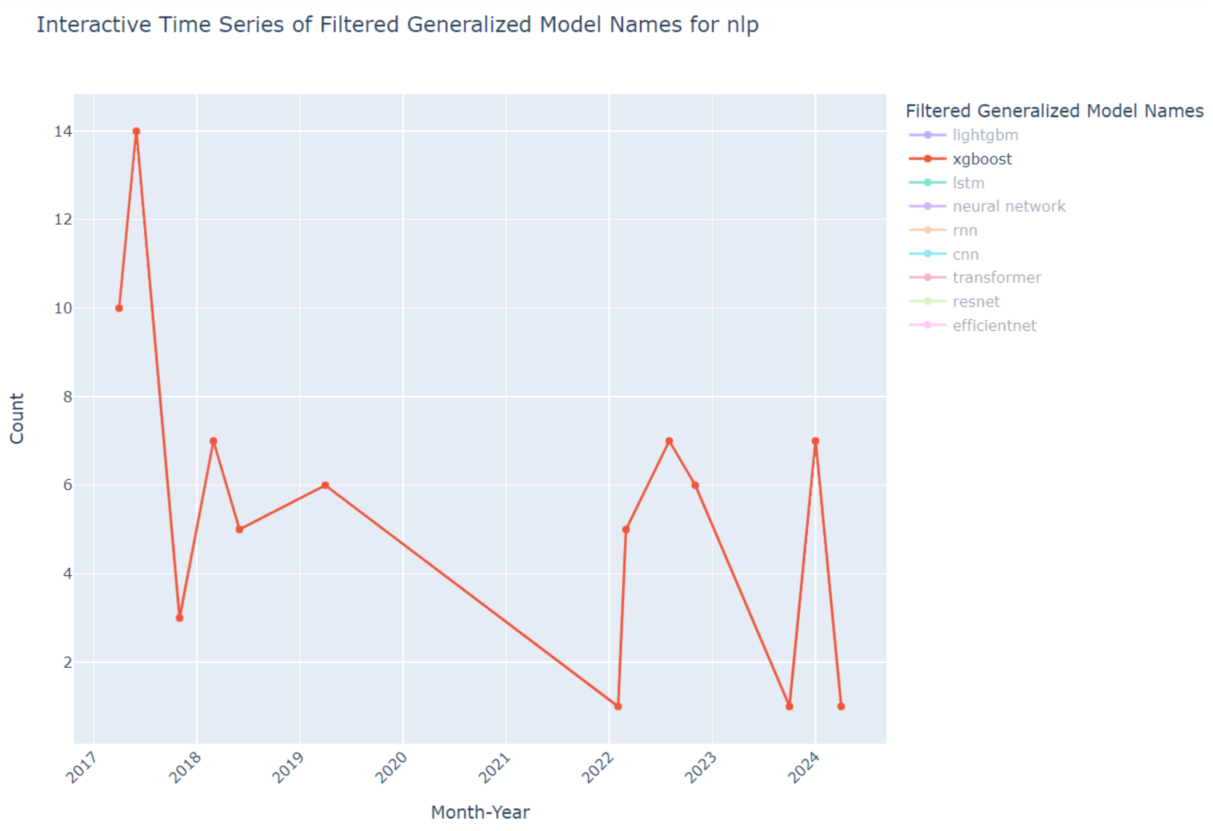

Figure 15: Time series plot for XGBoost stratified on the NLP competition tag

The plot in Figure 15 shows the monthly counts of XGBoost usage within NLP competitions from 2017 to 2024. XGBoost shows significant usage in the early years, with a peak around 2018, reaching a count of over 14. This early high usage indicates that XGBoost was explored as a robust model for handling structured features that might accompany NLP tasks, such as metadata or numerical data within language-based competitions. Following the peak in 2018, XGBoost usage exhibits a gradual decline, with counts steadily decreasing through 2021. This drop suggests that as NLP models like transformers became dominant, XGBoost's role diminished, likely due to its limitations in processing unstructured text data directly. From 2021 onwards, XGBoost usage remains relatively low, with occasional minor peaks but consistently reduced counts compared to its early usage. This low baseline reflects a shift away from gradient boosting methods in favor of architectures specifically designed for NLP tasks. The trend over time reveals that XGBoost's utility in NLP competitions is now limited to specific applications where its strengths align with the task requirements. Its early popularity suggests it was initially experimented with for a broader range of tasks, but the shift to NLP-specific models has curtailed its usage in recent years.

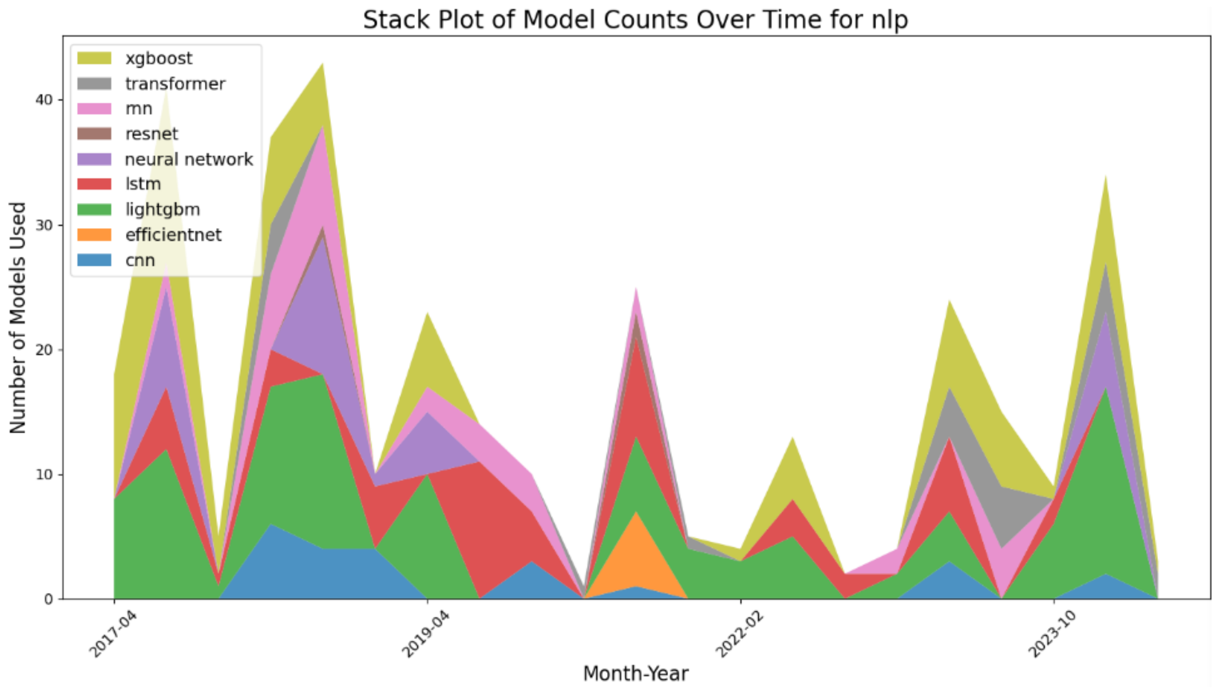

Figure 16: Stackplot for the NLP competition tag

Figure 16 reveals that XGBoost (yellow) and transformers (gray) have a prominent presence across multiple peaks, particularly after 2019. This suggests that while transformers are preferred for handling sequential language data due to their context-aware capabilities, gradient boosting models like XGBoost are commonly applied in tasks where structured data or feature engineering complements NLP. LSTMs (red) and neural networks (purple) appear consistently, particularly in the peaks before 2020. Their presence indicates that, prior to transformers' dominance, these models were foundational for NLP tasks, capturing sequential dependencies essential for language tasks. The plot shows multiple peaks where different models are used simultaneously, suggesting an experimental approach within NLP competitions. Notable spikes around 2017, 2020, and 2023 reflect a diversity of models applied to NLP tasks, implying that participants explore various architectures to address competition-specific needs. The recent years (2022–2024) continue to show strong representation for transformers.

5 Discussion & Conclusions

This section reflects on the study's findings, providing insights from the Kaggle data analysis, outlines limitations encountered and proposes directions for future research.

The objective of this study is to explore whether the ML and DL models employed by practitioners on Kaggle align with those proposed in academic research for similar tasks. Given the challenges encountered in obtaining comparable data from academic publications, our comprehensive analysis focused on extracting and analyzing model usage from Kaggle competitions.

5.1 Insights from Kaggle Data Analysis

5.1.1 Competition Specific Trends

Our evaluation of the ChatGPT-4o API demonstrated its effectiveness in extracting model names from unstructured text, achieving an F1 score of 0.85 and a Jaccard Index of 0.78. These metrics indicate a high level of accuracy, affirming the utility of the ChatGPT-4o API for large-scale data processing tasks where manual extraction would be impractical. The analysis of model usage across different competition tags revealed distinctive patterns in practitioners' preferences, aligning with the strengths and suitability of various models for specific tasks:

- **Computer Vision Tasks**: CNNs were overwhelmingly favored, particularly architectures like EfficientNet (Tan and Le, 2019), ResNet (He et al., 2016), and U-Net (Ronneberger et al., 2015). Time series analysis showed that EfficientNet gained rapid adoption following its introduction, peaking around 2021 before declining as newer models emerged. ResNet demonstrated early popularity but saw a gradual decline after 2021, while U-Net maintained consistent usage due to its effectiveness in image segmentation tasks.

- **NLP Tasks**: Transformer-based architectures (Vaswani et al., 2017), such as BERT (Devlin et al., 2019) and GPT models, were predominant. The time series data indicated that transformers became the preferred choice after 2019, replacing earlier models like LSTMs (Hochreiter and Schmidhuber, 1997). This shift reflects the superior performance of transformers in capturing contextual relationships in language data.

- **Tabular Data Tasks**: Boosting models like LightGBM (Ke et al., 2017) and XGBoost (Chen and Guestrin, 2016) emerged as the favored approaches. Both models showed consistent usage over time, with LightGBM gaining increased popularity due to its efficiency and performance advantages over earlier models like XGBoost . Neural networks were used less frequently in tabular data tasks, indicating a preference for models optimized for structured data.

The hierarchical clustering and Sankey diagram further illustrated the relationships between competition tags and model usage. Models were selected based on their alignment with task requirements, demonstrating that practitioners prioritize models that offer practical advantages, such as computational efficiency, ease of implementation, and strong performance on specific data types.

5.1.2 Temporal Trends and Model Adoption

The time series analyses provided insights into how model usage has evolved over time:

- **Adoption Lag**: There appears to be a lag between the introduction of new models in academic research and their widespread adoption by practitioners. Models like EfficientNet and transformer architectures gained prominence in Kaggle competitions a year or more after their initial publication in academic venues, possibly also due to long competition durations.
- **Model Longevity**: Established models like ResNet and XGBoost maintained usage over several years, highlighting their reliability and the value practitioners place on well-understood and supported models.
- **Emergence of New Architectures**: The rise of transformers in NLP and computer vision tasks signifies a shift in the landscape of preferred models, reflecting ongoing advancements in the field and practitioners' willingness to adopt models that offer significant performance improvements.

5.2 Implications of the Findings

Our findings suggest that practitioners on Kaggle tend to adopt models that are well-established, widely supported, and have demonstrated effectiveness in practical applications. While cutting-edge models proposed in academic research eventually make their way into practice, there is a discernible delay in their adoption. This lag can be attributed to factors such as the need for robust implementations, community validation, and the time required for practitioners to gain familiarity with new architectures.
The study highlights the importance of accessible dissemination of research findings. When academic models are accompanied by open-source implementations and comprehensive documentation, they are more likely to be adopted by practitioners. This underscores the role of the research community in facilitating the translation of theoretical advancements into practical tools.

5.3 Conclusions

Our analysis provides valuable insights into the models preferred by practitioners in real-world ML tasks on Kaggle. The successful extraction and analysis of model usage patterns demonstrate the effectiveness of using advanced language models like the ChatGPT-4o API for processing unstructured data.
While it was impossible to perform a direct comparison with academic research models due to data limitations, the findings suggest that practitioners prioritize models that offer a balance of performance, efficiency, and ease of use. The trends observed in model adoption over time indicate that while novel models from academic research do influence practice, their adoption is selective and influenced by practical considerations.
The study emphasizes the need for bridging the gap between theoretical advancements and practical implementations. By improving the accessibility of new models and fostering collaboration between researchers and practitioners, the ML community can enhance the impact of academic research on real-world applications.

6 Limitations & Future Work

This section discusses the study's limitations and future work, discussing faced obstacles, and potential future work.

Despite the contributions of this study, several limitations must be acknowledged. Firstly, our inability to directly analyze academic models posed a significant constraint. Due to obstacles in processing academic abstracts and the lack of explicit model naming conventions, it was impossible to construct a dataset of models proposed in academic research for direct comparison. This limitation restricts our ability to conclusively determine the alignment between academic proposals and practitioner adoption.
Secondly, the analysis focused on the winning solutions of Kaggle competitions, which may not capture the full diversity of models used by all participants. This focus potentially overlooks less common or emerging models that did not lead to winning entries. Considering a broader range of submissions could provide a more comprehensive understanding of practitioner behavior.
Thirdly, there was variability in the quality and detail of solution descriptions. Some descriptions lacked comprehensive information about the models used, which could affect the completeness and accuracy of the extracted data. Additionally, the temporal scope of our dataset reflects competitions up to early 2024 and may not capture the most recent trends in model usage, especially given the rapid evolution of the ML and DL fields (Sun et al., 2017).
Another limitation is the dependence on the ChatGPT-4o API for data extraction and cleaning. While the API demonstrated high accuracy, potential misinterpretations or extraction errors could exist, and reliance on a single tool introduces possible biases related to its training data

and limitations (Bender et al., 2021). Furthermore, the process of generalizing model names and categorizing competition tasks involved subjective decisions. Despite efforts to standardize and align authoritative sources, inherent biases could influence the categorization.

Building on our findings and acknowledging the limitations, future research can explore several avenues. Enhancing the extraction of information from academic publications is a critical area for future work. Developing or utilizing more sophisticated NLP tools tailored for academic texts could improve the extraction of model information from research papers. Access to full-text papers and advancements in NLP may facilitate better analysis (Cohan et al., 2020).

Including a broader range of solutions in the dataset is another potential direction. Expanding the dataset to encompass a wider array of competition submissions, not just the winning solutions, could provide a more comprehensive view of model usage among practitioners. This approach could reveal trends in experimentation and the adoption of emerging models.

Conducting a temporal analysis of model adoption by performing a longitudinal study would offer insights into how model preferences evolve over time. Such an analysis could illuminate the adoption rates of new models and the influence of academic research on practical applications. Additionally, surveying practitioners and researchers could provide qualitative insights into the factors influencing model selection, including perceptions of new academic models, barriers to adoption, and practical considerations (Zhang et al., 2022).

Future research could also extend the study to include other platforms where practitioners share their work, such as GitHub repositories, AI challenges, or industry case studies. This would broaden the understanding of model adoption in practice (Munaiah et al., 2017). Investigating mechanisms to bridge the gap between academia and industry is another crucial area. Enhancing the dissemination and accessibility of academic research, promoting open-source implementations, standardizing reporting practices, or fostering collaborations between researchers and practitioners could improve alignment (Pineau et al., 2020).

Finally, evaluating model performance in practice by analyzing models in real-world settings versus controlled academic environments could shed light on the practical viability of cutting-edge models and inform future research directions.

This study highlights the complex dynamics between academic research and practical application in ML and DL. By analyzing practitioners' model choices on Kaggle and observing temporal trends in model adoption, we have gained insights into how models transition from theory to practice. Addressing the limitations and pursuing the suggested future work could significantly enhance the understanding of this relationship. By fostering closer alignment between academia and industry, the ML community can accelerate innovation, improve model adoption, and ultimately advance the field in a way that benefits both researchers and practitioners.

7 Appendix

This section contains addition information and visualizations for the above sections.

7.1 Prompts

7.1.1 Model Extraction

Extraction Prompt: Extract only the names of ML and DL models mentioned in the following solution description, keeping only the names that are used as part of the final solution. List all names in their original format. Do not change the format (if the model name is abbreviated keep it that way, letter case, etc.). Return only a Python list with the names. Do not include any additional text.
Filtering Prompt: From the extracted list of names, exclude:

1. Terms related to pooling, activation functions, normalization, and operations.
2. Dataset names and hyperlinks.
3. Duplicates, ensuring each model/method name appears only once.
4. Packages/GitHub repos, mathematical operations.
5. Additional model information.

Return only a Python list with the filtered names. Do not include any additional text.

7.1.2 Standardize & Generalize Model Names

Standardize the following list of machine learning and deep learning model names according to the following rules:

1. Standardize variations of the same model to a single form. Make sure short forms of model names are changed into long form (e.g., 'xgb' and 'XGBoost' should both be 'xgboost', 'CatBoost' and 'CB' should be 'catboost', LGBM should be lightgbm, etc.).
2. For deep learning architectures, keep only the base architecture name (e.g., ResNet18 and ResNet50 should both be resnet).
3. For general architecture types, simplify to the basic form (e.g., '2D CNN' and '3D CNN' should be 'cnn', '2D Unet' should be 'unet', rnn model should be rnn, etc.).
4. Remove duplicates, ensuring each model name appears only once.

Return only a final unique Python list of the model names.

7.1.3 Additional Filtering Prompt

Filter the following list of machine learning (ML) and deep learning (DL) model and method names according to the following rules:

1. Remove anything none ML or DL related (e.g. Python package and library names, as well as operations such as pairwise ranking, or smoothing).
2. Remove things like paths, usernames for websites, etc.
3. Remove anything very general such as the 'deep learning', or machine learning.
4. Do not remove things such as names of language models, DL models and architecture types.

Keep the remaining elements of the list with the exactlty as they were originally. Return only a final Python list containing the filtered elements.

7.1.4 Clean Competition Tag Names

Clean the following list of machine learning (ML) and deep learning (DL) task tags according to the following rules:

1. Remove anything none ML or DL task related.
2. Remove names of general fields such as education, sports, mathematics, robotics, etc. as they are not task specific (task specific examples are e.g image classification).
3. Standardize the names of the tags into a simple form if neccesary.
4. If neccessary, merge two tags into one (e.g 'image', 'classification' can be 'image classification').
5. General ML fields such as computer vision or classical machine learning are fine.

Return only a final Python list without duplicates containing the cleaned tags.

7.1.5 Hierarchical Grouping

Given the following list of machine learning (ML) and deep learning (DL) models/architectures, categorize each model into its broader category or type. For example, if the model is 'efficient-net,' categorize it under 'cnn'. If the model is 'xgboost,' or 'catboost', 'lightgbm' categorize it under 'boosting', etc. For cases where a given model is a classical ML model and no specific category exists for it, you can label it as 'classical ml'. Similarly, group all models based on their broader ML/DL category. Return only a final Python list without duplicates containing the broader category/types of the models within the list.

7.1.6 Abstract Model Name Extraction

I am going to provide an abstract from a scientific paper. I want you to extract any Machine Learning/Deep Learning model names that were used in the abstract, provided that they've been mentioned in the abstract. Return a Python of model names, if nothing could be found return an empty list.

7.2 Visualizations

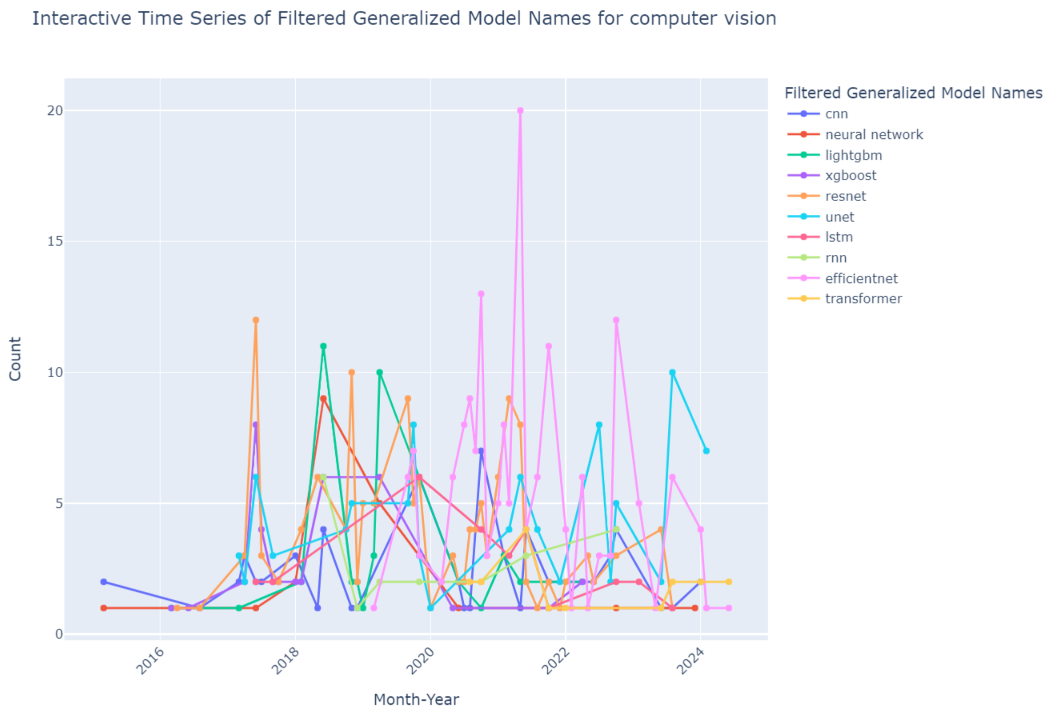

Figure 17: Time series plot of models for computer vision

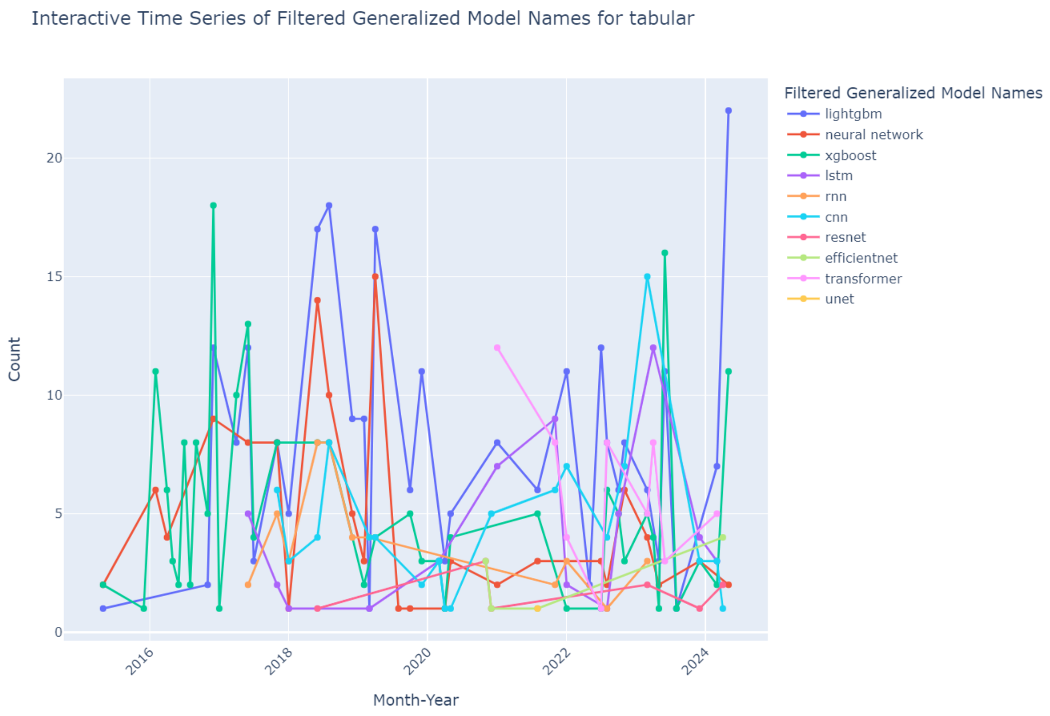

Figure 18: Time series plot of models for tabular data

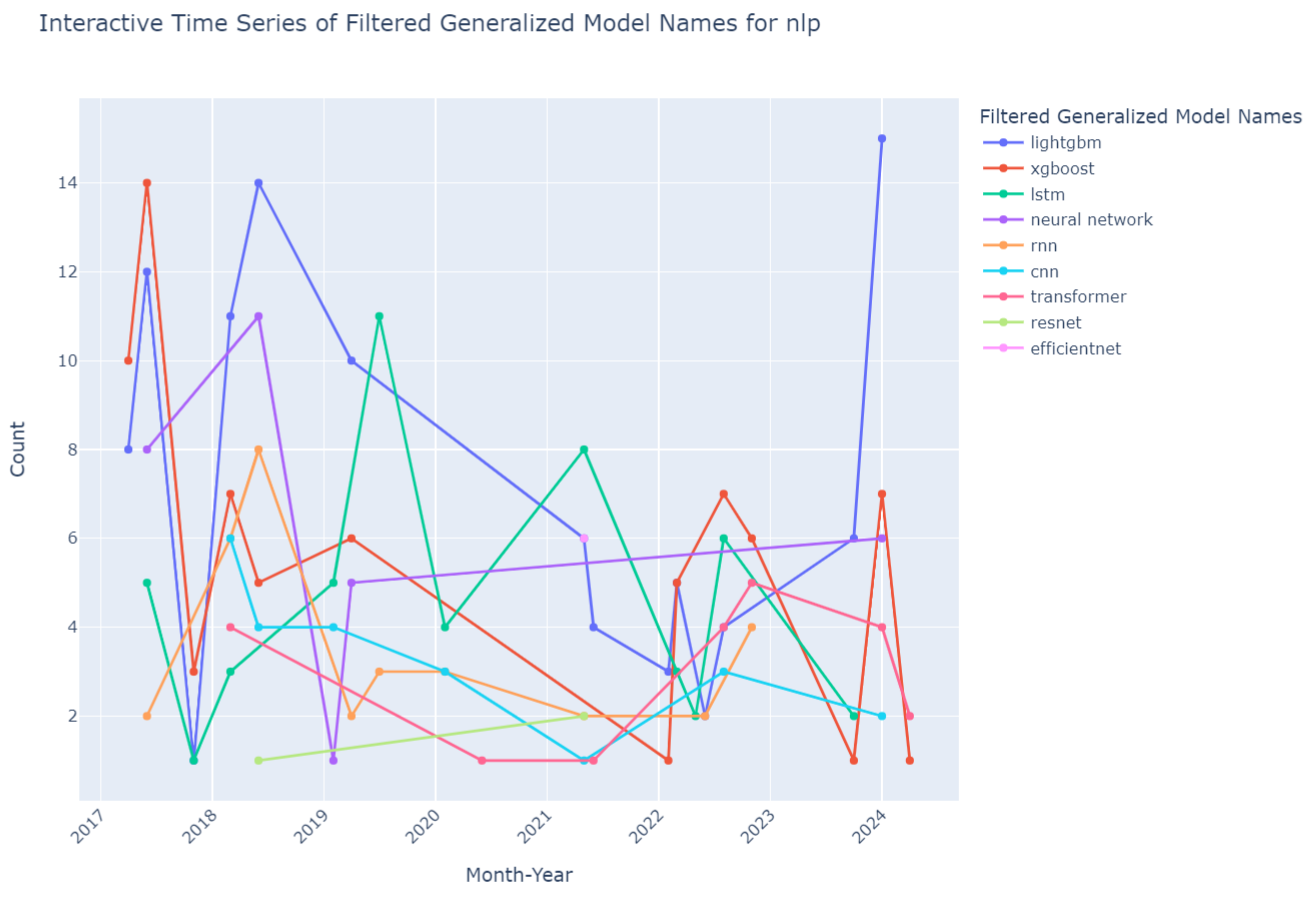

Figure 19: Time series plot of models for NLP

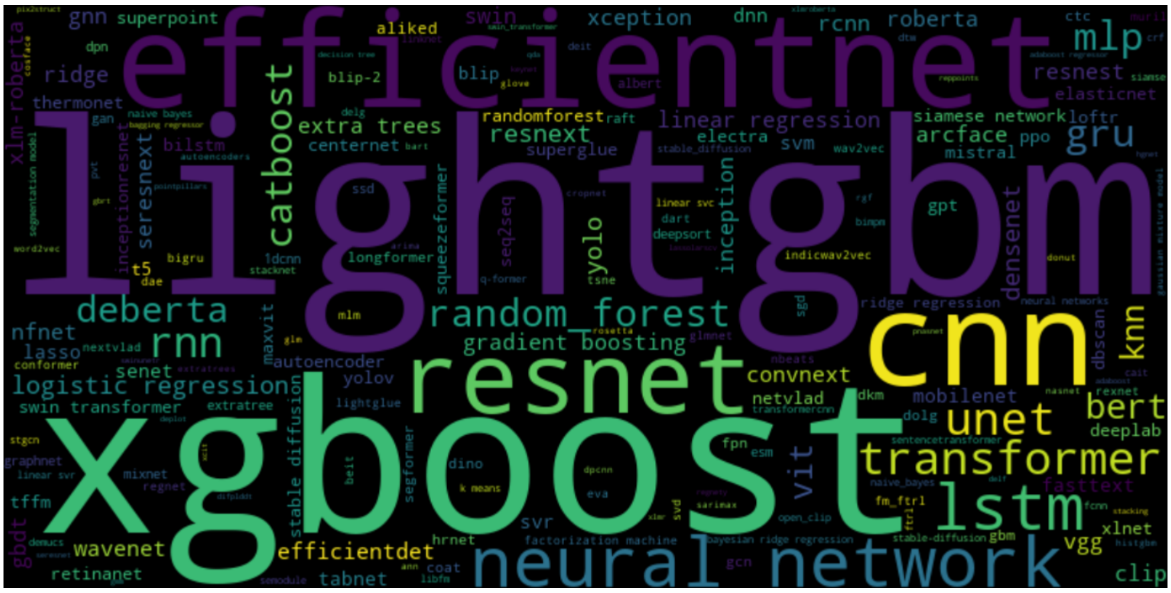

Figure 20: Wordcloud for model names

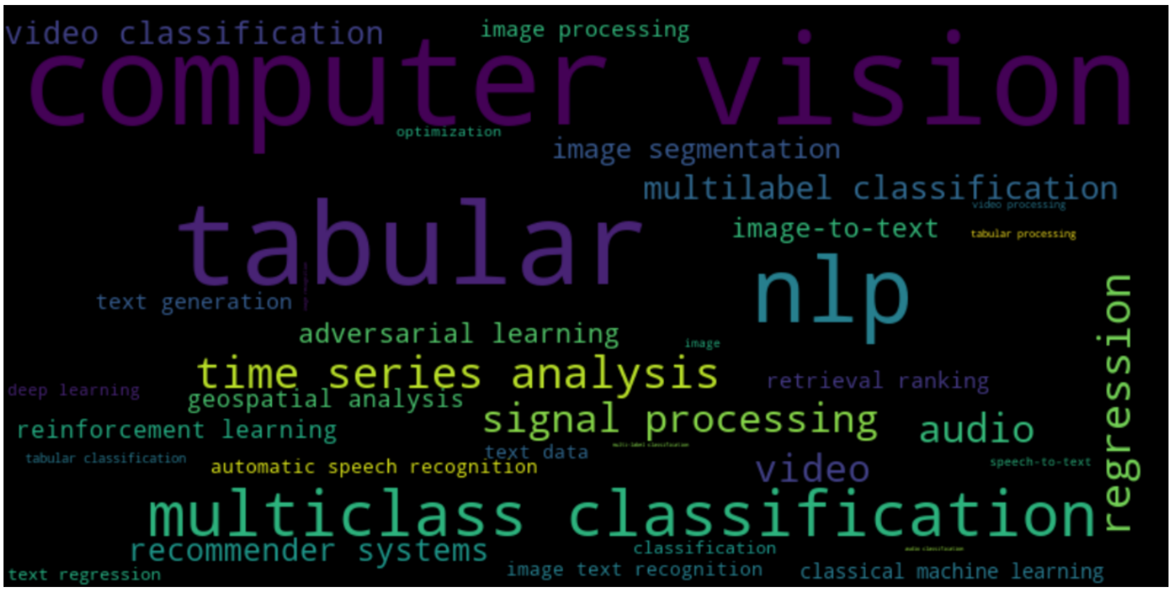

Figure 21: Wordcloud for competition tags

References

Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Bischl, B., Casalicchio, G., Feurer, M., Hutter, F., Lang, M., Mantovani, R. G., van Rijn, J. N., and Vanschoren, J. (2019). Openml benchmarking suites. *arXiv preprint arXiv:1708.03731v2 [stat.ML].*

Brown, T., Mann, B., Ryder, N., and et al. (2020). Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 785–794. ACM.

Cohan, A., Feldman, S., Beltagy, I., Downey, D., and Weld, D. (2020). SPECTER: Document-level representation learning using citation-informed transformers. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2270–2282, Online. Association for Computational Linguistics.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). 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*, pages 4171–4186.

Gooding, S., Kochmar, E., Yimam, S. M., and Biemann, C. (2021). Word complexity is in the eye of the beholder. In Toutanova, K., Rumshisky, A., Zettlemoyer, L., Hakkani-Tur, D., Beltagy, I., Bethard, S., Cotterell, R., Chakraborty, T., and Zhou, Y., editors, *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4439–4449, Online. Association for Computational Linguistics.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ke, G., Meng, Q., Finley, T., and et al. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, pages 3146–3154.

Munaiah, N., Kroh, S., Cabrey, C., and Nagappan, M. (2017). Curating github for engineered software projects. *Empirical Softw. Engg.*, 22(6):3219–3253.

OpenAI (2024). ChatGPT (november 2024 version). `https://openai.com`.

Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Larochelle, H. (2020). Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program).

Qin, Z., Han, C., Wang, Q., Nie, X., Yin, Y., and Xiankai, L. (2023). Unified 3d segmenter as prototypical classifiers. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S., editors, *Advances in Neural Information Processing Systems*, volume 36, pages 46419–46432. Curran Associates, Inc.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

Ruder, S., Vulić, I., and Søgaard, A. (2022). Square one bias in NLP: Towards a multi-dimensional exploration of the research manifold. In Muresan, S., Nakov, P., and Villavicencio, A., editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2340–2354, Dublin, Ireland. Association for Computational Linguistics.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J., and Dennison, D. (2015). Hidden technical debt in machine learning systems. In *Neural Information Processing Systems*, volume 28, pages 2503–2511.

Selenium Project (2024). *Selenium WebDriver*. `https://www.selenium.dev`.

Sun, C., Shrivastava, A., Singh, S., and Gupta, A. (2017). Revisiting unreasonable effectiveness of data in deep learning era.

Tan, M. and Le, Q. V. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 6105–6114.

Tor Project (2024). *Tor Project: Anonymity Online*. `https://www.torproject.org`.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. (2023). Llama: Open and efficient foundation language models. *Meta AI*.

Vaswani, A., Shazeer, N., Parmar, N., and et al. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Zhang, J. M., Harman, M., Ma, L., and Liu, Y. (2022). Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 48(1):1–36.

Zhang, Z., Xu, M., Jamasb, A., Chenthamarakshan, V., Lozano, A., Das, P., and Tang, J. (2023). Protein representation learning by geometric structure pretraining.

Statement of Originality

I, Davit Martirosyan, declare that this thesis, entitled *"Bridging the Gap Between Research and Practice: An Analysis of Machine Learning and Deep Learning Model Adoption on Kaggle"*, and the work presented within it are my own. I confirm the following:

- This work has not been previously submitted, either in part or in full, for any degree or diploma at any other university or institution.
- All sources of information, assistance, and inspiration have been appropriately acknowledged, and proper citations are provided where relevant.
- Any collaborative contributions have been clearly acknowledged and documented within the text.

I understand that failure to comply with these principles may lead to disciplinary actions by my institution.

Date: 15.11.2024
Location: Munich, November, 2024

Signature: [illegible]