Genre Prediction Using RNNs and LLM-Enhanced Video Game Review Data

Gabriel Young, Susan Gauch
Department of Electrical Engineering & Computer Science
University of Arkansas
Fayetteville, AR 72701, USA
Emails: gpy001@uark.edu, sgauch@uark.edu

***Abstract* — Large Language Models (LLMs) are powerful tools for engaging with textual data, carrying many advantages over classical Natural Language Processing (NLP) and Machine Learning (ML) approaches. However, a classical ML model can still be faster, more efficient to run, and accessible than an LLM. We seek to capture the benefits of LLM-based text comprehension and preserve them within a classical ML model through a hybrid approach. The LLM operates on text to identify relevant information and associations within our problem space, then the ML model trains on the LLM output. The model may learn from the LLM and provide a more efficient alternative to querying the LLM directly for future data. Using review data from the video game marketplace Steam, we conduct a series of experiments toward this end. We prompt the LLM to surface various information from the raw data and train Recurrent Neural Networks (RNNs) to predict a single genre of the games, "Role-Playing Game" ("RPG"). We then evaluate the performance of the trained RNN models on the raw data, checking for generalizability and performance loss/improvement. Results are promising. At baseline, using raw review data, a balanced (50% RPG, 50% non-RPG) dataset, and no LLM assistance, a shallow RNN can predict the genre under test with an average accuracy of 64.1%. The maximum accuracy of the LLM on this same dataset is 84.1%. Our other models under test lie between these two bounds and demonstrate merits from engaging the LLM during their training.**

Keywords - Classic ML; AI; LLM; NLP; Video Games; Reviews

I. INTRODUCTION

Large Language Models (LLMs) are able to solve or circumvent many of the greatest challenges in language comprehension. They are particularly useful for surfacing subtle, subtextual information from spoken and written language data. They are also user-friendly, requiring low effort on the part of a developer, data scientist, or layperson to leverage. They are now dominating the field of Natural Language Processing (NLP) and predictive modeling from language in popular use, enterprise applications, and, increasingly, research [7]. LLMs are not the most appropriate solution for every use case. An LLM is, by its nature, a generalist. It accepts many formats of input and answers a wide range of requests. This is by virtue of its having trained on large amounts of cross-functional data and a complex, computationally expensive network architecture [5]. For large-scale applications with hundreds of thousands, or millions, of users and a well-defined, domain-specific use case it can be an inefficient, or even infeasible, option [13]. Accessibility can also be an issue, since 3rd-party LLMs charge for services.

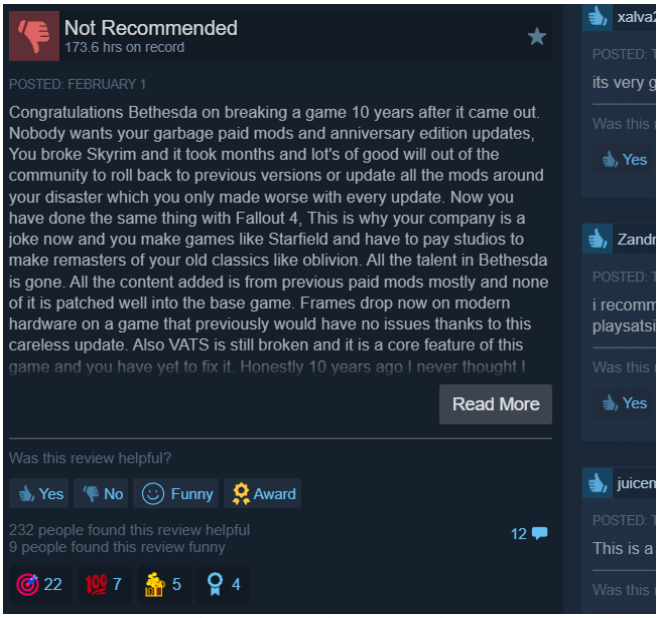

Figure 1. Example Steam Review.

One of the traditional approaches to predictive modeling is the bidirectional Recurrent Neural Network (RNN). An RNN must be more tailored to its problem space than an LLM and is generally less complex and less powerful, underperforming the LLM without a great deal of training and fine-tuning [7]. However, this means it is simpler to build and own, and much less resource intensive to run repeatedly [13].

The question to answer is this: is it possible to build a simple RNN, train it on LLM-modified data, and retain some of the benefits of the LLM's more capable text processing? We set out to explore this approach, using an LLM to generate summary data and surface key information from text, training RNNs on the LLMs output, and then evaluating the RNNs' performance.

For this, we need a difficult problem in written language comprehension on which both an LLM and RNN might perform and be compared. We chose to predict genre classifications for video games on the Steam service, based on user reviews (Figure 1). Steam is a popular gaming marketplace, the largest for online digital downloads. It makes sales, usage, site page data and review data for its games publicly available through a research API [2]. Among other information, each game listing on Steam includes a set of tags. The tags may be placed on a game's entry by developers and users and the top 20 tags are surfaced for viewing on the game's page [3]. Genres for each game are a subset of these tags belonging to Steam's list of recognized genres.

Because the tags, and by extension genres, are user-generated, there should be consistency between users' choices of genres for the game and references to a game's genres in the reviews. It should be possible, with sufficient informative review data, to predict genre for these games with some degree of accuracy.

The level of difficulty in this problem varies depending on the genre being classified. Some genres are more prevalent than others among games overall or are mentioned in reviews more often and more directly. Classifying genre is difficult in the case of the "Indie" genre. The classification pertains to the size of the development studio, a factor separate from the direct experience gained by playing the game. Consequently, it is mentioned infrequently by reviewers. By contrast, the "Sexual Content" genre is trivial to identify from user reviews, with frequent direct mentions.

In the best functional case, the interpretive advantages of the LLM on text data would be gained and then passed to the model, highlighting informative features of the text and cutting away uninformative or misleading features. In the worst case, a model trained on the LLM data will no longer generalize properly when exposed to raw text. A model that can only operate on the LLM output realizes no improvement over using the LLM by itself. However, if there is only acceptable (user and use-case determined) performance loss relative to using the LLM alone, the result is a lighter-weight, more efficient solution to the problem.

We are using simple RNNs for this experiment. It is very possible that more tuned models will achieve higher initial performance on the dataset and less performance gain from incorporating the LLM data. We are also predicting a single genre for games, and using data from a single source, so the scope of our conclusions will, necessarily, be limited.

The remainder of this article is organized as follows:

- background for RNNs, LLMs, and genre prediction with a discussion of related works
- a summary and analysis of our dataset

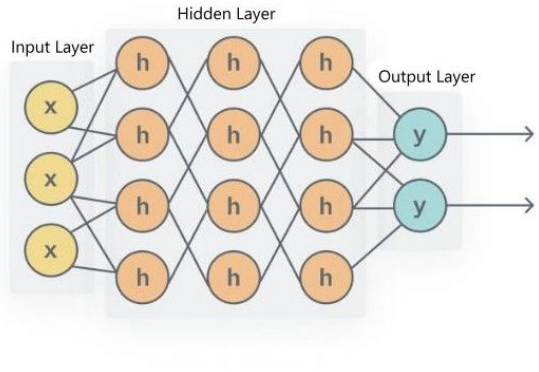

Figure 2. High-Level RNN Diagram.

- the design of our experiments and our different experiment cases
- an explanation of our experiment results
- an interpretation, the conclusions of our experiment results, and future works

II. RELATED WORK

A. Classic RNN

RNNs are a form of deep learning algorithm. "Recurrent" refers to the feedback pattern of information as it passes through the layers of the network, with the output of previous inputs being maintained as memory or "hidden state" at each node, then passed in as input to the node again alongside the next round of regular input. They have proven effective at discerning subtle connections between input and output in problems where order and context of the input matter. They process data sequentially across their layers (Figure 2) retaining information from previous steps by passing layer output back in as input during the next iteration [4].

Long Short-Term Memory (LSTM) is a specialized type of RNN capable of maintaining useful context in its hidden state for long periods, but also discarding used context from further calculations [10]. This capability is due to the structure of its nodes, which are modified with three "gates" to the node's hidden state: an input gate, output gate, and forget gate. The node can receive information, persist it where applicable, but also reject it after use. This innovation was designed to address the vanishing gradient problem of RNNs, wherein earlier layers of the network see much more intensive calculations during backpropagation than later layers. The ability to reject and forget information reduces the calculation complexity [10]. We use LSTM in these experiments because it has been shown to carry advantages in text processing, where context is often critical to correct interpretation and needs to be persisted until it is applied [10].

B. LLMs

An LLM is an extremely large deep learning model, trained on a vast amount of text data and a diverse array of subject matter. The data an LLM is trained on varies, sometimes according to its intended function but often based on what is accessible. Many are trained on sections of the Internet, with information that is free and publicly available. Their transformer architecture (Figure 3) allows for the processing of input data in parallel rather than sequentially, which is thought to enable their enhanced interpretive abilities [11].

LLMs have proven extremely capable at deriving meaning from language. They have been shown to navigate very complex language, with subtextual and contextual references, sarcasm, idiomatic speech, and linguistic artifacts [11]. Zhu et al. [7] examine several high-profile LLMs currently on the market and assess their capabilities for text comprehension and use in information retrieval (classically the domain of search engines). One of the areas of information retrieval cited as showing improvement under LLMs is query rewriting: refining, expanding, or otherwise modifying users' search queries to better surface relevant information [7]. This method of using LLMs to improve user text clarity is invoked similarly during these experiments.

C. Genre Prediction

In the recent work of Raj et al. [5], LLMs were leveraged for the multilabel prediction (overlapping, non-exclusive categories) of movie genres and compared against classical Machine Learning (ML) approaches [5]. Their results demonstrated that ChatGPT 3.5 consistently outperformed traditional classifiers (logistic regression, K-Nearest Neighbors (KNNs), Support Vector Machines (SVMs)), working off subtitle data for movies to predict genre [5].

Ströbel et al. [6] investigate the problem of genre prediction for literature from text, using a Gated Recurrent Unit (GRU) RNN. GRU is a variant of RNN that tends to see performance increases in smaller datasets [6]. They can achieve high rates of prediction accuracy (0.90) for the five disparate genres under test using a relatively "simple" RNN model. This work helps highlight the efficacy of RNNs in working with text for genre prediction and also supports the choice of simpler models with minimal text pre-processing as appropriate benchmarks in our own experiments [6].

D. Steam Data

Olmedilla et al. [8] undertake an exploration of Steam reviews and compare a few predictive modeling approaches. They train regression models, and a Bidirectional Encoder Representations from Transformers (BERT) model on the review data in order to predict a helpfulness score of the review. Helpfulness is a weighted vote score representing whether users who read the review found it informative. It is a user-defined and nebulous metric similar to the genres we are trying to classify in our experiments. The authors note only modest results from their model, highlighting the difficulties posed in learning from review text using a classical approach [8].

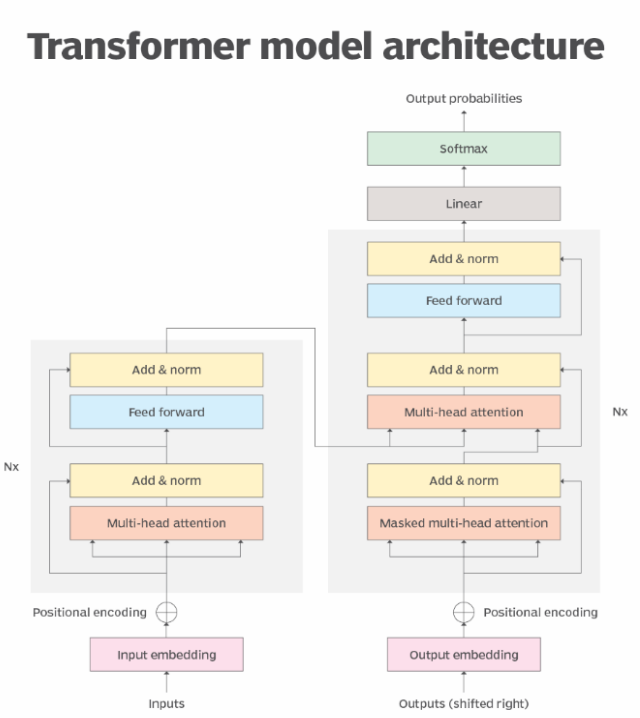

Figure 3. High-Level LLM Transformer Architecture.

E. LLM and RNN Hybrid Approaches

Chen et al. [9] engineered a hybrid LLM-Convolutional Neural Net (CNN) approach to financial modeling. Similarly to the approach we've taken for our experiments, they use LLMs to "first processes textual inputs to extract structured features" and subsequently "feed these features into a predictive model alongside numerical data." They realize consistent performance gains for the integrated solutions over either solution individually. They also note the criteria that must be considered during the design and approach to hybridized systems. "These architectures must balance competing objectives including predictive accuracy, computational efficiency, interpretability [...]" [9]. Our work extends on this consideration, seeking gains of efficiency with an acceptable cost to accuracy between the RNN-LLM hybrids and the LLM by itself.

III. DATASET

We began with the Steam Games Dataset, compiled in 2022 for all games on Steam at the time [1]. Each entry in the dataset represents a game, and each column contains information scraped from the game's page. The dataset contains information on both the genres and tags shown on the game's page at the time of compilation. Genres and tags for a game might have changed on the site since the data was gathered.

We created a dataset of 5000 Role-Playing Games (RPGs) and 5000 non-RPG games by taking random samples of gameIDs from the Steam Games dataset with "RPG" present or absent in their genre lists, respectively. We then queried the Steam API for the review data of these games, up

to 500 of their most recent reviews. Games for which we were unable to retrieve at least ten reviews were discarded from both samples. This resulted in a 7766-entry set.

The raw reviews were cleaned for processing by our RNN as well as the LLM. Common stop words ("a", "an", "for", etc.) were removed, and a minimum and maximum word length were enforced. We decided to consolidate our dataset to only those reviews with information relevant to the task, i.e. reviews with strong indicators about the presence or absence of the game's RPG classification. To assess this, we ran the full set of entries through the LLM with the following prompt:

- *"RPG stands for "Role-Playing Game". The genre often features character customization and rich narrative elements. Read the following reviews for this game and determine whether or not it is an RPG. If there is not enough information in the reviews to determine this, return an output of '[Insufficient Information]'. Reviews: {reviews}"*

Allowing the LLM to curate the dataset to only those reviews it finds informative might seem to introduce bias, advantaging the LLM. However, it has no knowledge of the true classifications for each game when it reads the reviews. The reduced dataset only includes entries for which the LLM had high confidence in its predictions. It can still, very confidently, misclassify.

After filtering, the dataset was drawn down to 2340 entries. The distribution of genre among those entries was 1122 RPG and 1218 non-RPG. We undersampled the majority class to balance the dataset, resulting in 2244 total entries.

This dataset was split 80-20 into testing and validation sets, both of which maintained the 50% class distribution.

IV. EXPERIMENTS

A. RNN Setup

The RNN was built using Python's tensorflow keras libraries [14]. For RNN classification, we need a dictionary and encoder to convert the words of our text into tokens. We use keras's TextVectorization layer object to accomplish this. Using its adapt function, we have it create a dictionary of 10,000 distinct words from our review data. When reviews or the LLM summaries are passed into this layer, they will be converted to numeric vectors based on the words present, their incidence, and their order [12].

We define our RNN as a sequential, single output, binary prediction model (Figure 4). The first layer is the encoder, and the second is a single bidirectional connected layer of 32 nodes. Bidirectional layers propagate the input forward and backwards through the layer, preserving sequential information about the tokens [4]. There is a subsequent activation layer, a dropout layer with a 50% drop rate to forestall overfitting, and finally a sigmoid activation layer. The model uses binary cross-entropy for its loss function and an Adam optimizer. Overall, the design is a minimally complex model that can still perform the desired function.

```
tf.keras.Sequential([
  tf.keras.layers.Embedding(
    input_dim=len(encoder.get_vocabulary()),
    output_dim=32),
  tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(32)),
  tf.keras.layers.Dense(32, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(1, activation="sigmoid")
])
```

Figure 4. Model Definition.

B. LLM Setup

The LLM used in these experiments is ChatGPT v4.1-mini, accessed through its batched input API. This version was chosen for this experiment for reasons of availability, reliability, popularity, and cost. The data is split into 500 entry batches, sent to the LLM service, and for each experiment prompt the entries are collected and processed back into the dataset as new columns.

C. Experiment Prompts

The RNN is trained on the LLM responses (except in the control experiments) for 10 epochs. The final model is evaluated over a verification set of the response data five times to produce an average accuracy. Then, it is evaluated in the same manner over the original raw data 10 times to test generalizability. Here, we also collect average precision, recall, and F1 scores.

The experiments and their prompts to the LLM are as follows:

- Experiment 1: RNN Control
  - No LLM used.
- Experiment 2: General Summary
  - "Write a summary of the following reviews for a video game. Reviews: {text}"
- Experiment 3: Surface Genre Information
  - "Write a summary of the following reviews for a video game. If possible, surface the name of the game and a few probable genres in which it might be classified. Reviews {text}"
- Experiment 4: Surface RPG Information
  - "Write a summary of the following reviews for a video game. If possible, surface the name of the game and whether or not it is of the RPG genre. RPG stands for 'Role-Playing Game'. The genre often features character customization and rich narrative elements. Reviews: {text}"
- Experiment 5: LLM Control
  - "RPG stands for 'Role-Playing game'. The genre often features character customization and rich narrative elements. Read the following reviews for this game and return an output of '[RPG]' if it is of the RPG genre, or '[Not An RPG]' if it is not. Reviews: {text}"

V. RESULTS

Table 1 shows the final binary validation accuracy of the models during training as well as the accuracy, precision, recall, and F1 scores when they are reintroduced to the raw text. 'Accuracy' here serves as our primary basis of comparison on model performance, with the precision and recall metrics serving supplemental information about the false negative and false positive rates, respectively. The F1 score, computed as the harmonic mean of precision and recall, conveys this same information as a single metric. We also show the training and validation curves of the models (Figure 5) for Experiment 4, the highest performing, compared against the training curve and model of Experiment 1, our control.

The more specific about RPG genre classification we were with our prompting, the more the models' performance approached the LLM's. There was better performance by the LLM-informed models on the raw data with the exception of Experiment 3. While that experiment saw the highest gains in false negative identification, it suffered on true positive identification, evidenced in the lower recall score and F1 scores.

VI. CONCLUSION AND FUTURE WORK

We compared several models predicting the RPG genre of games based solely on text: an RNN trained on generic summary data, an RNN trained on surfaced genre data, and an RNN trained on surfaced RPG genre data. The LLM's accuracy on the data was 84.1%, the upper bound for accuracy in our experiments. The models were capable of performing within 1-2% of accuracy of the LLM itself while validating with LLM summary text. While losses in accuracy were observed for all models when exposed to the raw review data, the models still realized significant gains of between 5% and 10% accuracy over the baseline RNN, our lower bound at 64.1%. This suggests that benefits of the LLM processing were preserved in the training and utilized in model prediction.

We conclude that there are significant advantages in using an LLM to process and predict from text over the simpler RNN models. We can also tentatively conclude that LLMs can successfully inform RNNs during training, with gains of accuracy on real data without fine-tuning or increasing the complexity of the RNN.

We would like to apply this same series of tests to the prediction of other, single genres in our dataset such as the "Adventure", "Puzzle", and "First Person Shooter (FPS)" genres of games and compare the results to establish generalizability across genres. These genres are non-exclusive, and a single game may belong to any or all of them. Expanding the modeling problem for multi-label classification makes the experiment more applicable to real-world use cases. If the results of these extensions support continued investigation, testing prominent available LLMs and comparing their utility in surfacing text information during model training will be another series of experiments.

We are also working to increase the size and representation of our dataset, which will enable further experimentation and grant advantages in reliability for our results.

TABLE I. EXPERIMENT RESULTS

Experiment	Final Model Avg Value Acc.	Final Model Raw Text Avg Value Acc.	Final Model Raw Text Avg Prec.	Final Model Raw Text Avg Recall	Final Model Raw Text Avg F1
1. No LLM	0.641	0.646	0.599	0.690	0.641
2. General Summary	0.794	0.719	0.701	0.763	0.729
3. Surface Genres	0.831	0.682	0.746	0.536	0.623
4. Surface RPG Genre	0.822	0.739	0.724	0.757	0.740
5. LLM Only	0.841	0.841	NA	NA	NA

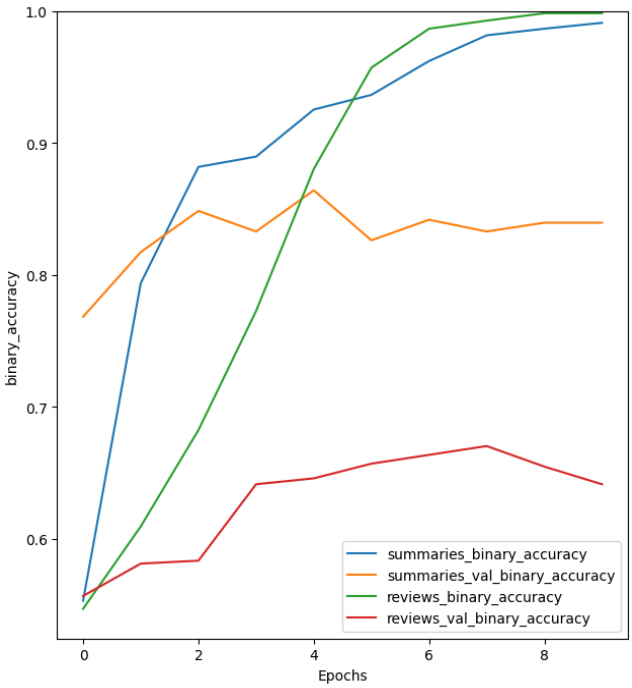

Figure 5. Experiment 4 Accuracy Curves vs. Experiment 1.

REFERENCES

[1] M. Bustos, "Steam Games Dataset", Kaggle, 2022, retrieved April, 2025 https://doi.org/10.34740/KAGGLE/DS/2109585

[2] Steamworks API Reference (Steamworks Documentation). 2026, retrieved May 2025 https://partner.steamgames.com/doc/api

[3] Steam Database Team, "SteamDB", retrieved March, 2026, https://steamdb.info

[4] C. Stryker, “Recurrent Neural Network (RNN)”. October 24, 2021, retrieved January, 2026, https://www.ibm.com/think/topics/recurrent-neural-networks

[5] S. Raj, S. Saha, B. Singh, and N. Pedanekar. “Demystifying chatgpt: How it masters genre recognition.” Natural Language Processing Journal, vol 14, 100198. 2026. https://doi.org/10.1016/j.nlp.2026.100198

[6] M. Ströbel, E. Kerz, D. Wiechmann, and Y. Qiao. “Text Genre Classification Based on Linguistic Complexity Contours Using A Recurrent Neural Network.” MRC@ IJCAI. July, 2018, pp. 56-63.

[7] Y. Zhu et al, “Large language models for information retrieval, a survey.” 2023, retrieved February, 2026, https://arxiv.org/abs/2308.07107

[8] M. Olmedilla, L. Espinosa-Leal, J. C. Romero-Moreno, and Z. Li, “Unveiling the Value of User Reviews On Steam: A Predictive Modeling Of User Engagement Approach Using Machine Learning”. In A. Abraham, G. C. Peng, P. Isaias, & P. Isaias (Eds.). Proceedings of the International Conferences on Big Data Analytics, Data Mining and Computational Intelligence 2024, BigDaCI 2024; Connected Smart Cities 2024, CSC 2024; and e-Health 2024, EH 2024. pp. 43-49

[9] S. Chen, S. Ren, and Q. Zhang, “Hybrid Architectures that Combine LLMs and Predictive Analytics for Next-Generation Financial Modeling. Mathematical Modeling and Algorithm Application”, 2025, pp 31-43.

[10] M. E. Peters et al. “Deep contextualized word representations,” in Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers), M. A. Walker, H. Ji, and A. Stent, Eds. Association for Computational Linguistics, 2018, pp. 2227–2237.

[11] L. Xu et al. “Prompting large language models for recommender systems: A comprehensive framework and empirical analysis." 2024, arXiv:2401.04997.

[12] Keras Team, “Keras documentation: Multi-GPU distributed training with TensorFlow.” Keras.io, 2020, retrieved January, 2026, https://keras.io/guides/distributed_training_with_tensorflow/

[13] A. P. Behera, J. P. Champati, R. Morabito, S. Tarkoma, and J. Gross, “Towards efficient multi-llm inference: Characterization and analysis of llm routing and hierarchical techniques.” 2025, arXiv preprint arXiv:2506.06579.

[14] Keras, Home - Keras Documentation, Keras.io; Keras, 2019, retrieved May, 2025, https://keras.io/