Dual-Model Distillation for Efficient Action Classification with Hybrid Edge-Cloud Solution

Timothy Wei *
Saratoga High School
timswei@gmail.com

Hsien Xin Peng *
Ridge High School
maxpeng123678@gmail.com

Elaine Xu
The Harker School
elainexuyz@gmail.com

Bryan Zhao *
Saratoga High School
bryanzhao08@gmail.com

Lei Ding
University of California Santa Cruz
lding25@ucsc.edu

Diji Yang †
University of California Santa Cruz
dyang39@ucsc.edu

Abstract

As Artificial Intelligence models, such as Large Video-Language models (VLMs), grow in size, their deployment in real-world applications becomes increasingly challenging due to hardware limitations and computational costs. To address this, we design a hybrid edge-cloud solution that leverages the efficiency of smaller models for local processing while deferring to larger, more accurate cloud-based models when necessary. Specifically, we propose a novel unsupervised data generation method, Dual-Model Distillation (DMD), to train a lightweight switcher model that can predict when the edge model's output is uncertain and selectively offload inference to the large model in the cloud. Experimental results on the action classification task show that our framework not only requires less computational overhead, but also improves accuracy compared to using a large model alone. Our framework provides a scalable and adaptable solution for action classification in resource-constrained environments, with potential applications beyond healthcare. Noteworthy, while DMD-generated data is used for optimizing performance and resource usage in our pipeline, we expect the concept of DMD to further support future research on knowledge alignment across multiple models.

1 Introduction

Recent advances in Artificial Intelligence (AI) have shown that increasing model size leads to substantial performance gains across various domains, from language models [3, 8] to video-language models [11, 18]. Large-scale models, with billions of parameters, consistently achieve state-of-the-art results on various benchmarks [19]. However, these models can be challenging to deploy when translating benchmark performance into real-world applications [13] due to their high runtime costs.

One critical area where this limitation becomes evident is in action classification for elderly care [15]. A task such as predicting falls through non-intrusive monitoring [12, 5] is a critical yet complex task that benefits from large-scale video-language models [5]. Despite their high accuracy, these

*Work done during internship at the University of California, Santa Cruz
†Corresponding to: dyang39@ucsc.edu

Workshop on Video-Language Models at 38th Conference on Neural Information Processing Systems (NeurIPS 2024).

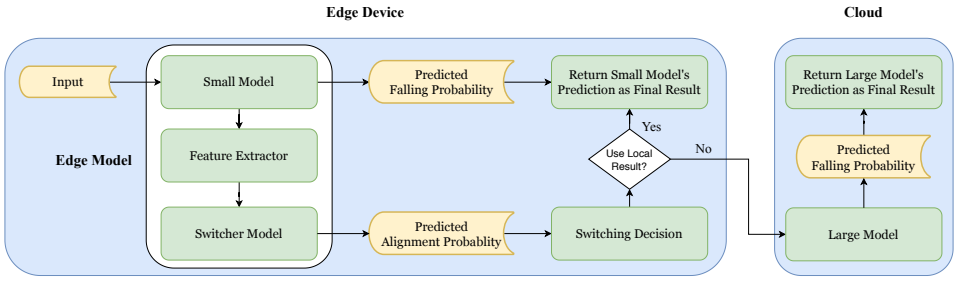

Figure 1: Overview of our hybrid edge-cloud solution

oversized models are not feasible for deployment on common elderly care appliances due to hardware constraints [1]. To address this, we propose a hybrid edge-cloud solution where small models run locally on edge devices while larger models deployed in the cloud are accessed only when necessary. However, the core technical challenge here remains: how to effectively align the knowledge and capabilities of the small and large models to ensure seamless collaboration between them. While Model or Knowledge Distillation techniques [7, 6] have been extensively explored, their primary goal is to compress the knowledge of large models into smaller ones. Yet, this process often falls short in collaborative systems where both small and large models have unique strengths. The smaller model, after distillation, may not fully exploit its distinct capabilities or optimally collaborate with its larger counterpart [14].

To overcome these limitations, we introduce Dual-Model Distillation (DMD), a novel approach to enhance cooperation between small and large models. Unlike traditional distillation, which focuses on unidirectional knowledge transfer, DMD treats both models as teacher models to co-teach and generate synthetic data to train a student model (i.e., the switcher model). The DMD process eliminates the need for additional human-labeled data by leveraging the agreement or disagreement between the two teacher models' zero-shot inferences to create training data. Through instruction-supervised finetuning [2], the switcher model learns to make optimal decisions about when to invoke the larger model. Experiments conducted on the dataset used in Fall Detection - Eldercare Robot datase [4] demonstrate the effectiveness of our approach. Surprisingly, our pipeline not only demands less computational overhead than just using the large model, resulting in 39.5% cost savings, but also outperforms the isolated large model by 4.6% in terms of F1 score. Furthermore, we anticipate that the DMD concept will provide substantial opportunities for future research on knowledge integration across multiple deep learning models.

2 Methodology

As shown in Figure 1, our hybrid edge-cloud solution introduces a companion switcher model using the hidden features of the input data extracted from the small model, which serves as the foundation of its reasoning capacity. With the help of the switcher model, our edge model can not only predict the falling probability but also accurately estimate the probability that the large model's result aligns with its own. By incorporating this switcher model, the edge model deliberates when to invoke the cloud model, thereby keeping system expenses under a pre-defined budget. In our case, a threshold of alignment probability, which yields optimal performance in predicting the falling probability during training, determines when to offload data to the large model during inference time. Regarding more complex decision-making scenarios, we expect that a more sophisticated utility-based threshold design [17] can also be applied in the future.

Dual-Model Distillation To give the switcher model an understanding of the capabilities of both models, we propose a novel training data generation method. First, we pass all the raw data through the small model and extract its hidden representation together with its prediction. Then, we trigger another run-through using the large model to generate the ground truth of alignment, or agreement, between the predictions of the small and large models. It is important to underscore the generalizability of this method, as for any of two deep learning models with the same I/O format, the DMD process can generate the above-described data from any raw data with only the required initial input and final output (which is provided in most datasets). During the subsequent instruction finetuning, the

Approach	F1 (%)	Large Model (%) [3]	Time (s)	Energy (kJ)
Small model only [9]	58.2	0	18.25	2.32
Large model only [10]	87.5	100	663.1	190.73
Uncertainty-based system [14]	76.1	60	386.91	113.04
Ours	**92.1**	60	405.16	115.36

Table 1: Results of different approaches.

switcher model learns the dual-model distilled knowledge. At inference time, using its grasp of both models' capability, the switcher is able to decide when to utilize the power of the large model.

3 Experiments

We conducted the experiments on the Fall Detection - Eldercare Robot [4] dataset, with more details provided in Appendix A.1. We used a Vision-and-Language Transformer (ViLT) [9] as the small model, chosen for its computational efficiency and relatively small size (less than 500 MB). For the large model, we utilized LLaVA-NeXT-Video-7B [10], which is a well-performing open-source video-language model. Following the DMD process described in Section 2, we generated data where the inputs consist of the last hidden layer of the small model for each image input in the raw data, and the outputs indicate whether the small model and large model agree on their falling predictions. To minimize the computing on edge devices, we design a lightweight switcher model implemented as a Multi-Layer Perceptron (MLP) as an ad-hoc layer on top of the small model. The switcher model is trained on the DMD-generated data for the intelligent model switching. More implementation details are available in Appendix A.3.

3.1 Results

For performance comparison, we conducted experiments and included the results of three baselines. As reported in Table 1, reliance solely on the ViLT (Small model only) resulted in an F1 score of 58.2%, while LLaVA-NeXT-Video (Large model only) got 87.5%. In addition, we reproduced a popular uncertainty-based switcher model approach [14] on our task. Specifically, we extracted the falling probability in the final output layer of ViLT and used that as the metric to determine whether to defer judgment to the larger model. This baseline got 76.1% when offloading 60% computing to the large model. Our system achieves 92.1%, which is 33.9%, 4.6%, and 16.0% higher than the traditional approach, large model-only approach and the uncertainty-based approach respectively.

Performance Analysis To gain a deeper understanding of the performance, we manually adjust the threshold for offloading decisions to large models as described in A.4. As shown in Figure 3, our system consistently outperforms the traditional uncertainty-based system regardless of the offload threshold. Furthermore, it suggests that solely calling the large model may be *less optimal* than calling it only when the switcher model indicates to do so, as the switcher model knows better about both models' capabilities from the DMD process. Therefore, we conclude that the switcher model offers a low-cost, lightweight solution to increase fall detection accuracy while optimizing efficiency.

Computational Cost Evaluation We also evaluated the computational cost of our model in terms of energy consumption and runtime compared to using a large model only solution, following the method described in Appendix A.5. As illustrated in Figure 4, our switcher model solution, which defers judgment to the large model 60% of the time (corresponding to the switcher model's behavior on the test data), achieved a significant reduction in energy consumption. Specifically, the energy used dropped from 0.0530 kWh to 0.0320 kWh, a 39.5% reduction in energy usage. Also, as shown in Figure 5, the time taken for processing decreased from 663.1 seconds to 405.16 seconds, a 38.9% reduction in time taken. Thus, our system not only improves detection quality but also enhances detection latency, energy efficiency, and overall computational cost.

[3] Large Model (%) refers to the percentage of test data instances that were processed using the large model.

4 Conclusion and Future Works

In this paper, we introduced a novel Dual-Model Distillation (DMD) method to develop a lightweight hybrid edge-cloud solution that efficiently balances computational cost and accuracy, enabling deployment on resource-constrained edge devices with support from a large model in the cloud. Although our framework is designed for long video inputs, our current experiments are limited to single-frame video inputs. In future work, we plan to transition to video-based inputs and test our method on more diverse action classification datasets. Overall, we expect our framework to be adaptable and applicable to other real-world applications beyond fall detection. Moreover, we anticipate that DMD will advance the research frontier of knowledge alignment across multiple models in a variety of domains.

Acknowledgments and Disclosure of Funding

We would like to thank Evan Li and Ryan Tellado for their insightful feedback and contributions to the development of this paper.

References

[1] Jiasi Chen and Xukan Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019.

[2] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

[3] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[4] Ahmad Elwaly and Ahmed Abdellatif. Fall detection - eldercare robot, February 2023. URL https://www.kaggle.com/datasets/elwalyahmad/fall-detection.

[5] Ahmad Elwaly, A Abdellatif, and Y El-Shaer. New eldercare robot with path-planning and fall-detection capabilities. *Applied Sciences*, 14(6):2374, 2024.

[6] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[7] Geoffrey Hinton. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[8] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[9] Wonjae Kim, Bokyung Son, and Ildoo Kim. Vilt: Vision-and-language transformer without convolution or region supervision, 2021.

[10] Feng Li, Renrui Zhang, Hao Zhang, Yuanhan Zhang, Bo Li, Wei Li, Zejun Ma, and Chunyuan Li. Llava-next-interleave: Tackling multi-image, video, and 3d in large multimodal models. *arXiv preprint arXiv:2407.07895*, 2024.

[11] Bin Lin, Bin Zhu, Yang Ye, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023.

[12] Saturnino Maldonado-Bascon, Cristian Iglesias-Iglesias, Pilar Martín-Martín, and Sergio Lafuente-Arroyo. Fallen people detection capabilities using assistive robot. *Electronics*, 8(9):915, 2019.

[13] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2024.

[14] Taman Narayan, Heinrich Jiang, Sen Zhao, and Sanjiv Kumar. Predicting on the edge: Identifying where a larger model does better. *arXiv preprint arXiv:2202.07652*, 2022.

[15] Bartosz Sawik, Sławomir Tobis, Ewa Baum, Aleksandra Suwalska, Sylwia Kropińska, Katarzyna Stachnik, Elena Pérez-Bernabeu, Marta Cildoz, Alba Agustin, and Katarzyna Wieczorowska-Tobis. Robots for elderly care: Review, multi-criteria optimization model and qualitative case study. In *Healthcare*, volume 11, page 1286. MDPI, 2023.

[16] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7464–7475, 2023.

[17] Jian Wang and Yi Zhang. Utilizing marginal net utility for recommendation in e-commerce. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1003–1012, 2011.

[18] Shengqiong Wu, Hao Fei, Leigang Qu, Wei Ji, and Tat-Seng Chua. Next-gpt: Any-to-any multimodal llm. In *Forty-first International Conference on Machine Learning*, 2023.

[19] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

A Appendix and Supplemental Materials

A.1 Datasets

We used the Fall Detection - Eldercare Robot dataset for training, validation and testing. It includes 1061 images, each having a data entry in YOLO v7 PyTorch format [16]. The data entry includes a binary fall/no-fall classification label, the coordinates of the center of the bounding box containing the person, and the width and height of the bounding box. The creators of the dataset oriented pixel data automatically using EXIF-orientation stripping and stretched to resize it to 640x640. When exported, each image had a 50% probability of being laterally inverted and a one-in-three chance of undergoing one of the following operations: a 90-degree clockwise rotation, a 90-degree anti-clockwise rotation, or none at all. Of the 1061 images, we follow the commonly used split, i.e., 742 for training, 212 for validation, and 106 for testing.

A.2 Data Generation and Formatting using Dual-Model Distillation

In order to prepare and standardize our data for training, we implemented the DMD process as described in this section. As discussed in Section 2, the only information we needed to train the switcher model is the nodes of the last hidden layer for each given input and whether the large and small models agree, referred to as the alignment between the models. We used our novel DMD framework to refine these data for training: by inputting each image into both models with a prompt telling them to report if a fall occurred, we extracted their outputs and recorded whether the models agreed. The values of the nodes in the last hidden layer of ViLT were extracted and saved accordingly. Ultimately, our refined data consisted of only three fields, making the DMD process compatible with any image or video dataset. An example of the formatted data after the DMD process is given as following.

```
[
    {
        "image_path": "path_to_image/img.jpg",
        "last_hidden_layer": [
            1.369998574256897,
            ...
```

```
        ],
        "label": 1
    },...
]
```

A.3 Implementation Details

Model Design To make our switcher model lightweight yet effective, we implemented a Multi-Layer Perceptron (MLP) with input size 1536, two hidden layers of size 512 and 128, and an output layer of size 1. Due to this structure, it can be easily implemented in a local system with little additional cost or memory. However, we also found that by using the hidden representation of the last hidden layer from the small model, the MLP demonstrates the capability of understanding the reasoning of the small model, therefore making better alignment predictions.

Training Details As for the training settings, we used a learning rate of 0.0001 and a dropout rate of 0.3. Using the refined data mentioned above, we calculated the Binary Cross-Entropy loss (BCELoss) followed by the Logits function (which applies the BCELoss function after the Sigmoid function), and updated weights through backpropagation. The training early stopped at 100 epochs, where the model achieved an F1 score of 79.2% and an accuracy of 82.2% stably.

A.4 Threshold for Offload Decision

To determine the optimal threshold for deferring judgment to the large model at inference time, we analyzed the switcher model's performance on the training data.

We collected the results through our switcher model after all the training data was processed. We sorted these results in ascending order of alignment probability and split the results into 10 "buckets". Using this "buckets" strategy on the training set, we then plotted the graph of the combined model F1 score against the percentage of data deferred to the large model. For comparison purposes, we also plot the baseline (a standard uncertainty-based switcher model) in Figure2.

By observing the training curve of the combined model, we chose the optimal threshold value 60%, shown as the peak of the curve in Figure 2, for testing. For the baseline, uncertainty-based system, we also used 60%, because the baseline uncertainty doesn't have a clear peak to indicate how much data should be offloaded to the large model. Keeping this threshold consistent helps us compare performance fairly. The corresponding results are shown in Table 1.

To explore how the threshold value influences the final testing performance, we further applied our "buckets" strategy to test data for both methods and analyzed the results. The corresponding results are shown in Figure 3. As a note, this analysis of accuracy against percent of data deferred was used extensively in [14] and can be extended to any framework involving a switcher model.

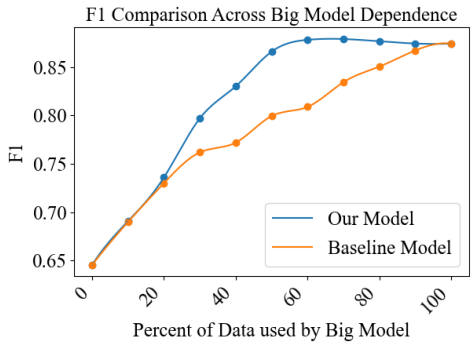

Figure 2: Combined and baseline model accuracy against the percentage of data deferred to the large model in the training data split

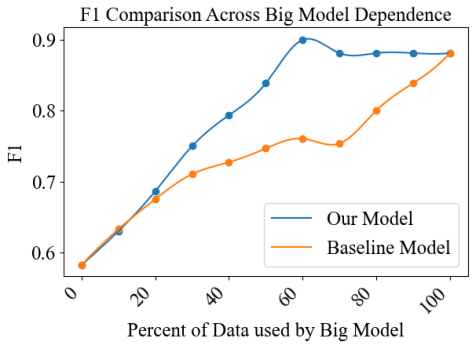

Figure 3: Combined and baseline model accuracy against the percentage of data deferred to the large model in the testing data split

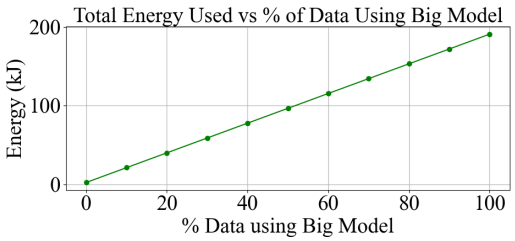

Figure 4: Total energy consumed against the percentage of data deferred to big model

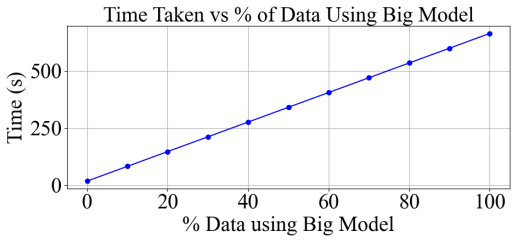

Figure 5: Runtime of combined model against percentage of data deferred to big model

A.5 Computational Cost Evaluation

Finally, we compared our model's computational cost to our baselines. We measured the total energy consumption of our system using NVIDIA's `nvidia-smi` tool and tracked the time taken for processing with Python's `time` library on our training dataset. To explore the tradeoff in computational cost for performance, we graphed both energy consumption and time against the percentage of test cases where the large model was involved. Our simplistic analysis can be applied to other metrics of computational cost and to analysis of other models. The results are illustrated in Figure 4 and Figure 5.