Dynamic Finetuning Of Multi-Task Multi-Modal Models: A Case Study On Comic Mischief Detection

Cullen Anderson
University Of Massachusetts Amherst
cyanderson@umass.edu

Sukruth Rao
Michigan State University
sukruth.j.rao@gmail.com

Abstract

Multitask Multimodal Models have seen significant advancement in recent years. However, there are many challenges in effectively training a multitask model and a variety of dynamic training methods to address these issues. It is not always clear which method is superior or why performance differences arise between multitask models and their single task counterparts. We focus on the problem of applying dynamic training methods to finetuning multimodal multitask models. In particular, we finetune a SOTA pretrained multimodal model, HICCAP, on the task of Comic Mischief Detection. We survey existing dynamic training methods for multitask learning and present two simple extensions of methods, based on dynamic loss reweighting and curriculum learning, that achieve SOTA performance on Comic Mischief Detection.

1 Introduction

There has been a proliferation of excitement in multimodal learning and multitask learning in recent years. Both areas enable the leveraging of more information within data and are crucial for solving nuanced problems and for building more general purpose AI agents. Many key challenges remain within both fields. While multitask learning has been shown to yield performance benefits with improved data efficiency, it can also sometimes lead to worse performance. To yield improved performance with curriculum learning, it is crucial to employ *dynamic training strategies*. Such methods dynamically control the training process by overseeing how data is sampled or by reweighting loss functions. These methods are especially important within multitask learning, where they help to facilitate even training across all tasks.

Despite the growing amount of literature in dynamic training strategies, they have not yet been widely adopted in practice beyond naive approaches. We aim to bridge this gap by examining a variety of dynamic training methods on the finetuning of multimodal multitask models. In particular, we study the problem of Comic Mischief Detection and evaluate different methods over the SOTA model for this task, HICCAP.

Structure We begin by introducing the problem of Comic Mischief Detection, the dataset, and the HICCAP model in Section 2. We additionally motivate our study based on the performance of the current SOTA model on this problem. We breifly discuss multitask learning in Section 3 detail several dynamic training methods in Section 4, giving the reader necessary background knowledge and intuitions on the nuances of multitask learning. We evaluate these methods on the Comic Mischief Detection dataset in Section 6. We explain avenues of future work in Section 8. We include additional experiments and plots in Appendix A and B.

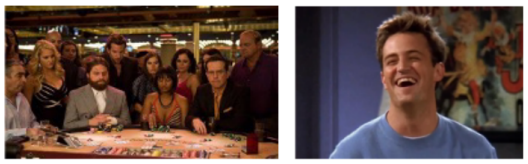
2 Comic Mischief Detection

Certain forms of online media have been linked to numerous negative effects in children. As a result, detecting questionable content in online media is an important goal that can be supported by AI. Detecting comic mischief, which includes violence, adult content, slapstick, or sarcasm with humor, is an important subcategory of this problem. Such content is difficult to detect as it requires the capturing of subtle details across modalities. Recently, (Baharlouei et al., 2024) developed both a novel dataset and a multimodal, multitask model for this problem which achieves SOTA results.

2.1 Dataset

(Baharlouei et al., 2024) introduced a novel dataset for Comic Mischief Detection. Comic Mischief is divided into four labels corresponding to gory humor, slapstick humor, mature humor, and sarcasm. Data is gathered from YouTube and consists of three modalities: dialogue (transcription

(a) Gory Humor (b) Slapstick



(c) Mature Humor (d) Sarcasm

Figure 1: Comic Mischief Dataset Examples

of spoken dialogue), sound, and video. In total, the dataset consists of 4,478 60 second clips from 1,179 videos. Figure 1 displays examples of images from the dataset belonging to each category of comic mischief.

2.2 HICCAP Model

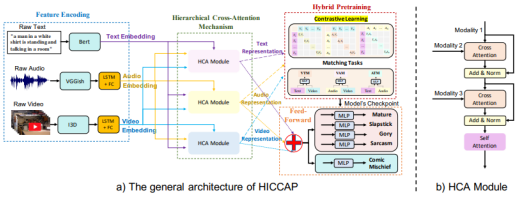


Figure 2: HICCAP Model Architecture

The "HIerarchical Cross-Attention model with CAptions" (HICCAP) is a multimodal model used to detect comic mischief in online videos. For ease of reference, we display the model architecture in Figure 2. It uses a pretrained BERT model to encode text (Devlin et al., 2019), a pretrained VGGish model to encode audio (Hershey et al., 2017), and a pretrained I3D model to encode video (Carreira and Zisserman, 2018). Each encoding is then further processed by RNNs and Fully Connected layers. Crucially, the embedding for each modality is further processed in the Hierarchical Cross Attention (HCA) modules to enhance the representation of all the modalities by capturing dependencies across all modalities. Finally, the embeddings for each modality are concatenated and passed on to task-specific output heads.

HICCAP was pretrained using a hybrid approach that combines matching tasks and contrastive learning methods. This combined pretraining framework unifies the various modalities into a single representation space, allowing the model to understand and capture the complex relationships between video, audio, and text. Finally, the model is fine-tuned for comic mischief detection, with task specific heads for all four labels - gory humor, slapstick humor, mature humor, and sarcasm. The original paper also trains a single task classification model, that detects whether or not a piece of content has comic mischief in it.

2.3 Motivation

Method	F1 MH	F1 GH	F1 SH	F1 S	Macro F1
HICCAP per task	79.44	73.59	43.80	74.42	67.81
Multi-task HICCAP	76.83	65.73	62.34	74.93	69.95

Figure 3: HICCAP Model Original Results

While (Baharlouei et al., 2024) achieved SOTA results and demonstrated improved performance of multi-task learning over single-task and multi-label models, notable questions remain. We report the performance of HICCAP given in the original paper on Comic Mischief Detection over single task and multi task models in Figure 3. While Macro F1-Score improves through multi-task learning, individual performance changes vary significantly. The F1-Score for slapstick humor sees a whopping 18.54% gain. Meanwhile, sarcasm sees a marginal 0.51% gain while both mature humor and gory humor degrade in performance, with 2.61% and 7.86% drops respectively. This suggests that certain tasks train better alongside others, while others train better alone. This is a common issue within multi-task learning. We aim to investigate why this issue occurs and whether performance can be improved through the utilization of different training methods tailored to multi-task learning. In addition to evaluating the problem of comic mischief detection we hope to shed light on the general applicability of dynamic training methods to finetuning multitask multimodal models.

3 Multi Task Learning

Multitask learning (MTL) is a subfield of machine learning in which a shared model learns multiple tasks simultaneously. This has been shown to be capable of yielding improved performance over single task models (Lu et al., 2020) through improved

data efficiency, reduced overfitting through shared representations, and fast learning by leveraging auxiliary information. Despite these potential benefits, employing multitask learning is non trivial, and can often degrade performance. For a more comprehensive survey see recent surveys (Crawshaw, 2020; Yu et al., 2024).

4 Dynamic Training Methods

There are several methods to train multitask models which we call dynamic training methods. Some of these methods are unique to multitask learning, while others emerged in the past and have since been generalized to multitask learning. We classify and provide overviews of several methods here, detailing naive approaches, their flaws, and how modern approaches improve upon them. We evaluate these methods on the finetuning of HICCAP over the comic mischief detection dataset. Due to time constraints, we did not evaluate all methods, and make special note of the methods which we do not implement.

4.1 One Task At A Time

Some dynamic training methods only ever train on one task at a time. In other words, they only ever backpropogate on the loss function for one task a time. Different methods control how tasks are iterated through.

Never Look Back Training Naively, train each task until convergence before moving on to the next task. Continue until all tasks have been covered. This additionally requires that the order of tasks be specified beforehand, which may lead to differences in performance. This naive method illustrates a key issue to multitask learning: *catastrophic forgetting*. After training on later tasks, the model is likely to diverge on earlier tasks and attain much worse performance despite converging earlier.

Round Robin Training Cycle through tasks in a specified interval and ordering. This naive approach combats the catastrophic forgetting issue by ensuring that the model never stops seeing training data from any task. However, this approach induces overfitting on easier tasks. It is likely for there to be easier tasks that train faster, meaning that they will continue to train despite having already converged while harder tasks are still training.

Dynamic Stop And Go (Lu et al., 2020)

Cycle through tasks, but dynamically assign each task a "stop" and "go" mode. When a task is in "go" mode, it is not converged and will continue to train. When a task is in "stop" mode, it is converged and will not train to reduce overfitting. If a task diverges while in "stop" mode, return the task to "go" mode and return to training. This method addresses both issues in the previous two naive methods.

Curriculum and Anti-Curriculum Learning Assign an ordering of difficulties to tasks and train from easiest to hardest. To prevent catastrophic forgetting, we start with a subset of easier tasks and gradually add harder tasks to it. Anti-curriculum learning does exactly the opposite of this, training from hardest to easiest. As described in (Wang et al., 2021), curriculum learning methods involve a difficulty measurer, which ranks the data by difficulty, and a training scheduler, which utilizes this difficulty ranking to control how data is fed into the model during the training process. This very general framework was introduced in (Bengio et al., 2009) and has seen significant improvement and adaptations. For surveys on this field see (Wang et al., 2021; Soviany et al., 2022). In its broadest sense, all methods described in this section operate in the spirit of curriculum learning. However, we restrict the terminology curriculum learning here to refer only to this more narrow definition.

Difficulty Based Drawing Probabilistically draw training batches based on task difficulty. Task difficulty is difficult to define and calculate in general, and heuristics such as data size and expert knowledge are often employed in practice (Hu and Singh, 2021). This broad framework may motivate many other approaches. We do not evaluate this method.

4.2 All Tasks At Once

Alternatively, we can train all tasks simultaneously within a single training batch. To achieve this, we define a new loss function as the (possibly weighted) sum of task specific loss functions. That is, for task specific loss functions, $\mathcal{L}_i$, and corresponding weights, w_i, we define the new loss function as $\mathcal{L} = \sum_I w_i \mathcal{L}_i$. This is prone to two significant issues which help illustrate the difficulty of multi task learning in general. First, loss functions may lay on different scales, causing tasks with larger losses to overwhelm the training process compared to tasks with smaller losses. Second, gradients may point in different directions. As a re-

sult, minimizing the loss on one task may increase the loss on other tasks. Although taking the sum of task specific loss functions attempts to average out these effects, there may still be destructive interference, especially when losses lay on different scales. A naive solution to these issues is to treat the task losses as hyperparamaters, although the methodology of doing this isn't always clear. The original HICCAP model was trained through this process using manually tuned weights.

To combat the previously discussed issues, there are several methods that work by manipulating losses and gradients. We consider these as dynamic training methods because, in effect, they control the scheduling of tasks, as described in (Crawshaw, 2020). We also consider dynamic difficulty sampling alongside these loss reweighting methods.

Gradnorm (Chen et al., 2018)

Gradnorm sets loss weights as learnable parameters with a separate optimization objective. This objective aims to place gradients on a common scale and to dynamically adjust gradient norms so different tasks learn at a similar rate. Loss weights can either be dynamically adjusted throughout training, or learned ahead of time through an initial training run. We implement the first strategy, which is slightly more competitive with significantly less time cost.

PCGrad and Uncertainty Weighting (Yu et al., 2020; Kendall et al., 2018)

There are several other methods in this spirit. Notably, PCGrad (Yu et al., 2020) directly manipulates gradients to point in a similar direction. Much of our exposition on the intuitive foundation behind all tasks at once based training methods come from this work's analysis. There are several other methods in this spirit. An earlier work (Kendall et al., 2018) shows the importance of task specific weighting and proposed one of the first methods in this line, called Uncertainty Weighting, which considers the uncertainty between tasks.

Inverse Loss Task Sampling (Piergiovanni et al., 2023) Control the portion that each task makes up of each batch through a difficulty metric. Specifically, have each task make up $\frac{\mathcal{L}_i}{\mathcal{L}}$ of each batch where $\mathcal{L}$ and $\mathcal{L}_i$ are training losses updated every k iterations. We note that this is a general strategy that can be adapted to different difficulty metrics. Additionally, it may be combined with loss reweighting methods.

4.3 Defining Task Groupings

One area of multitask learning focuses on analyzing which tasks to train together. Rather than needing to employ dynamic training methods to train tasks together and to resolve conflicts between them, these methods area determine which tasks train together best. A naive approach is to choose tasks to maximize cosine similarity between pairs of gradients between tasks. More sophisticated approaches have been devised to improve upon this baseline both in performance and time complexity. Recently, Fifty et al. 2021 devised a method that groups tasks based on pairwise inter-task affinity scores, which intuitively capture the effect of one tasks's gradient on another's. Prior to this Standley et al. 2020, approximates task groupings based on pair-wise task performance. Task grouping methods are complementary with loss reweighting based methods and future work is needed to integrate them. We do not evaluate these methods but note that this area is especially helpful when selecting auxiliary tasks to benefit performance across the main tasks.

4.4 Novel Methods

Inverse Loss Task Weights This method provides a novel way to weight task specific losses. Inspired by the batch sampling based approach in (Piergiovanni et al., 2023), we utilize the same loss ratio to weight the loss for each task, rather than to weight the proportion of each task within a batch. Define task weights, $w_i = \frac{\mathcal{L}_i}{\mathcal{L}}$. As training progresses, continue to adjust these weights accordingly. Tasks with higher losses are assigned more weight, directing the model to focus more on improving those tasks. This dynamic adjustment ensures that the model does not neglect any task and continuously strives to balance performance across all tasks.

Dynamic Curriculum Learning We define a simple difficulty measurer and training scheduler for curriculum learning. The algorithm starts with an assessment phase followed by a training phase. In the assessment phase, the curriculum is dynamically determined based on the difficulty of each task. In this phase, the model trains all downstream tasks for a fixed number of epochs, while collecting performance data. At the end of this phase an assessment is made based on the F1-Scores of the tasks. The curriculum is prepared with easier tasks coming first and more difficult tasks gradually getting incorporated. Next, each of these tasks are

assigned a fixed number of epochs for training. It is important to note that as new tasks are added for training, the previously included tasks are allowed to continue training. The net result is that the final part of training ends up including all the tasks. During training, when multiple tasks are involved, the combined loss is determined by assigning a weight to each task. We determine these weights using the inverse loss task weights previously discussed.

Dynamic Anti-Curriculum Learning Dynamic Anti-Curriculum Learning is identical to the previously described Dynamic Curriculum Learning with the exception of the order of tasks in the Curriculum. In this strategy the difficult tasks are trained before the easier ones.

5 Implementation

The previous implementation (RiTUAL-UH, 2024) of HICCAP architecture (Baharlouei et al., 2024) maintained independent code bases for binary and multi tasks. This meant the training, evaluation, and testing had to be done separately consuming excessive computational time and resources. Further all parts of the HICCAP architecture were tightly coupled resulting in a monolithic software. This posed significant problems when adding a new fine-tuning strategy as this required changing the heart of training and evaluation code. Our goal was to experiment with a number of fine-tuning strategies at short notice. This called for rapid turn around for the development and experimentation of the code. To achieve this, we redesigned the software architecture from scratch. The new software architecture mirrors the original HICCAP model completely in implementation. The salient features of the new architecture are:

- Modularized software components with well defined interfaces
- Simultaneous training, evaluation, and testing for all downstream tasks
- Pluggable fine-tuning strategies

The following software components are designed as Pytorch Neural Network modules with well defined interfaces to interconnect them:

- Feature Encoding
- Hierarchical Cross Attention
- Binary Tasks
- Multi Tasks

The modularized approach helped in putting together a combined model covering all the downstream tasks. The resulting structure allowed for simultaneous training, evaluation, and testing of all tasks in a single pass. A number of fine-tuning strategies are implemented using the interfaces offered by the software architecture. These interfaces are:

- **start_epoch**(): Invoked every time a new epoch is started.
- **end_epoch**(): Invoked at the end of each epoch.
- **start_batch_iter**(): Invoked at the beginning of a batch.
- **end_batch_iter**(): Invoked at the end of a batch.
- **process_eval**(): Invoked to process the evaluation results after each epoch.
- **backward**(): Entry point for backpropagation.
- **process_batch_eval**(): Invoked to process the results of evaluation within an epoch if performed. This is applicable to only certain fine-tuning strategies.
- **get_batch_eval_iter_count**(): Indicates the number of batches after which an evaluation must be performed. This is applicable to only certain fine-tuning strategies.

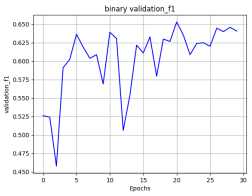
The above interfaces allowed for a number of fine-tuning strategies to be integrated seamlessly without changing the mainline code for training, evaluation, and testing.

6 Experiments

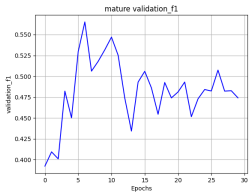
We evaluate dynamic training methods over finetuning the HICCAP model. In Table 1 we include final testing results over several methods. The Binary Task Model (BTM) column contains F1-Scores over a single task binary model. The Multi Task Model (MTM) column contains Macro F1-Scores for each strategy trained over all tasks. We additionally report task specific validation F1-Score histories. We employ the same experimental conditions as the original paper: a batch size of 16; an AdamW optimizer with a weight decay of 0.02, betas set at the default values of (0.9, 0.999), and eps at 1e-8; and PyTorch's adaptive learning rate scheduler which takes in validation Macro F1-Score, adjusts

Strategy	Epochs	BTM F1-Score	MTM Macro F1-Score
Fixed Task Weights	30	73.38 (unweighted)	69.34
Per-Batch Round Robin	30	79.27	66.13
Dynamic Stop-and-Go	35	74.33	67.95
Curriculum Learning	30	77.18	61.43
Anti Curriculum Learning	35	76.36	61.86
Dynamic Task Weights	40	79.31	70.66
Dynamic Anti Curriculum Learning	40	76.54	69.07
Dynamic Curriculum Learning	50	78.38	70.96

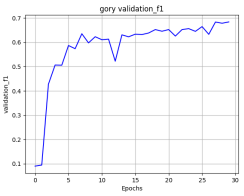
Table 1: Results of Fine-Tuning Strategies



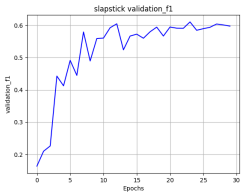
(a) Binary Validation F1-Score



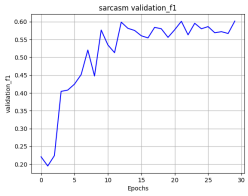
(b) Mature Validation F1-Score



(c) Gory Validation F1-Score



(d) Slapstick Validation F1-Score



(e) Sarcasm Validation F1-Score

Figure 4: Fixed Weights F1-Scores

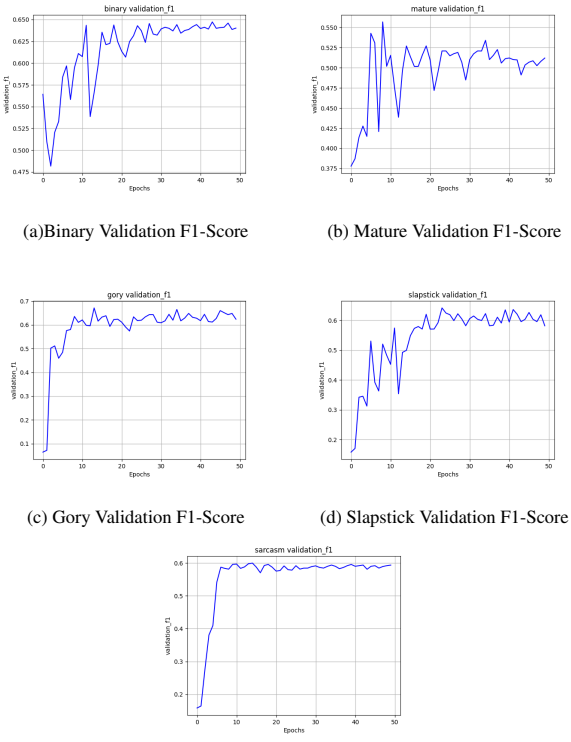
the learning rate by a factor of 0.5, and maintains a minimum learning rate of 1e-8. We note that because each data item has labels for every task, whenever we accumulate losses we utilize every label for each piece of data.

We first replicate the original paper's results over a multitask model and a naive finetuning strategy. The original paper trains tasks all at once and assigns fixed task-specific loss weights learned ahead of time. We treat these results as our baseline and report them in Figure 4. For the binary task the F1-Score was 73.38 and the average F1-Score for multi tasks was 69.34. These scores are marginally different from the ones reported in the paper.

We evaluate **Round Robin Training**, where we switch tasks at every batch. In the Per-Batch Round Robin strategy, the backpropagation for the binary task is always performed whereas the backpropagations for multi tasks are performed in a round robin manner. This gave the binary tasks an adequate chance for learning. This is reflected in a high F1-Score of 79.27 for the binary task. Since only one of the multi-tasks are selected for backpropagation for each batch, it may have led to inadequate learning which is showcased in a moderate F1-Score of 66.13. We note that, perhaps surprisingly, Round Robin Training is competitive with all other methods.

In the **Dynamic Stop-and-Go** strategy, all tasks are started in "GO" mode during training and the training continues normally. After a fixed number of batches (16 batches in our experiment), an evaluation is performed. If the current loss for a task exceeds the minimum of the last two recorded losses by more than 0.1%, the task is marked "converged", and its mode is switched to "STOP". If a task is in "STOP" mode and the current loss surpasses the best recorded loss by at least 0.5%, the task is marked "diverged", and its mode is switched back to "GO". While in "STOP" mode, the backpropagation for a task is performed only at a fixed interval of batch iterations (8 in our experiment). For the binary task the F1-Score was 74.33 and the average F1-Score for multi tasks was 67.95. While this strategy can improve performance of tasks prone to overfitting, it might interrupt the learning process for tasks that could benefit from continued training.

In **Naive Curriculum Learning**, the algorithm starts by training on the 'Binary' task first. Subsequently more complex tasks like 'Gory,' 'Sarcasm,', 'Slapstick', and 'Mature' are introduced in stages. For the binary task the F1-Score was 77.18 and the average F1-Score for multi tasks was 61.43. Anti-Curriculum Learning starts by training on the "Mature" task followed by "Slapstick", "Sarcasm", "Gory", and "Binary" tasks in that order. In this case, the F1-Score obtained for the binary task was 76.36 and the average F1-Score obtained for multi tasks was 61.86. Curriculum Learning and Anti-Curriculum Learning produced the two lowest multi-task scores, possibly due to their rigid task introduction sequences. The fixed arrangement of tasks may have led to overfitting on simpler tasks, making it harder for the model to adapt to more



(a)Binary Validation F1-Score

(b) Mature Validation F1-Score

(c) Gory Validation F1-Score

(d) Slapstick Validation F1-Score

(e) Sarcasm Validation F1-Score

Figure 5: Dynamic Curriculum Learning F1-Scores

complex ones later.

The **Inverse Loss Task Weights** strategy achieved an F1-Score of 79.31 for the binary task and an average F1-Score of 70.66 for multi tasks. This strategy outperformed the Weighted Strategy, which might be due to its adaptive capabilities. Unlike the fixed weights used in the Weighted Strategy, the Dynamic Weighted Strategy continuously adjusts the task weights based on their losses determined during backpropagation. This helped the underperforming tasks to instantaneously get higher weights, potentially leading to more balanced and efficient learning across all tasks. The adaptability nature of this algorithm might have helped the model optimize its training process, contributing to the improved overall performance.

Similarly, the newly conceived **Dynamic Curriculum Learning** and **Dynamic Anti-Curriculum Learning** also showed superior performance compared to their static counterparts. Dynamic Curriculum Learning resulted in an F1-Score of 77.56 for the binary task and an average F1-Score of 69.60 for multi-tasks. Dynamic Anti-Curriculum Learning achieved an F1-Score of 78.38 for the binary task and an average F1-Score of 70.96 for multi-tasks. The improvement can be attributed to their ability to dynamically determine the order of tasks for learning and dynamically adjust their weights. This adaptability allowed for a more balanced and sensitive training process. This ensured that the model effectively learned from a variety of tasks, leading to enhanced overall performance.

7 Conclusion

We evaluate a variety of dynamic training methods to finetune the multitask HICCAP model on comic mischief detection. We devise two simple extensions of existing algorithms that yield improved SOTA performance on comic mischief detection. These are inverse task weighting, which weights task specific losses through a simple and efficient method, and dynamic curriculum learning, a simple method of determining a curriculum for curriculum learning via an initial assessment phase.

8 Future Work

We note that this is preliminary work and further experiments must be done to draw more conclusive results. First, we need to more clearly lay out and compare experimental results and task specific performance. This comparison is necessary to determine how different tasks compare to their single task model counterparts and to note trends in how well different tasks train together. We must also perform ablations to better analyze the performance of different dynamic training methods and to understand how changes in performance arise. We also believe that our work can extend more generally beyond the use case of comic mischief detection and hope to evaluate our methods on wider classes of models and on multitask benchmarks. Another possible area for expansion would be to incorporate auxiliary tasks into the training mix. We believe that by training on adjacent tasks to comic mischief detection, such as emotion recognition, we can likely further push the comic mischief detection SOTA. Introducing auxiliary tasks also introduces important general technical challenges for dynamic training methods that warrant further investigation. We are also interested in further developing dynamic training methods that specifically take into account the multimodal and finetuning assumptions to achieve better performance.

Acknowledgments

We would like to thank Dr. Thamar Solorio and Dr. Hugo Jair Escalante for their support and guidance throughout this project. This work was done on site at INAOE in Puebla, Mexico in partnership with University of Houston as part of the International Research Experience for Students (IRES) program. This work was made possible by the support of the National Science Foundation (NSF).

References

Elaheh Baharlouei, Mahsa Shafaei, Yigeng Zhang, Hugo Jair Escalante, and Thamar Solorio. 2024. Labeling comic mischief content in online videos with a multimodal hierarchical-cross-attention model. In *Proceedings of the Language Resources and Evaluation Conference (LREC)*. European Language Resources Association (ELRA).

Y. Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. volume 60, page 6.

Joao Carreira and Andrew Zisserman. 2018. Quo vadis, action recognition? a new model and the kinetics dataset. *Preprint*, arXiv:1705.07750.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. 2018. GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks. *arXiv preprint*. ArXiv:1711.02257 [cs].

Michael Crawshaw. 2020. Multi-Task Learning with Deep Neural Networks: A Survey. *arXiv preprint*. ArXiv:2009.09796 [cs, stat].

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

Christopher Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. 2021. Efficiently Identifying Task Groupings for Multi-Task Learning. *arXiv preprint*. ArXiv:2109.04617 [cs].

Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. 2017. Cnn architectures for large-scale audio classification. *Preprint*, arXiv:1609.09430.

Ronghang Hu and Amanpreet Singh. 2021. Unit: Multimodal multitask learning with a unified transformer. *Preprint*, arXiv:2102.10772.

Alex Kendall, Yarin Gal, and Roberto Cipolla. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *Preprint*, arXiv:1705.07115.

Jiasen Lu, Vedanuj Goswami, Marcus Rohrbach, Devi Parikh, and Stefan Lee. 2020. 12-in-1: Multi-Task Vision and Language Representation Learning. *arXiv preprint*. ArXiv:1912.02315 [cs].

AJ Piergiovanni, Weicheng Kuo, Wei Li, and Anelia Angelova. 2023. Dynamic pretraining of vision-language models.

RiTUAL-UH. 2024. Comic mischief prediction. https://github.com/RiTUAL-UH/Comic-Mischief-Prediction/tree/main.

Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. 2022. Curriculum learning: A survey. *Preprint*, arXiv:2101.10382.

Trevor Standley, Amir R. Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. 2020. Which tasks should be learned together in multi-task learning? *Preprint*, arXiv:1905.07553.

Xin Wang, Yudong Chen, and Wenwu Zhu. 2021. A survey on curriculum learning. *Preprint*, arXiv:2010.13166.

Jun Yu, Yutong Dai, Xiaokang Liu, Jin Huang, Yishan Shen, Ke Zhang, Rong Zhou, Eashan Adhikarla, Wenxuan Ye, Yixin Liu, Zhaoming Kong, Kai Zhang, Yilong Yin, Vinod Namboodiri, Brian D. Davison, Jason H. Moore, and Yong Chen. 2024. Unleashing the Power of Multi-Task Learning: A Comprehensive Survey Spanning Traditional, Deep, and Pretrained Foundation Model Eras. *arXiv preprint*. ArXiv:2404.18961 [cs].

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient Surgery for Multi-Task Learning. *arXiv preprint*. ArXiv:2001.06782 [cs, stat].

A Additional Experiments

Here we evaluate some methods not described in the main paper. We leave these experiments to the appendix as they are either tangential to the main methods or yielded inconclusive results.

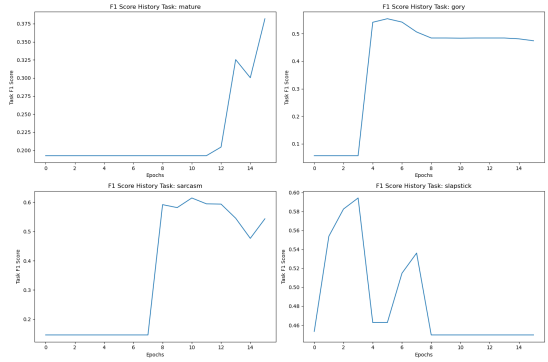
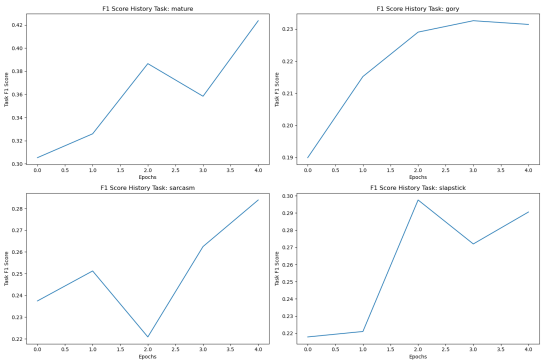


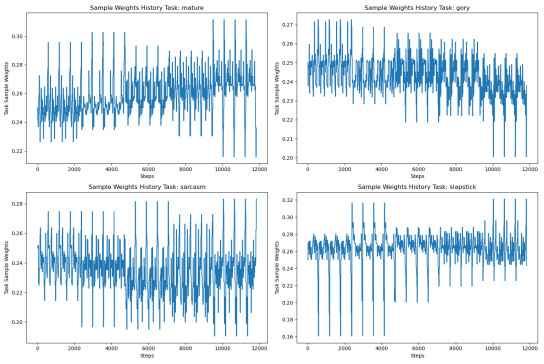
Figure 6: Never Look Back Training: Task-Specific F1 Scores

We evaluate **Never Look Back Training** where we train each task for 4 epochs before moving onto the next task. We define the order of training as slapstick, gory, sarcasm, mature. We informally define this ordering from easiest to hardest using the final F1 scores reported in the original HICCAP paper for both single task and multi task models (the ordering agrees over both paradigms). Task specific validation F1 scores are shown in Figure 6. We achieve the following test F1 scores: mature: 44.67, gory: 8.33, sarcasm: 47.04, slapstick: 0.00, macro: 25.01. As expected, this is not a viable strategy, with catastrophic forgetting being clearly exhibited. Notably, some tasks are more affected by catastrophic forgetting than others. For example, F1 score over gory tasks, which is trained second, only sees an initial performance drop from about 0.55 to 0.48 and then remains mostly stable after 8 epochs of training on other tasks. Meanwhile, F1 score over sarcasm, the first task, is about 0.6 after its training period, and quickly drops to about 0.45 after only 4 epochs of training on gory. Further work in understanding this phenomenon is an important line of future work in multi-task learning.

We evaluate **Inverse Loss Difficulty Sampling** over 5 epochs and report validation F1-Score and sampling weight histories in Figure 7. We achieve the following test F1 scores: mature: 48.18, gory: 9.01, sarcasm: 32.7, slapstick: 24.7, macro: 28.67. We note that despite using the same batch size across experiments, each training step here utilizes less data. Because each item in our dataset has



(a) Task-Specific F1 Scores

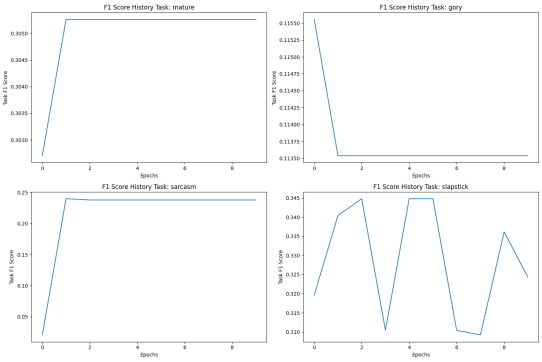


(b) Sampling Weights

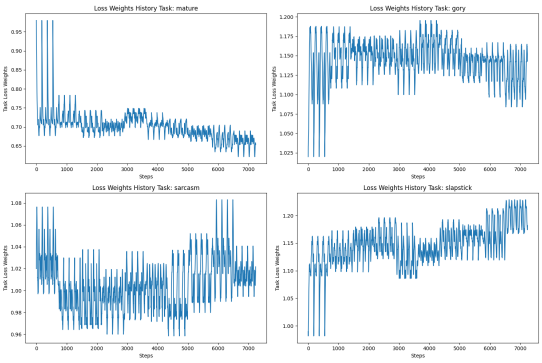
Figure 7: Inverse Loss Difficulty Sampling

labels for every class, we generally accumulate the losses for every task for each item of data. However, the nature of this method doesn't allow for this, instead forcing us to treat different labels for the same input data as different points of training data. We enforce a minimum of 2 data points per task per batch and resample data every 20 iterations. This is in contrast to the experimental setup in the original paper, which enforces a minimum of 4 data points per task per batch and resamples every 100 iterations. We make these modifications due to our lower batch size and running time. Like the original paper, we do not weight task-specific losses, but note that this is a possible area of improvemet within the algorithm. We find that despite the positive trend, F1-Scores remain lower than after training for the same number of epochs using fixed loss weights. We also see that sampling weights are highly volatile and do not notice clear trends or major deviations from a uniform split. Due to the lower number of training epochs and discrepancies in how data is handled, these results are inconclusive.

We evaluate **GradNorm** over 10 epochs and report validation F1-Score and loss weight histo-



(a) Task-Specific F1 Scores



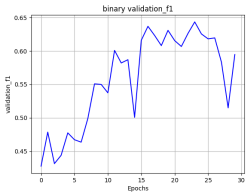
(b) Loss Weights

Figure 8: GradNorm

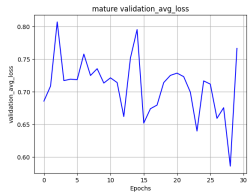
ries in Figure 8. We achieve the following test F1 scores: mature: 51.11, gory: 6.04, sarcasm: 23.78, slapstick: 10.90, macro: 10.90. Mature, gory, and sarcasm all freeze training after 1 epoch, with gory exhibiting a decrease in performance (though changes for mature and gory are marginal). Slapstick exhibits no clear training pattern. We also present the loss weight history, but there is no clear connection between these weights and the results given. We believe that these results are not reflective of the Gradnorm algorithm, but rather due to a bug somewhere in the implementation.

B Additional F1-Scores for Fine-Tuning Strategies

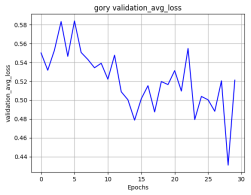
Here we include additional F1-Score histories for methods discussed in Section 6.



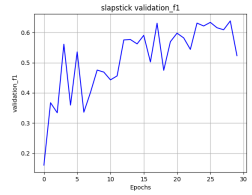
(a) Binary Validation F1-Score



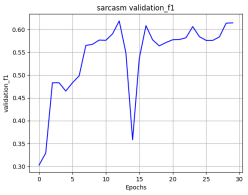
(b) Mature Validation F1-Score



(c) Gory Validation F1-Score

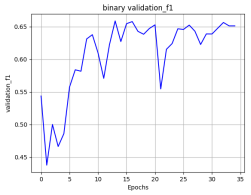


(d)Slapstick Validation F1-Score

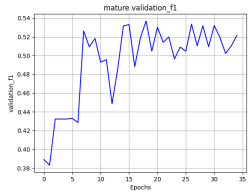


(e) Sarcasm Validation F1-Score

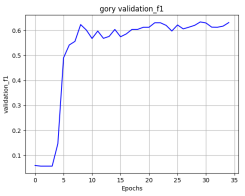
Figure 9: Round-Robin F1-Scores



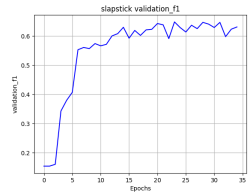
(a) Binary Validation F1-Score



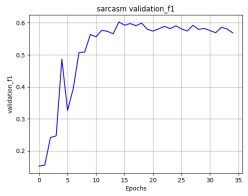
(b) Mature Validation F1-Score



(c) Gory Validation F1-Score

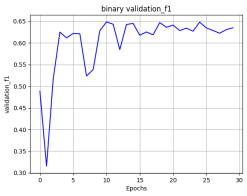


(d) Slapstick Validation F1-Score

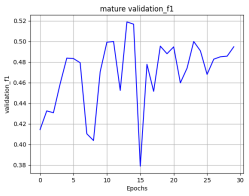


(e) Sarcasm Validation F1-Score

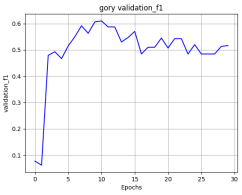
Figure 10: Dynamic Stop-and-Go F1-Scores



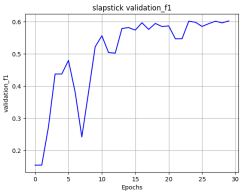
(a) Binary Validation F1-Score



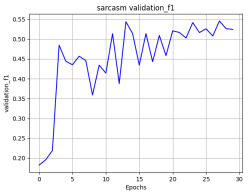
(b) Mature Validation F1-Score



(c) Gory Validation F1-Score

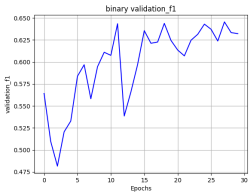


(d) Slapstick Validation F1-Score

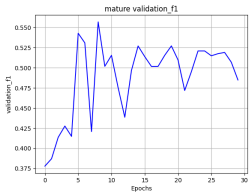


(e) Sarcasm Validation F1-Score

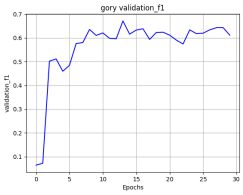
Figure 11: Curriculum F1-Scores



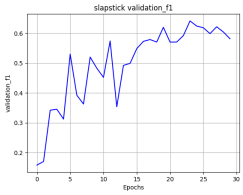
(a) Binary Validation F1-Score



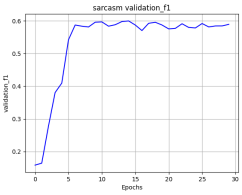
(b) Mature Validation F1-Score



(c) Gory Validation F1-Score

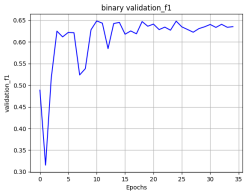


(d) Slapstick Validation F1-Score

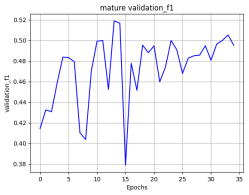


(e) Sarcasm Validation F1-Score

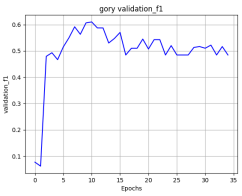
Figure 13: Dynamic Weights F1-Scores



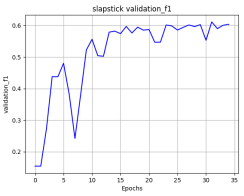
(a) Binary Validation F1-Score



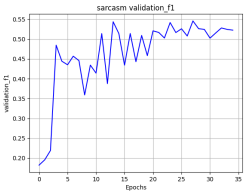
(b) Mature Validation F1-Score



(c) Gory Validation F1-Score

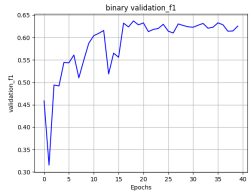


(d) Slapstick Validation F1-Score

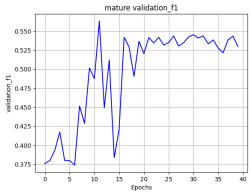


(e) Sarcasm Validation F1-Score

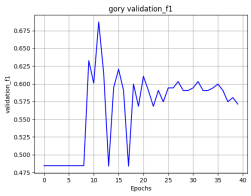
Figure 12: Anti Curriculum F1-Scores



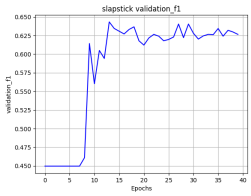
(a) Binary Validation F1-Score



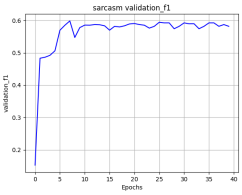
(b) Mature Validation F1-Score



(c) Gory Validation F1-Score



(d) Slapstick Validation F1-Score



(e) Sarcasm Validation F1-Score

Figure 14: Dynamic Anti Curriculum F1-Scores