Improving Baseline Performance on TRIP: Pre-Training, Backbone Substitution, and Multi-Task Learning Approaches

Shion Matsumoto
matsumos@umich.edu

Viraj Lunani
vlunani@umich.edu

Kevin Yan
yankevn@umich.edu

1 Introduction

Despite advances in natural language processing (NLP) on a variety of tasks, it is unclear as of yet *how* machines are understanding the tasks with which they are prompted to learn. To this end, Tiered Reasoning for Intuitive Physics (TRIP) was introduced as a benchmark for physical commonsense reasoning (Storks et al., 2021). The dataset consists of pairs of similar stories differing only by one sentence, where one of the stories is implausible as a result. The goal is to jointly identify 1) the plausible story, 2) pair of conflicting sentences in the implausible story, and 3) the physical state conflict that leads to the implausibility. By design, TRIP's dense annotations and multi-tiered evaluation strategy allow the evaluation of models at various levels of understanding – an approach orthogonal to modern datasets that feature superficial labels for a large number of samples.

As part of proposing TRIP, (Storks et al., 2021) presented a baseline model, which is able to identify the plausible story with up to 78% accuracy but jointly support its prediction with the correct conflicting sentence pair and corresponding physical state conflict only up to 11% of the time (Fig. 1. Specifically, despite a 73.6% accuracy on story classification, the baseline model struggles greatly with identifying the correct conflicting sentence pair. Even when the conflicting sentence pair is correctly identified, the model is only able to verify its prediction with the correct physical states half of the time.

This indicates that, though the baseline model's performance could be increased arbitrarily on the end-task loss (story classification), similar gains in conflict and physical state predictions require additional investigations into mechanisms that promote coherent reasoning. As such, there is substantial room for improvement with regards to the ability of the model to provide *verifiable* predictions on this new benchmark through incorporating recent advances in pre-training, language model architectures, and learning paradigms.

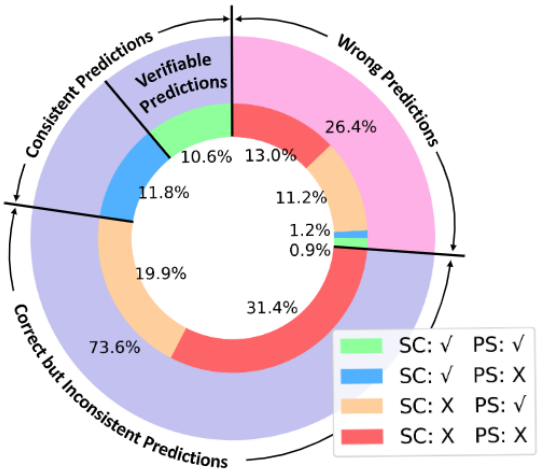


Figure 1: Breakdown of baseline model performance on TRIP (Storks et al., 2021).

2 Related Work

Briefly, the original TRIP architecture proposed by (Storks et al., 2021) consists of a series of modules (Fig. 2)[1]:

- Contextual Embedding: pre-trained transformer-based language model such as BERT that calculates an embedding of the input sentence and entity of interest
- Physical State (Precondition & Effect) Classifiers: feedforward classification heads for predicting the physical state of a given entity in a sentence
- Conflict Detector: a transformer followed by a feedforward layer to predict pairs of conflicting sentences in a story given the previously

[1] See the original article for an in-depth explanation of each module.

calculated contextual embedding and physical state predictions for each sentence-entity pair

- Story Choice Predictor: summation of the Conflict Detector outputs to predict story most likely to have a conflict

When attempting to improve the downstream performance of models on a specific task, pre-training models on separate datasets in a similar domain are a popular approach. Specifically for TRIP, (Ma et al., 2022) found that pre-training a RoBERTa-Large model on SQuAD 2.0 (Rajpurkar et al., 2018) provided a noticeable improvement on downstream performance.

Another popular approach is experimenting with various language models, which have been trained on natural language tasks such as sentence generation, question answering and more. One popular language model that has led to state of the art results is GPT-2 (Radford et al., 2019). With over 1.5B parameters and trained on a data set of millions of web pages known as WebText, GPT-2 has demonstrated its generalizability to many language processing tasks. Another popular language model is BART (Lewis et al., 2020), a transformer with a bidirectional (BERT-like) encoder and an autoregressive (GPT-like) decoder. It was trained by corrupting text with a noising function and then learning a model to reconstruct the original text. It matches the performance of other popular language models such as RoBERTa, and has achieved state of the art results on question answering and summarization tasks.

There has been extensive work in using these language models for commonsense reasoning such as the work done in (Shwartz et al., 2020). The work here deals with an unsupervised learning model that asks information seeking questions to gain an understanding of the world around it. The researchers used GPT-2 along with other transformers as a backbone for their model, and it is important to note that GPT-2 led to the best results for their task.

A final approach for consideration is multi-task learning. Models in NLP are often trained to learn multiple related tasks simultaneously as it enables them to capture both generalized and complementary knowledge. However, this often introduces a difficult optimization problem as there can be multiple objectives that cannot easily be consolidated into a single objective. As such, multi-task learning (MTL) in NLP has been used for tasks involving multiple languages, styles of supervision, and, as in our case, levels of features.

The simplest optimization-level approach to MTL is to linearly combine loss functions as in (Storks et al., 2021); however, more complex approaches can be used to provide better performance. One approach is to dynamically adapt the relative importance of individual tasks during the course of training (Song et al., 2020). Another is to modify conflicting gradients as proposed in PCGrad (Yu et al., 2020), where, in the case of two tasks, "gradient surgery" is performed by projecting one gradient $\boldsymbol{g}_i$ onto the normal plane of the other gradient $\boldsymbol{g}_j$. This eliminates conflicting components of the gradients from interfering with progress on the optimization objective.

3 Approaches

We proposed three distinct approaches to improve the baseline performance on TRIP: pre-training on datasets in a similar domain, substituting different language models to calculate the contextual embedding, and reformulating the optimization objective based on insights from MTL in NLP. If we were to refer to the original figure of the model architecture from (Storks et al., 2021), we modified the "Contextual Embedding" module and reformulated the objective function shown at the bottom (Fig. 2).

3.1 Pre-Training

The original model (Storks et al., 2021) used three separate pretrained language models: BERT, RoBERTa, and DeBERTa and did not pre-train their models on any other dataset, whereas (Ma et al., 2022) observed success pre-training a RoBERTa-Large model on SQuAD 2.0 (Rajpurkar et al., 2018). The SQuAD dataset tests whether a system can answer reading comprehension questions as well as abstain from questions that cannot be answered given the provided paragraph. This task is notably similar to the reasoning requirement of the TRIP, which is why we are interested in exploring whether pre-training our model on SQuAD, among others, could improve upon the baseline performance.

In total, we experimented with models pretrained on SQuAD 2.0 (Ma et al., 2022), PIQA (Bisk et al., 2020), and RACE (Lai et al., 2017). We hypothesized that using models pretrained on datasets similar to TRIP may yield higher downstream performance once further fine-tuned. Most notably,

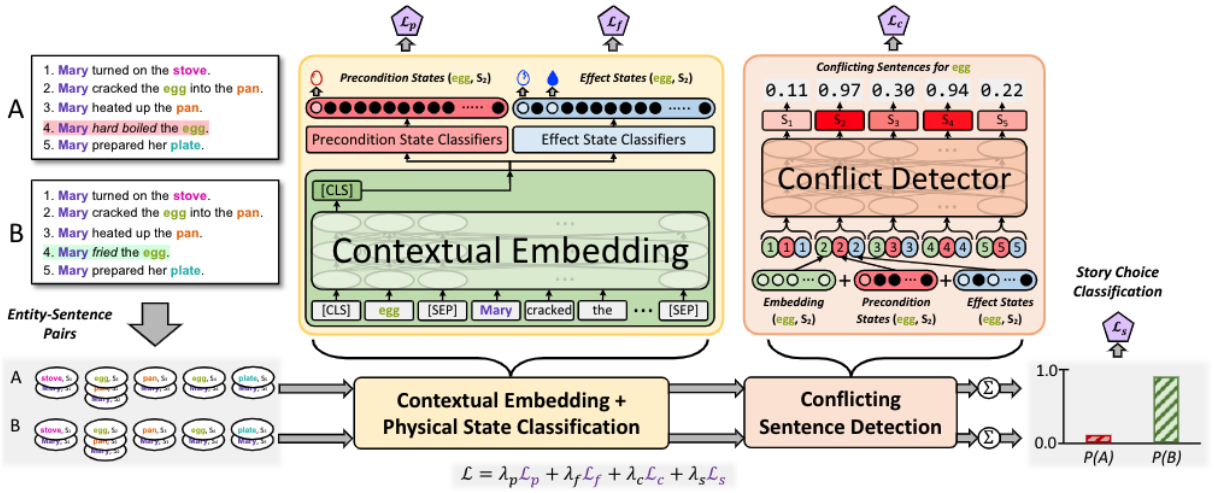


Figure 2: Baseline TRIP architecture (Storks et al., 2021).

SQuAD 2.0, PIQA, and RACE all deal in the domain of textual understanding. PIQA focuses on multiple-choice answering, which requires a level of sequential logic and understanding. RACE is based on English comprehension questions meant for middle school and high school students in China.

We used pretrained models available on Hugging Face for our experiments. We ended up using the following models: sledz08/finetuned-bert-piqa, deepset/roberta-large-squad2, LIAMF-USP/roberta-large-finetuned-race. Implementing the models was as simple as modifying the existing data structures to include them as potential pre-trained models for use. We were careful to pick models that did not change the model backbone, i.e. sticking to BERT, RoBERTa, and DeBERTa models only.

3.2 Contextual Embedding Model Substitution

The baseline model experimented with three different pre-trained language models to calculate the contextual embedding: BERT, RoBERTa, and DeBERTa. These are popular choices for many general purpose natural language processing tasks as they are trained on a large corpus that translates well to most general purpose language tasks.

We propose a model architecture that experiments with two additional language model backbones: GPT-2 and BART. GPT-2 is a model that was trained for the task of text generation; however similar to BERT, this language model learns an inner representation of the English language that is applicable to a wide variety of tasks. Note that GPT-2 is a very large language model so we use both the large version and the distilled version to see how this affects the performance of the models. BART is a model that was also trained for text generation, but works well on comprehension tasks, so we analyze how this generalizes to the TRIP dataset.

We experiment with these two state-of-the-art language transformers in conjunction with the pre-training to evaluate the change in performance compared to the three baseline language models used in the baseline model.

3.3 Multi-Task Learning

Given the hierarchical evaluation strategy of TRIP, we explored two sub-approaches: dynamic weight adjustment and gradient manipulation. In these approaches, we refrained from modifying the hyper-parameters for the best-performing baseline model to illuminate the effects of multi-task learning approaches on model performance.

As alluded to in Sec. 1, we chose this approach as we observed physical state prediction to be a major bottleneck in the baseline model. Of the stories where the story choice was classified correctly, the majority had incorrectly predicted physical states. Seeing as though the the conflict detection is directly affected by the physical state predictions (outputs of the precondition and effect state classifiers are inputs to the Conflict Detector module as shown in Fig. 2), we sought to improve the performance of the model from the bottom-up.

By allowing the models to attend to various levels of the signal over the course of training, we hypothesized that they will be encouraged to learn

consistent and *verifiable* mappings as opposed to those that simply satisfy the end-level task.

3.3.1 Dynamic weight adjustment

In its current formulation, the loss function is a weighted sum of the four task losses

$$\mathcal{L} = \lambda_p \mathcal{L}_p + \lambda_f \mathcal{L}_f + \lambda_c \mathcal{L}_c + \lambda_s \mathcal{L}_s \quad (1)$$

where an optimal, static weight for each of the losses was found manually beforehand (Storks et al., 2021). Though this encourages the model to attend to the task's signal at various levels, it was found to be insufficient to attain high-levels of consistency and verifiability despite relatively high end-task performance. As such, we implemented approaches from MTL, namely dynamic weight adjustment (Song et al., 2020; Nishino et al., 2019) to improve the baseline model's performances on `TRIP`.

Gamma training schedule Following the loss function introduced in (Song et al., 2020), we dynamically adjusted the weights associated with the conflict detection λ_c and story choice classification λ_s. With some abuse of notation, λ_c and λ_s are updated as

$$\lambda_c = \max\left(\min\left(\frac{\mathcal{L}_c}{\mathcal{L}_{p,f}} \cdot \lambda_c, \beta_c\right), 0.01\right), \quad (2)$$

$$\lambda_s = \max\left(\min\left(\frac{\mathcal{L}_s}{\mathcal{L}_{p,f}} \cdot \lambda_s, \beta_s\right), 0.01\right), \quad (3)$$

where $\lambda_c = \lambda_s = 0.1$ to begin and $\lambda_p, \lambda_f = 1$. The suggested values for β_c and β_s were 1, however, we experimented $\beta_c = \beta_s = 3$ following observation of the model's performance on the default value. Note that the task loss weights are updated according to previous epoch's task losses.

This schedule forces the model to focus on optimizing the difficult (lower-level) tasks, which was identified as the bottleneck in the baseline model, prior to the end-level task. For example, when $\mathcal{L}_c$ is large, β_c is the task loss weight. However, as $\mathcal{L}_c$ decreases (faster relative to $\mathcal{L}_p$ or $\mathcal{L}_f$), so does $\frac{\mathcal{L}_c}{\mathcal{L}_{p,f}}$ until it crosses the β_c threshold, at which point λ_c decreases indefinitely, allowing the model to attend to the lower-level losses $\mathcal{L}_p$ and $\mathcal{L}_f$. The same intuition can be applied to $\mathcal{L}_s$.

Sigmoid training schedule We additionally trained and evaluated the model based on the scheduling strategy proposed by (Nishino et al., 2019), which dynamically (and explicitly) adapts the weights of the loss function to focus on easy tasks at the beginning and difficult tasks at the end. The weight λ_i for the p-th epoch is calculated as

$$\lambda_i(p) = \frac{\lambda_i^{\text{const}}}{1 + \exp\big((p_i^{\text{th}} - p)/\alpha\big)}, \quad (4)$$

where λ_i^{const} and p_i^{th} are task-specific hyperparameters and α is the temperature. We experimented with various values for λ_i^{const} and p_i^{th}, which are summarized in Table 1. For all experiments, we set $\alpha = 0.9$.

Schedule	Parmaters (p, f, c, s)	
	λ_i^{const}	p_i^{th}
1	$0.4, 0.4, 0.2, 0.1$	$0.0, 0.0, 5.0, 8.0$
2	$0.4, 0.4, 0.2, 0.1$	$0.0, 2.0, 5.0, 8.0$
3	$0.4, 0.4, 0.2, 0.1$	$2.0, 2.0, 5.0, 8.0$
4	$0.4, 0.4, 0.2, 0.0$	$0.0, 0.0, 2.0, -$
5	$0.4, 0.4, 0.2, 0.0$	$0.0, 0.0, 3.0, -$
6	$0.4, 0.4, 0.2, 0.0$	$0.0, 0.0, 4.0, -$
7	$0.4, 0.4, 0.2, 0.0$	$0.0, 0.0, 5.0, -$
8	$0.4, 0.4, 0.2, 0.0$	$0.0, 0.0, 6.0, -$
9	$0.4, 0.4, 0.2, 0.0$	$0.0, 0.0, 8.0, -$
10	$0.4, 0.4, 0.2, 0.0$	$0.0, 0.0, 10.0, -$

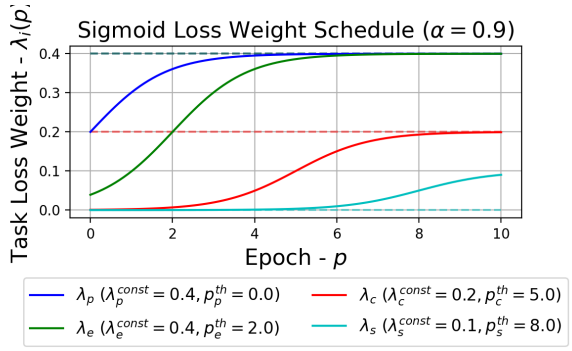
Table 1: Sigmoid task loss weight schedule parameters. The corresponding values are listed in the order precondition, effects, conflict, story. Schedules 4 to 10 do not have a p_s^{th} as $\lambda_s^{\text{const}} = 0$.

We chose to explore the method from (Nishino et al., 2019) in addition to (Song et al., 2020), as it allows for finer-grained control on the schedule of the lower-level tasks as well – a feature not available with (Song et al., 2020).

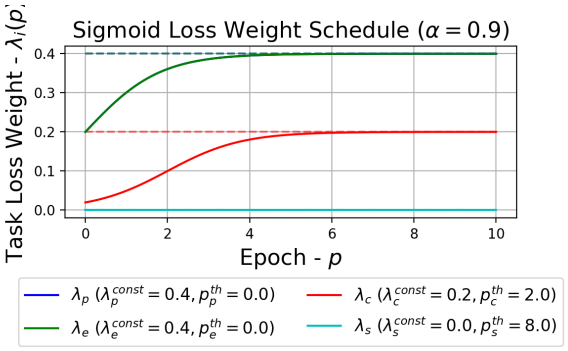
The task loss weights for four schedules are shown in Fig. 3 and shows the evolution of the weights as a function of the epoch: at the beginning of training, the physical state prediction losses are weighted much more heavily compared to the conflict and story prediction loss. We hypothesized that this would encourage the model to first understand the physical states presented in the data, then use that understanding to predict sentence conflicts and classify stories. This is reflected in the progression towards an equal weighting scheme as training progresses.

3.3.2 Gradient manipulation

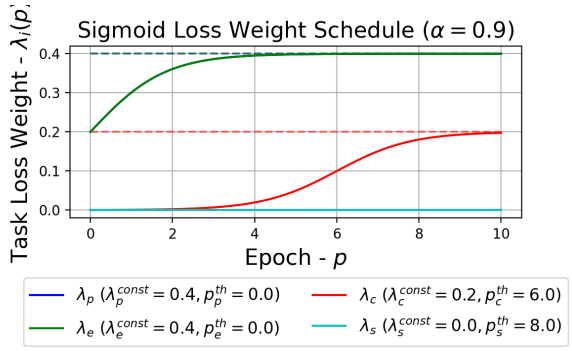
Though we were unable to obtain results using this approach, we elected to write a short description for



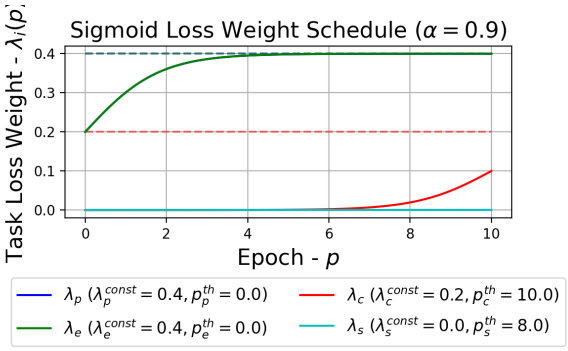
(a) Schedule 2.



(b) Schedule 4.



(c) Schedule 8.



(d) Schedule 10.

Figure 3: Sigmoid task loss weight schedules for Schedules 2, 4, 8, and 10. The weight $\lambda_i(p)$ associated with each task loss term as a function of the epoch p. The dashed lines show the static task loss weights used in the baseline model corresponding to the dynamic task loss weights of the same color. The schedules are visualized as sigmoid functions though they are piecewise linear functions in reality purely for visual purposes. Note that the λ_p and λ_e curves are perfectly overlapping for Schedules 4, 8, and 10.

future exploration. This approach adapts `PCGrad` (Yu et al., 2020) to the setting of three of more tasks, where the gradients for the three lower-level task losses ($\boldsymbol{g}_p, \boldsymbol{g}_e, \boldsymbol{g}_c$) are projected onto the normal of the gradient of the end-level task loss ($\boldsymbol{g}_p$), if their cosine similarity is negative (Eq. 5):

$$\boldsymbol{g}_i' = \begin{cases} \boldsymbol{g}_i - \dfrac{\boldsymbol{g}_i \cdot \boldsymbol{g}_p}{\|\boldsymbol{g}_p\|^2} \boldsymbol{g}_p, & \text{if } \langle \boldsymbol{g}_i, \boldsymbol{g}_p \rangle < 0 \\ \boldsymbol{g}_i, & \text{otherwise} \end{cases} \tag{5}$$

where $\boldsymbol{g}_i$ is the gradient of the loss associated with task i, and $\langle \cdot, \cdot \rangle$ denotes the inner product. Once the modified gradients $\boldsymbol{g}_i'$ are calculated, they are summed and passed to standard optimizers, such as SGD or Adam, to update the weights of the model (Yu et al., 2020). A pictorial representation of `PCGrad` is show in Figure 4.

In our implementation, we chose the gradient of the story prediction loss $\boldsymbol{g}_s$ to be the base task upon which gradient surgery is performed. We hypothesized that this would allow the model to perform well on the more difficult tasks (predicting the conflicting sentences and physical states) whilst maintaining its baseline performance on story choice classification. We used a publicly available PyTorch implementation of `PCGrad`[2] to implement this method.

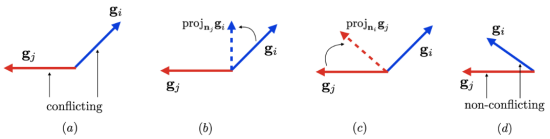


Figure 4: A pictorial representation of `PCGrad` from (Yu et al., 2020). In (a), $\boldsymbol{g}_i$ and $\boldsymbol{g}_j$ are conflicting and can be manipulated as shown in (b) and (c). If $\boldsymbol{g}_i$ and $\boldsymbol{g}_j$ are not conflicting as in (d), they are not manipulated.

Though this scheme falls under MTL, we considered it to be an orthogonal approach to dynamic weight adjustment as it relies on smoothing the

[2] https://github.com/WeiChengTseng/Pytorch-PCGrad

overall learning process instead of prioritizing individual tasks to boost the model's performance.

3.4 Datasets

We plan on using the `TRIP` dataset hosted on Hugging Face[3] for all of our experiments. As mentioned in 3.1, we experimented with PIQA, SQuAD 2.0, and RACE.

4 Results

In this section, we will share the results of our three approaches, namely using models pretrained on different datasets, substituting different model backbones, and augmenting our models with multi-task learning.

All approaches were implemented on top of the authors' original implementation[4] and is publicly available on GitHub[5]. All training and testing were performed on the Great Lakes HPC Cluster [6]. The training and testing of each model took roughly three hours to complete.

4.1 Pre-Training

We show the test set results of using models pretrained on different datasets in Table 2. Unfortunately, it seems like while the results are certainly still quite high, they do not surpass the results achieved by the baseline model in (Storks et al., 2021).

While it should be noted that there is a small amount of variance across runs due to inherent randomness of the models, the relatively superior performance of the baseline model remains consistent.

There are several potential reasons for these results. The first is that perhaps pretraining on the chosen "similar" datasets do not actually yield significant transferrable benefits to TRIP. While this is definitely something to consider, it's difficult to evaluate and also seems unlikely, as (Ma et al., 2022) achieved excellent performance using a RoBERTa-large model pretrained on SQuAD 2.0. A second reason is that the pretrained models were simply not fine-tuned enough. Reusing the ideal hyperparameter configuration for a RoBERTa-large model may not be the ideal configuration for a RoBERTa-large model pretrained on SQuAD 2.0, thus leading to subpar results. Furthermore, the same hyperparameter configuration was used on a BERT model pretrained on PIQA. More hyperparameter tuning is almost certainly necessary to optimize the BERT model when compared to a RoBERTa model. A final reason is the pretrained models used may not be optimally fine-tuned to begin with. The BERT model pretrained on the PIQA dataset in particularly is not extremely popular on Hugging Face, which may or may not indicate quality. The SQuAD and RACE models are much more popular, but it's still unclear how optimally trained they are.

Dataset	Accuracy	Consistency	Verifiability
Baseline	**72.9**	**22.2**	**10.6**
PIQA	70.9	17.4	4.6
SQuAD 2.0	70.1	19.7	5.7
RACE	73.5	21.4	6.3

Table 2: Test set accuracy, classification, and verifiability for models pretrained on different datasets. Models are taken from Hugging Face's public listings. All metrics are in percentages.

4.2 Contextual Embedding Model Substitution

While we were able to achieve better than random accuracy with the GPT-2 and BART models, we were unfortunately unable to outperform the baseline models presented in the original paper. Table 3 shows the test set results for BART, DistilGPT2, and GPT2-Large models.

Schedule	Accuracy	Consistency	Verifiability
Baseline	**72.9**	**22.2**	**9.1**
BART	63.2	8.2	2.1
DistilGPT2	57.3	3.2	0
GPT2 Large	60.3	5.2	1.8

Table 3: Test set accuracy, classification, and verifiability for the BART, DistilGPT2, and GPT2-Large models. All metrics are in percentages.

The results do not outperform the baseline model. A discussion of why we believe this to be the case follows below.

4.3 Multi-Task Learning

Gamma training schedule The accuracy, consistency, and verifiability for the gamma training

[3] https://huggingface.co/datasets/sled-umich/TRIP

[4] https://github.com/sled-group/Verifiable-Coherent-NLU

[5] https://github.com/matsumotosan/Verifiable-Coherent-NLU

[6] https://arc.umich.edu/greatlakes/

schedule on the test set are summarized in Table 4.

Schedule	Accuracy	Consistency	Verifiability
Baseline	72.9	**22.2**	**9.1**
$\gamma(1)$	76.4	8.0	3.1
$\gamma(3)$	**77.8**	7.7	1.7

Table 4: Test set accuracy, classification, and verifiability for gamma schedules. $\gamma(1)$ denotes the gamma schedule with $\beta_s = \beta_c = 1$ and similarly for $\gamma(3)$. All metrics are in percentages.

The gamma-weighted training schedule performed poorly, as it performed roughly equally to the baseline model in terms of accuracy but was worse in consistency and verifiability by several factors. Looking at the values for γ_c and γ_s for the training steps (Fig. 5), we see that both oscillate rapidly over the course of training indicating that $\mathcal{L}_c$ and $\mathcal{L}_s$ failed to stabilize relative to $\mathcal{L}_p$ and $\mathcal{L}_s$. We observed a similar pattern for the case when $\beta_c = \beta_s = 3$.

Sigmoid training schedule The accuracy, consistency, and verifiability for the sigmoid task loss weight schedules on the test set are summarized in Table 5. Overall, the sigmoid schedule yielded much better results compared to the gamma training schedule. Every schedule we tested had higher accuracy than the baseline model and three had higher consistency. Three schedules had the same or higher verifiability compared to baseline. Overall, Schedule 10 had the best metrics, outperforming the baseline model by 10.0%, 11.7%, and 2.9% in accuracy, consistency, and verifiability, respectively.

Overall, models trained under the sigmoid training schedule performed comparatively or better than the baseline model.

Schedule	Accuracy	Consistency	Verifiability
Baseline	72.9	22.2	9.1
1	82.6	13.7	3.4
2	81.8	12.8	6.3
3	81.8	14.2	6.0
4	77.5	17.9	8.5
5	75.5	21.1	6.6
6	79.5	19.1	5.4
7	82.3	28.8	10.0
8	82.9	21.9	8.8
9	81.2	30.8	9.1
10	**82.9**	**33.9**	**12.0**

Table 5: Test set accuracy, classification, and verifiability for sigmoid schedules. All metrics are in percentages.

5 Discussion

5.1 Pre-Training

The biggest implication from our pre-training results is that improving baseline performance is not as simple as using models pretrained on "similar" datasets. More work and research is required in selecting, training, and evaluating models in order to beat existing models' performance. It's very clear now that using models pretrained on different datasets should be treated as an optimization rather than an approach for significant performance gain.

The largest limitation of our results from using models pretrained on different datasets was that minimal hyperparameter tuning was performed. Given the limited time and resources, we tried to focus instead of trying many different models with different datasets rather than optimizing each model we used. Other datasets we investigated included ProPara, ReCAM, ReClor, and AdversarialQA. These datasets did not have existing pretrained models on Hugging Face, and our attempts to create our own pretrained models were unsuccessful in the time frame given. A big factor was that conflicting version requirements and a restrictive development environment on Great Lakes made it extremely difficult to manually train any sort of model. As a result, we only have results from the PIQA, SQuAD 2.0, and RACE datasets.

The most obvious avenue for future work is to perform hyperparameter tuning on each of the mentioned models in hopes that they will yield results that surpass the baseline. Another obvious area for future work is to experiment with more datasets. It is still unclear how models pretrained on ReCAM, ReClor, AdversarialQA, and especially ProPara will perform on TRIP. There are, of course, many other promising datasets to pre-train on as well.

5.2 Contextual Embedding Model Substitution

From our results, it can be seen that changing the contextual embedding models did not result in an increase in performance in comparison with the baseline model.

We believe that this is because, as discussed

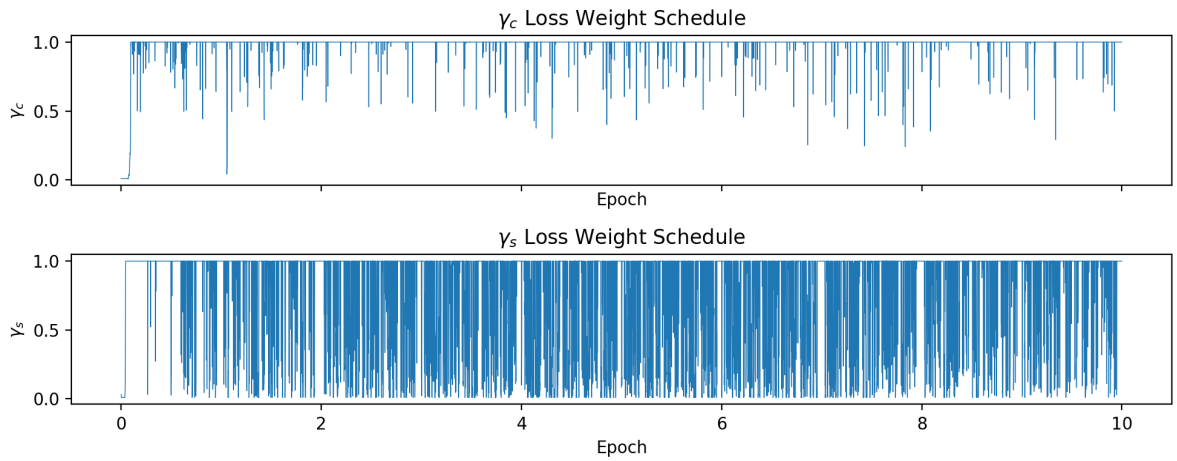


Figure 5: Values of (top) γ_c and (bottom) γ_s over the course of training.

above, these models were originally trained for text generation purposes. While they supposedly have an understanding of the English language, the inherent difference in task training means that these models may not be well–suited for TRIP out-of-the-box. We were mildly surprised by this result as there have been examples that have used these backbones for question answering and commonsense reasoning tasks, which are pretty similar to the task at hand. However, as mentioned in (Shwartz et al., 2020), the model was used in an unsupervised learning framework. In this paper the model asks "information seeking" questions, which is a task that is better suited for a text generation model.

5.3 Multi-Task Learning

In exploring the effect of adapting two MTL methods for TRIP, we found that a strategy as simple as modifying the weights associated with task losses is sufficient to produce significant improvements in performance. Specifically, with the sigmoid training schedule, we found that focusing only on the lower-level tasks for the majority of training, and factoring in consistency in the last few epochs (completely ignoring story choice loss throughout training) yielded the best performance. This is in line with the original findings (Storks et al., 2021), which found that omitting story choice loss yielded the best performance. In retrospect, we hypothesize that choosing to omit story classification loss may be a logical choice given that story choice merely serves as a proxy for the model's conflict detection – backpropagating the story choice loss gradient may have been introducing undue noise into the training of the model. Additionally, our results indicate that learning to correctly predict physical states, as expected, is critical to verifiability and consistency. This is demonstrated by the increase in consistency and verifiability the later consistency is introduced into loss function (Schedules 4 to 10).

The gamma training schedule did not yield as promising results as it dramatically reduced the model's consistency and verifiability. We believed that the weights associated with story choice classification and consistency prediction would decrease over the course of training as it decreased relative to the physical state predictions. However, our results indicate that their relative magnitude continued to oscillate over all epochs. Though we are unsure of why

An obvious next step with the MTL approach is to resolve the runtime errors for gradient surgery. This would greatly diversify the MTL approaches as it could be combined with the dynamic task loss weight adjustment strategies to exploit the advantages provided by each of the approaches, namely the smoothening of the overall learning process and the ability to focus on tasks of varying level over the training process. It is unclear if such a combination would yield better results; however, given the relatively unexplored field of MTL, such an interaction would be valuable given rising interests in the development of models with multiple objectives.

6 Conclusion

While NLP models have achieved increasingly better performance in a variety of tasks, commonsense reasoning remains a challenging task. Being able to provide coherent reasoning for predictions remains

an area where humans triumph over machines. In this project, we attempted to improve upon the baseline performance on a novel benchmark TRIP by incorporating models pretrained on a variety of datasets (which we deemed to be similar to TRIP), substituting the contextual embedding model with other popular language models, and reformulating the proposed objective with strategies from MTL.

We found swapping out the model backbone yielded significantly lower results, despite our best efforts at fine-tuning, and models pretrained on different datasets yielded results similar to the baseline. Ultimately, we found the most success with augmenting the model with MTL as it comfortably surpassed the baseline performance.

References

Yonatan Bisk, Rowan Zellers, Ronan Le bras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: Reasoning about Physical Commonsense in Natural Language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding Comprehension Dataset From Examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kaixin Ma, Filip Ilievski, Jonathan Francis, Eric Nyberg, and Alessandro Oltramari. 2022. Coalescing Global and Local Information for Procedural Text Understanding. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 1534–1545, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Toru Nishino, Shotaro Misawa, Ryuji Kano, Tomoki Taniguchi, Yasuhide Miura, and Tomoko Ohkuma. 2019. Keeping Consistency of Sentence Generation and Document Classification with Multi-Task Learning. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3193–3203, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. Technical report, OpenAI.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Vered Shwartz, Peter West, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Unsupervised Commonsense Question Answering with Self-Talk. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4615–4629, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wei Song, Ziyao Song, Lizhen Liu, and Ruiji Fu. 2020. Hierarchical Multi-task Learning for Organization Evaluation of Argumentative Student Essays. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 3875–3881, California. International Joint Conferences on Artificial Intelligence Organization.

Shane Storks, Qiaozi Gao, Yichi Zhang, and Joyce Chai. 2021. Tiered Reasoning for Intuitive Physics: Toward Verifiable Commonsense Language Understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4902–4918, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient Surgery for Multi-Task Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 5824–5836. Curran Associates, Inc.