Generating Auxiliary Tasks with Reinforcement Learning

Judah Goldfeder*
Department of Computer Science
Columbia University
jag2396@columbia.edu

Matthew So*
Department of Computer Science
Columbia University
ms5513@columbia.edu

Hod Lipson
Department of Mechanical Engineering
Columbia University
hod.lipson@columbia.edu

Abstract

Auxiliary Learning (AL) is a form of multi-task learning in which a model trains on auxiliary tasks to boost performance on a primary objective. While AL has improved generalization across domains such as navigation, image classification, and NLP, it often depends on human-labeled auxiliary tasks that are costly to design and require domain expertise. Meta-learning approaches mitigate this by learning to generate auxiliary tasks, but typically rely on gradient based bi-level optimization, adding substantial computational and implementation overhead. We propose RL-AUX, a reinforcement-learning (RL) framework that dynamically creates auxiliary tasks by assigning auxiliary labels to each training example, rewarding the agent whenever its selections improve the performance on the primary task. We also explore learning per-example weights for the auxiliary loss. On CIFAR-100 grouped into 20 superclasses, our RL method outperforms human-labeled auxiliary tasks and matches the performance of a prominent bi-level optimization baseline. We present similarly strong results on other classification datasets. These results suggest RL is a viable path to generating effective auxiliary tasks.

1 Introduction

Auxiliary Learning (AL) is a technique by which learned or pre-labeled auxiliary tasks are provided as an additional objective to a network during its training with the intended goal of improving the network's performance on a desired primary task. Auxiliary Learning can be thought of as a sub-field of multi-task learning, in which the objective of the training is to improve the main network's performance on the primary task while the auxiliary tasks regularize the training [Caruana, 1997, Ponti, 2021]. It has been demonstrated that the inclusion of auxiliary tasks during training improves generalization and network performance on unseen samples across a large range of domains, including speech recognition, navigation, and image classification [Jaderberg et al., 2016, Goyal et al., 2019, Mirowski et al., 2016, Liebel and Körner, 2018, Toshniwal et al., 2017]. Even small, tangentially related tasks have been shown to provide significant support to the main task [Liebel and Körner, 2018]. The intuition is that using the auxiliary task pushes the network to learn a shared representation of the data that guards against overfitting on the primary task [Liu et al., 2019].

*These authors contributed equally to this work

2nd Workshop on Aligning Reinforcement Learning Experimentalists and Theorists (ARLET 2025).

A historic weakness of Auxiliary Learning has been the need for additional human labeling during the creation of supervised auxiliary tasks. This requires a large amount of human effort and domain expertise for each auxiliary task. Furthermore, in the multi-task context, we know that poor task selection can ultimately harm primary task performance [Gururangan et al., 2020]. Therefore, the manner in which primary and auxiliary tasks should be optimally combined during the weight update procedure can be ambiguous and require expert knowledge.

Meta Auxiliary Learning (MAXL) attempts to alleviate the problem of auxiliary task labeling through the procedural generation of an auxiliary task that optimizes the performance on the given primary task [Liu et al., 2019]. The MAXL framework works by organizing the inputs of the primary task into hierarchical subclasses for each primary class using an additional label generation network. As such, MAXL is one of several approaches for dynamic label generation that run into the Bi-Level Optimization problem [Navon et al., 2020, Chen et al., 2023]. Bi-Level Optimization, which is at the heart of many Meta Learning procedures, arises here as the gradients of the label network are calculated with respect to the performance of the main network on the primary task, resulting in a Hessian-inverse vector calculation and increased implementation complexity.

In this work, we will attempt to bypass the need for this complex Bi-Level Optimization altogether by training a Reinforcement Learning (RL) agent to learn the auxiliary task selection (RL-AUX). Our approach involves encapsulating the main network's training loop (for both the primary and auxiliary tasks) within a Reinforcement Learning environment. We then expose the input data points as the environment's state and the network's performance on the main task as the reward. The environment's action space will enable the RL agent to provide auxiliary labels per data point. An agent trained in this environment should therefore learn the auxiliary task labeling that maximizes the performance of the main network on the primary task. We focus our work on commonly used image classification datasets, primarily CIFAR-100. Our experiment results reveal that the RL approach performs as well as the state-of-the-art MAXL approach with less model customization for the Hessian calculation.

After this initial success with RL labeling of the auxiliary task, we extend the work to explore loss weight assignment per data point. Several techniques have been explored in the literature to optimize auxiliary task weighting but none attempt to dynamically select the auxiliary task labels and also the weights at a sample-level [Kung et al., 2021, Grégoire et al., 2023, Abbas and Tap, 2019, Chennupati et al., 2019].

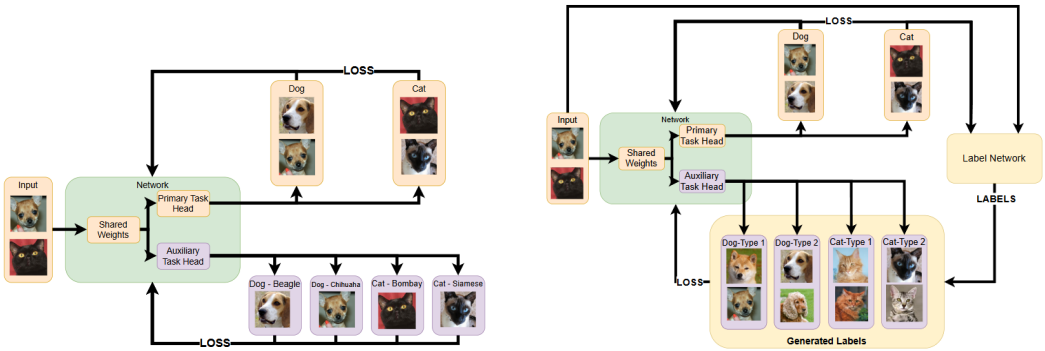
To this end, this paper also presents a weight-aware version of MAXL (WA-MAXL) and a weight-aware version of our Reinforcement Learning architecture (WA-RLAUX). Both are shown to have significant improvement over their statically weighted counterparts while not requiring any additional hyperparameters or inducing a significant scaling cost.

In summary, this paper accomplishes two things that have not yet been presented in the literature to our knowledge. Firstly, we show that Reinforcement Learning can be used for auxiliary task selection while retaining the performance gains of the main network on the primary task. Moreover, it can do this while providing linear scaling with respect to the number of parameters in the main network. Secondly, we demonstrate methods to dynamically learn sample-level loss weights and an auxiliary task at the same time in both the Bi-Level Optimization and Reinforcement Learning approaches. We see significant performance improvements in both domains.

2 Related Work

Auxiliary and Multi-Task Learning in Vision Multi-task Learning (MTL) is a well-studied and widely-used Machine Learning method to have a network learn multiple tasks simultaneously [Caruana, 1997, Liebel and Körner, 2018, Zhang and Yang, 2017]. Auxiliary Learning (AL) is a special case of MTL, in which there is one primary task of importance and one or more auxiliary tasks that support the performance of the main task [Liu et al., 2019]. MTL and AL has been shown to improve target task performance compared to networks that are trained on a single task, particularly in low data contexts [Standley et al., 2019, Zhang and Yang, 2017]. There has been extensive work using Multi-task and Auxiliary Learning in the vision domain, such as the auxiliary classifiers in GoogleLeNet [Szegedy et al., 2014], multi-task cascaded convolutional networks [Zhang et al., 2016], state-of-the-art performance on three vision tasks using one convolutional network [Eigen and Fergus, 2014], and many other examples [Strezoski et al., 2023, Kokkinos, 2017, Rasmus et al., 2015]. Figure

1a represents a common, simple example of an auxiliary task in a classification task setting where the auxiliary task is provided by human labeling.



(a) Classic Auxiliary Task Setup (b) Label Network Auxiliary Task Setup

Figure 1: Sample Auxiliary Task Approaches

Task Weighting The standard loss formulation in Auxiliary Learning is given by:

$$\mathcal{L}_{total}(\theta) = \mathcal{L}_{primary}(\theta) + \sum_{i=1}^{k} \lambda_i \mathcal{L}_{aux}^{(i)}(\theta)$$

Where k is the number of auxiliary tasks, λ_i represents the weight of the task i, and the network is parametrized by θ. The Taskonomy framework provides a methodology to investigate the relationship between candidate auxiliary tasks and map their relationship based on their contribution to target task performance [Zamir et al., 2018]. [Standley et al., 2019] took this a step further and demonstrated that not all possible tasks are helpful for learning. As such, several papers in the field have attempted to learn optimal λ_i loss weights on a per-task basis . [Kendall et al., 2017] introduced the idea of using homoscedastic aleatoric uncertainty to weight known auxiliary tasks. The auxiliary loss in their framework is given by:

$$\mathcal{L}(\theta) = \frac{1}{2\sigma_{primary}^2}\mathcal{L}_{primary}(\theta) + log(\sigma_{primary}) + \sum_{i=1}^{k}[\frac{1}{2\sigma_i^2}\mathcal{L}_{aux}(\theta) + log(\sigma_{aux}^{(i)})]$$

Where the σ values represent task-specific uncertainty noise. [Liebel and Körner, 2018] extended this work but enforced positive regularization values to achieve improved results [Gong et al., 2019]. GradNorm demonstrated strong results by normalizing gradient magnitudes on a per-task basis [Chen et al., 2017] . Task weighting as a Pareto multi-objective optimization was also attempted [Sener and Koltun, 2018].

While the previously mentioned approaches attempt to weight on a per-task basis, SLGrad presents a sample-level weighting approach for known auxiliary tasks that scales each sample on the cosine-similiarity of the sample's loss gradient and the primary task's validation gradient [Grégoire et al., 2023]. Auxilearn also presents a framework to learn sample-specific weights for a known task using Bi-Level Optimization with implicit differentiation [Navon et al., 2020].

Task Generation MAXL started the label network paradigm for dynamic task generation in the Auxiliary Learning space using Bi-Level Optimization [Liu et al., 2019]. The MAXL framework trains a label network to create an auxiliary task that optimizes the main network's performance on the target primary task. Figure 1b shows how the setup of a label network is used to generate the auxiliary task. Bi-Level Optimization approaches attempt to find the optimal primary network weights θ^* and auxiliary labeling network weights ϕ^* that satisfy:

$$\phi^* = \arg\min_{\phi} \mathcal{L}_{aux}(\theta^*(\phi)) \text{ s.t. } \theta^*(\phi) = \arg\min_{\theta} \mathcal{L}_{primary}(\theta, \phi)$$

The gradient update of the label network is given by:

$$\nabla_\phi \mathcal{L}_{aux}(\theta^*(\phi)) = -\nabla_\theta \mathcal{L}_{aux} \cdot (\nabla^2_\theta \mathcal{L}_{primary})^{-1} \cdot \nabla_\phi \nabla_\theta \mathcal{L}_{primary}$$

Auxilearn, another approach using Bi-Level Optimization, attempted to use Neumann approximation to optimize this complex update [Navon et al., 2020]. Other Bi-Level Optimization approaches in task creation involve generating features/samples on the fly and finding useful "questions" as general value functions [Veeriah et al., 2019, Chen et al., 2023].

There have been several attempts at task generation without Bi-Level Optimization. These approaches involve selecting task objectives from a predefined or procedurally generated pool of tasks. Approaches include using Beta-Bernoulli multi-armed bandit framing [Guo et al., 2019], search over unified taxonomy [Dery et al., 2023], and trial-and-error search over generated features [Rafiee et al., 2023].

Combining Auxiliary Learning and Reinforcement Learning The majority of work combining Reinforcement Learning with Auxiliary Learning involves constructing auxiliary tasks for RL agents to perform better on their assigned task [Jaderberg et al., 2016, Riedmiller et al., 2018, Narvekar et al., 2017].

There has been limited work in using Reinforcement Learning to improve the performance of networks through Multi-task and Auxiliary Learning. AutoAugment attempts to use RL to find optimal data augmentation strategies during training to improve a network's classification performance [Cubuk et al., 2018]. In [Fan et al., 2018], the researchers experiment using an RL teacher to select which data points from the training set should be used to train a main network.

3 Problem Formulation

This work focuses on using Reinforcement Learning to design auxiliary tasks for image classification. Let our dataset be represented as $D = \{(x_i, y_i)\}_{i=1}^N, x \in X, y \in Y$. We construct a policy $\pi_\phi : S \to A$, parameterized by ϕ, that maps a state $s \in S$ to an action $a \to A$. In practice, the state is an input value of the dataset, i.e. $S = X$, and the action space defines another classification problem. We will train a main network f, parameterized by θ, that has two output heads $f^\theta_{primary} : X \to Y$ and $f^\theta_{aux} : X \to A$. We will train the network with the following loss:

$$\mathcal{L}_{total}(\theta) = \mathcal{L}_{primary}(\theta) + \lambda \mathcal{L}_{aux}(\theta) \tag{1}$$

Where λ is the weight of the auxiliary task. We can pose the total loss per-sample as:

$$\ell_{total}(x_i, y_i) = \ell_{primary}(f^\theta_{primary}(x_i), y_i) + \lambda \ell_{aux}(f^\theta_{aux}(x_i), \pi_\phi(x_i)) \tag{2}$$

We define $l_{primary}$ and l_{aux} separately, as the primary and auxiliary task do not need to use the same metric in our framework.

Reinforcement Learning Approach The policy π_ϕ will be iteratively updated through an RL approach. To do this, we encapsulate the learning of the main network within an environment. Figure 2 provides an overview of how the training is set up.

The policy π_ϕ provides a single auxiliary label for one data point in each step. A reward is provided to the policy after it labels B_T samples, which is the training batch size for the main network. To calculate the reward, the environment trains the main network on the batch that the policy has labeled. The environment trains the network on both the primary and auxiliary tasks using the loss given by Equation 2. Then, a randomized evaluation batch of size B_R is selected from the training sample. The reward provided to the policy will be the negative loss of the main network on the primary task over the evaluation batch with an added batch-wise entropy loss factor. The entropy loss factor $\mathcal{H}(\pi_\theta(x)_{(b)})$ per evaluation batch b is given by:

$$\mathcal{H}(\pi_\theta(x)_{(b)}) = \sum_{k=1}^{K} \pi_\theta(x)^k_{(b)} \log \pi_\theta(x)^k_{(b)}, \pi_\theta(x)^k_{(b)} = \frac{1}{B_R} \sum_{n=1}^{B_R} \pi_\theta(x)^k_{(b)}[n] \tag{3}$$

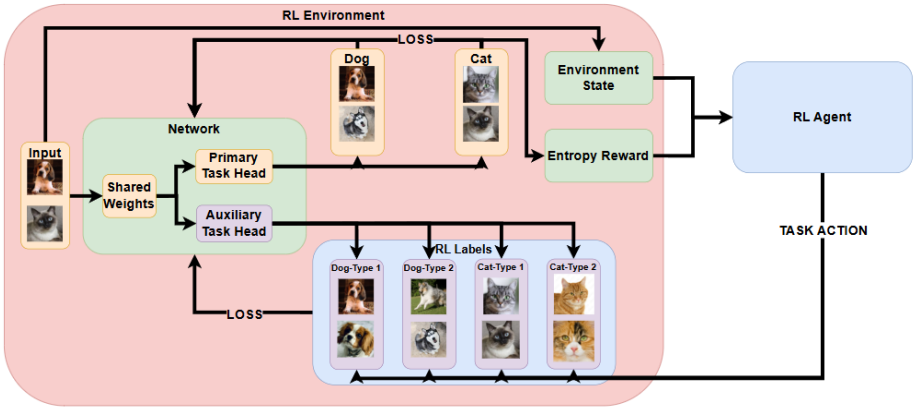


Figure 2: Reinforcement Learning Auxiliary Task Setup

Where K is the number of auxiliary classes. This idea is inspired by [Liu et al., 2019], and is useful to guard against the degenerate case where each primary task class has one and only one auxiliary class.

The total reward is therefore constructed as:

$$Reward = \frac{-1}{B_R} \sum_{i=1}^{B_R} l_{primary}(f^{\theta}_{primary}(x_{(b),i}), y_{(b),i}) + \mathcal{H}(\pi_\theta(x)_{(b)}) \tag{4}$$

Almost any RL algorithm/architecture can be used to define and update π_θ. In practice, we used a customized Proximal Policy Optimization approach [Schulman et al., 2017].

Label Hierarchy Optimization MAXL employs hierarchy-constrained label generation through the use of a Masked Soft-max [Liu et al., 2019]. In other words, each class in the primary task will have some fixed number of subclasses in the auxiliary task.

If z are the logits generated from an input (x_i, y_i) and y_i represents the integer index of the primary label, then the following gives the probability for auxiliary label k:

$$p_k = \frac{\exp(z_k) m_k}{\sum_{j=1}^{K} \exp(z_j) m_j} \tag{5}$$

$$m_k = \begin{cases} 1 \text{ if } \psi \cdot y_i \leq k < \psi \cdot (y_i + 1) \\ \text{else } 0 \end{cases} \tag{6}$$

Where K is the number of auxiliary classes, z represents raw logits, and $m \in \{0, 1\}^K$ acts as the mask. MAXL employs a hierarchy factor ψ that dictates how many subclasses a primary task class will have in the auxiliary task.

Focal loss is used after the Masked Soft-max to promote the use of the entire auxiliary label space. We have adopted this Masked Soft-max focal loss calculation approach for the Reinforcement Learning setup.

Weight-Adjusted Approaches After initial successes with the RL task generation approach mentioned, we turned our attention to learning sample-level weight adjustment in both the Bi-Level Optimization and RL approaches. Our per-sample loss from Equation 2 is adjusted slightly, so that the weighting is sample-specific:

$$\ell_{total}(x_i, y_i) = \ell_{primary}(f^{\theta}_{primary}(x_i), y_i) + \lambda_i \ell_{aux}(f^{\theta}_{aux}(x_i), \pi_\phi(x_i)) \tag{7}$$

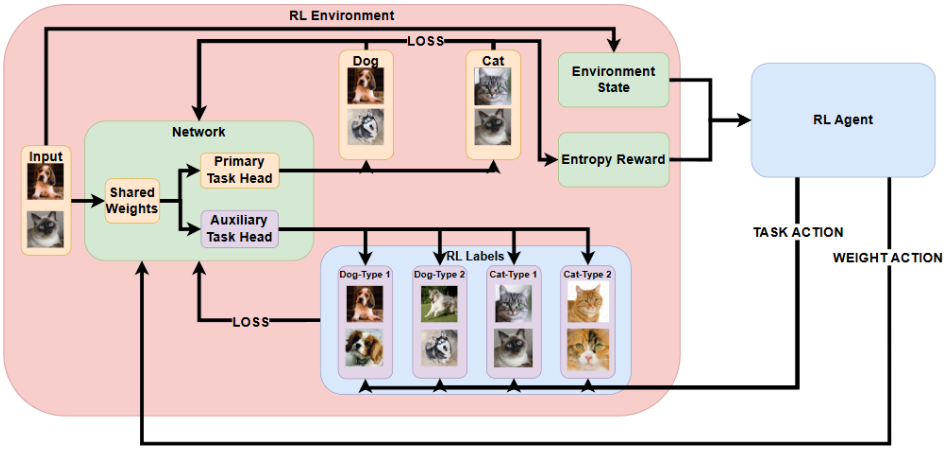


Figure 3: Weight-Aware Reinforcement Learning Auxiliary Task Setup

For the Reinforcement Learning approach, we added another component to the action space, to enable the agent to select a weight for the given data point. The Weight-Adjusted Reinforcement Learning architecture is presented in Figure 3.

For the Bi-Level Optimization approach, we took MAXL's open-source implementation and added a weight-output head to the labeling network such that it also outputs a weight parameter for a given sample.

4 Methodology

RL Environment Implementation We implemented the main network environment (see Figure 2) using OpenAI's Gymnasium framework [Brockman et al., 2016]. Our custom environment is configured to be agnostic of the primary task, type of main network, type of RL agent, system hardware, optimizer/scheduler, etc. These parameters can be set during instantiation, making our environment a robust platform to prototype Auxiliary Task training on different architectures and datasets rapidly.

Agent Interaction Our environment is written to operate with an observation space that provides a single data point (image). The action it expects back is the RL agent's labeling of the last provided image. Therefore, on start-up, the environment samples a single batch from the training dataloader and provides a single data point (image) from the batch to the RL agent as the observation. The other data points in the batch are held for use as future observations. The RL agent then selects the action (label) for the observation and provides it to the environment.

This continues back-and-forth until the environment has stored labels for each data point in the training batch (of size B_T). Once this happens, the main network is trained on that batch for both the primary and auxiliary tasks simultaneously using Equation 2. Then an evaluation batch (of size B_R) is sampled, and the reward is calculated as in Equation 4 and provided to the agent. A single episode consists of every single image in the training set being seen, labeled, and trained on once. An episode is equivalent to one epoch of training data.

Training Modes In addition to the standard OpenAI Gymnasium interface, our custom environment has additional features to support our unique training. This includes training evaluation functions, main network persisting, metric capturing, and so on.

Most importantly, our environment has been implemented to support two modes of training. They are:

- **Train RL Agent Mode** - The agent is sampled as per their training algorithm. Throughout a training epoch, the weights of the RL agent and main network are both updated. The agent

is pushed towards learning the policy that most improves the main network's performance on the primary task. At the end of the epoch, the main model is reverted to its starting state.

- **Train Main Network Mode** - The main model is trained with a static version of the auxiliary task. This is done by deterministically sampling the RL agent's policy for each image. Then the main model is trained on both tasks. As an optimization, the reward calculation is skipped as the agent is not being updated in this mode. At the end of the epoch, the main model is saved (and becomes the new "canonical model" for future RL Agent training).

Weight Selection The current version of Stable-Baselines3's $MultiInputPolicy$ does not support combining discrete label actions ($MultiDiscrete$) and continuous scalar actions (Box) simultaneously [Raffin et al., 2021]. As such, when implementing the weight-aware version of the RL approach, we treat weight selection in the action space as a 21-class classification problem. Each class represented an evenly spaced value between 0 and 1, as in $\{0, 0.05, 0.1, 0.15, ..., 0.95, 1\}$. The selected unscaled weight w_u, is then used to get the true selected weight as:

$$2^{10w_u - 5} \tag{8}$$

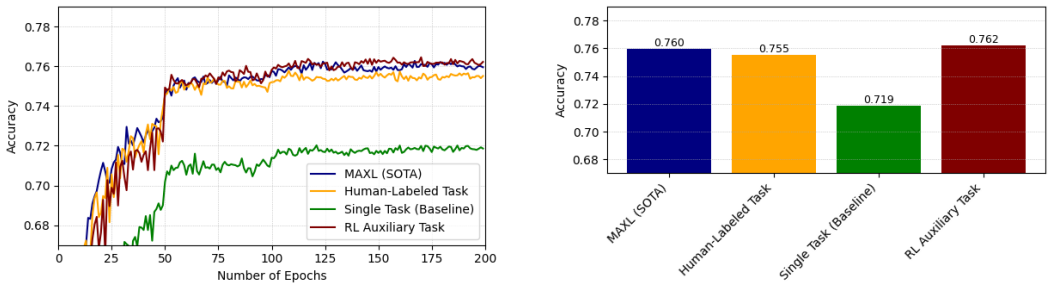
This gives us a range of $[2^{-5}, 2^5]$ that the agent can select from. The weight-aware environment asks the agent to provide both a selection for the auxiliary task labeling and the weighting of the sample.

In the Weight-Aware MAXL implementation, the label network outputs a scalar value $[0, 1]$ and is scaled the same as in Equation 8.

Datasets Cifar-10 and Cifar-100 have become ubiquitous benchmark datasets in the Auxiliary Learning space, both of which contain 60,000 images spread across 10 and 100 classes, respectively [Grégoire et al., 2023, Chen et al., 2023, Fan et al., 2018, Cubuk et al., 2018, Liu et al., 2019, Navon et al., 2020, Krizhevsky, 2009]. Cifar-100 provides a 20 superclass hierarchy. Each of the 100 classes in the dataset is mapped into one of the 20 Superclasses such that each superclass is composed of 5 of the 100 classes [Krizhevsky, 2009]. Table 1 in the Appendix shows how the Superclasses are constructed. This is a very useful human-labeled auxiliary task benchmark, against which we can compare our proposed methodology. SVHN (Street View House Numbers) is also a prominent test dataset in the literature [Netzer et al., 2011]. It is composed of 600,000 real-world images of address digits with 10 classes (0-9). Our experiments will explore how the main network performance changes on these datasets as we introduce RL-created auxiliary tasks. For further training details, see Appendix B. For network details, see Appendix C.

5 Results and Analysis

Reinforcement Learning vs MAXL To judge the performance benefits of our RL-based Auxiliary Task generation approach, we compare it to: (1) a network trained with a MAXL-generated Auxiliary Task, (2) a Human-Labeled Auxiliary Task (see Section 4), and (3) a network trained without an auxiliary task. We will test these approaches on the CIFAR-100 20-Superclass dataset. In this experiment, the auxiliary tasks being tested are equally weighted to the primary task.



(a) Test Accuracy over Epochs Plot

(b) Test Accuracy Bar Graph

Figure 4: Cifar100 20-Superclass Performance of Auxiliary Task Methods

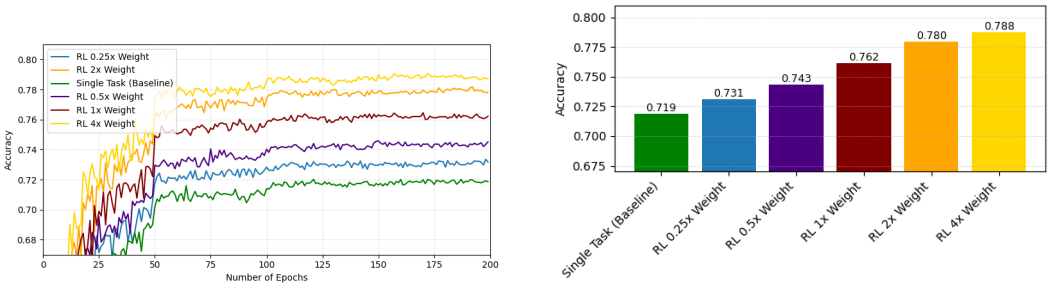
Figure 4 demonstrates that our RL-based auxiliary task generation approach (76.186%) performs just as well (if not slightly better) than MAXL's Bi-Level Optimization approach (75.983%). Both

perform better than the human-labeled auxiliary task (75.525%). All of these three are significantly better than the single-task baseline (71.864%).

In Section 5, we will report results of the RL-based approach on additional datasets.

It should be noted that all trainings were redone for this paper. The values reported here are consistent with what the MAXL paper reported.

Weight Adjustment Ablation Next we wanted to observe how tuning the task-level weight hyperparameter λ affects the performance of the main network on the primary task. To do this, we retrained the RL Agent and the main network on the Cifar100 20-Superclass task with $\lambda = 0.25, 0.5, 1, 2, 4$ and observed the results.



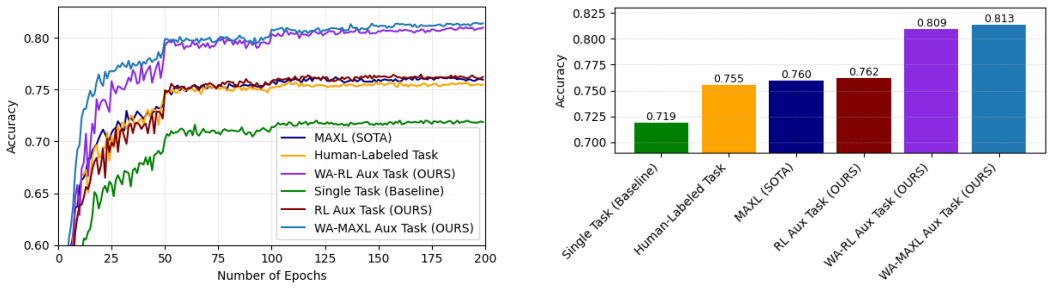
(a) Test Accuracy over Epochs Plot

(b) Test Accuracy Bar Graph

Figure 5: Cifar100 20-Superclass Auxiliary Task Weight Adjustment Ablation

Figure 5 shows that the performance of the main network on the primary task is sensitive to the weighting of the auxiliary task in the loss. Inspired by the clarity of the results from this ablation study, we turned our attention to methods that could learn sample-level weighting and auxiliary task labels jointly. We would like to be able to find an ideal weighting setup without an expensive hyperparameter search. This is what gave rise to our Weight-Aware MAXL and Weight-Aware RL approaches.

Weight-Aware Approach Experiment Now we wish to test the performance of our weight-aware approaches on the 20-Superclass CIFAR100 task. As is clear in Figure 6, our Weight-Aware MAXL (WA-MAXL) and Weight-Aware RL approach (WA-RL) provide significant performance improvement over the other methods. On the 20-Superclass Cifar100 task, WA-MAXL is the highest performer with 81.3% accuracy, followed closely by WA-RL with 80.9% accuracy. The non-weight-aware approaches hover in the 76% range, while the single task network is just below 72%



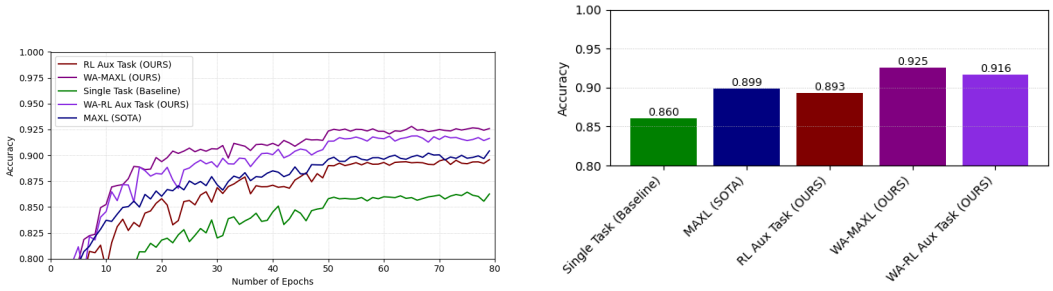
(a) Test Accuracy over Epochs Plot

(b) Test Accuracy Bar Graph

Figure 6: Cifar100 20-Superclass Weight-Aware Experiment Results

We also performed experiments on the CIFAR10 dataset over 80 epochs. Figure 7 shows that while weight-based approaches perform better, the performance gain is not quite as drastic.

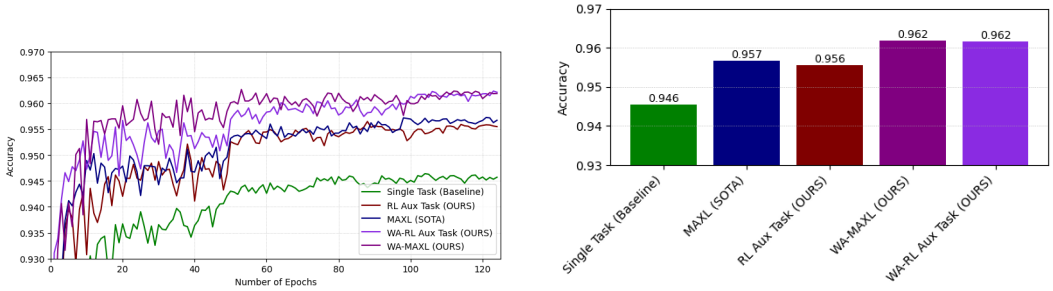
Lastly, we test our approaches on the SVHN dataset over 125 epochs. As we can see in Figure 8, once again, the weight-aware approaches perform best. Meanwhile, the regular RL and MAXL approaches



(a) Test Accuracy over Epochs Plot (b) Test Accuracy Bar Graph

Figure 7: Cifar10 Weight-Aware Experiment Results

still provide performance improvements over the baseline. There is a total improvement of about 1.5%.



(a) Test Accuracy over Epochs Plot (b) Test Accuracy Bar Graph

Figure 8: SVHN Weight-Aware Experiment Results

In summary, these experiments clearly demonstrate our framework's ability to improve a main network's performance on a target primary task.

6 Conclusion

This work demonstrates that an RL agent can be used for auxiliary task generation without requiring the cost or complexity of Bi-level Optimization. We see that our RL-based approach outperforms a human-labeled auxiliary task and performs comparably to MAXL, one of the most prominent approaches in the space, across several benchmarks. We also demonstrate that sample-level auxiliary weights can be learned simultaneously with the auxiliary task selection. Both the Weight-Aware MAXL and RL approaches perform significantly better than their Weight-Unaware counterparts without requiring any additional hyperparameters or scaling cost. We hope to continue exploring the efficacy of RL in the Auxiliary Learning space in the future.

References

Waseem Abbas and Murtaza Tap. Adaptively weighted multi-task learning using inverse validation loss. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1408–1412, 2019. doi: 10.1109/ICASSP.2019.8683776.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016. URL http://arxiv.org/abs/1606.01540.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, Jul 1997. ISSN 1573-0565. doi: 10.1023/A:1007379606734. URL https://doi.org/10.1023/A:1007379606734.

Hong Chen, Xin Wang, Yuwei Zhou, Yijian Qin, Chaoyu Guan, and Wenwu Zhu. Joint data-task generation for auxiliary learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 13103–13114. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/2a91fb5a4c03e0b6d889e1c52f775480-Paper-Conference.pdf.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *CoRR*, abs/1711.02257, 2017. URL http://arxiv.org/abs/1711.02257.

Sumanth Chennupati, Ganesh Sistu, Senthil Kumar Yogamani, and Samir A. Rawashdeh. Multinet++: Multi-stream feature aggregation and geometric loss strategy for multi-task learning. *CoRR*, abs/1904.08492, 2019. URL http://arxiv.org/abs/1904.08492.

Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *CoRR*, abs/1805.09501, 2018. URL http://arxiv.org/abs/1805.09501.

Lucio M. Dery, Paul Michel, Mikhail Khodak, Graham Neubig, and Ameet Talwalkar. AANG : Automating auxiliary learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vtVDI3w_BLL.

David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *CoRR*, abs/1411.4734, 2014. URL http://arxiv.org/abs/1411.4734.

Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Learning to teach. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=HJewuJWCZ.

Ting Gong, Tyler Lee, Cory Stephenson, Venkata Renduchintala, Suchismita Padhy, Anthony Ndirango, Gokce Keskin, and Oguz H. Elibol. A comparison of loss weighting strategies for multi task learning in deep neural networks. *IEEE Access*, 7:141627–141632, 2019. doi: 10.1109/ACCESS.2019.2943604.

Priya Goyal, Dhruv Mahajan, Abhinav Gupta, and Ishan Misra. Scaling and benchmarking self-supervised visual representation learning. *CoRR*, abs/1905.01235, 2019. URL http://arxiv.org/abs/1905.01235.

Emilie Grégoire, Hafeez Chaudhary, and Sam Verboven. Sample-level weighting for multi-task learning with auxiliary tasks, 2023. URL https://arxiv.org/abs/2306.04519.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. Autosem: Automatic task selection and mixing in multi-task learning. *CoRR*, abs/1904.04153, 2019. URL http://arxiv.org/abs/1904.04153.

Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. *CoRR*, abs/2004.10964, 2020. URL https://arxiv.org/abs/2004.10964.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *CoRR*, abs/1611.05397, 2016. URL http://arxiv.org/abs/1611.05397.

Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. *CoRR*, abs/1705.07115, 2017. URL http://arxiv.org/abs/1705.07115.

Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5454–5463, 2017. doi: 10.1109/CVPR.2017.579.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009. URL https://api.semanticscholar.org/CorpusID:18268744.

Po-Nien Kung, Sheng-Siang Yin, Yi-Cheng Chen, Tse-Hsuan Yang, and Yun-Nung Chen. Efficient multi-task auxiliary learning: Selecting auxiliary data by feature similarity. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 416–428, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.34. URL https://aclanthology.org/2021.emnlp-main.34/.

Lukas Liebel and Marco Körner. Auxiliary tasks in multi-task learning. *CoRR*, abs/1805.06334, 2018. URL http://arxiv.org/abs/1805.06334.

Shikun Liu, Andrew J. Davison, and Edward Johns. Self-supervised generalisation with meta auxiliary learning. *CoRR*, abs/1901.08933, 2019. URL http://arxiv.org/abs/1901.08933.

Shuying Liu and Weihong Deng. Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, pages 730–734, 2015. doi: 10.1109/ACPR.2015.7486599.

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J. Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to navigate in complex environments. *CoRR*, abs/1611.03673, 2016. URL http://arxiv.org/abs/1611.03673.

Sanmit Narvekar, Jivko Sinapov, and Peter Stone. Autonomous task sequencing for customized curriculum design in reinforcement learning. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2536–2542, 2017. doi: 10.24963/ijcai.2017/353. URL https://doi.org/10.24963/ijcai.2017/353.

Aviv Navon, Idan Achituve, Haggai Maron, Gal Chechik, and Ethan Fetaya. Auxiliary learning by implicit differentiation. *CoRR*, abs/2007.02693, 2020. URL https://arxiv.org/abs/2007.02693.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Andrea Ponti. Multi-task learning on networks. *CoRR*, abs/2112.04891, 2021. URL https://arxiv.org/abs/2112.04891.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Banafsheh Rafiee, Sina Ghiassian, Jun Jin, Richard Sutton, Jun Luo, and Adam White. Auxiliary task discovery through generate-and-test. In Sarath Chandar, Razvan Pascanu, Hanie Sedghi, and Doina Precup, editors, *Proceedings of The 2nd Conference on Lifelong Learning Agents*, volume 232 of *Proceedings of Machine Learning Research*, pages 703–714. PMLR, 22–25 Aug 2023. URL https://proceedings.mlr.press/v232/rafiee23a.html.

Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund, and Tapani Raiko. Semi-supervised learning with ladder network. *CoRR*, abs/1507.02672, 2015. URL http://arxiv.org/abs/1507.02672.

Martin Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom van de Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. Learning by playing solving sparse reward tasks from scratch. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4344–4353. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/riedmiller18a.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://arxiv.org/abs/1707.06347.

Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. *CoRR*, abs/1810.04650, 2018. URL http://arxiv.org/abs/1810.04650.

Aviv Shamsian, Aviv Navon, Neta Glazer, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. Auxiliary learning as an asymmetric bargaining game, 2023. URL https://arxiv.org/abs/2301.13501.

Trevor Standley, Amir Zamir, Dawn Chen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? *CoRR*, abs/1905.07553, 2019. URL http://arxiv.org/abs/1905.07553.

Gjorgji Strezoski, Nanne van Noord, and Marcel Worring. Matte: Multi-task multi-scale attention. *Computer Vision and Image Understanding*, 228:103622, 2023. ISSN 1077-3142. doi: https://doi.org/10.1016/j.cviu.2023.103622. URL https://www.sciencedirect.com/science/article/pii/S1077314223000024.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL http://arxiv.org/abs/1409.4842.

Shubham Toshniwal, Hao Tang, Liang Lu, and Karen Livescu. Multitask learning with low-level auxiliary tasks for encoder-decoder based speech recognition. *CoRR*, abs/1704.01631, 2017. URL http://arxiv.org/abs/1704.01631.

Vivek Veeriah, Matteo Hessel, Zhongwen Xu, Richard L. Lewis, Janarthanan Rajendran, Junhyuk Oh, Hado van Hasselt, David Silver, and Satinder Singh. Discovery of useful questions as auxiliary tasks. *CoRR*, abs/1909.04607, 2019. URL http://arxiv.org/abs/1909.04607.

Amir R. Zamir, Alexander Sax, William B. Shen, Leonidas J. Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. *CoRR*, abs/1804.08328, 2018. URL http://arxiv.org/abs/1804.08328.

Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multi-task cascaded convolutional networks. *CoRR*, abs/1604.02878, 2016. URL http://arxiv.org/abs/1604.02878.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *CoRR*, abs/1707.08114, 2017. URL http://arxiv.org/abs/1707.08114.

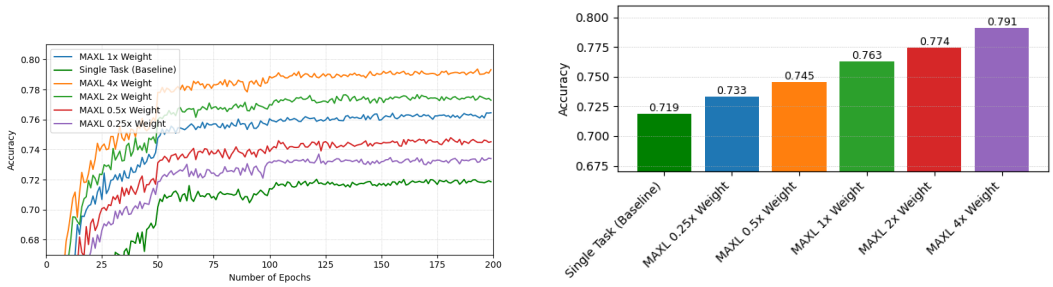
A Appendix / supplemental material

Superclass	Classes
Aquatic mammals	beaver, dolphin, otter, seal, whale
Fish	aquarium fish, flatfish, ray, shark, trout
Flowers	orchids, poppies, roses, sunflowers, tulips
Food containers	bottles, bowls, cans, cups, plates
Fruit and vegetables	apples, mushrooms, oranges, pears, sweet peppers
Household electrical devices	clock, computer keyboard, lamp, telephone, television
Household furniture	bed, chair, couch, table, wardrobe
Insects	bee, beetle, butterfly, caterpillar, cockroach
Large carnivores	bear, leopard, lion, tiger, wolf
Large man-made outdoor things	bridge, castle, house, road, skyscraper
Large natural outdoor scenes	cloud, forest, mountain, plain, sea
Large omnivores and herbivores	camel, cattle, chimpanzee, elephant, kangaroo
Medium-sized mammals	fox, porcupine, possum, raccoon, skunk
Non-insect invertebrates	crab, lobster, snail, spider, worm
People	baby, boy, girl, man, woman
Reptiles	crocodile, dinosaur, lizard, snake, turtle
Small mammals	hamster, mouse, rabbit, shrew, squirrel
Trees	maple, oak, palm, pine, willow
Vehicles 1	bicycle, bus, motorcycle, pickup truck, train
Vehicles 2	lawn-mower, rocket, streetcar, tank, tractor

Table 1: Cifar-100 20 Superclass and Corresponding Single Classes

A.1 MAXL Weight Ablation Study

Similarly to how we conducted the ablation study to see the effect of changing the auxiliary task weight λ on the RL setup, we conducted the same experiment for MAXL.



(a) Test Accuracy over Epochs Plot

(b) Test Accuracy Bar Graph

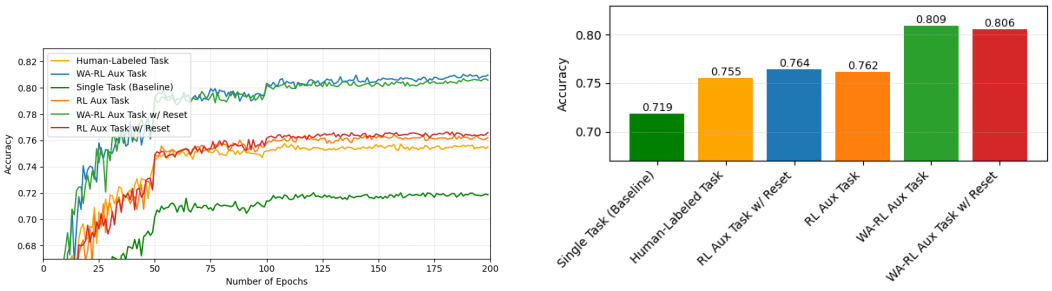
Figure 9: Cifar100 20-Superclass Auxiliary Task Weight Adjustment Ablation on MAXL

Figure 9 shows a similar impact as in the RL case.

A.2 Impact of Reset Ablation

Instead of resetting the RL environment at the end of every epoch, we reset it at the end of every batch during the RL Agent training portion of the cycle. The main difference here is that over the course of one training epoch for the RL Agent, the main weights are constantly reset back to the canonical model.

The results in Figure 12 show that resetting the environment per batch does not yield different outcomes.

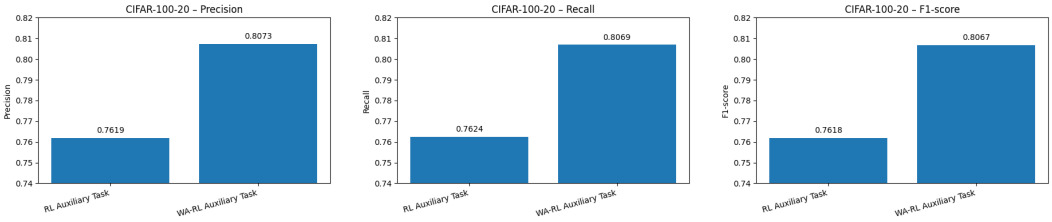


(a) Test Accuracy over Epochs Plot (b) Test Accuracy Bar Graph

Figure 10: Cifar100 20-Superclass Auxiliary Task Environment Reset Ablation

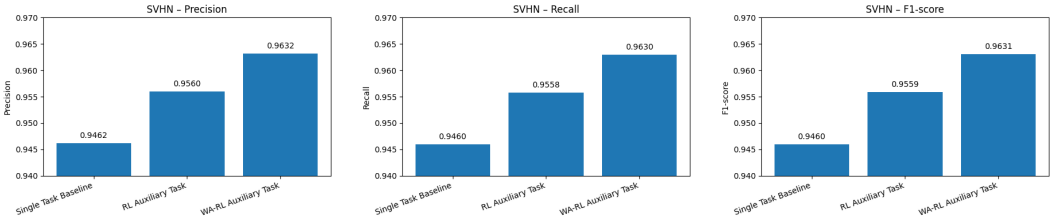
A.3 Precision, Recall and F1 Score

Accuracy is the main metric captured in the Auxiliary Learning literature. Most papers do not reference Precision, Recall, and F1 score in their analysis. For the sake of completeness, we have included some comparisons of these metrics. We include the Precision, Recall and F1 score for the RL approach on the 20-Superclass CIFAR100 problem. We also include comparisons of the baseline, the regular RL approach and the Weight-Aware RL approach on the SVHN dataset.



(a) Precision Comparison (b) Recall Comparison (c) F1 Score Comparison

Figure 11: 20-Superclass CIFAR100 Precision, Recall and F1 Score Comparison



(a) SVHN Precision Comparison (b) SVHN Recall Comparison (c) SVHN F1 Score Comparison

Figure 12: SVHN Precision, Recall and F1 Score Comparison

B Training

The training procedure was standardized in all experiments, unless otherwise noted. The training strategy involved alternating the training between one epoch of RL Agent training followed by one epoch of main network training. The two supported training modes mentioned in Section 4 facilitated this alternating training strategy. The environment was reset (dataloader was reshuffled, canonical main network was loaded, global state cleared, etc.) between batches. Figure 13 visualizes this workflow.

SGD with a learning rate scheduler was used as the optimizer to train the main network. The standard transformations (cropping, normalization, etc.) for each dataset were applied. Early stopping

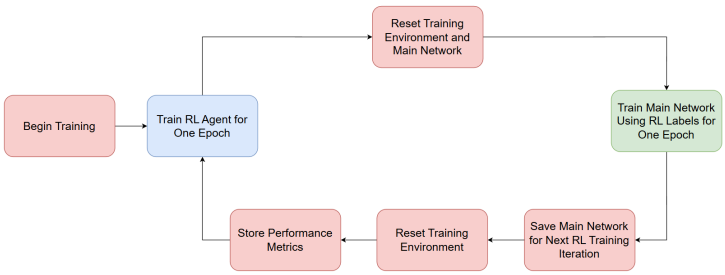


Figure 13: Alternating Training Approach

was used to obtain performance metrics. The following are the default hyperparameters that were used in all training, unless otherwise stated: Training Batch Size(B_T)=100, Evaluation Batch Size(B_R)=256,Feature Extraction Dimensions=256, Number of Epochs=200, Primary Learning Rate=0.01, PPO Learning Rate=0.0003, PPO Entropy Coefficient=0.01, Scheduler Step Epochs=50, Scheduler Gamma=0.5, Auxiliary Task Weight(λ)=1, Hierarchy Factor(ψ)=5.

The training was conducted on several 2080Ti GPUs simultaneously. Each experiment took between 12-36 hours to run. About 800-900 GPU hours were required to conduct this research.

C Networks

C.1 Main Network

As it is widely used in the Auxiliary Learning literature, we focus our experiments on a custom implementation of the VGG16 architecture [Liu and Deng, 2015, Navon et al., 2020, Liu et al., 2019, Chen et al., 2017, Shamsian et al., 2023]. The primary difference is that there are two classifier heads (for the primary and auxiliary tasks). Each head is comprised of two linear layers. The first projects the features from the final convolutional layer to a 512-dimensional representation, and the second maps this representation to the respective task's output dimension.

C.2 Agent Network

We used a customized Proximal Policy Optimization (PPO) Algorithm to learn the auxiliary task provided to the main network [Schulman et al., 2017]. PPO is an on-policy method that was created by OpenAI. It attempts to prevent large policy changes by optimizing:

$$L^{\text{CLIP}}(\theta) = \mathbb{E}_t\Big[\min\big(r_t(\theta)A_t,\ \text{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)\ A_t\big)\Big], \tag{9}$$

where

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)} \tag{10}$$

and A_t denotes the advantage.

For ease of integration with our Gymnasium environment, we implemented the policy using the open source Stable-Baselines3 framework [Raffin et al., 2021]. Our implementation mirrors the main network as it uses the same architecture (VGG16) as the backbone.

Concretely, our implementation is composed of 3 custom components:

- Feature Extractor Network - VGG16 architecture that takes in an image and converts it to a 256-dimensional representation.
- Action Network - Linear mapping layer that takes in 256-dimensional representation from the Feature Extractor Network and outputs a vector of the size of the auxiliary task dimension. The output represents the agent's auxiliary label selection for the given image.
- Value Network - Linear mapping layer that takes in 256-dimensional representation from the Feature Extractor Network and outputs a single value, representing the value of the observation.