# Reinforcement Learning for Data Manipulation

Anurag Patil
*Oden Institute for Computational Engineering and Sciences*
*Univeristy of Texas at Austin*
Austin, U.S.A
anuragrpatil@utexas.edu

Neha Agarwal
*Department of Electrical and Computer Engineering*
*University of Texas at Austin*
Austin, U.S.A
neha.agarwal@utexas.edu

***Abstract*—Coming up with data manipulation techniques like data weighting schemes and data augmentation schemes is a commonly encountered problem in supervised learning. However, current data augmentation implementations are manually designed, with the scheme formulated separately for each unique dataset. Adaption of these scheme, especially in scenarios with low data regime or imbalanced classes, can significantly improve the accuracy of the supervised learning model. Reinforcement learning can help design an agent that can come up with these different data manipulation schemes for any input dataset. Thus allowing for an automatic way of handling difficult datasets for supervised learning.**

***Index Terms*—Data weighting, Low data regime, imbalanced classes.**

## I. Introduction

In this project, we formulate a reinforcement learning framework which automates the data weight process to improve the efficiency of learning a classification problem. Furthermore, by changing the reward function it can automate different data manipulation techniques like data augmentation, data weighting, etc. Handling imbalanced classes and low data regime is a commonly encountered problem in supervised learning. Our reinforcement learning framework is well suited to handle such tasks. Data augmentation helps in training the model for the invariance in the data domain.

The formulation as a reinforcement learning problem is as follows: The state space is the image vector and the corresponding label associated with it. For one particular state i.e an image, a policy is used to find the probabilities of the image belonging to all the different classes. The action is choosing the class with highest probability among these. The data manipulation is learnt via an intrinsic reward. The intrinsic reward parameters act as the data weights. Using this intrinsic reward, the extrinsic reward, which is the accuracy of the classification model, is calculated. Policy gradient based methods are used to find this policy.

To facilitate this, we have reproduced the results from the paper: Learning data manipulation for augmentation and weighing *(citation here)*. Our implementation goes beyond the implementations adopted in the paper. We have implemented a diverse set of policy gradient based methods and evaluated the results. The focus of this project was to understand the implement the formulation that the authors discussed, and to explore various algorithms in Reinforcement Learning and apply the suitable ones here..

There is rich line of literature which show equivalence between the data in supervised learning and the reward function in reinforcement learning. A popular paper on the topic was AutoAugment: Learning Augmentation Strategies from Data *(citation here)*. This paper discusses learning the data augmentation scheme. The paper highlights the need for automating the process of data augmentation. For example, horizontal flipping of images during training is an effective data augmentation method on CIFAR-10, but not on MNIST, due to the different symmetries present in these datasets *(citation here)*. Hence, we need to devise a way of designing data-augmentation suitable for each unique dataset. The drawback of this paper was that it did not allow for a smooth transition between different data manipulation techniques, like data weighting and data augmentation. This is achieved by the model implemented in our project. It uses reinforcement learning, where the formulation of the the intrinsic reward reflects the data manipulation scheme that we want to achieve. The literature review section discusses these in more depth.

The experiments consider the CIFAR10 dataset. Results have been shows for different policy gradient methods for two different setting, Low Data regime and Imbalanced Classes regime. For both these setting, the following policy gradient methods have been implemented: Reinforce, Proximal Policy Gradient and Soft Actor Critic.

github repo here

## II. RL Framework

The environment has the **x** and **y** (images and labels) $\in$ $D$, $D$ being the training dataset, as the state-action space. The environment gives an extrinsic reward for every correct classification which is the accuracy of our supervised learning model. There is separate intrinsic reward environment which is responsible for coming up with the data weighting scheme. The intrinsic reward environment has **x** and **y** (images and labels) $\in$ $D^v$, where $D^v$ is the validation dataset.

### A. State Space

The state space is discrete. The state variables consist of the parameters of the CIFAR10 images, where the images are RGB (3 channels) 32x32 images. For the low data regime,

the training dataset the consisted of uniformly sampled 400 images from the original 50000 images. For the imbalanced data regime, the training set consisted of 1000 images sampled from one class, and 100 images sampled from another class. Please refer to the Experiments section for more details.

### B. Action Space

The action space is the set of 10 classes in the CIFAR 10 dataset. The action space is discrete with values between 1 to 10.

### C. Extrinsic Reward Function

For every correctly classified image, the reward given is the inverse of the state space, i.e. the number of images being considered. Thus, the total reward is the number of correctly classified images divided by the total number of images.

### D. Intrinsic Reward Function

For every correctly classified image, a weight is assigned as the intrinsic reward. The final extrinsic reward is obtained by an additive transform using the intrinsic reward. Thus, the model assigns more importance to the correctly classified images, and updates the policy gradients and the extrinsic policy accordingly. This is extremely important in the two settings considered here. *insert the equation for model given in paper here.*

### E. Transition Functions

The state transition functions are the Data Loaders. The probability of each transition is 1.

### F. Policies

*Extrinsic Policy*: The extrinsic policy is to predict the correct label given an image from the training dataset. The parameters of the extrinsic policy are represented by $\theta$
*Intrinsic Reward Policy*: There is a variety of intrinsic reward policy formulations that can be used. For example, a simple formulation would weigh the correctly classified image by some constant amount. The paper discusses a superior approach. The reward is computed using the validation dataset, by evaluating the performance of the network iteratively and then finding the weights according to the correctly classified data-points. This is found to be more robust. The weighting scheme is designed as a parameterized model, with the weights being the parameters. The weights are initialized once in the beginning, and are updated at each step iteratively. The author has chosen to implement a parametric approach with gradient based update to ensure a smooth learning for the parameters, and to preserve historical information. The parameters of the intrinsic reward are represented by $\phi$

The following figures provide a schematic of the reinforcement learning framework:

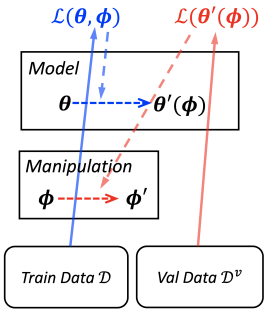


Fig. 1. The reinforcement learning framework

## III. Literature Review

There is much literature on using reinforcement learning models to improve performance of supervised learning models.

### A. Data Augmentation

We understand the idea of Data Augmentation from other implementations as well [3]. The RL algorithm they are using is a SoftMax predictor, which is very close to our algorithm. They have defined the problem of finding the best policy as a search problem, where the search space is (16 X 10 X 11) X (16 X 10 X 11) big (there are 16 possible operations, with input magnitudes in the discrete range of [1,10] and probability of taking each action in the discrete range of [1,11]. Since, each policy consists of 2 actions, we take the square). We don't use the same policy search formulation.

### B. Transfer Learning

We consider transfer learning [3,6] as a stretch goal in our project. Given a dataset (a dataset of images in our case), we should be able to learn policies to augment this dataset, and then the same policies should be applicable if the input data is sourced from another image repository.

### C. Dual Reward

The RL model we propose has two reward functions. The extrinsic reward is the performance (as defined by accuracy) of the supervised learning model that carries out image classification based on the weighted dataset, D, which is produced by the RL model. Since this reward is obtained from outside the environment of the RL model, we define this reward as extrinsic reward. The intrinsic reward is what is observed from within the environment. The paper defines the intrinsic reward as a function of the weights computed. Intrinsic reward is defined as the function which helps the agent achieve the behavior that optimises the extrinsic reward function [1, 2].

### D. Formulating Intrinsic Reward

The paper first proposes a delta function like reward function, with reward being +1 if augmented data point is in the original dataset, or negative infinity otherwise. But later, we change the reward function to a function of the weight parameters of the model. The reason for this change is that a delta-like reward function is extremely restricted, and disables any exploration of the model beyond the training data [4].

### E. Policy Gradient Methods

Our RL model is a policy gradient model. We learnt about such modeling in Chapter 13 [5]. The state space is the entire dataset, which is huge. We, therefore, do not model our RL based on the value function. Instead we choose policy parameters, and use gradient based methods to find the optimal policy by optimizing the value of the parameters. We use a maximum likelihood function to find the optimal policy.

### F. Maximum Likelihood Modelling

MLE modelling suffers from exposure bias [4,7]. Since the model is exposed to only the training dataset, using which it finds the value for its parameters, it could potentially overfit the training dataset. When the testing dataset differs from training dataset, the model could under perform due to overfitting. The other paradigm of modelling is Bayesian models, which do not typically suffer from overfitting as they find a probability distribution function for their parameters, rather than the actual values. However, the MLE model is better suited to our problem scenario. For the data weighting case, a logistic regression based binary classification problem is considered. To formulate the problem for this case the supervised learning ideas are discussed in Chapter 4[7]. In Addition to the multiclass and binary logistic regression maximum likelihood formulation and the methods to solve them like gradient descent and second order methods like newton Raphson, methods like perceptron are also discussed.

### G. Meta Learning

This helps to draw a parallel between the importance and the semi-supervised learning based methods discussed in [8]. The meta-learning based methods [12] like automatic differentiation, the online approximation for it are discussed. Most importantly, these methods are discussed in the context of deep learning and experiment similar to ours. This allows us to study the convergence and performance of our reinforcement learning based implementation with the popular school of meta-learning based methods. The method to find the solution to the Bayesian perspective of maximizing the posterior distribution with a reinforcement learning based technique is seen in [9]. They have found two off-policy based algorithms with good performance. This method will be studied as an extension due to the inherent drawback of exposure bias associated with the maximum likelihood formulation.

### H. Others

Adapting the proposed method to the data augmentation case will be studied as an extension. To draw a parallel between the reinforcement learning based formulation and the supervised learning based methods, the work [10] is considered. It also discusses bootstrapping which is popularly used for data manipulation to improve the performance in the supervised setting. Thus, a discussion from this also benefits our primary formulation as well as a parallel to bootstrapping would be considered. [11] approaches the problem with a very interesting concept. They model the data augmentation technique as a sequence of individual transformative functions. Their input dataset is unlabelled, and enables easy transfer learning. Although, their approach is quite different from ours, it definitely is an interesting read.

## IV. Algorithms

We have implemented various algorithms, and compared the accuracy of each with that obtained by the authors. We had proposed that we will carry out data weighting scheme, and pursue the data augmentation case as an extension. Due to paucity of time, we were not able to take up data augmentation. The settings for data weight have been studied thoroughly.

We have an input dataset of 60,000 images, uniformly distributed across 10 classes. We construct our training and validation datasets from this input dataset. Our project experiments with two scenarios, low data regime and imbalanced label regime. In the first scenario, we randomly pick 40 datapoints from each class to construct the training dataset and 2 data points from each class to construct the validation dataset.

### A. Policy Gradient Methods

*1) Proximal Policy Optimization (PPO):* Proximal Policy Approximation implements a policy gradient update such that the at every step, it tries to minimize the cost function in such a way that the deviation from the old policy is relatively small. The biggest advantage of PPO compared to other methods is that, not only is it easy to implement the algorithm and tune the hyper parameters, at the same time, the algorithm does not lose out on the sample complexity.

We have implemented PPO. A part of the code was taken from Github [**?**].

We considered implementing PPO as an extension of Actor-Critic method discussed in Chapter 13 because of its state of the art performance. Our implementation adapts PPO for a discrete and large state space, and adds in the factor of intrinsic reward and the updates associated with it. The sequence of these updates is discussed shortly after the details of the model.

Model: Neural Networks were used to evaluate the state values and the values of the actions. The Resnet18 model was used for the critic model and Resnet34 was used for the actor model. The last layer of the Resnet34 model was changed

to return 10 actions corresponding to the 10 classes for low data regime. This layer is changed to return only two actions for the imbalanced class problem. The actions sampled were clamped by a clamping factor of 0.1. The intrinsic reward was parameterized with weights assigned to every data point in the dataset. The updates for the parameters were done by weighting them by alpha. A minibatch size of 4 images was considered everywhere

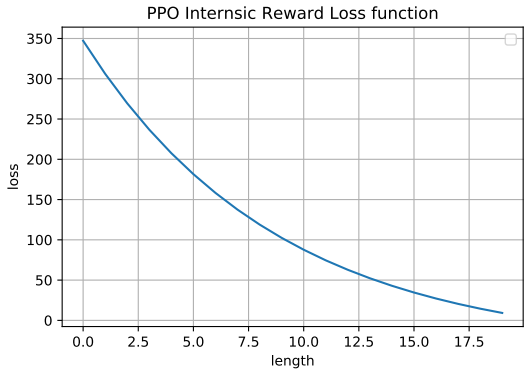


Fig. 2. The intrinsic reward loss function in PPO vs the runs

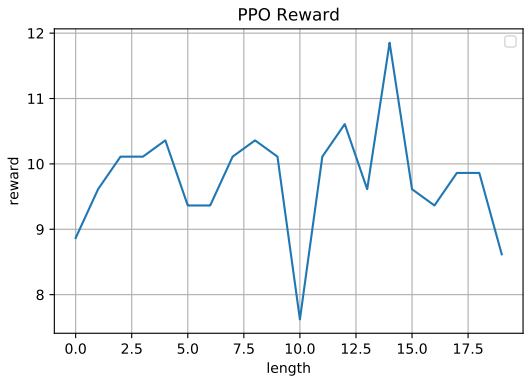


Fig. 3. The sum of weighted extrinsic rewards in PPO vs the runs

The sequence of updates is as follows. First, the images and true labels are loaded from the data loader. Using the image, the value is found from the critic model. The actions and the log probabilities are found from the actor model. At the end of the episode, using the network for the actor model, the weights are obtained through the validation dataset. Using these, the discounted weighted extrinsic rewards are found. The $\theta$ is updated sequentially using the PPO update and $\phi$ is updated according to the intrinsic reward model.

*2) Reinforce:* Reinforce is the most basic Monte-Carlo policy gradient algorithm. Reinforce suffers from high variance and learns slowly due to it being a Monte-Carlo method. But, it has good convergence and is easy to implement. It is considered as the baseline in our project for comparison of the results.

Model: The same actor model as illustrated in PPO was used here. The actions were obtained using an $\alpha$ of and $\gamma$ of $0.9$.

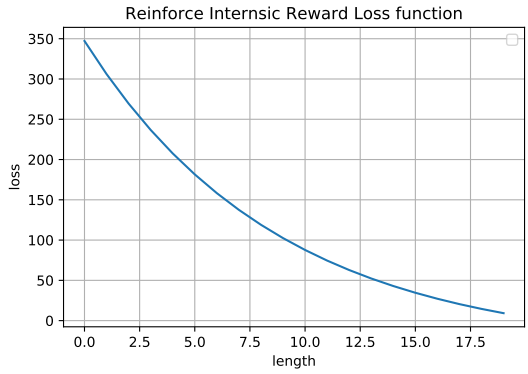


Fig. 4. The intrinsic reward loss function in Reinforce vs runs

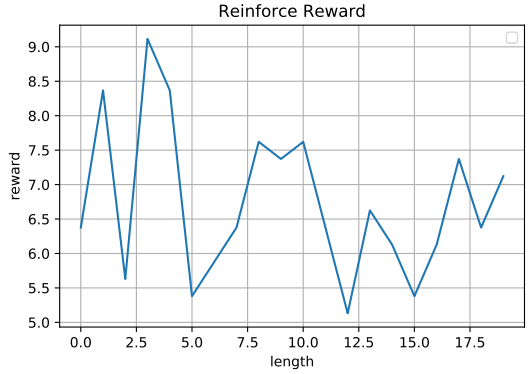


Fig. 5. The sum of weighted extrinsic rewards by Reinforce vs the runs

Note: Some weights blew up exponentially, and hence we got NaN values for some actions.

The algorithm was not robust and was not able to converge to an optimal policy.

*3) One Step Actor-Critic:* There are two components in Actor-Critic methods, namely, actor and critic. As against only actor methods like Reinforce, here the actions taken by the actor are guided by the direction given by critic. This is by considering V or Q in the update equations.
One step Actor-Critic was again implemented using the same actor and critic models as discussed in PPO.

One Step Actor-Critic was implemented with the same space and reward function representations as that of Reinforce.

Model parameters: $\alpha =$ and $\gamma = 0.99$

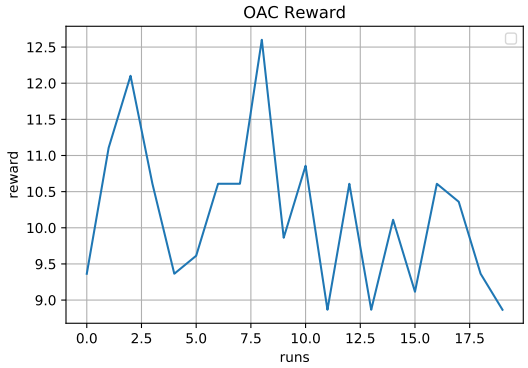


Fig. 6. The intrinsic reward loss function in one step actor critic vs the runs

## V. Analysis for low data regime

The results came close to the ones presented by the author. The authors have not elaborated on the formulation of intrinsic reward properly. Hence, it is difficult to draw an exact parallel between their results and ours. We have presented different ways of formulating the intrinsic reward and their results.

We implemented three formulations of the intrinsic rewards. In the first approach, we assign a constant increased weight corresponding to every class based on the training dataset. In the second formulation, we assign the same constant based on the validation dataset instead of the training dataset. The third formulation assigns a random integer weight between a certain range for every correct classification based on the validation dataset. We found the third formulation to be the best. But, still the authors models are found to perform better.

We tested out these three policy gradient implementations for the two scenarios (low data regime and imbalanced label regime) first by considering transfer learning, and then without transfer learning, but by using a pre-trained implementation of Resnet34.

For the low data regime, as it can be seen for reinforce, the intrinsic reward loss function is found to be decreasing. It is the intrinsic reward loss function that we are interested in, as the goal is to learn the data weighting scheme for this low data regime. The extrinsic reward

## VI. Setting and Results for the imbalanced classes

In the second setting, we consider only two classes, but with an imbalance of data points between the two. We randomly pick 1000 datapoints from class 2, with {20, 50, 100} datapoints sampled from class 1. To construct the validation set, we randomly pick 10 data points from each class.

### A. Policy Gradient Method Results:

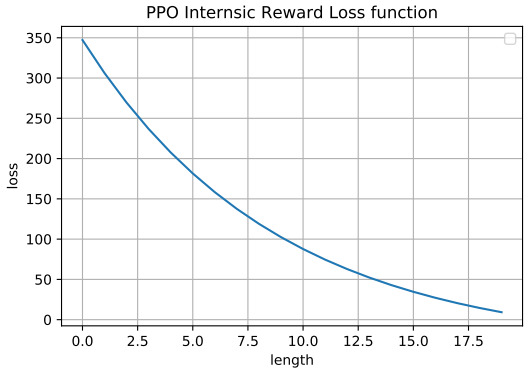


Fig. 7. The intrinsic reward loss function in PPO vs the runs

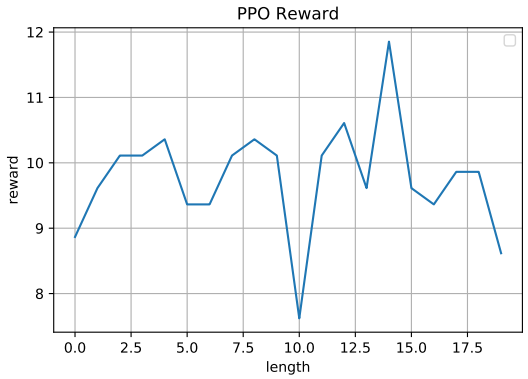


Fig. 8. The sum of weighted extrinsic rewards in PPO vs the runs

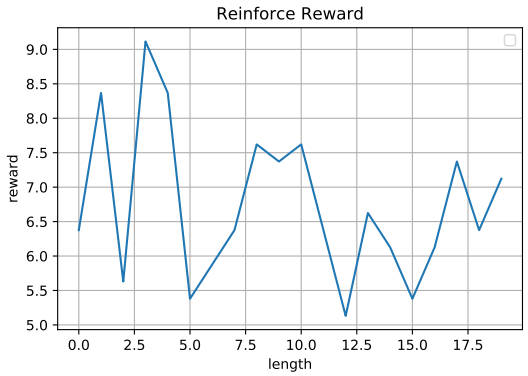


Fig. 9. The intrinsic reward loss function in Reinforce vs runs

#### 1) Reinforce Results::

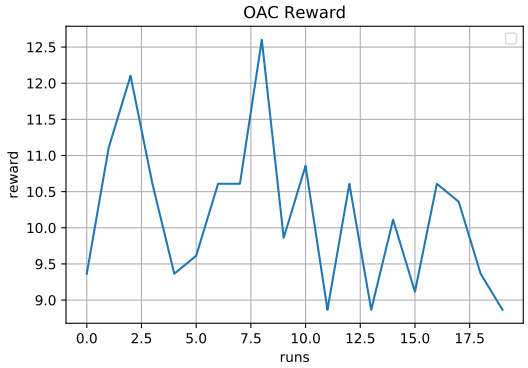


Fig. 10. The intrinsic reward loss function in one step actor critic vs the runs

*2) One Step Actor-Critic Results::*

## VII. Analysis for the imbalanced classes

Again, The results came close to the ones presented by the author. We have used the third formulation of the reward for this. Again had the authors provided a clearer representation of the intrinsic reward formulation the results could have been much better.

We tested out these three policy gradient implementations for the this setting first by considering transfer learning, and then without transfer learning, but by using a pre-trained implementation of Resnet34.

**discussion for imbalanced data**

## VIII. Conclusion

This project encompasses many concepts of Reinforcement Learning. It draws a good distinction between supervised learning and reinforcement learning, and helps us understand how we can borrow ideas from one domain into another. We implemented three formulations for this project. However, the results we arrived at do not match with that of the paper. Since, the paper does not discuss its implementation strategy in detail, we believe that we can not draw a one-to-one correspondence between their results and ours. Our intrinsic loss function always converges, under all formulations.

## References

[1] Satinder Singh ; Richard L. Lewis ; Andrew G. Barto ; Jonathan Sorg. Intrinsically Motivated Reinforcement Learning: An Evolutionary Perspective. IEEE Transactions on Autonomous Mental Development

[2] Zeyu Zheng, Junhyuk Oh, Satinder Singh. On Learning Intrinsic Rewards for Policy Gradient Methods. https://arxiv.org/abs/1804.06459 .

[3] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, Quoc V. Le. AutoAugment: Learning Augmentation Policies from Data. https://arxiv.org/abs/1805.09501

[4] Bowen Tan, Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, Eric Xing. Connecting the Dots Between MLE and RL for Sequence Prediction. https://arxiv.org/abs/1811.09740

[5] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. Second Edition

[6] Mark Ho, David Abel, Tom Griffiths, Michael L. Littman. The Value of Abstraction. https://psyarxiv.com/6fm9a/

[7] Christopher Bishop. Pattern Recognition and Machine Learning

[8] Mengye Ren, Wenyuan Zeng, Bin Yang, Raquel Urtasun. Learning to Reweight Examples for Robust Deep Learning. https://arxiv.org/abs/1803.09050

[9] Abdolmaleki A, Springenberg JT, Tassa Y, Munos R, Heess N, Riedmiller M. Maximum a posteriori policy optimisation. arXiv preprint arXiv:1806.06920. 2018 Jun 14.

[10] Wei JW, Zou K. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. arXiv preprint arXiv:1901.11196. 2019 Jan 31.

[11] Alexander J. Ratner, Henry R. Ehrenberg, Zeshan Hussain, Jared Dunnmon, Christopher Ré. Learning to Compose Domain-Specific Transformations for Data Augmentation. https://arxiv.org/pdf/1709.01643.pdf

[12] Lilian Weng. https://lilianweng.github.io/lil-log/2019/06/23/meta-reinforcement-learning.html