Online learning with binary feedback for multi-class problems

Evan Lucas, Steven Whitaker, Timothy C. Havens
Michigan Technological University
Houghton, MI 49930
eglucas, sjwhitak, thavens @mtu.edu

***Abstract*—Online learning methods often focus on data selection, as human labeling is a noted bottleneck in resources needed to train a model. This work chooses to focus on reducing the human effort necessary for providing labels by attempting to usefully utilize a binary feedback method where the human indicates whether the prediction is correct or incorrect. More specifically, this work investigates methods for using and labeling training data in the absence of complete information. Various methods to generate labels in response to human feedback are proposed and then compared. These methods are tested on a variety of common classification tasks and results showing their usefulness are presented. Although the maximum accuracy achieved is not as high, the methods presented allow a model to learn faster, in terms of number of human interactions required.**

***Index Terms*—binary feedback, online learning**

I. Introduction

When a supervised learning model is applied to a new domain or set of data, it is necessary to obtain new labeled training data. The volume of data required can depend on many factors, including similarities to other data sets, but performance without task-specific data (often called the zero-shot case) almost invariably lags behind a model trained on the new domain. Labeling of data can be expensive and some domains of data can require substantially more effort to label. For example, in image classification it would typically take more effort for a human to determine what object is in an image than for the same individual to decide which of the digits 0-9 are written in a box.

In addition to making feedback easier, there are other applications, such as recommender systems, that only receive binary feedback, either actively or passively. In the active case, a user might give a suggestion a "thumbs-up" or "thumbs-down". Or in the passive case, a video streaming service may only know whether or not a user chooses to watch a suggested movie within a given timespan. [1]

The focus of this work is to make better use from the limited human feedback on a partly pre-trained model for online learning. We propose using binary feedback, where the human tells the model if a prediction is correct or incorrect, as a method for creating additional training data. After a model makes a prediction on new unlabeled data, it queries the human for feedback on whether this is a correct prediction or not. It is obvious that for correct predictions, this example can be directly considered as training data. However, when a prediction is indicated as incorrect, the prediction cannot be used directly. In this paper, we show how an online learning algorithm can use both correct and incorrect indications from human feedback.

Although this work focuses on the online learning problem, where the incoming data is rate limited and used in entirety, it is also helpful to consider some of the findings of studies in active learning. Active learning generally assumes a sufficiently large body of training data that can be searched to find training examples that will most quickly teach the model. Although the implementations are substantially different, the usefulness of a given file will depend on the same things. In [2], they classify samples with three criteria to determine if they will be a good choice to train on: that the samples are learnable, worth learning, and not yet learnt. These concepts were applied to the work in this paper by trying to balance the inclusion of correctly identified samples with a strategy for using incorrectly identified samples.

Other work, such as [3] and [4], identifies the human bottleneck imposed by requiring multi-class feedback and suggests that binary human feedback can be used on a multi-class problem. The procedure proposed by Joshi et al. to exploit binary feedback starts as expected: by asking the user if the predicted class is correct. If a prediction is correct, the sample can be added to the training data. If a prediction is incorrect, the model follows an active learning strategy to re-query the user if the next-most-probable class is correct until a correct answer is found or an early stopping criteria is triggered. They propose using an expected *value of information* (VOI) metric to determine whether additional user information is likely to provide useful information to the model or will just take up human interaction time without expected benefit. Joshi et al. also use VOI as an input to an active-learning strategy to select training samples for which labels will provide the maximum amount of information to the model.

Another work in the area of binary feedback for multi-class problems approach the problem in a way familiar to those who have played the word game "20 Questions", where one player asks yes-or-no questions that become increasingly specific. Hu [5] uses repeated binary questions regarding partial labels to sort through multiple classes and narrow in on a specific unique class label. In contrast to Hu's method, the method proposed in this paper only seeks feedback once per sample. Furthermore, the proposed method only asks whether

the prediction is correct, which does not require building a knowledge base of partial labels.

Human feedback has also been extensively used in NLP, generally through the concept of reinforcement learning. [6] Human feedback is often binary, in the form of a preference between a pair of outputs, and is generally used to train a reward model that is then optimizes the model. The method of *Proximal Policy Optimization* (PPO) [7] has been applied to the problems of improving summarization, text generation, and following human language instructions [8–10].

The rest of this paper is organized as follows. An overview of the methods are introduced in Section II. The initial efforts to demonstrate this method and understand its limitations using a data set consisting of two dimensional points belonging to equally spaced clusters and a simple fully connected neural network are presented in Section III-A. Model and data complexity are increased and effects of the proposed method are studied using a simple *convolutional neural network* (CNN) with the MNIST and EMNIST data sets in Section III-B. Finally, the method is applied to a pre-trained *residual network* (ResNet) with the CIFAR10 data set in Section III-C. We summarize in Section IV.

II. Methods

We start with a model that is partly trained on some quantity of labeled data. At each generation of training a batch of new data is introduced to our model, which generates a set of predictions. A human is queried as to whether the predictions are correct or incorrect. Correct predictions can be used as additional training samples with the correctly classified hard label, but for incorrectly labeled predictions a new approach is needed. We propose a variety of ways that utilize this feedback. Our approach is predicated on the idea that a new label can be created for the queried sample that represents the uncertainty left by the user indicating that the inference is incorrect. Then the algorithm trains with this new label.

The methods proposed are using a uniform soft label, a predicted soft label from model outputs, a hard label sampled from the model outputs, and, last, using the measured misclassification probabilities to produce a label. An example of this work flow is shown in Figure 1.

The four feedback methods initially considered consist of the following:

1) Uniform soft labels: The uniform soft label approach creates a uniform probability distribution, where the incorrect class is set to zero. This approach assumes that there is no useful knowledge available or gained outside of knowing one incorrect class. This is depicted in Figure 2a. This approach could be adapted for imbalanced class problems by using the prior distribution of the classes in the fully labeled initial training set.
2) Modified soft labels: This approach assumes that the model has partly learned the classes and reuses the model outputs as a training label, after removing the incorrect class. This is done by setting the logit of the incorrect output to be a large negative number and rebalancing the probabilities with a softmax function. An example of what this would look like for the distribution in Figure 1 is shown in Figure 2b.
3) Sampled hard labels: The idea behind this method is to provide a hard label that is potentially correct, by sampling from the soft labels predicted by the model. Similar to the other methods, the incorrect class is removed, the probabilities are rebalanced to sum to 1, and a uniform random variable is generated to sample from the resulting distribution. One possibility for a sampled hard label is shown in Figure 2c.
4) Conditional prior: The final approach considered was to use a held-out data set to estimate the conditional probabilities for an incorrectly classified sample conditioned on its incorrect class. Knowing that a sample was incorrectly classified as a given class, the distribution of true labels from other samples misclassified with the same label are substituted for the soft labels used in training. An example of this might look something like

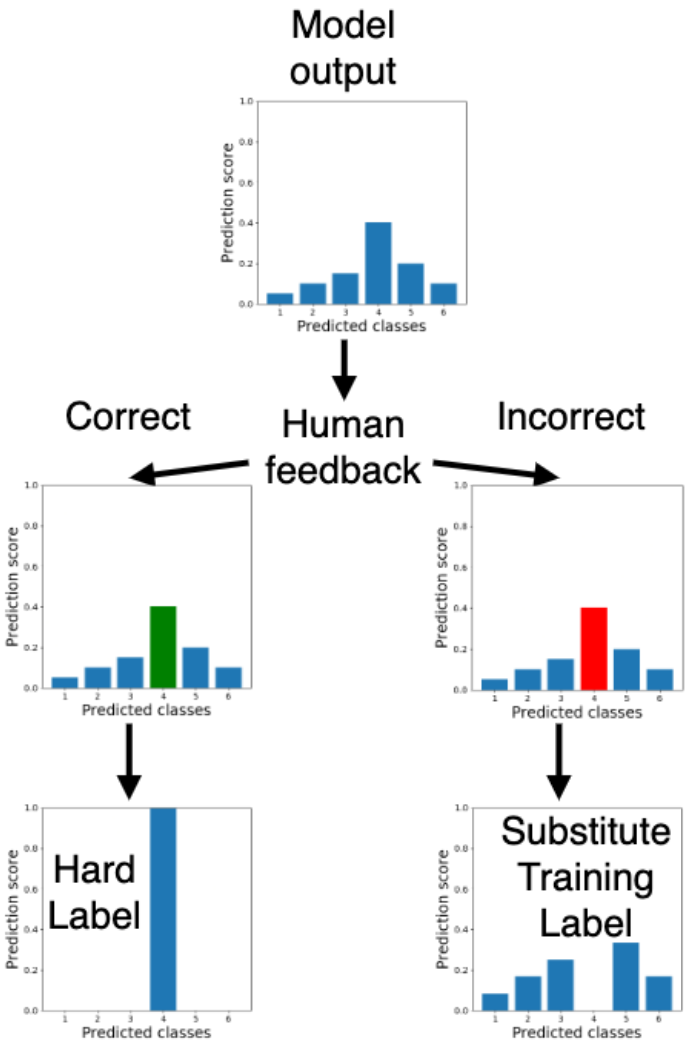
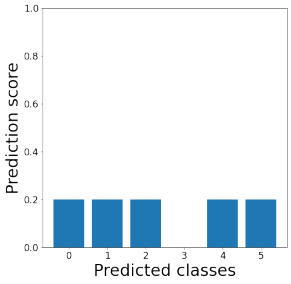
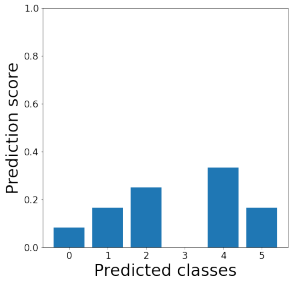


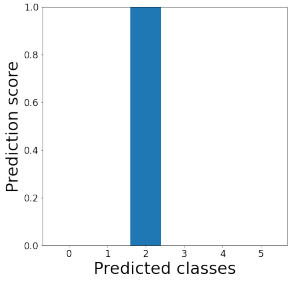
Fig. 1: Binary feedback with soft-labeling in multi-class data



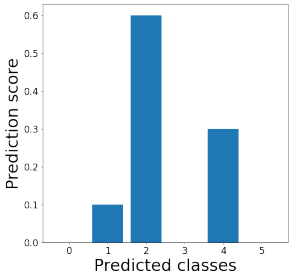
(a) Uniform soft label



(b) Modified inference soft label



(c) Sampled hard label



(d) Conditional prior soft label

Fig. 2: Visualization of different training label substitutes that can be created with negative feedback.

what is seen in Figure 2d.

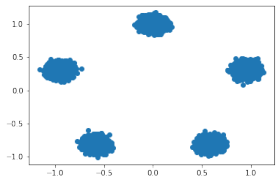
These approaches allows immediate use of the incorrectly labeled prediction without additional human feedback. Other approaches to utilizing binary feedback rely on continually asking for additional binary feedback until the correct label is found or the expected value of information gained by asking falls below a threshold [5]. Following this work, the number of binary feedbacks required to get a correct hard label are tracked for most of the experiments performed.

III. Results

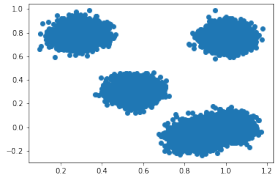
Results are shared first for the point cluster identification problem, which is a toy example that runs quickly in order to test many iterations of different strategies. The difficulty of the problem, along with the complexity of the model used, are increased for the other results subsections, which use MNIST and CIFAR10 with more complex models.

A. Point cluster identification

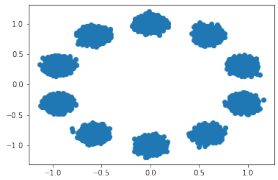
To provide a very simple starting point to test this method, a simple data set was generated by creating clusters of points in two dimensions. Point cluster centers were located on a unit circle, with equal spacing for most tests, and points were generated in a uniform distribution around each center. Some sample distributions can be seen in Figure 3. A surplus of points were generated for each experiment and randomly sampled as needed.



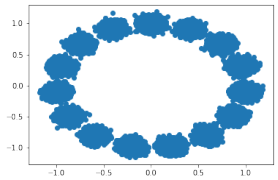
(a) Five points, evenly spaced



(b) Five points with random centers



(c) Ten points, evenly spaced



(d) Fifteen points, evenly spaced

Fig. 3: Samples of point cloud data used for initial investigations.

The model used for these investigations was a fully connected neural network with a single hidden layer consisting of 16 nodes. The input layer takes in the two dimensional coordinates of the point and the output layer is the same size as the number of classes used in that test. To minimize influences from more advanced optimizers, *stochastic gradient descent* (SGD) was selected as the optimizer of choice. Hyperparameters that allowed the model to learn were manually selected for the base model. To provide an upper and lower limit to gauge success of the method, two models were run in parallel with the model using binary feedback; one with full labels and another that only trained on samples that the model had correctly classified.

For each point experiment, a batch size of 20 samples was used for each generation as well as the fully labeled pre-training batch. Thirty runs with random initializations were performed to reduce the effects of randomness. For the base model, the number of binary feedbacks to obtain a label using exhaustive search was counted by finding the rank of the correct output label in the model prediction. For example, referencing Figure 2b, if Class 2 were the correct class, it would take three interactions of yes or no questions with a human labeler to determine the correct class.

Starting with the simplest case where binary feedback is still partial feedback, three clusters of points were generated and each model was trained on the samples. The resulting average test accuracy from 30 runs is presented in Figure 4 as a function of total number of new training files evaluated and used. It is clear that the uniform feedback performs best out of all the feedback methods, followed by the sampled and conditional prior methods. Interestingly, using only files that received good feedback appears to have little to nothing to offer the model. The authors hypothesize that this is due to the simple nature of the data set, where little is learned once

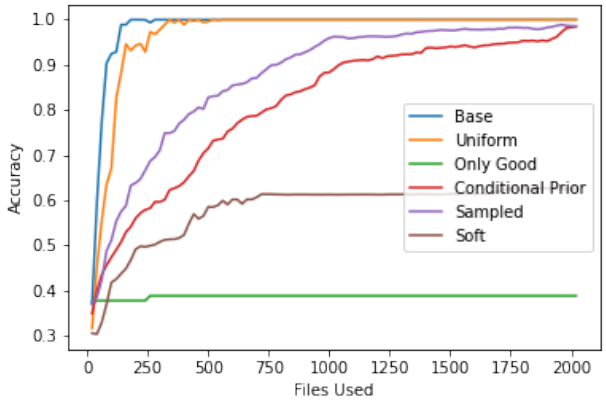


Fig. 4: Average model performance for different feedback strategies with the three points data

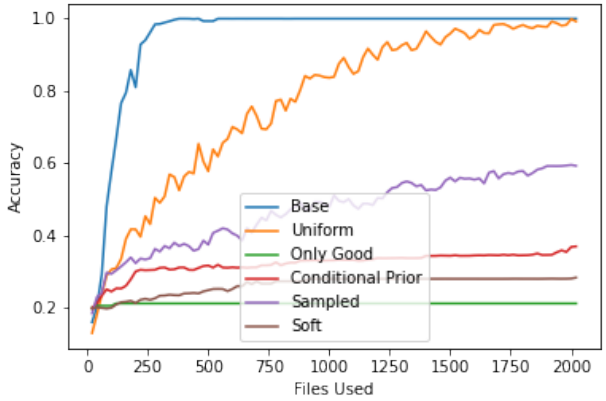


Fig. 5: Average model performance for different feedback strategies using five point clusters with equal spacing

a class has been learned by the model and the good data only reinforces the existing model decision boundaries. As will be seen in the other results sections, the good data offers more when the data space is more complex.

To gradually extend these methods to harder problems, the same 30-run experiment was repeated with data sets containing five, ten, and fifteen evenly spaced points. These are shared in Figures 5, 7, and 8 respectively. The five point experiment was repeated with randomly spaced points that allowed overlap, which is included as Figure 6. It can be observed that as the base model performance degrades, so does the partial feedback models. As before, the uniform partial feedback performs best out of all of the different feedback methods considered. Again, the use of only correctly labeled files is found to not be helpful.

To understand the level of human interaction required to train such a model with full labels, the number of interactions was estimated using the method previously described and averaged over the 30 runs. An example of this for the points data using ten clusters is included in Figure 9, where it can

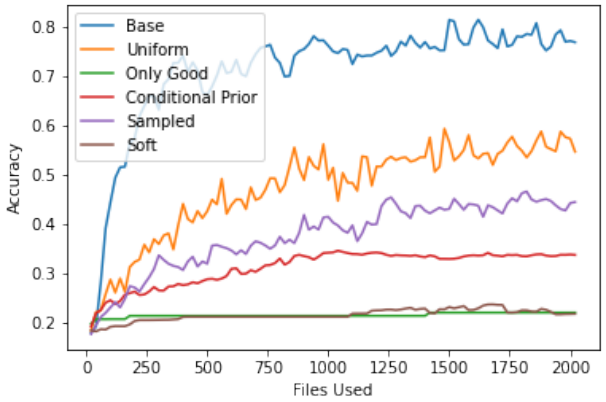


Fig. 6: Average model performance for different feedback strategies using five point clusters with random spacing

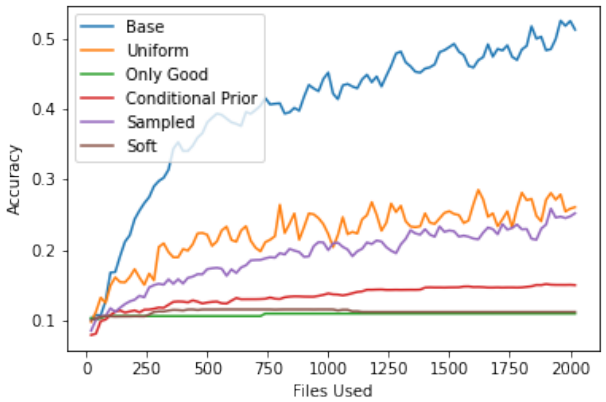


Fig. 7: Average model performance for different feedback strategies using ten point clusters

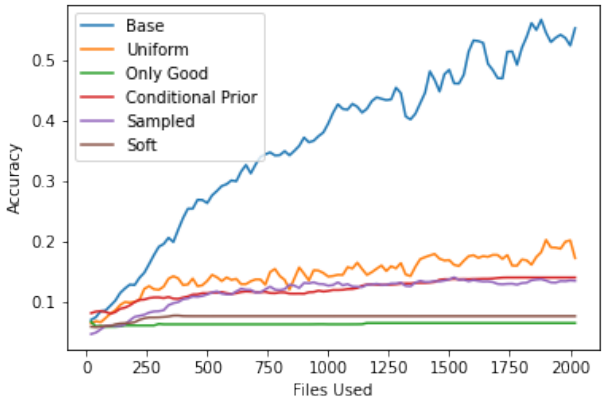


Fig. 8: Average model performance for different feedback strategies using fifteen point clusters

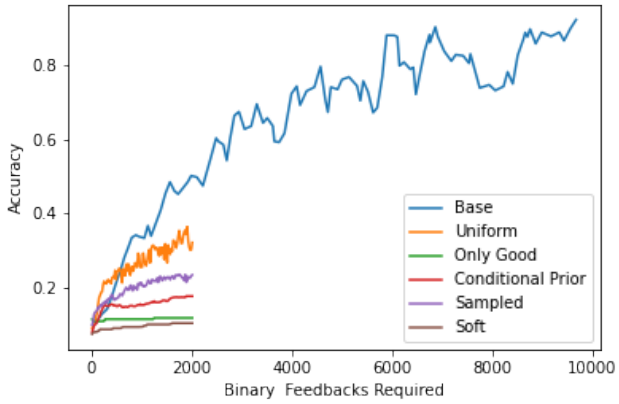


Fig. 9: Average model performance for different feedback strategies using ten point clusters, plotted against binary feedbacks

be seen that the model requiring full labels requires nearly five times as many interactions to train. Although it achieves a higher accuracy ultimately, the uniform feedback achieves a higher accuracy faster in terms of feedbacks.

1) Ablation study: A small ablation study was performed using the point data to understand the influence of different aspects of the proposed method. As with the other tests using the points data, 30 runs were performed and averaged. A held-out test set was used to compute all accuracies to prevent memorization. Because random data selection was used, the model using only positive feedback and the full label model was run alongside the online binary feedback model and reported in the ablation results. For purposes of this test, the soft label learning rate was given priority for determining learning rates in the binary feedback model: so the full label model ran with this learning rate as did the model that trained on good labels (from the positie feedback) and soft labels (from the negative feedback) together. The base model is considered to be the model described in the previous section, which separates files with positive and negative feedback so that it can use a higher learning rate for soft labels than hard labels (8e-4 and 8e-5 respectively), and uses SGD with momentum of 0.95.

Going through the ablation study, some of the findings appear obvious, while others require deeper thought. It has been demonstrated analytically that reducing the learning rate is necessary for convergence on a solution [11], so it is not a surprising finding that removing momentum causes performance degradation. Interestingly, changing the learning rates between the good data and the soft or estimated label data doesn't appear to hurt the binary feedback method. However, it does appear to be necessary to separate the good data from the soft label data so that separate optimizers can be used. Additionally, using the good label data and the soft label data together was found to be optimal for this simple data and model. Finally, as one would expect, as the data becomes

TABLE I: Ablation study: Highest average accuracy achieved with various model configurations using ten cluster point data

Model Description	**Only-Good Data**	**Binary Feedback**	**Full Labels**	**Feedback Type**
Base model	0.117	0.364	0.924	U
-Momentum	0.095	0.174	0.255	U
-Momentum, $LR_{good} > LR_{soft}$	0.102	0.265	0.883	U
$LR_{good} > LR_{soft}$	0.109	0.364	0.984	U
Class separated training	0.109	0.224	0.942	Samp.
-Training with negative feedback only	0.117	0.254	0.928	Samp.
Combined good and bad training	0.112	0.237	0.955	U
Overlap allowed	0.110	0.285	0.525	U

harder (or impossible) to linearly separate, performance of the binary feedback method degrades along with that of the base model.

B. MNIST

The MNIST data set [12] was selected due to its use as a benchmarking dataset. [13] A *convolutional neural network* (CNN) was used as a model. The model consisted of two convolutional layers, a flattening operation, and two fully connected layers. Each convolutional layer contained eight five-by-five filters with a ReLU activation and two-by-two maximum value pooling before the next layer. The first fully connected layer used a ReLU activation and contained 100 nodes, which was followed by the output layer which used linear activations. A softmax was applied to the model outputs to determine the class, as having the soft label outputs was useful for generating some types of training labels. SGD was used as the optimizer and batch sizes of 40 files were found to be a good number to demonstrate the impact of the proposed method. Learning rates of 5e-2 and 2.5e-2 were used for the negative feedback and positive feedback respectively. The base model was tested with both learning rates, but ultimately used the higher learning rate. Momentum of 0.95 was used for all runs. The larger batch size was chosen to help prevent an initial generation that would have zero correct files, which would give a strong disadvantage to any test that was unfortunate enough to have an unlucky random initialization included in its runs.

Following the method set in the previous section, 30 runs of the model with each type of feedback were performed and averaged. Due to computational constraints, the base model and model using correctly identified files were not rerun with each binary feedback method. The results of this are shared in Figure 10. Interestingly, all of the feedback strategies and the model using only the correctly identified files all learn faster than the base model. The conditional prior method,

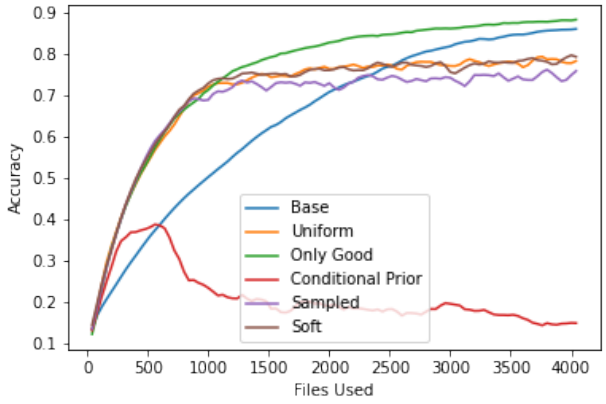


Fig. 10: Average model performance for different feedback strategies using MNIST

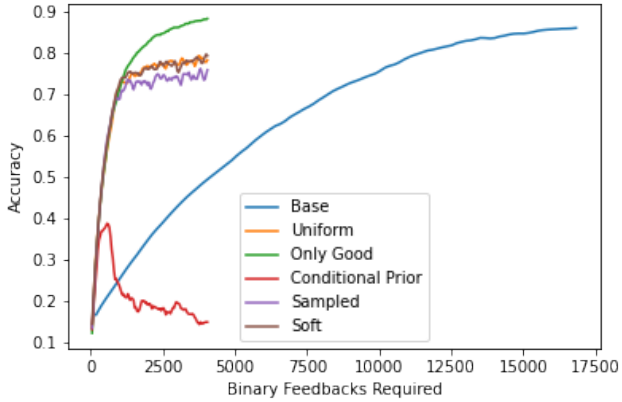


Fig. 11: Average model performance for different feedback strategies using MNIST, plotted against binary feedbacks

although initially successful, appears to stop working rather quickly. The models using the negative feedback appear to be limited in terms of maximum accuracy capable; this is similar to the the results found using the simple point cluster data where they appeared to reach a maximum accuracy below that of the full label model. The advantages of working with partial feedback are made clear by considering the number of feedbacks required, as shown in Figure 11. Every single method considered achieves it's maximum in far fewer simulated human interactions than the full feedback model would require, assuming binary feedback given until a full label was reached.

The hypothesis put forward by the authors for the improved performance of the model using only the correctly identified files is that the feature space is complex enough that it hasn't already learned all of the features necessary to correctly classify that class. Because the model architecture is a CNN that shares filters between classes, improving the filters for a given class will generally help improve other classes as well.

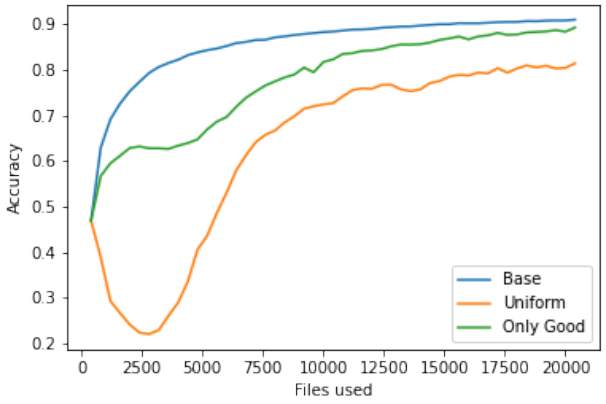


Fig. 12: Average model performance for different feedback strategies using CIFAR10

C. CIFAR10

A final experiment was conducted using CIFAR10 [14] with a pre-trained ResNet18 model [15]. Similarly to the previous experiments, 30 runs were performed with randomly sampled files and averaged. Due to its success in previous trials, only the uniform feedback method was considered. The results of this trial are shared in Figure 12. For this investigation, 400 samples were used to fine-tune a ResNet18 model pre-trained on ImageNet [16] and 400 samples were used for each update generation. Much like the MNIST case, it appears that training on only files that have received good feedback provides the most useful information to the model. The large drop in performance seen by the uniform model was investigated and appears to be related to learning rate. The authors were unable to eliminate this dip without hurting maximum accuracy later on and suspect that an optimizer specifically developed for this type of feedback may be necessary to optimally use this type of training label on a model of this type. Plotting the accuracy curves against number of feedbacks again, Figure 13 shows that the use of only correctly labeled files allows a high accuracy to be achieved with fewer human feedbacks.

IV. Conclusions

In this work, we proposed a method for utilizing negative binary feedback in an online learning scenario. It is shown that in some cases, using the samples receiving negative feedback in conjunction with the ones that receive positive feedback outperform models utilizing only positive feedback. Interestingly, in more complicated cases, using only the files with positive feedback gives the best outcome.

Future work could include testing this on more complicated models and data sets, as well as extending the binary feedback concept to problems that would require several bits of data to fully respond in a binary fashion; for example an object detection problem would require a bit for the accuracy of the object classification, a bit each for center coordinates, and another bit each for bounding box dimensions.

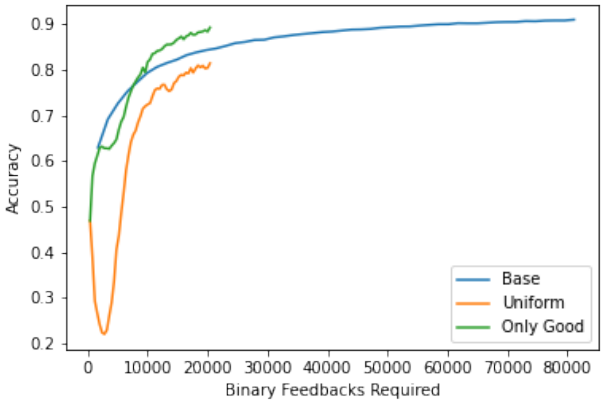


Fig. 13: Average model performance for different feedback strategies using CIFAR10, plotted against binary feedbacks given

References

[1] M. Volkovs and G. W. Yu, "Effective latent models for binary feedback in recommender systems," in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015, pp. 313–322.

[2] S. Mindermann, J. Brauner, M. Razzak, M. Sharma, A. Kirsch, W. Xu, B. Höltgen, A. N. Gomez, A. Morisot, S. Farquhar *et al.*, "Prioritized training on points that are learnable, worth learning, and not yet learnt," *arXiv preprint arXiv:2206.07137*, 2022.

[3] A. J. Joshi, F. Porikli, and N. Papanikolopoulos, "Breaking the interactive bottleneck in multi-class classification with active selection and binary feedback," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 2995–3002.

[4] A. J. Joshi, F. Porikli, and N. P. Papanikolopoulos, "Scalable active learning for multiclass image classification," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2259–2273, 2012.

[5] P. Hu, Z. C. Lipton, A. Anandkumar, and D. Ramanan, "Active learning with partial feedback," in *International Conference on Learning Representations*, 2018.

[6] V. Uc-Cetina, N. Navarro-Guerrero, A. Martin-Gonzalez, C. Weber, and S. Wermter, "Survey on reinforcement learning for language processing," *arXiv preprint arXiv:2104.05565*, 2021.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[8] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving, "Fine-tuning language models from human preferences," *arXiv preprint arXiv:1909.08593*, 2019.

[9] N. Stiennon, L. Ouyang, J. Wu, D. Ziegler, R. Lowe, C. Voss, A. Radford, D. Amodei, and P. F. Christiano, "Learning to summarize with human feedback," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3008–3021, 2020.

[10] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray *et al.*, "Training language models to follow instructions with human feedback," *Preprint*, 2022.

[11] H. Robbins and S. Monro, "A stochastic approximation method," *The annals of mathematical statistics*, pp. 400–407, 1951.

[12] Y. LeCun, "The mnist database of handwritten digits," *http://yann. lecun. com/exdb/mnist/*, 1998.

[13] L. Deng, "The mnist database of handwritten digit images for machine learning research [best of the web]," *IEEE signal processing magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[14] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[16] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.