# Meta-Learning For Regression Via Data Re-Weighting

Ansh Sharma, anshgs2@illinois.edu
Stefan Ivanovic, stefan4@illinois.edu
CS 598 LTL - Fall 2022

## Abstract

*For many regression problems in machine learning, one ultimately only cares about accurate prediction in some restricted range. For instance, one may have a scoring function and the ultimate objective is to find the data points which have the highest predicted score. This can be viewed similarly to a class imbalance in which the balance of labels in the training set does not match the test set. Existing methods have corrected by class imbalance by reweighting training data points according to the validation loss [4]. Inspired by this, we apply a meta-learning approach to reweight training data according to validation accuracy. We formulate a transfer learning approach that utilizes unweighted data for training the feature extractor, and weighted data for training the final regression layer. Since our final linear layer is applied for regression rather than classification, we are able to utilize the closed-form solution to weighted least squares, resolving some mathematical issues.*

*We demonstrate our method on the task of predicting ages from facial images. Our test set goal is predicting ages within the specific range 14 to 21, which may be useful for law enforcement purposes. Our method dramatically outperforms training on all of the data and is competitive with training on a validation set already restricted to the relevant ages. We further demonstrate the ability of the model to learn effectively from a very restricted but discriminative set learned by reweighting. This demonstrates that the approach may be extremely useful in cases where one does not know how to restrict the training data a priori but does have a validation data set representative of the testing data, as well as generally powerful even in cases where we do have the ability to restrict.*

## 1. Introduction

There are many situations in machine learning regression in which the final goal is to produce accurate predictions in some range of Y values. One example of this is in finance in which one may train on investments with a wide range of outcomes with the goal of predicting which investments will have highly positive outcomes. In this case, one does not care about accurately differentiating poor investments from very poor investments. Instead, one is only concerned with accurately predicting the highly profitable portion of the Y-value range. Another example is drug discovery. In drug discovery, one may train on a large set of chemicals with widely varying binding affinities to some protein. However, the final goal is to predict which chemicals have a very high binding affinity. Again, one is not concerned about accurately differentiating low and very low binding affinities.

In this paper, we work with the task of age prediction from facial images. The training data consists of facial images ranging from age 1 to 100+ from the UTKFace data set [6]. However, the test objective is to accurately predict facial images for ages 14 to 21. Age prediction in this age may be very useful for law enforcement purposes (monitoring illegal behavior and abuse). In order to achieve Our method utilizes a smaller validation of images with ages from 14 to 21 to learn weightings of the training data which allow for effective regression on the restricted test set.

Figure 1. UTKFace Data Set

## 2. Related Work

Data reweighting and selection has been successfully utilized in various domains in the past based on the intuition that it can help avoid training set bias. Prior works have explored reweighting data to work around noise in data labels [5] as well as on class imbalance between the train and test sets [4]. However, most prior works in data reweighting, as well as within meta-learning, tend to focus on classification tasks. Our paper differs from these prior works as we focus on regression as our primary task, though within a setting that can be viewed as analogous to class imbalance.

Our work is also related to the Teacher-Student model paradigm [1] and curriculum learning paradigm. In these frameworks, prior works have explored utilizing a secondary teacher/mentor network to augment the training procedure as a form of guidance beyond just the loss function. For instance, MentorNet( [3]) explores designing a curriculum iteratively with a secondary model that takes in information about the student's performance on tasks to improve its curriculum. Although our work also explores the impact of reweighting data given information about the model's current performance, our work differs from curriculum design fairly heavily in that we provide a single static weighting as opposed to a fluid curriculum.

Finally, our work also broadly related to the field of meta-learning. Standard meta-learning tasks such as in MAML [2] aim to find features that are easily adaptable to novel tasks without significant computation for the update. Our work approaches a similar problem but from the opposite direction - as opposed to selecting for highly discriminative features, we aim to select highly discriminative training examples that would allow for strong test performance given a fixed feature set.

## 3. Method

Our method starts with a training data set $D_{\text{train}}$, a testing data set $D_{\text{test}}$ and a validation data set $D_{\text{valid}}$ which is representative of the testing data. The training data points are assigned weights $W$ which are set in order to mimic the validation loss. The immediately straightforward for setting the weights $W$ is using the validation loss gradients as well as the gradient of the training loss for each data point to determine which training points are beneficial in each iteration. This was experimented with, giving generally worse unstable results - the details of this are described in further detail in Appendix A. We will now describe the more successful approach, taking a more indirect path towards optimization.

### 3.1. Primary Approach

The regression model for predicting ages from facial images consists of a feature extractor function $f_\psi$ and a linear regression layer $g_\phi$. Initially, the feature extractor and regression layer are trained together on all of the training and validation data (but not the test set data) using gradient descent. We will use $f_\phi$ to refer to this trained feature extractor and $g_{\phi_0}$ to refer to this trained regression layer. This feature extractor then allows us to convert high dimensional images into vector representations which can be used for age prediction.

After finding this feature extractor, the regression layer can be optimized with various different weightings. Let $N$ be the number of training data points and $M$ be the dimension of the dimension of the feature vectors produced by $f_\psi$. Let $R_i$ be the ith data point facial image. Define $X$ as the matrix with $N$ by $M+1$ matrix with $X_ij = f_\psi(R_i)[j]$ for $j \leq M$, and $X_iM = 1$ for all $i$. The last column of $X$ consisting of all ones simply allows for a Y-intercept. This matrix $X$ is the "design matrix", a matrix of input data points for the linear regression problem. Let $W$ be an $N$ by $N$ diagonal matrix with $W_ii$ set as the weighting assigned to data point $i$. Let $\lambda$ be the ridge regression regularization factor. The solution to weighted ridge regression is then the below equation.

$$\phi^*(W) = (X^T W X + \lambda I)^{-1} X^T y$$

For baselines in the paper, we set $W$ as either all ones, or ones for data points with ages between 14 and 21 and zeros otherwise.

For our meta-learning approach, we split off part of the training data containing faces within the desired age range to act as validation data and utilize this to determine $W$. Define $L_{\text{valid}}(\phi)$ to be the $L2$ loss on the validation data as a function of the regression model parameters $\phi$. We then wish to set $W$ to minimize $L_{\text{valid}}(\phi^*(W))$. However, directly optimizing $W$ simply results in heavy over-fitting to the validation set. Instead, we define a function $h_\theta$ which outputs weightings for each data point. In theory $h_\theta$ can take any information about the training data points as an input. However, in practice we find it works best by utilizing $Y_i$ and the initial age prediction found on all the data $g_{\phi_0}(f_\psi(R_i))$.

Define $W_\theta$ as the vector of weight outputs produced by $h_\theta$ on all of the training data. This allows us to calculate the validation loss as a function of $\theta$ as $L_{\text{valid}}(\phi^*(W_\theta))$. We optimize the parameters $\theta$ of our weighting function $h_\theta$ using gradient descent in order to minimize this loss. Define $\theta^*$ as this optimized $\theta$. Our final predictions are given by $g_{\phi^*(W_{\theta^*})}(f_\psi(R_i))$ (for training, validation and testing data).

## 4. Experiments

### 4.1. Dataset Setup

The full UTKFace data set [6] consists of 9780 images, some of which are visualized in Figure 1. Each image is 200 pixels by 200 pixels and had three color channels with values ranging from 0 to 255. The images were downsampled by a factor of 5 to 40 by 40 images to make the machine learning require less computational resources. This was done by averaging the pixel value across 5 by 5 groups of pixels. Additionally, the color channels were normalized to range from 0 to 1. In all experiments, the testing data was formed by first selecting 1000 random images and then restricting to images with ages between 14 and 21. In experiments with a validation data an additional 1000 images were removed from the training data and then restricted to

ages between 14 and 21. The training set consists of all images not in the testing or validation data set, possibly age restricted depending on the experiment. In the cases where the training data is not age restricted (our meta-learning approach and the *All* baseline), the entirety of the training data is utilized, while the *Restricted* baseline is trained on only the subset of the training data that falls within the ages 14 to 21.

### 4.2. Model Architecture

As mentioned in section 3, we subdivide our problem into a few steps. The actual classifier requires first training a feature classifier with standard optimization techniques and then solving for the closed form solution for the linear classifier using Ridge Regression. We choose a 2-layer convolutional neural network with kernel size 7 and feature maps of depth 32 and 16 respectively to serve as our feature extractor, and train it with a linear head on the entire training dataset to learn feature representations.

In order to gain a more robust evaluation of our method, we employ cross-validation with 8 test/train splits, and as such train a separate feature extractor for each of the splits as to avoid data leakage between the test and train data. Both our baselines simply train a linear layer on top of the feature extractor using Ridge Regression with ridge parameter 5e-3 - note that as such there is no additional learning that needs to be done for either of these methods as Ridge Regression has a closed form solution.

For the weight predictor component of our approach, we utilize a small linear network that takes in the predicted age from the feature extractor (with the linear head it was trained with) and the true age as inputs. We find that the specific architecture of this network does not change results too much given that the parameter count isn't excessively high, and settle on a network with one hidden layer with dimension 20. For the final outputs, we do notice reasonable variation in performance depending on the non-linearity utilized. We observe the best performance when the final non-linearity in the weight predictor network is $f(x) = \frac{1}{x^2+1}$. Alternate bounding functions based around tanh and sigmoid seemed to result in inferior results and greater instability. We train each model for 1000 epochs with gradient descent, using ADAM as our optimizer with a learning rate of 1e-4.

### 4.3. Age Prediction Accuracy

We observe strong results from our method with comparison to both baselines, as shown below in Figure 2 and Table 1. We leave out the results for the all baseline in our visualization as they are orders of magnitude off from the other two, and thus make it quite difficult to view the differences between our method and the restricted baseline when included.

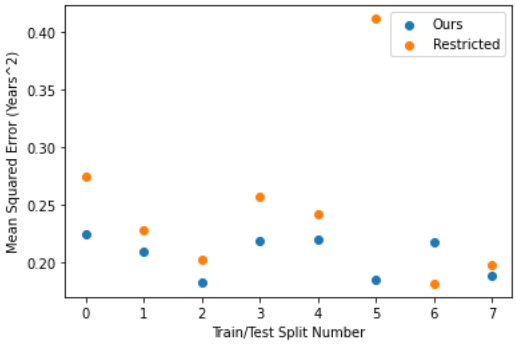


Figure 2. Comparison of MSE for Reweighted and Restricted

| Method | MSE |
|---|---|
| All | $7.955 \pm 1.344$ |
| Restricted | $0.249 \pm 0.067$ |
| Reweighted (Ours) | $\mathbf{0.206 \pm 0.016}$ |

Table 1. Method Results

Our method is not only able to massively outperform the naive all baseline by orders of magnitude, it is effectively always competitive with the restricted baseline and tends to outperform by at least 10 - 15% on nearly all of the train/test splits that we evaluated. This makes it particularly useful in scenarios where creating an extensive restricted dataset may be difficult but acquiring a small clean validation set is not difficult. It's performance improvements are also promising in that it is able to reasonably consistently beat out the restricted baseline in 7 out of our 8 splits, suggesting that the re-weighting method holds significant merit for even tasks where a restricted dataset may exist.

### 4.4. Learned Weighting Results

Beyond just demonstrating the efficacy of our approach, we also seek to provide an understanding of how the learned weightings improve performance, and observe fairly surprising results. As opposed to what one might intuitively expect, the model learns to select a fairly sparse subset of the training data, with significant portions not even within the desired target range, yet still manages to avoid overfitting to the validation set.

We provide heat map visualizations of the weightings learned over various train/test splits in Figure 3. Note that each appears to select for a curve with fairly thin boundary of a couple years at most. In most of the outputs that we tested with, the curves would be semantically similar, generally following an inverse relationship between the predicted and true ages. Despite these semantic similarities, the produced curves for each model were still quite different with regard to exact shape and endpoints, suggesting that the learned reweightings were truly optimized for the fea-

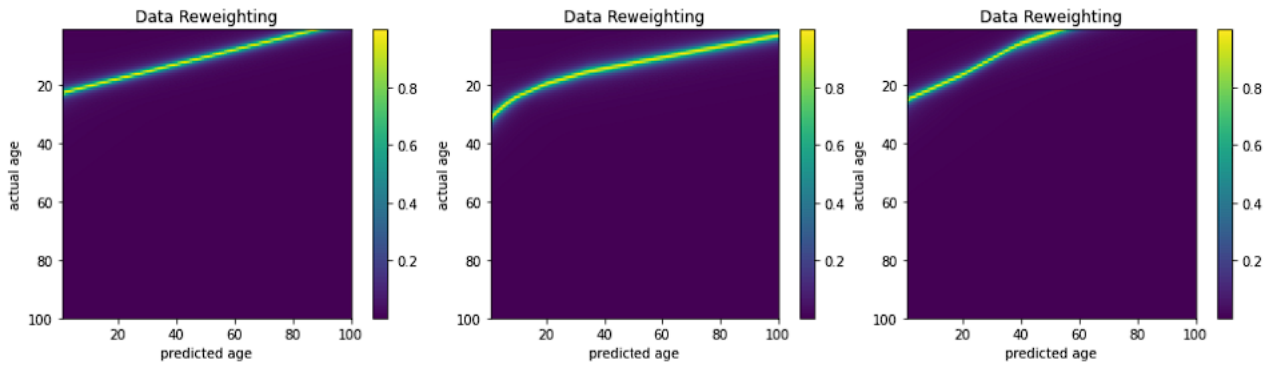


Figure 3. Learned Weighting Heat Maps for 3 Train/Test Splits

tures learned by the feature extractor as opposed to solely picking high quality datapoints that would help any model.

This result was quite interesting to us as we would have expected that the optimal weighting would include most of the relevant data with small portions of the misclassified non-relevant data. However, the model appears to find optimal weightings for the linear layer with just a small number of highly discriminative samples along some form of learned boundary. To give some sense of scale, we typically observe that only around 200 to 400 out of the initial 8000 train datapoints are given a weight of greater than 0.1, which is less than half of the size of the restricted train dataset size.

With this intuition in mind, we can attempt to reorient our understanding of what the learned weightings accomplish. Rather than simply utilizing most of the relevant data and reweighting small portions of the remaining data as one might expect, our model is able to learn a highly discriminative boundary that surgically corrects the inadequacies of the original feature extractor while still avoiding overfitting even with an extremely small number of samples used to train the linear classifier head.

## 5. Conclusion

In regression, one is often concerned with accurate prediction only within some range of Y values. We introduced a generalized technique to learn data re-weightings for regression tasks with such domain mismatches between train and test data. This technique utilizes a feature extractor trained on all the training data and uses validation data to determine training data re-weightings that allow the regression layer to perform well on the testing data. Our reweighting model utilizes information about the feature extractor's deficiencies on a validation set to assign weights to the training data that correct these deficiencies when used to train the linear regression layer head. We demonstrate the technique's strong capabilities in identifying discriminative boundaries to extract the most utility from the provided feature extractor in a single linear layer, both vastly outperforming training on all data by orders of magnitude and reasonably outperforming training on solely restricted data. Our experiments also provide insight into how training on even a low number of datapoints alongside a learned discriminative boundary can help alleviate data imbalance between the test/train sets. One possible direction for future work could be exploring more complex functions for assigning weights to data points, rather than using simple two-layer neural networks. Another direction for future work could by applying our method to situations in which the test domain shift is more complex than restricting to an a-priori known set of Y values.

## References

[1] Yang Fan, Fei Tian, Tao Qin, Xiang-Yang Li, and Tie-Yan Liu. Learning to teach. *arXiv preprint arXiv:1805.03643*, 2018. 2

[2] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017. 2

[3] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. Mentornet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *International conference on machine learning*, pages 2304–2313. PMLR, 2018. 2

[4] Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pages 4334–4343. PMLR, 2018. 1

[5] Jun Shu, Qi Xie, Lixuan Yi, Qian Zhao, Sanping Zhou, Zongben Xu, and Deyu Meng. Meta-weight-net: Learning an explicit mapping for sample weighting. *Advances in neural information processing systems*, 32, 2019. 1

[6] Zhifei Zhang, Yang Song, and Hairong Qi. Age progression/regression by conditional adversarial autoencoder. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5810–5818, 2017. 1, 2

## 6. Appendix A: An Alternative Approach

In this appendix, an approach we previously used is described. We explore some of the substantial mathematical issues and practical difficulties that we encountered while attempting it.

This approach is very general and does not require a separate treatment of the feature extractor and regression layer. Define $f_\theta$ now as the full neural network which inputs facial images and outputs age predictions. Define $L_i(\theta)$ as the L2 loss on training data point $i$ as a function of the network parameters $\theta$. Define $L_{\text{valid}}(\theta)$ as the average L2 loss on the entire validation data set as a function of the network parameters $\theta$. Define $W_i$ as the below dot product.

$$W_i = \frac{dL_i(\theta)}{d\theta} \cdot \frac{dL_{\text{valid}}(\theta)}{d\theta}$$

Conceptually, $W_i$ is positive if minimizing the loss of the training data point $i$ "pushes" the network parameters in a direction that reduces the validation loss. Similarly, $W_i$ is negative if minimizing the loss of the training data point $i$ "pushes" the network parameters in a direction that increases the validation loss. Thus, in a sense, $W_i$ is positive for beneficial data points and negative for detrimental data points. In fact, $W_i$ is the derivative of the validation loss in the next iteration with respect to the weight assigned to data point $i$ evaluated at the current weight being zero. The actual weighting used in each iteration is $ReLU(W_i)$ where $ReLU(x) = x$ if $x > 0$ and $ReLU(x) = 0$ if $x \leq 0$. This $ReLU$ modification is used in order to remove negative weights which may result in over-fitting to the validation data.

### 6.1. Failures Observed in Proposed Method

We initially sought to explore our method on a very simple case to verify its validity for debugging purposes in the future. However, we found that it generally failed on the trivial case of a piecewise affine function with two pieces. Specifically, we set $X$ to be $100$ data points ranging from $-0.5$ to $0.5$. We set $Y$ to be $0$ at $X = -0.5$. We set the slope of $Y$ to be $1$ and the slope of $Y$ to be $4$ when $X > 0.4$ as shown in 4.

The variable $Y$ has no discontinuities as a function of $X$. We defined the validation loss and the test loss to both be the mean squared error on $10$ data points with $X$ ranging from $0.4$ to $0.5$. The optimal solution is for the model to only put weight on data points with $X > 0.4$. This was meant to be a simple test for debugging our approach. However, after it failed, very careful investigation instead allowed us to determine a fundamental mathematical flaw with using data point gradients. Interestingly, this mathematical flaw seems to only be an issue in practice when using simple models such as linear models. Therefore, we were still able to achieve good results on an image regression task described in the third section.

The mathematical flaw expressed itself in the form of y-intercept gradients massively interfering with slope gradients, resulting in misleading signals. As a "hack" solution to this, we modified our task definition to predict the difference in $Y$ values between data points rather than predicting individual $Y$ values. This change in the definition made the y-intercept irrelevant (it cancels out when calculating differences in $Y$ values) and allowed us to somewhat progress. This solution allowed the model to achieve perfect results (finding a slope of $4$) on our previously described $1$ dimensional regression task.

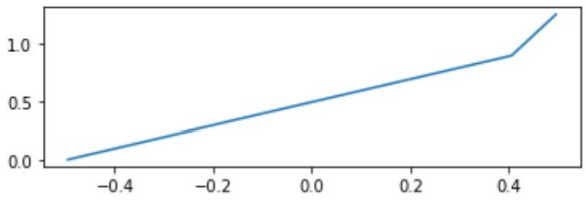


Figure 4. Piecewise Affine Test Case

Additionally the modification, allowed us to move on to two dimensional cases. Specifically, we let $X_1$ and $X_2$ range from $0$ to $1$. Then we defined $Y = 2(X_1)(X_2 - 0.5)$. The test set goal is to identify the top $10\%$ highest $Y$ value data points. Our model is a simple linear model. Not using meta-learning results in a slope of $0$ in $X_1$ (since the positive slope for high $X_2$ exactly cancels the negative slope for low $X_2$). However, applying meta-learning allows the model to ignore data points with low $X_2$ and high $X_1$, resulting in a positive slope for both variables and correctly identifying the highest $Y$ values as coming from high values of $X_1$ and $X_2$. However, the method already started running into substantial issues on similar types of problems using a linear model but with $60$ dimensional inputs. In general, it seems our optimization procedure is difficult and "likes" to run into local optima, cycles, and other major issues. However, as described in the third section, using a high dimensional convolutional neural network avoids these issues.

### 6.2. Mathematical Explanation

The hope our are approach is that a data point having a loss gradient in the same direction as the validation loss gradient is the same as it being beneficial to put some weight on that data point. A more precise formulation of this is as follows. If the data point $x$ has a loss gradient in the wrong direction (with respect to the validation loss gradient) then adding an infinitesimal amount of that data point to our training data should be detrimental. In formal mathematical language we have the following.

**Hypothesis 1.** *Let $\theta*$ be the optimal parameters on the training data set $D$ with data points given weights $W$. Let $d$ be some new data point we are considering adding. Let $g_d$ be the gradient of the training loss function on that data point with respect to the parameters. Let $g_v$ be the validation loss gradient. Let $\theta_\delta$ be the optimal parameters on the training data set $D \cup \{d\}$ with data points given weights $W \cup \{\delta\}$. In other words, $\theta_\delta$ is the optimal solution when the data point $d$ is added with weight $\delta$. Let $L(\theta)$ be the mean squared error validation loss of any model parameters $\theta$. Let the dot product of $g_d$ and $g_v$ be negative, so adding $d$ appears to be detrimental. Then,* $\lim_{\delta \to 0}(L(\theta*) - L(\theta_\delta))/\delta \geq 0$.

There is also the reverse hypothesis about data points with "good" gradients being beneficial to add to the training data with an infinitesimal weight $\delta$. The reason for letting $\delta$ approach $0$ is that we are only using gradient information, and therefore could only hope to have guarantees on infinitesimal effects to our network parameters.

**Theorem 1.** *Hypothesis 1 is false.*

*Proof.* For the sake of saving time since this is not a theory/proof course, we will just sketch the proof. Imagine the data point $d$ moves the slope in the direction that decreases the validation loss and $d$ moves the y-intercept in a direction that increases the validation loss. Then, by shifting the $X$-scale (multiplying all $X$ values by some constant) one can shift the scale of slope gradients. Therefore, one can change whether or not the dot product of $g_d$ and $g_v$ is positive or negative. With sufficiently large $X$ values, one can force the dot product to be negative even if $\lim_{\delta \to 0}(L(\theta*) - L(\theta_\delta)) > 0$. Even if one normalizes $X$, one either the training or validation set, one can still have the same problem due to the other one not being normalized. This can't be fixed because one can't simultaneously normalize both.

Additionally, there are failure cases which have nothing to do with normalization. Specifically, there exists cases where the slope/intercept gradient $g_d$ is positive despite the slope/intercept in $\theta_\delta$ being larger than the slope/intercept in $\theta*$ even as $\delta \to 0$. This happens as a consequence of the optimal slope/intercept being dependent upon the current intercept/slope and so the gradient is not all-ways in the same direction as the direction of the optimal solution even for infinitesimal $\delta$.

This demonstrates a potential flaw with using our method. However, in practice, only seems to be catastrophic when using simple models. Intuitively, only using data points that help the validation loss is a form of restricting the gradient directions. When using a simpler model, these restrictions can completely prevent effective optimization. However, when using high-parameter models, there are alternative beneficial directions in which model parameters can move.

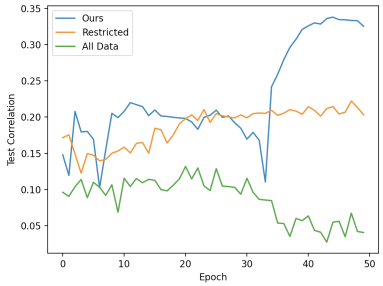


Figure 5. Correlation Curves for Direct Reweighting Approach

We observe in practice that the model is able to learn reasonably in this method. A training plot over 50 epochs is provided in Figure 5 where the initial CNN is fine tuned on either our re-weighting of the data, a fully restricted dataset, or the entire dataset, and we can observe the seemingly improved performance of our method over both baselines. However, stability is highly lacking, as demonstrated by the sharp dips and jumps in the correlation curve, suggesting that alternative methods with higher stability could show more promise as we explore in the main section of the paper.

In contrast, we provide a visualization of one sample's test correlation curve in Figure 6 with the method we explore in the main body of the paper, consisting of a much smoother curve and higher correlation over 200 epochs. We do note however that correlation ended up being a relatively more unstable measure, so we eventually pivoted to using a standard mean squared error loss as reported in the main body of our paper. Correlation was initially proposed as the evaluation metric due to concerns that having an unrestricted training dataset for our and the all baseline methods would unfairly penalize them for learning on out of domain predictions while the restricted dataset would avoid that, but we found in practice that our method continues to outperform the restricted baseline even with this disadvantage.

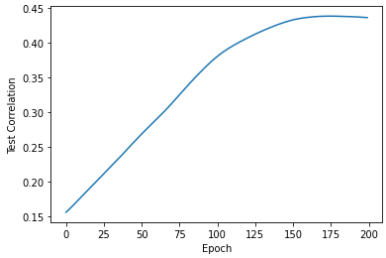


Figure 6. Correlation Curve for Indirect Reweighting Approach