Proximal Regularization of Deep Residual Neural Networks Applied to High-dimensional Data

Anonymous authors
Paper under double-blind review

Abstract

Residual neural networks (ResNets) have become widely used as they allow for smooth and efficient training of deep neural network architectures. However, when trained on small, noisy and high-dimensional data, ResNets may suffer from overfitting due to the large amount of parameters. As a solution, a range of regularization methods have been proposed. One promising approach relies on the proximal mapping technique which is computationally efficient since it can be directly incorporated into the optimization algorithm. However, the performance of ResNets with various convex or non-convex proximal regularizers remains under-explored on high-dimensional data. In our study, we develop a stochastic adaptive proximal gradient ResNet method that can handle both convex and non-convex regularizers that range from L_0 to L_∞. Moreover, we evaluate the prediction performance in a supervised regression setting on three high-dimensional genomic data sets from mice, pig and wheat. Traditional sparse linear proximal gradient methods are also implemented with the same regularizers and evaluated for comparison. Experimental results demonstrate that a ResNet with 18-layers and $L_{\frac{1}{2}}$ regularization outperforms other configurations on both mice and pig datasets, as well as the sparse linear proximal gradient methods across all the datasets. For the wheat data, a 15-layer ResNet configuration achieves the lowest test mean squared error. These findings highlight the effectiveness of our regularized adaptive proximal gradient ResNet method and its potential for prediction tasks on high-dimensional genomic data.

1 Introduction

ResNets were introduced to address the difficulties associated with training of very deep neural networks (He et al., 2016a). The main innovation of ResNets are skip connections which enable the network to learn residual functions with reference to the layer inputs instead of learning unreferenced functions. This architecture is beneficial in addressing the vanishing and exploding gradient problems. ResNets have emerged as a powerful tool for nonlinear regression and classification, demonstrating robustness to extrapolation and handling "out-of-distribution" data samples effectively (Bondarenko, 2021; Ge et al., 2022). By incorporating shortcut residual connections, it has become possible to train much deeper multilayer perceptrons (MLPs) and convolutional neural networks (CNNs) than was previously possible. ResNets have achieved record low error rates, some important examples include Wide ResNet (Zagoruyko, 2016), ResNeXt (Xie et al., 2017) and PyramidNet (Han et al., 2017). However, the ResNet architecture is on its own not sufficient to reduce the generalization error because of overfitting.

To improve the performance of ResNets, different regularization methods that place more coherent constraints on the model parameters have been proposed (Miyato et al., 2018; Santos & Papa, 2022). Data augmentation techniques artificially increase the diversity of training data, helping to increase the robustness of the model (Shorten & Khoshgoftaar, 2019). Early stopping halts training when the validation loss ceases to improve, preventing the model from overfitting (Prechelt, 1998). Label smoothing softens hard labels, preventing overconfidence in predictions and improving generalization (Müller et al., 2019). The dropout randomly deactivates neurons during training, forcing networks to learn redundant features, which reduces overfitting (Srivastava et al., 2014). Other widely used regularization methods include stochastic gradient descent

(Zhang et al., 2021) and weight decay (Krogh & Hertz, 1991). Batch normalization techniques allow for smooth gradient flow and stable training (Ioffe, 2015; Santurkar et al., 2018). Unfortunately, the regularization effect of these approaches is generally too weak on high-dimensional input data.

High-dimensional data analysis is a rapidly growing field of research that focuses on data with a large number of input variables, often much larger than the sample size. For example, high-throughput measurements in genomics often contain thousands or millions of variables, such as single nucleotide polymorphism (SNP) markers and gene expression data, for each individual observation (Davey et al., 2011; de Los Campos et al., 2013). The interest in applying deep learning (DL) to various regression and classification problems is increasing dramatically in many areas, so also in genomics (Eraslan et al., 2019; Yue et al., 2023). However, when dealing with genomic data, the challenge of fitting ResNets becomes even more pronounced, not only because of the high dimensionality but also because of the inherent noise within the data. Traditional data augmentation and training-based regularization methods can mitigate some overfitting, but they often lack effectiveness and are not well suited for structured constraints specific to genomic prediction. On the other hand, proximal methods can easily enforce sparsity and other structural constraints on model weights, which is particularly beneficial for high-dimensional genomic data (Fan et al., 2024).

Proximal mappings offer a principled way of efficient regularization on each residual block of a ResNet architecture that provides a coherent iterative optimization framework (Bai et al., 2018). Adaptive proximal gradient methods allow dynamic adjustments to learning rates while simultaneously incorporating regularization terms, effectively balancing model complexity, weight sparsity and predictive accuracy (Yun et al., 2020). This dual capability makes them particularly well-suited for high-dimensional genomic datasets, where many input variables are redundant. In this study, we address two primary objectives: (1) Develop stochastic adaptive proximal gradient methods for ResNets incorporating a broad range of regularization functions. (2) Conduct a comprehensive evaluation of the impact of network size and the various regularization functions on predictive accuracy in high-dimensional genomic prediction of continuous phenotypes. By systematically exploring network size and employing different regularization strategies, we aim to identify optimal configurations that effectively balance model complexity with predictive performance.

2 Related Work

To improve the training and performance of a deep neural network, the issues of shallow versus deep networks have been extensively discussed in machine learning (Larochelle et al., 2007). Some studies have tried to make ResNets as thin as possible by increasing the depth via bottleneck blocks (Zagoruyko, 2016). ResNets address the depth problem by allowing the construction of very deep architectures without suffering from vanishing or exploding gradient issues, which typically limits gradient flow during backpropagation in DL. Additionally, identity mappings in residual blocks have been shown to accelerate training in very deep networks, as they allow gradients to pass through unchanged across multiple layers (He et al., 2016b). Wide ResNets were designed to prioritize width while reducing depth (Zagoruyko, 2016). These wide blocks allow ResNets to capture high-dimensional features efficiently by compressing the number of parameters in intermediate layers.

Training-based regularization methods play an important role in enhancing generalization and robustness of ResNets. Dropout (Srivastava et al., 2014) is one of the most popular methods for reducing overfitting which works by randomly deactivating a fraction of neurons during each forward pass. Early stopping (Prechelt, 1998) has shown to be a simple but powerful technique, stopping training at the point where validation error starts to increase. Data augmentation (Shorten & Khoshgoftaar, 2019) can enrich the training dataset by applying transformations such as rotations, cropping, or flipping of data, which introduces variability and prevents memorization of the training set. Label smoothing (Müller et al., 2019) and stochastic depth (Huang et al., 2016) have also been shown to improve ResNets' performance, particularly in settings where overfitting is of concern. Furthermore, ShakeDrop (Yamada et al., 2019) is a dynamic regularization strategy for deep ResNet architectures, which injects stochasticity into the training process by perturbing residual blocks. These methods have been crucial for training deep networks.

Normalization techniques have been introduced to improve gradient flow and stabilize training. Batch Normalization (BatchNorm) (Ioffe, 2015; Wu et al., 2024) has become standard in CNNs, improving both

the speed and the stability of training. BatchNorm works by normalizing the activations of the input layer during training, which prevents small changes in weight updates from being amplified between layers. In contrast, layer normalization (LayerNorm) (Ba, 2016) and weight normalization (WeightNorm) (Salimans & Kingma, 2016) are designed for recurrent neural networks (RNNs) and fully connected architectures, where traditional normalization techniques often fail. Based on the Dropout approach, (Liu et al., 2023) proposed a regularization method for deep neural networks that penalizes the trace of the Hessian. However, normalization techniques can be disrupted by stochastic training methods, such as stochastic gradient descent (SGD) and dropout, leading to a high variance in training error (Bousquet & Elisseeff, 2002).

Adaptive gradient methods are another important way to optimize deep neural networks, especially when fixed learning rate schedules under-perform (Zhao & Huang, 2023). Although the traditional SGD method is suitable for many problems, it often struggles with slow convergence due to its sensitivity to learning rate selection. Adaptive gradient methods like Adam (Parikh & Boyd, 2014; Lee & Lee, 2019), RMSProp (Yang et al., 2020), and Adagrad (Duchi et al., 2011) have been developed to dynamically adjust learning rates, enabling faster convergence and improving stability during training (Duchi et al., 2011). Adagrad modifies the learning rate for each parameter based on the historical sum of squared gradients. This adjustment prevents large updates to frequently occurring features, ensuring that less frequent features receive more significant updates. However, Adagrad's learning rate tends to shrink excessively over time, limiting its effectiveness for long-term training (Duchi et al., 2011). RMSProp addresses this issue by maintaining a moving average of squared gradients, which decays exponentially over time. This prevents learning rates from decaying too quickly (Tieleman & Hinton, 2012). Adam further builds on RMSProp by incorporating momentum, which helps smooth out noisy gradient updates. Adam maintains two moving averages: one for the gradient (first moment) and one for the squared gradient (second moment), which are used to adapt the learning rates for each parameter. Hence, Adam benefits from both momentum-based and adaptive learning methods, making it one of the most widely used optimizers in deep learning (Kingma, 2014). Using the non-linear self-adaptive mechanism, self-adaptive gradient descent search algorithm (SaGDSA) can guide the whole search process and facilitate escape from local optima (Xue et al., 2022). The adaptive gradient methods have been effective in various deep learning tasks, especially in applications to high-dimensional data (Becker et al., 2019).

Proximal regularization techniques are powerful techniques for improving the generalization and performance of neural networks. Unlike traditional gradient descent, which updates parameters based solely on smooth gradients, proximal gradient methods incorporate a regularization step that allows for non-smooth constraints to be imposed on the model parameters (Beck, 2017; Klambauer et al., 2017). ProxQuant reformulates quantized neural network training as a regularized optimization problem, allowing the use of proximal gradients to optimize quantized weights efficiently (Bai et al., 2018). Proximal methods have also been integrated with adaptive first-order methods like Adam and RMSProp, enhancing their ability to handle non-smooth regularizers (Lee & Lee, 2019). By implementing the adaptive step size scheme from the optimization methods Adagrad and Adam, AdaProx (Melchior et al., 2019) avoids the need for explicit computation of the Lipschitz constants and reduce per-iteration cost. Proximal methods allow the regularization to be applied separately from the gradient update, thus enabling the model to handle non-differentiable points and achieve better generalization while maintaining computational efficiency (Becker et al., 2019). Weighted proximal operators have been proposed to incorporate the adaptive learning rates in Adam and RMSProp (Ukil et al., 2021). Based on the idea of weight decay, (Yang et al., 2022) presented Pathprox which provides an alternative proximal gradient algorithm for network training with effective regularization.

Unlike traditional regularization methods, which primarily focus on data augmentation or weight decay, our method enables more precise control over model complexity by adjusting regularization terms dynamically, thereby reducing redundancy in the learned features and improving the model's generalization ability. Our paper present an efficient stochastic adaptive proximal gradient ResNet method, which incorporates a wide range of convex and non-convex regularizers, from L_0 to L_∞, into the optimization process. The motivation behind this approach is to address the overfitting issue that arises in high-dimensional, noisy data by using proximal mappings to enforce structural constraints on the model's parameters.

3 Methods

3.1 Problem Formulation

ResNets can be susceptible to overfitting when trained on small, noisy or high-dimensional datasets due to their deep architecture and large number of parameters (Hinton et al., 2012). Regularization is an essential strategy to mitigate this issue by imposing constraints on the network's structure and enhancing generalization (Ukil et al., 2021). The optimization problem for training networks with regularization can be formalized as:

$$\underset{\theta\in\mathbb{R}^d}{\text{minimize}} \quad \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda\mathcal{R}(\theta), \tag{1}$$

where $\theta \in \mathbb{R}^d$ denotes the network parameters, $\mathcal{L}(\theta)$ is the empirical loss function, $\lambda > 0$ is a regularization parameter and $\mathcal{R}(\theta)$ is a data-independent regularizer, which encourages more effective models and leads to a better estimation (Bousquet & Elisseeff, 2002). Proximal gradient methods are particularly well-suited for solving such optimization problems as they incorporate regularization terms, including non-differentiable ones, directly into the training process (Liang et al., 2023). These methods can ensure that the network parameters are within reasonable bounds and prevent them from becoming excessively large, which would otherwise have a negative impact on predicting unseen data.

Minimizing the above objective function defined is typically accomplished using stochastic first-order optimization techniques applied to the k-th random mini-batch of the data:

$$\underset{\theta\in\mathbb{R}^d}{\text{minimize}} \quad \mathcal{F}(\theta) = \sum_{k=0}^{n-1} \mathcal{F}_k(\theta). \tag{2}$$

3.2 Adaptive Proximal Gradient Descent Methods for ResNets with Various Regularizer

Adaptive proximal gradient descent methods (Duchi et al., 2011; Tieleman & Hinton, 2012) adjust the learning rate based on past gradient information, introducing a form of preconditioning by scaling gradients based on their historical magnitudes. By combining proximal updates with adaptive preconditioning, the recently introduced ProxGEN offers a robust and flexible optimization method that addresses complex regularization of both non-smooth and non-convex regularization terms in neural network training (Yun et al., 2021). Based on the ADAM optimizer (Kingma, 2014) and the idea of diagonal preconditioners (Pock & Chambolle, 2011), the preconditioner for ProxGEN can be formulated as $C_t = \sqrt{\beta_2 C_{t-1} + (1-\beta_2) g_t^2}$, where g_t is the gradient of a stochastic mini-batch at iteration t and $\beta_2 \in [0, 1)$. Given a non-empty proximal mapping

$$\text{prox}_f(\theta) = \arg\min_z \{f(z) + \frac{1}{2}\|z-\theta\|^2\}, \tag{3}$$

the update rule for the weight parameters then becomes

$$\theta_{t+1} = \text{prox}^{C_t+\delta I}_{\alpha_t\lambda\mathcal{R}(\cdot)}(\theta_t - \alpha_t(C_t+\delta I)^{-1} m_t), \tag{4}$$

where $m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$ represents the first-order momentum, θ_t is the parameter vector at iteration t, $\lambda > 0$ is the penalty factor, α_t is the adaptive learning rate and θ_{t+1} is the new parameter vector at iteration $t+1$. The small constant δ ensures the positive definiteness of the matrix $C_t + \delta I$. One of the key challenges with the momentum m_t in above equation is the initialization bias that occurs when m_{t-1} is set to zero at the start of optimization. When g_t is large, this bias will hinder convergence, especially early in training. The bias correction method in ADAM is designed to mitigate this initialization bias using the decay rates β_1 and β_2 for the first and second moments, respectively. Algorithm 1 outlines the full ProxGEN method for a general choice of differentiable loss and closed-form proximal regularization functions.

We now take a look at how to generalize the regularization function $\mathcal{R}(\theta)$. The closed-form solution for a general L_q regularization problem can be written as

$$\hat{\theta} = \arg\min_\theta \{(\theta - z)^2 + \lambda\|\theta\|^q\}, \tag{5}$$

Algorithm 1 Adaptive Stochastic Proximal Gradient Descent for ResNets with Regularization

Require: Stepsize sequence $\{\alpha_t\}_{t=1}^{T}$, momentum parameters $\beta_1 \in [0,1)$, $\beta_2 \in [0,1)$, regularization parameter λ, small constant $\delta > 0$, regularization function $\mathcal{R}$ (e.g., L_q, MCP, SCAD).

1: **Initialize:** Network parameters $\theta_1 \in \mathbb{R}^p$, momentum $m_0 = 0 \in \mathbb{R}^p$, preconditioner matrix $C_0 = 0 \in \mathbb{R}^{p\times p}$.

2: **for** $t = 1, 2, \ldots, T$ **do**

3: Compute the gradient on a random mini-batch: $g_t \leftarrow \nabla_\theta \mathcal{L}(\theta_t)$

4: Update the momentum: $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$

5: Update the preconditioner: $C_t \leftarrow \sqrt{\beta_2 C_{t-1} + (1-\beta_2) g_t^2}$

6: Update the weight parameters using the proximal operator:

$$\theta_{t+1} = \text{prox}_{\alpha_t \lambda \mathcal{R}}^{C_t + \delta I}\left(\theta_t - \alpha_t (C_t + \delta I)^{-1} m_t\right)$$

7: **end for**

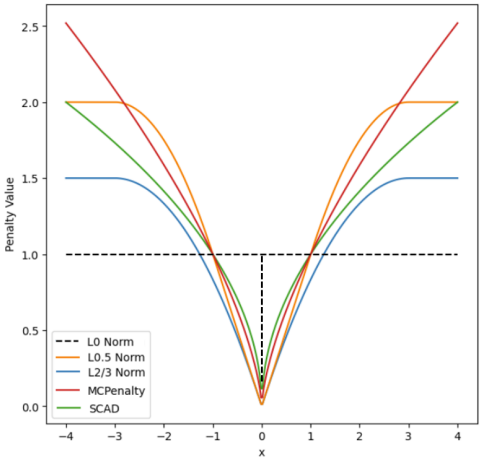
where $\mathcal{R}(\theta) = \|\theta\|^q$ represents the L_q penalty function for $0 \leq q \leq \infty$. We extend the application of sparse L_q regularization beyond the conventional range $0 \leq q \leq 1$ to encompass cases where $q > 1$, and therefore end up with regularizers with the following penalty functions: $L_0, L_{\frac{1}{2}}, L_{\frac{2}{3}}, L_1, L_{\frac{4}{3}}, L_{\frac{3}{2}}, L_2, L_3, L_4$ and L_∞. In addition, we also explore the smoothly clipped absolute deviation (SCAD) (Mehranian et al., 2013) and the minimax concave penalty (MCP) (Zhang, 2010) regularizers. This set of regularizers provides deeper insights into the behavior of various proximal operators and their effects on ResNets in the case of high-dimensional genomic datasets. Figure 1 compares these non-convex and convex penalty functions and their corresponding proximal operators for a single parameter $\theta \in \mathbb{R}$. The mathematical details of the different regularization functions and their proximal operators are provided in the Appendix A.

3.3 Convergence Guarantees

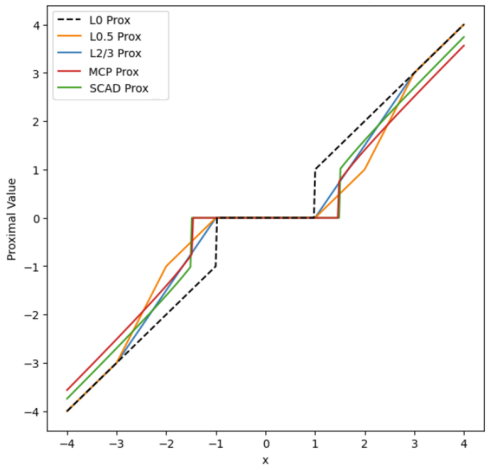
In this study, we consider three different kinds of penalty functions: the convex norm functions L_q where $q \geq 1$, the non-convex fractional quasinorm functions $L_{\frac{1}{2}}$ and $L_{\frac{2}{3}}$ which descend strictly towards the zero vector from all directions, and the L_0 penalty function which is non-convex and has a zero gradient at all points where it is differentiable, and is therefore problematic. In existing research work (Yun et al., 2020), it has been shown that the ProxGEN family of stochastic proximal gradient descent methods converge in mean to a stationary point, which however might not be a global minimum. For the L_0 penalty function, the stationary point will not be a global minimum unless further narrow restrictions are made on the number of non-zero components and the initial parameter vector is chosen well enough.

For the adaptive proximal methods using ResNets with various regularizations, the goal is to find an ϵ-stationary point for the minimization problem (1) where ϵ is the required precision. To derive the convergence bound, some mild conditions were assumed to hold in (Yun et al., 2020): $L-$smoothness $(\mathbf{C-1})$, bounded variance $(\mathbf{C-2})$, final step-vector conditions $(\mathbf{C-3})$, and sufficient positive-definitess $(\mathbf{C-4})$. Among these conditions, $(\mathbf{C-1})$, $(\mathbf{C-2})$ are standard in convergence analysis for optimization algorithms designed for deep learning such as ADAM (Bai et al., 2018; Melchior et al., 2019). The conditions $(\mathbf{C-3})$ and $(\mathbf{C-4})$ are plausible, since it is well-known that the parameter vector of an overparametrized neural network hardly changes from the initial point during the training (Du et al., 2018). The details of guaranteed convergence for the minimization problem (1) with non-convex and convex penalty functions are provided in the Appendix B. Based on Theorem 1 for general convergence and Corollary 1 for constant mini-batch sizes shown in (Yun et al., 2020), we make the following remarks about our results and their relationship with prior work:

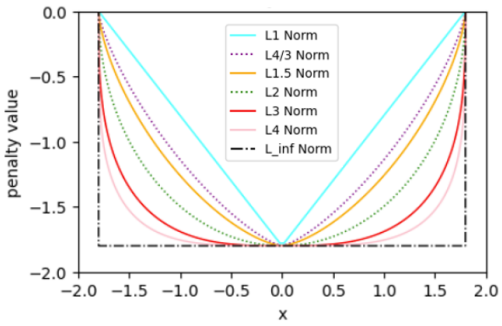
- **On convergence results.** Corollary 1 in (Yun et al., 2020) states a computational complexity of order $\mathcal{O}(1/\epsilon^4)$ for SGD to find ϵ-stationary points, assuming the four conditions $(\mathbf{C-1})$ to $(\mathbf{C-4})$ for non-convex cases. Regarding the convex regularizers, the convergence to a global minimum is well-established under standard assumptions. For high-dimensional genomic data, the upper bounds for the norms of the step size and the gradient should have small positive constant values such that



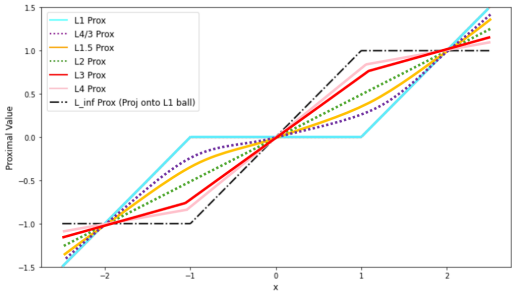
(a) Non-convex penalty functions.



(b) Proximal operators of non-convex penalty functions.



(c) Convex penalty functions.



(d) Proximal operators of convex penalty functions.

Figure 1: Comparison of non-convex and convex penalty functions and their proximal operators.

$\|\theta_{t+1} - \theta_t\| \leq D$ and $\|g_t\| \leq G$, where $D > 0$ and $G > 0$. This helps to avoid slow convergence and to adjust the gradient's variation.

- **Advantages of adaptive proximal methods.** According to Theorem 1 in (Yun et al., 2020), the constant γ, which affects the convergence, depends on the constants Q_1, Q_2, and Q_3 that are proportional to $1/\gamma$. For the genomic data, we set $C_t = \sqrt{(1-\beta_2)\sum_{\tau=1}^{t}\beta_2^{t-\tau} g_\tau \odot g_\tau}$, with $\beta_2 \in [0, 1)$. With the stepsize α_t we get

$$Q_i \propto \frac{1}{\gamma} = \frac{\sqrt{(1-\beta_2)\sum_{\tau=1}^{t}\beta_2^{t-\tau}\|g_\tau\|_2^2} + \delta}{\alpha_t} \leq \frac{G+\delta}{\alpha_t}, \tag{6}$$

  where g_τ is the gradient at time τ. If g_τ is very small, then $1/\gamma$ will be quite small due to the variation of α_t in the training. This coincides with the convex regret theory for adaptive gradient methods (Duchi et al., 2011; Kingma, 2014), which also holds for the various regularizers on high-dimensional genomic data.

- **Implications of condition $(\mathbf{C}-\mathbf{4})$ for genomic prediction.** Condition $(\mathbf{C}-\mathbf{4})$ enforces that the Hessian matrix is positive-definite. This ensures that the model does not get stuck into saddle points or flat regions of $\mathcal{F}(\theta)$, which is a significant issue for high-dimensional genomic datasets. This doesn't however apply to the L_0 penalty function, which is notoriously difficult to approximate, even

when the algorithm is allowed to "cheat" with regard to both the sparsity of the solution output and the accuracy of the resulting linear combination (Foster et al., 2015).

- **On mini-batch in Corollary 1 in (Yun et al., 2020).** Under the same assumptions regarding the sample size of mini-batches as in Theorem 1, the guarantee of convergence given by Corollary 1 is crucial for applications to high-dimensional genomic data, where mini-batching allows the methods to scale efficiently to large datasets while still maintaining convergence.
- **Connection to second-order methods.** The adaptive proximal gradient methods provide a flexible ground, approximating second-order behavior through decay rates and proximal updates, but at a decreased computational cost. These methods allow for efficient handling of complex, non-convex loss functions for high-dimensional data. Their adaptive nature makes them more robust to noise and the complexity of genomic data than pure first-order methods.

4 Implementation Details

4.1 Network Architectures

We adopt a highly flexible design following ResNets. Our network consists of basic block ResNets from 5 layers to 18 layers and of bottleneck block ResNets beyond 18 layers. Figure 2 shows the basic block and the bottleneck block used in this work. A residual block can be formulated as

$$X_{l+1} = X_l + \mathcal{E}(X_l, \theta_l), \tag{7}$$

where X_l and X_{l+1} are the input and the output of the l-th unit in the network, respectively, and $\mathcal{E}$ is a sequence of 1D convolution, batch normalization and ReLU activation, followed by another 1D convolution and batch normalization. As described in (He et al., 2016a), a ResNet consists of sequentially stacked residual blocks. For the the basic block, it contains two consecutive 1×1 convolutions with batch normalization and ReLU proceeding convolution, where a stride of 2 is used for dimension reduction in the 1×1 convolution. For the bottleneck block, ResNet utilizes a 1×1 or a 3×3 convolution to adjust the number of channels, where a stride of 1 is applied to the other convolutions to improve computational efficiency. In comparison to previous architectures (Zagoruyko, 2016), the use of varying strides and residual blocks mitigates the loss of information at each layer.

4.2 Implementation Details

We consider several important tasks for regularized training: (1) finding the optimal configurations of network for different high-dimensional genomic data, (2) discovering the best regularizer over various real-world data, and (3) comparing the performance of networks with the traditional sparse proximal methods. The code for all the algorithms are available online at: https://github.com/angelYHF/Adaptive-gradient-methods.

Training ResNets. We consider training ResNets on three real genomic datasets using various regularizers. Toward this, we use the objective function with convex and non-convex L_q regularization: $\mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda \sum_{j=1}^{p} |\theta_j|^q$, where $0 \leq q \leq \infty$. We train the parameters with the closed-form proximal mappings introduced in Appendix A. Throughout the experiments, we consider ADAM as a representative of ProxGen with $\beta_1 = 0.1$ and $\beta_2 = 0.999$ in Algorithm 1. For the adaptive proximal gradient methods with different regularizers, a ResNet-15 model was initially employed, as it typically offers a balanced trade-off between model complexity and the risk of overfitting. During the training, the validation mean squared error (MSE) was monitored closely. If the model failed to adequately capture the data complexity, increasing the network size was considered. Conversely, if overfitting persisted despite regularization efforts, a smaller network with fewer layers was evaluated. Additionally, hyperparameter tuning was conducted to determine optimal values for the learning rate, batch size and regularization parameters.

Training traditional sparse proximal models. For the purpose of comparison, we use a linear regression model with various regularizers, where the penalty function varies from L_0 to L_∞. By using the closed-form solutions for different regularizers, we use the proximal gradient method to find the best regularization factor

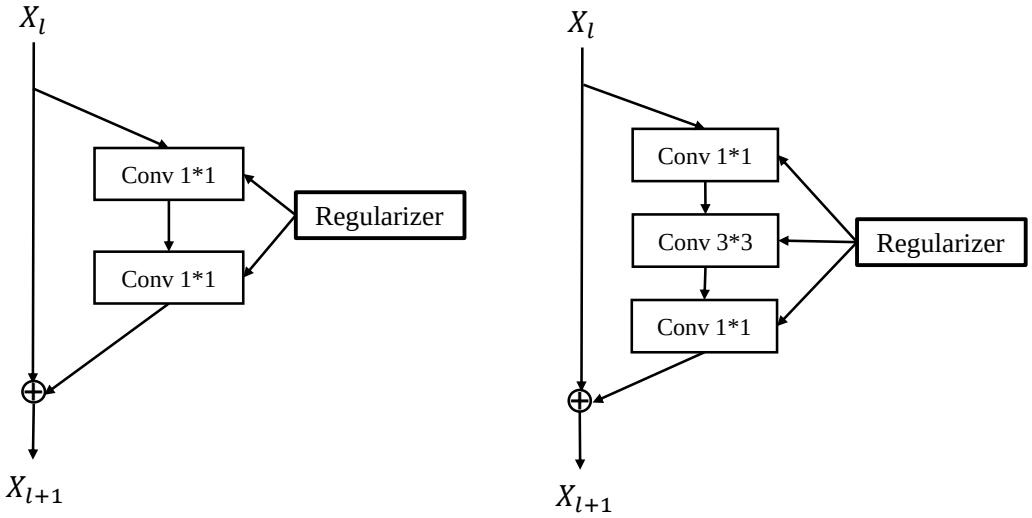


Figure 2: Residual Functions $\mathcal{E}$. Left panel: a basic block used in ResNets with 5 to 18 layers. Right panel: a bottleneck block with three convolutional layers used in ResNets with 20 to 25 layers. Batch normalization and ReLU activation precede each convolution (omitted for clarity).

and the test MSE based on Bayesian optimization (Frazier, 2018). Performance metrics and evaluation frameworks were aligned with earlier work (Fan et al., 2024), enabling a thorough comparison of results.

Tuning parameters. To achieve the best configurations, we utilized Bayesian optimization to find the best network configurations. The proposed approach involved initially dividing the data randomly into 5-fold cross-validation sets. To improve the computation efficiency, models were executed in parallel across each fold, with the test MSE calculated independently. The BO process was terminated by monitoring the improvement in model performance over the iterations. To ensure diverse exploration of the parameter space, the Tree-structured Parzen Estimator (TPE) method was used within BO to leverage the model's inherent stochasticity while integrating new recommendations (Frazier, 2018). The BO was implemented using the Hyperopt library (Bergstra et al., 2013).

Parallel computing. All models were implemented using the PyTorch framework, and experiments were run on five NVIDIA GPUs to accelerate training. Parallelization across multiple folds was employed during cross-validation to enhance computational efficiency. The final model configurations were determined by running BO for a maximum of 100 iterations, with convergence considered achieved when the improvement in validation MSE was less than 10^{-5} for five consecutive iterations.

5 Experimental Results

5.1 Material

Mice data: The first data of this study is the mice data. It is a part of the BGLR package in R (Pérez & de Los Campos, 2014), but originally comes from the Wellcome Trust (http://gscan.well.ox.ac.uk) and has been used for whole-genome regression in several earlier studies (Legarra et al., 2008; Okut et al., 2011). It consists of genotypes and phenotypes of 1,814 mice. Each mouse was genotyped at 10,346 single nucleotide polymorphisms (SNPs) that were coded as 0, 1 and 2. Here we use two continuous traits, body length (BL) and body mass index (BMI).

Pig data: The largest tabular data set in our study is the pig data (Cleveland et al., 2012) which contains 3534 individuals with high-density genotypes and continuous phenotypes of five anonymized traits. After

cleaning some missing data, we finally obtained 2314 samples, where each sample contains 52,843 SNPs. The data was anonymised by randomising the map order and recording the SNP identities.

Wheat data: The wheat data set originates from CIMMYT's Global Wheat Program and is a publicly available in the BGLR package (Pérez & de Los Campos, 2014). It comprises 599 wheat lines from the CIMMYT Global Wheat Program evaluated in four international environments representing four basic agroclimatic regions (mega-environments). The wheat lines were genotyped using 1,447 Diversity Array Technology (DArT) markers. As a quality control, all the markers with a minor allele frequency below 0.05 were eliminated, and any missing genotypes were imputed once using samples from the marginal distribution of marker genotypes. Following these procedures, the dataset was reduced to 1,279 DArT markers.

5.2 Comparison of Adaptive Proximal Methods on Different Datasets

Mice data: In the mice data, we observed a consistent improvement in test MSE as the depth of the ResNet increased up to 18. For instance, in the case of the L_0 regularizer, the test MSE decreased consistently from ResNet-5 (0.144 $\pm$ 0.003 for Trait 1 and 0.148 $\pm$ 0.003 for Trait 2) to ResNet-25 (0.136 $\pm$ 0.002 for Trait 1 and 0.139 $\pm$ 0.001 for Trait2). A similar trend was observed for most other regularizers, indicating that deeper networks generally improve performance. However, this improvement would not continue beyond ResNet-18, particularly for regularizers L_2 and L_3, where no significant performance gains were observed for ResNet-18 and ResNet-25. This demonstrates that deeper networks can improve feature learning but may not always result in better prediction after a specific depth.

We can also notice that different regularizers impact performance in varying degrees. For Trait 1, the best performance was achieved with $L_{\frac{1}{2}}$ regularization on ResNet-18 (Trait 1: 0.130 $\pm$ 0.001, Trait 2: 0.131 $\pm$ 0.002). However, for Trait 2, the worst result was obtained with the SCAD on ResNet-18 (Trait 1: 0.138 $\pm$ 0.002, Trait 2: 0.142 $\pm$ 0.002). The performance differences across regularizers are not too big, but non-convex penalties like MCP and SCAD tend to perform slightly worse than $L_{\frac{1}{2}}$ and L_2 for both traits across all network depths. The choice of regularizer and network depth also has a significant impact on the genomic prediction, with $L_{\frac{1}{2}}$ and L_2 showing the most consistent results across different ResNet depths for both traits. However, deeper networks, particularly those with more than 18 layers, may not provide substantial improvements over shallower ones when appropriate regularization was applied.

Pig data: The pig data is the largest among the three datasets used in this study, comprising of five traits. Table 1 summarizes the test MSE (mean $\pm$ stddev) of ResNet-18 with various regularizers on this data. We can clearly see that there was a significant improvement in performance across most regularizers for all five traits when the depth of the ResNet increased from 5 to 18 layers. For instance, with the $L_{\frac{1}{2}}$ regularizer, the test MSE decreased consistently from ResNet-5 (Trait 1: 0.140 $\pm$ 0.003, Trait 2: 0.143 $\pm$ 0.002, Trait 3: 0.140 $\pm$ 0.004, Trait 4: 0.138 $\pm$ 0.003, Trait 5: 0.147 $\pm$ 0.004) to ResNet-18 (Trait 1: 0.129 $\pm$ 0.005, Trait 2: 0.130 $\pm$ 0.004, Trait 3: 0.129 $\pm$ 0.004, Trait 4: 0.126 $\pm$ 0.001, Trait 5: 0.133 $\pm$ 0.002). Similar trends were observed for the other regularizers, indicating that increasing the number of layers improves ResNet performance by reducing test MSE. However, increasing the network depth from 18 to 25 layers showed only marginal gains: Trait 5 with L_1 regularizer improved slightly from 20 layers (test MSE: 0.143 $\pm$ 0.001) to 25 layers (test MSE: 0.142 $\pm$ 0.003), and Trait 4 with $L_{\frac{1}{2}}$ regularizer showed a small improvement from 18 layers (test MSE: 0.131 $\pm$ 0.004) to 25 layers (test MSE: 0.130 $\pm$ 0.001). Despite the larger size of the pig dataset, deeper residual networks did not consistently lead to further performance gains beyond 18 layers.

For the different regularizers, $L_{\frac{1}{2}}$ achieved the best performance on ResNet-18 for all traits from the pig data. After 15 layers, the performance differences among the regularizers narrow down, with regularizers such as $L_{\frac{4}{3}}$, $L_{\frac{3}{2}}$, and SCAD yielding slightly higher test MSE values than $L_{\frac{1}{2}}$. Nonetheless, deeper networks, particularly beyond 18 layers, offered no substantial improvements over shallower models (e.g., 5 layers) when effective regularization was applied.

Wheat data: The wheat dataset that consists of four traits, was analyzed using various regularizers applied to ResNets of different depths. Table 2 shows the test MSE (mean $\pm$ stddev) of ResNet-15 with different regularizers on wheat data. Among the ResNet architectures tested (ResNet-5, ResNet-10, ResNet-15, ResNet-18, ResNet-20, and ResNet-25), ResNet-15 yielded the best overall performance for the wheat data.

Table 1: Test MSE (mean ± stddev) of ResNet-18 with Different Regularizers on Pig Data. **Bold** values indicate the phenotype predictions yielding the lowest test MSE.

Regularizer	**Trait 1**	**Trait 2**	**Trait 3**	**Trait 4**	**Trait 5**
L_0	0.136 ± 0.003	0.137 ± 0.001	0.136 ± 0.002	0.132 ± 0.004	0.144 ± 0.003
$L_{\frac{1}{2}}$	**0.129 ± 0.005**	**0.130 ± 0.004**	**0.129 ± 0.004**	**0.126 ± 0.001**	**0.133 ± 0.002**
$L_{\frac{2}{3}}$	0.131 ± 0.002	0.132 ± 0.005	0.131 ± 0.003	0.127 ± 0.003	0.135 ± 0.001
MCP	0.137 ± 0.004	0.138 ± 0.003	0.137 ± 0.004	0.134 ± 0.001	0.144 ± 0.003
SCAD	0.138 ± 0.002	0.139 ± 0.004	0.138 ± 0.002	0.134 ± 0.005	0.145 ± 0.003
L_1	0.135 ± 0.001	0.136 ± 0.002	0.135 ± 0.003	0.132 ± 0.002	0.143 ± 0.001
$L_{\frac{4}{3}}$	0.135 ± 0.002	0.136 ± 0.004	0.135 ± 0.005	0.132 ± 0.002	0.143 ± 0.001
$L_{\frac{3}{2}}$	0.136 ± 0.002	0.137 ± 0.002	0.136 ± 0.002	0.132 ± 0.003	0.144 ± 0.002
L_2	0.133 ± 0.003	0.134 ± 0.003	0.133 ± 0.004	0.130 ± 0.001	0.140 ± 0.005
L_3	0.132 ± 0.003	0.133 ± 0.002	0.132 ± 0.003	0.128 ± 0.001	0.136 ± 0.003
L_4	0.131 ± 0.002	0.132 ± 0.003	0.132 ± 0.004	0.127 ± 0.003	0.136 ± 0.001
L_∞	0.134 ± 0.002	0.135 ± 0.001	0.134 ± 0.002	0.131 ± 0.003	0.142 ± 0.001

The test MSE results reveal that the $L_{\frac{1}{2}}$ regularizer consistently outperformed others, achieving the lowest MSE across all four traits (Trait 1: 0.132 ± 0.002, Trait 2: 0.131 ± 0.002, Trait 3: 0.130 ± 0.003, Trait 4: 0.130 ± 0.002). This suggests that $L_{\frac{1}{2}}$ regularization provides effective model generalization and is particularly suited for ResNet-15 on this dataset. In comparison, other regularizers like SCAD and MCP resulted in higher test MSE values, indicating less effective performance.

While the differences between the regularizers were not very large, several observations can still be made: (1) The regularizers L_0, $L_{\frac{2}{3}}$, and $L_{\frac{4}{3}}$ performed similarly across the traits, but slightly worse than $L_{\frac{1}{2}}$. This pattern was consistent with the performance seen in other traits, highlighting the robustness of $L_{\frac{1}{2}}$. (2) The regularizers SCAD and MCP demonstrated higher test MSEs, particularly for Trait 1 (SCAD: 0.144 ± 0.005, MCP: 0.143 ± 0.005). These findings suggest the importance of selecting an appropriate regularization method for the wheat data. While deeper networks generally provided more capacity, for the wheat data, ResNet-15 found the right balance between model complexity and performance. Deeper models such as ResNet-18 and ResNet-25 offered no significant improvement over ResNet-15, demonstrating the importance of model architecture selection alongside regularization.

Table 2: Test MSE (mean ± stddev) of ResNet-15 with Different Regularizers on Wheat Data. **Bold** values indicate the phenotype predictions yielding the lowest test MSE.

Regularizer	**Trait 1**	**Trait 2**	**Trait 3**	**Trait 4**
L_0	0.142 ± 0.001	0.137 ± 0.003	0.134 ± 0.005	0.135 ± 0.002
$L_{\frac{1}{2}}$	**0.132 ± 0.002**	**0.131 ± 0.002**	**0.130 ± 0.003**	**0.130 ± 0.002**
$L_{\frac{2}{3}}$	0.135 ± 0.001	0.132 ± 0.005	0.132 ± 0.003	0.132 ± 0.002
MCP	0.143 ± 0.005	0.137 ± 0.003	0.135 ± 0.002	0.136 ± 0.002
SCAD	0.144 ± 0.005	0.138 ± 0.004	0.136 ± 0.001	0.137 ± 0.002
L_1	0.140 ± 0.003	0.136 ± 0.001	0.134 ± 0.001	0.134 ± 0.003
$L_{\frac{4}{3}}$	0.142 ± 0.002	0.136 ± 0.003	0.134 ± 0.004	0.134 ± 0.001
$L_{\frac{3}{2}}$	0.142 ± 0.001	0.137 ± 0.001	0.134 ± 0.002	0.135 ± 0.003
L_2	0.139 ± 0.004	0.135 ± 0.003	0.133 ± 0.002	0.134 ± 0.001
L_3	0.136 ± 0.002	0.133 ± 0.003	0.133 ± 0.002	0.134 ± 0.001
L_4	0.135 ± 0.005	0.132 ± 0.001	0.133 ± 0.002	0.133 ± 0.002
L_∞	0.140 ± 0.002	0.135 ± 0.001	0.133 ± 0.003	0.134 ± 0.003

5.3 Adaptive Proximal Methods vs. Traditional Sparse Proximal Methods

5.3.1 Performance Comparison

For the comparison purpose, we conducted the traditional sparse proximal method with different forms of regularization from L_0 to L_∞ on the same datasets. The evaluation of adaptive proximal gradient methods utilizing ResNets with various regularizers against traditional sparse proximal methods across the mice, pig, and wheat datasets reveals their significant performance distinctions. On the mice dataset, the adaptive proximal methods employing ResNet architectures, particularly ResNet-18 with the $L_{\frac{1}{2}}$ regularizer, achieved the lowest test MSE values (Trait 1: 0.130 ± 0.001, Trait 2: 0.131 ± 0.002). In contrast, the traditional sparse proximal method with the same $L_{\frac{1}{2}}$ regularizer yielded higher test MSE (Trait 1: 0.162 ± 0.001, Trait 2: 0.173 ± 0.003). This substantial reduction in MSE highlights the superior model generalization capabilities of adaptive proximal methods facilitated by deeper ResNet architectures. Similarly, on the larger pig dataset encompassing five traits, adaptive proximal methods with ResNet-18 and $L_{\frac{1}{2}}$ regularization demonstrated superior performance (Trait 1: 0.129 ± 0.005 to Trait 5: 0.133 ± 0.002) compared to traditional sparse proximal methods (Trait 1: 0.161 ± 0.002 to Trait 5: 0.166 ± 0.002). The adaptive proximal methods not only achieved lower MSEs but also exhibited better consistency across multiple traits, whereas traditional methods showed more variable MSEs across traits. On the wheat data, which includes four traits, ResNet-15 with the $L_{\frac{1}{2}}$ regularizer achieved the best performance (Trait 1: 0.132 ± 0.002 to Trait 4: 0.130 ± 0.002). In contrast, the traditional sparse proximal linear method with the same regularizer showed higher MSEs (Trait 1: 0.155 ± 0.004 to Trait 4: 0.151 ± 0.001). These results demonstrate the advantage of adaptive proximal methods in handling high-dimensional genomic data. The adaptive proximal methods leveraging ResNets significantly enhanced performance beyond what traditional methods could achieve with the same regularization.

5.3.2 Model Depth and Complexity

Model depth: For the adaptive proximal methods with ResNets, increasing network depth helps capture more complex relationships between features, as seen with ResNet-18 often yielding the best results on the mice and pig data and with ResNet-15 obtaining the smallest MSE on the wheat data. This depth enables the model to learn more intricate patterns in the data that traditional methods might miss. Regularizers like $L_{\frac{1}{2}}$ are able to adapt to the depth of the network, resulting in improved performance with increasing depth. The improvement in MSE as ResNet depth increases is observed across most regularizers, suggesting that deeper networks enhance performance. In contrast, traditional sparse proximal methods, which rely on linear models, lack the capacity to learn hierarchical or complex interactions between features. These methods do not benefit from additional depth, as they operate with simple penalty terms that limits their capacity to capture more complex feature interactions.

Complexity: The computational complexity for traditional sparse proximal methods is $O(n \cdot p + p)$ per iteration, where n represents the number of samples and p the number of features. In comparison, adaptive proximal gradient methods utilizing ResNets have a higher computational complexity due to the deeper network architecture and additional complexity introduced by the proximal operator for regularization. The complexity of adaptive gradient methods using ResNets is influenced by several factors, such as network depth, width, and the specific nature of the applied regularization. Each forward pass in a ResNet is computationally intensive, particularly for deeper networks, and the proximal gradient step further adds complexity based on the number of features and the nature of the proximal operator. The overall computational complexity of adaptive proximal gradient methods is $O(L \cdot n \cdot d^2 + n \cdot p + p)$, where L is the number of layers, K is the number of residual blocks per layer, and d is the number of blocks per layer. It is obvious that adaptive proximal methods have much higher computational complexity than traditional sparse proximal methods.

6 Conclusion

In this study, we examined the influence of network size and regularization function on ResNets with an application to high-dimensional genomic data. Extensive experimental results indicate that increasing the depth of ResNets generally enhances prediction accuracy, as deeper networks are more capable of modeling

the complex, hierarchical structures within genomic data. Specifically, ResNet-18 consistently outperformed shallower networks on mice and pig data. While ResNet-15 achieved the best prediction accuracy than other architectures on the wheat data. These illustrate the benefits of deeper architectures in handling high-dimensional feature interactions, which are critical for genomic prediction tasks. Furthermore, regularization function choice emerged as a significant factor in the ResNets' performance. Particularly the $L_{\frac{1}{2}}$ regularizer demonstrated robust performance improvements for deep architectures. Additional comparison with traditional sparse proximal methods also demonstrates the advantages of adaptive proximal methods with ResNets for high-dimensional genomic data.

References

Jimmy Lei Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Yu Bai, Yu-Xiang Wang, and Edo Liberty. Proxquant: Quantized neural networks via proximal operators. *arXiv preprint arXiv:1810.00861*, 2018.

Amir Beck. *First-order methods in optimization*. SIAM, Philadelphia, 2017.

Stephen Becker, Jalal Fadili, and Peter Ochs. On quasi-Newton forward-backward splitting: proximal calculus and convergence. *SIAM Journal on Optimization*, 29(4):2445–2481, 2019.

James Bergstra, Dan Yamins, David D Cox, et al. Hyperopt: A Python library for optimizing the hyperparameters of machine learning algorithms. *SciPy*, 13:20, 2013.

Ivan Bondarenko. More layers! End-to-end regression and uncertainty on tabular data with deep learning. *arXiv preprint arXiv:2112.03566*, 2021.

Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, Cambridge, 2004.

Wenfei Cao, Jian Sun, and Zongben Xu. Fast image deconvolution using closed-form thresholding formulas of lq (q= 12, 23) regularization. *Journal of Visual Communication and Image Representation*, 24(1):31–41, 2013.

Matthew A Cleveland, John M Hickey, and Selma Forni. A common dataset for genomic analysis of livestock populations. *G3: Genes| Genomes| Genetics*, 2(4):429–435, 2012.

John W Davey, Paul A Hohenlohe, Paul D Etter, Jason Q Boone, Julian M Catchen, and Mark L Blaxter. Genome-wide genetic marker discovery and genotyping using next-generation sequencing. *Nature Reviews Genetics*, 12:499–510, 2011.

Gustavo de Los Campos, John M Hickey, Ricardo Pong-Wong, Hans D Daetwyler, and Mario PL Calus. Whole-genome regression and prediction methods applied to plant and animal breeding. *Genetics*, 193: 327–345, 2013.

Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(7), 2011.

G Eraslan, Z Avsec, J Gagneur, and F J Theis. Deep learning: new computational modelling techniques for genomics. *Nature Reviews Genetics*, 20:389–403, 2019.

Yuhua Fan, Ilkka Launonen, Mikko J Sillanpää, and Patrik Waldmann. Evaluation of sparse proximal multi-task learning for genome-wide prediction. *IEEE Access*, pp. 52665–51675, 2024.

Dean Foster, Howard Karloff, and Justin Thaler. Variable selection is hard. In *Conference on Learning Theory*, pp. 696–709. PMLR, 2015.

Peter I Frazier. Bayesian optimization. In *Recent advances in optimization and modeling of contemporary problems*, pp. 255–278. Informs, Maryland, 2018.

Fang Ge, Ying Zhang, Jian Xu, Arif Muhammad, Jiangning Song, and Dong-Jun Yu. Prediction of disease-associated nsSNPs by integrating multi-scale ResNet models with deep feature fusion. *Briefings in Bioinformatics*, 23(1):bbab530, 2022.

Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 5927–5935, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 630–645. Springer, 2016b.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Yaohua Hu, Xinlin Hu, and Xiaoqi Yang. On convergence of iterative thresholding algorithms to approximate sparse solution for composite nonconvex optimization. *Mathematical Programming*, pp. 1–26, 2024.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 646–661. Springer, 2016.

Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in Neural Information Processing Systems*, 30, 2017.

Anders Krogh and John Hertz. A simple weight decay can improve generalization. *Advances in Neural Information Processing Systems*, 4, 1991.

Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine learning*, pp. 473–480, 2007.

Sangkyun Lee and Jeonghyun Lee. Compressed learning of deep neural networks for opencl-capable embedded systems. *Applied Sciences*, 9(8):1669, 2019.

Andrés Legarra, Christele Robert-Granié, Eduardo Manfredi, and Jean-Michel Elsen. Performance of genomic selection in mice. *Genetics*, 180(1):611–618, 2008.

Haoran Liang, Enbin Song, Zhujun Cao, Weiyu Li, and Tingting Wang. A proximal gradient method for regularized deep neural networks. In *2023 42nd Chinese Control Conference (CCC)*, pp. 8757–8762. IEEE, 2023.

Yucong Liu, Shixing Yu, and Tong Lin. Hessian regularization of deep neural networks: A novel approach based on stochastic estimators of Hessian trace. *Neurocomputing*, 536:13–20, 2023.

Abolfazl Mehranian, Hamidreza Saligheh Rad, Arman Rahmim, Mohammad Reza Ay, and Habib Zaidi. Smoothly clipped absolute deviation (SCAD) regularization for compressed sensing MRI using an augmented Lagrangian scheme. *Magnetic Resonance Imaging*, 31(8):1399–1411, 2013.

Peter Melchior, Rémy Joseph, and Fred Moolekamp. Proximal adam: robust adaptive update scheme for constrained optimization. *arXiv preprint arXiv:1910.10094*, 2019.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(8):1979–1993, 2018.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in Neural Information Processing Systems*, 32, 2019.

Hayrettin Okut, Daniel Gianola, Guilherme JM Rosa, and Kent A Weigel. Prediction of body mass index in mice using dense molecular markers and a regularized neural network. *Genetics Research*, 93(3):189–201, 2011.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

Paulino Pérez and Gustavo de Los Campos. Genome-wide regression and prediction with the BGLR statistical package. *Genetics*, 198(2):483–495, 2014.

Thomas Pock and Antonin Chambolle. Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *2011 International Conference on Computer Vision*, pp. 1762–1769. IEEE, 2011.

Nicholas G Polson, James G Scott, and Brandon T Willard. Proximal algorithms in statistics and machine learning. *Statistical Science*, 30(4):559–581, 2015.

L Prechelt. Early stopping-but when?, neural networks: tricks of the trade, lectur e notes in computer science, 1998.

Didier Quirin, Valerio Salvucci, Moto Kawanobe, Travis Baratcart, and Takafumi Koseki. Closed form minimum infinity-norm resolution for single-degree kinematically redundant manipulators. In *IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society*, pp. 002892–002897. IEEE, 2015.

Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in Neural Information Processing Systems*, 29, 2016.

Claudio Filipi Gonçalves Dos Santos and João Paulo Papa. Avoiding overfitting: A survey on regularization methods for convolutional neural networks. *ACM Computing Surveys (CSUR)*, 54(10s):1–25, 2022.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *arXiv preprint arXiv:1805.11604*, 2018.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):1–48, 2019.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1): 1929–1958, 2014.

Tijmen Tieleman and Geoffrey Hinton. Rmsprop: Divide the gradient by a running average of its recent magnitude. coursera: Neural networks for machine learning. *COURSERA Neural Networks Mach. Learn*, 17, 2012.

Arijit Ukil, Leandro Marín, and Antonio J Jara. L1 and L2 regularized deep residual network model for automated detection of myocardial infarction (heart attack) using electrocardiogram signals. In *CIKM Workshops*, 2021.

Huaxuan Wu, Bingxi Gao, Rong Zhang, Zehang Huang, Zongjun Yin, Xiaoxiang Hu, Cai-Xia Yang, and Zhi-Qiang Du. Residual network improves the prediction accuracy of genomic selection. *Animal Genetics*, 2024.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1492–1500, 2017.

Yu Xue, Yankang Wang, and Jiayu Liang. A self-adaptive gradient descent search algorithm for fully-connected neural networks. *Neurocomputing*, 478:70–80, 2022.

Yoshihiro Yamada, Masakazu Iwamura, Takuya Akiba, and Koichi Kise. Shakedrop regularization for deep residual learning. *IEEE Access*, 7:186126–186136, 2019.

Liu Yang, Jifan Zhang, Joseph Shenouda, Dimitris Papailiopoulos, Kangwook Lee, and Robert D Nowak. PathProx: A proximal gradient algorithm for weight decay regularized deep neural networks. *arXiv preprint arXiv:2210.03069*, 2022.

Yang Yang, Yaxiong Yuan, Avraam Chatzimichailidis, Ruud JG van Sloun, Lei Lei, and Symeon Chatzinotas. Proxsgd: Training structured neural networks under regularization and constraints. In *International Conference on Learning Representations (ICLR) 2020*, 2020.

T Yue, Y Wang, L Zhang, C Gu, H Xue, W Wang, Q Lyu, and Y Dun. Deep learning for genomics: From early neural nets to modern large language models. *International Journal of Molecular Sciences*, 24:15858, 2023.

Jihun Yun, Aurelie C Lozano, and Eunho Yang. A general family of stochastic proximal gradient methods for deep learning. *arXiv preprint arXiv:2007.07484*, 2020.

Jihun Yun, Aurélie C Lozano, and Eunho Yang. Adaptive proximal gradient methods for structured neural networks. *Advances in Neural Information Processing Systems*, 34:24365–24378, 2021.

Sergey Zagoruyko. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021.

Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. *Annals of Statistics*, 38 (2):894–942, 2010.

Weijing Zhao and He Huang. Adaptive orthogonal gradient descent algorithm for fully complex-valued neural networks. *Neurocomputing*, 546:126358, 2023.

Appendix

A Proximal Mappings of The Regularization Functions

Here we present the proximal operators for the regularizers L_0 to L_∞ and how they are combined with the ADAM preconditioners presented in Section 3. We also consider the cases of MCP and SCAD regularization. Recall that the minimization problem for L_q regularizers is

$$\arg\min_{\theta}\{\frac{1}{2}\left\|\theta-\hat{\theta}_t\right\|^2_{C_t+\delta I}+\lambda\sum_{j=1}^{p}|\theta_j|^q\}, \tag{8}$$

for which a solution is obtained by iterating

$$\begin{aligned}\hat{\theta}_t &= \theta_t - \alpha_t (C_t + \delta I)^{-1} m_t, \\ \theta_{t+1} &= \text{prox}^{C_t+\delta I}_{\alpha_t \lambda \mathcal{R}(\cdot)}(\hat{\theta}_t).\end{aligned} \tag{9}$$

By denoting the i-th coordinate of θ_t as $\theta_{t,i}$ and the i-th diagonal entry of the preconditioner $[C_t]_{ii}$ as $C_{t,i}$, the formulation (9) is coordinate-wise decomposable and the preconditioner matrix C_t is diagonal, so we get

$$\begin{aligned}\theta_{t+1,i} &= \arg\min_{\theta_i} \{\frac{1}{2}(C_{t,i} + \delta I)(\theta_i - \hat{\theta}_{t,i})^2 + \alpha_t \lambda |\theta_i|^q\} \\ &= \arg\min_{\theta_i} \{(\theta_i - \hat{\theta}_{t,i})^2 + \frac{2\alpha_t \lambda}{C_{t,i} + \delta I} |\theta_i|^q\},\end{aligned} \tag{10}$$

where

$$\hat{\theta}_{t,i} = \theta_{t,i} - \alpha_t \frac{m_{t,i}}{C_{t,i} + \delta I}. \tag{11}$$

A.1 L_0 Regularization

When $q = 0$ in Equation (5), the regularization corresponds to the L_0 penalty function, which penalizes the number of non-zero entries of the parameter vector. The L_0 penalty is non-convex and promotes unbiased sparsity in the estimated parameters $\hat{\theta}$. In this case, the closed-form proximal operator can be obtained via hard-thresholding given by

$$\text{prox}_{\lambda \mathcal{R}(\cdot)}(\theta) = \begin{cases} \{0\}, & |\theta| < \sqrt{2\lambda}, \\ \{0, \theta\}, & |\theta| = \sqrt{2\lambda}, \\ \{\theta\}, & |\theta| > \sqrt{2\lambda}. \end{cases} \tag{12}$$

The closed-form proximal mappings given above for L_0 regularization can be combined with the preconditioner $C_{t,i}$ which leads to the following update rule (Yun et al., 2020):

$$\theta_{t+1,i} = \begin{cases} \hat{\theta}_{t,i}, & \text{if } \left|\hat{\theta}_{t,i}\right| > \sqrt{\frac{2\alpha_t \lambda}{C_{t,i}+\delta I}}, \\ 0, & \text{if } \left|\hat{\theta}_{t,i}\right| < \sqrt{\frac{2\alpha_t \lambda}{C_{t,i}+\delta I}}, \\ \left\{0, \hat{\theta}_{t,i}\right\}, & \text{if } \left|\hat{\theta}_{t,i}\right| = \sqrt{\frac{2\alpha_t \lambda}{C_{t,i}+\delta I}}. \end{cases} \tag{13}$$

A.2 $L_{\frac{1}{2}}$ Regularization

For $q = \frac{1}{2}$, the penalty corresponds to the $L_{\frac{1}{2}}$ quasinorm which encourages sparsity similarly to the L_0 penalty, but with smoother behavior near zero. This form of non-convex regularization applies shrinkage on the regression coefficients, but yields more sparse solutions than the L_1 regularizer. According to (Cao et al., 2013), the proximal mapping is as follows:

$$\text{prox}_{\lambda \mathcal{R}(\cdot)}(\theta) = \begin{cases} \frac{2}{3}|\theta|\left(1 + \cos\left(\frac{2}{3}\pi - \frac{2}{3}\varphi_\lambda(\theta)\right)\right), & \text{if } \theta > p(\lambda), \\ 0, & \text{if } |\theta| \le p(\lambda) \\ -\frac{2}{3}|\theta|\left(1 + \cos\left(\frac{2}{3}\pi - \frac{2}{3}\varphi_\lambda(\theta)\right)\right), & \text{if } \theta < p(\lambda) \end{cases} \tag{14}$$

where $\varphi_\lambda(\theta) = \arccos\left(\frac{\lambda}{4}\left(\frac{\theta}{3}\right)^{-\frac{3}{2}}\right)$ and $p(\lambda) = \frac{\sqrt[3]{54}}{4}(\lambda)^{\frac{2}{3}}$. Combined with the preconditioner, the i-th coordinate update becomes:

$$\theta_{t+1,i} = \begin{cases} \frac{2}{3}\left|\hat{\theta}_{t,i}\right|(1 + \cos(\frac{2}{3}\pi - \frac{2}{3}\varphi_\lambda(\hat{\theta}_{t,i}))), & \text{if } \hat{\theta}_{t,i} > p(\lambda) \\ 0, & \text{if } \left|\hat{\theta}_{t,i}\right| \le p(\lambda) \\ -\frac{2}{3}\left|\hat{\theta}_{t,i}\right|(1 + \cos(\frac{2}{3}\pi - \frac{2}{3}\varphi_\lambda(\hat{\theta}_{t,i}))), & \text{if } \hat{\theta}_{t,i} < -p(\lambda) \end{cases} \tag{15}$$

where

$$\varphi_\lambda(\hat{\theta}_{t,i}) = \arccos\left(\frac{\alpha_t \lambda}{4(C_{t,i} + \delta I)}\left(\frac{\left|\hat{\theta}_{t,i}\right|}{3}\right)^{-\frac{3}{2}}\right), \quad p(\lambda) = \frac{\sqrt[3]{54}}{4}\left(\frac{2\alpha_t \lambda}{C_{t,i} + \delta I}\right)^{\frac{2}{3}}. \tag{16}$$

A.3 $L_{\frac{2}{3}}$ Regularization

When $q = \frac{2}{3}$, the non-convex penalty includes a fractional function that encourages sparsity while being smoother than the L_0 penalty. By leveraging the properties of fractional quasinorms to balance sparsity and shrinkage, the proximal operator in this case is (Cao et al., 2013)

$$\text{prox}_{\lambda\mathcal{R}(\cdot)}(\theta) = \begin{cases} \left(\frac{|A|+\sqrt{\frac{2|\theta|}{|A|}-|A|^2}}{2}\right)^3, & \text{if } \theta > \frac{2}{3}\sqrt[4]{3\lambda^3} \\ 0, & \text{if } |\theta| \leq \frac{2}{3}\sqrt[4]{3\lambda^3} \\ -\left(\frac{|A|+\sqrt{\frac{2|\theta|}{|A|}-|A|^2}}{2}\right)^3, & \text{if } \theta < -\frac{2}{3}\sqrt[4]{3\lambda^3} \end{cases} \tag{17}$$

where

$$|A| = \frac{2}{\sqrt{3}}\lambda^{\frac{1}{4}}\left(\cosh\left(\frac{\phi}{3}\right)\right)^{\frac{1}{2}},\ \phi = \text{arccosh}\left(\frac{27\theta^2}{16}\lambda^{-\frac{3}{2}}\right). \tag{18}$$

As in the $L_{\frac{1}{2}}$ regularization case above, this regularizer is coordinate-wise separable and it is sufficient to update the sub-problems for each coordinate as:

$$\theta_{t+1,i} = \begin{cases} \left(\frac{|A|+\sqrt{\frac{2\left|\hat{\theta}_{t,i}\right|}{|A|}-|A|^2}}{2}\right)^3, & \text{if } \hat{\theta}_{t,i} > \frac{2}{3}\sqrt[4]{3\lambda^3} \\ 0, & \text{if } \left|\hat{\theta}_{t,i}\right| \leq \frac{2}{3}\sqrt[4]{3\lambda^3} \\ -\left(\frac{|A|+\sqrt{\frac{2\left|\hat{\theta}_{t,i}\right|}{|A|}-|A|^2}}{2}\right)^3, & \text{if } \hat{\theta}_{t,i} < -\frac{2}{3}\sqrt[4]{3\lambda^3} \end{cases} \tag{19}$$

where

$$|A| = \frac{2}{\sqrt{3}}\left(\frac{2\alpha_t\lambda}{C_{t,i}+\delta I}\right)^{\frac{1}{4}}\left(\cosh\left(\frac{\phi}{3}\right)\right)^{\frac{1}{2}},\ \phi = \text{arccosh}\left(\frac{27\hat{\theta}_{t,i}^2}{16}\left(\frac{2\alpha_t\lambda}{C_{t,i}+\delta I}\right)^{-\frac{3}{2}}\right). \tag{20}$$

A.4 MCP Regularization

When use MCP in Equation (5), it offers a penalty that encourages sparsity while avoiding the bias introduced by the traditional L_1 penalty. Unlike L_1 regularization, MCP adaptively applies less shrinkage to larger coefficients, providing a balance between low prediction error and unbiased estimation. The proximal mapping of the MCP penalty has a closed-form following (Hu et al., 2024):

$$\text{prox}_{\lambda\mathcal{R}(\cdot)}(\theta) = \begin{cases} 0, & |\theta| \leq \lambda \\ \frac{\text{sign}(\theta)(|\theta|-\lambda)}{1-\frac{1}{a}}, & \lambda < |\theta| \leq a\lambda \\ \theta, & |\theta| > a\lambda \end{cases} \tag{21}$$

where $a > 0$ is the parameter, which helps the unbias estimation by reducing the penalty applied to larger coefficients. For the genomic data, we set $a = 2$ for the robust models. Integrated with the preconditioner defined, the i-th coordinate update method is as follows:

$$\theta_{t+1,i} = \text{sign}(\hat{\theta}_{t,i})\text{min}\left\{\frac{a\ \text{max}\left\{\left|\hat{\theta}_{t,i}\right| - \frac{\alpha_t\lambda}{C_{t,i}+\delta I},\ 0\right\}}{a-1}, \left|\hat{\theta}_{t,i}\right|\right\}. \tag{22}$$

A.5 SCAD Regularization

When utilizing the SCAD penalty in Equation (5), it introduces a piecewise linear form, encouraging smaller coefficients to shrink towards zero, while applying less penalty on larger coefficients, thus preserving important features. This adaptive property helps to reduce bias in parameter estimates and avoids over-penalizing significant variables. According to (Hu et al., 2024), the proximal mapping of the SCAD penalty has the closed-form:

$$\text{prox}_{\lambda\mathcal{R}(\cdot)}(\theta) = \begin{cases} \text{sign}(\theta)\max\{|\theta-\lambda|, 0\}, & \text{if } |\theta| \leq 2\lambda \\ \frac{(a-1)\theta - \text{sign}(\theta)a\lambda}{a-2}, & \text{if } 2\lambda < |\theta| \leq a\lambda \\ \theta, & \text{if } |\theta| > a\lambda, \end{cases} \tag{23}$$

where $a > 2$ is the parameter, which controls the shape of the penalty and influences the rate at which the penalty decreases for large coeffients. We set $a = 3.7$ in our cases, which has shown reliable performance for genomic prediction. Based on the above proximal operator and the defined preconditioner, the i-th coordinate update rule is:

$$\theta_{t+1,i} = \begin{cases} \text{sign}(\hat{\theta}_{t,i})\max\left\{\left|\hat{\theta}_{t,i}\right| - \hat{\lambda}_i, 0\right\}, & \text{if } \left|\hat{\theta}_{t,i}\right| \leq 2\hat{\lambda}_i \\ \frac{(a-1)\hat{\theta}_{t,i} - \text{sign}(\hat{\theta}_{t,i})a\hat{\lambda}_i}{a-2}, & \text{if } 2\hat{\lambda}_i < \left|\hat{\theta}_{t,i}\right| \leq a\hat{\lambda}_i \\ \hat{\theta}_{t,i}, & \text{if } \left|\hat{\theta}_{t,i}\right| > a\hat{\lambda}_i \end{cases} \tag{24}$$

where $\hat{\lambda}_i = \frac{\alpha_t\lambda}{C_{t,i}+\delta I}$, which explicitly includes the dependence on λ.

A.6 L_1 Regularization

For $q = 1$, the regularization corresponds to the L_1 norm, which is based on the sum of absolute values that encourages sparsity by driving small coefficients towards zero while also applying some shrinkage to large coefficient. Unlike the L_0 penalty function, which directly counts non-zero elements, L_1 provides a convex relaxation that allows for efficient optimization while still promoting sparsity. The proximal mapping of L_1 regularization can be given as (Parikh & Boyd, 2014):

$$\begin{aligned} \text{prox}_{\lambda\mathcal{R}(\cdot)}(\theta) &= \text{sign}(\theta)\max(|\theta| - \lambda, 0) \\ &= \text{sign}(\theta)(|\theta| - \lambda)_+, \end{aligned} \tag{25}$$

where $(\cdot)_+$ denotes the positive part. Combining it with the preconditioner, the i-th coordinate update scheme is:

$$\theta_{t+1,i} = \text{sign}(\hat{\theta}_{t,i})\left(\left|\hat{\theta}_{t,i}\right| - \frac{\alpha_t\lambda}{C_{t,i}+\delta}\right). \tag{26}$$

A.7 $L_{\frac{4}{3}}$ Regularization

When $q = \frac{4}{3}$, the regularization is a norm. It is softer than the L_1 norm, but it will not promote the sparsity. The proximal mapping of $L_{\frac{4}{3}}$ is presented in (Polson et al., 2015) as

$$\text{prox}_{\lambda\mathcal{R}(\cdot)} = \theta + \frac{4\lambda\kappa}{32^{\frac{1}{3}}}\left((\chi-\theta)^{\frac{1}{3}} - (\chi+\theta)^{\frac{1}{3}}\right), \tag{27}$$

where $\chi = \sqrt{\theta^2 + 256\kappa^3/729}$ and $\kappa = 1$ is a scaling factor that adjusts the strength of the $L_{\frac{4}{3}}$ norm's impact. Combined with the preconditioner, the i-th coordinate update rule is

$$\theta_{t+1,i} = \hat{\theta}_{t,i} + \frac{8\kappa\alpha_t\lambda}{32^{\frac{1}{3}}(C_{t,i}+\delta I)}((\chi-\hat{\theta}_{t,i})^{\frac{1}{3}} - (\chi+\hat{\theta}_{t,i})^{\frac{1}{3}}), \tag{28}$$

where $\chi = \sqrt{\hat{\theta}_{t,i}^2 + 256\kappa^3/729}$.

A.8 $L_{\frac{3}{2}}$ Regularization

For $q = \frac{3}{2}$, the penalty corresponds to the $L_{\frac{3}{2}}$ norm which places more emphasis on shrinkage compared to L_1 norm. This regularization avoids over-penalization of large coefficients, offering a more refined control over model complexity. According to (Polson et al., 2015), the proximal mapping is given as

$$\text{prox}_{\lambda\mathcal{R}(\cdot)} = \theta + 9\kappa^2 \text{sgn}(\theta) \left(1 - \sqrt{1 + 16\lambda |\theta| / (9\kappa^2)}\right) /8, \tag{29}$$

where $\kappa = 1$ is the parameter which controls the degree of shrinkage applied to the coefficients. Combined with preconditioner defined, the i-th coordinate update rule is as

$$\theta_{t+1,i} = \hat{\theta}_{t,i} + 9\kappa^2 \text{sgn}(\hat{\theta}_{t,i}) \left(1 - \sqrt{1 + \frac{32 \left|\hat{\theta}_{t,i}\right| \alpha_t \lambda}{9\kappa^2 (C_{t,i} + \delta I)}}\right) /8. \tag{30}$$

A.9 L_2 Regularization

When $q = 2$, the penalty corresponds to the L_2 norm which applies a quadratic penalty on the parameters and encourages smoothness by shrinking the coefficients towards zero in a continuous manner. Unlike the penalties from L_0 to L_1, the L_2 norm does not promote sparsity but rather emphasizes minimizing the magnitude of the coefficients. According to (Parikh & Boyd, 2014), the proximal mapping of the L_2 norm is

$$\text{prox}_{\lambda\mathcal{R}(\cdot)} = (1 - \frac{\lambda}{\|\theta\|_2})_+ \theta, \tag{31}$$

where $(\theta)_+ = \max\{0, \theta\}$. Then the i-th coordinate update with preconditioner is as follows:

$$\theta_{t+1,i} = (1 - \frac{2\alpha_t \lambda}{(C_{t,i} + \delta I) \left\|\hat{\theta}_{t,i}\right\|_2})_+ \hat{\theta}_{t,i}. \tag{32}$$

A.10 L_3 Regularization

When $q = 3$, the penalty corresponds to the L_3 norm. It is a higher-order norm that encourages more shrinkage on larger coefficients. This form of convex regularization reduce the risk of overfitting by dampening high-variance components more effectively, resulting in a model that is more resistant to noise. The proximal mapping of L_3 regularization is defined following (Polson et al., 2015) as

$$\text{prox}_{\lambda\mathcal{R}(\cdot)} = \text{sgn}(\theta)(\sqrt{1 + 12\lambda\kappa |\theta|} - 1)/(6\kappa), \tag{33}$$

where $\kappa = 1$, which helps adjust the level of shrinkage applied to the larger weights. With the preconditioner, the i-th coordinate update becomes

$$\theta_{t+1,i} = \text{sgn}(\hat{\theta}_{t,i}) \frac{\sqrt{1 + \frac{24\kappa\alpha_t\lambda\left|\hat{\theta}_{t,i}\right|}{C_{t,i}+\delta I}} - 1}{6\kappa}. \tag{34}$$

A.11 L_4 Regularization

When $q = 4$, the penalty is the L_4 norm, which provides a smoother penalty function, encouraging a more gradual reduction in the coefficients of less important features. The proximal mapping of L_4 regularization, given by (Polson et al., 2015), is

$$\text{prox}_{\lambda\mathcal{R}(\cdot)} = \left(\frac{\chi + \lambda\theta}{8\kappa}\right)^{\frac{1}{3}} - \left(\frac{\chi - \lambda\theta}{8\kappa}\right)^{\frac{1}{3}}, \tag{35}$$

where $\kappa = 1$ impacts the strength of the regularization applied to the coefficients, and $\chi = \sqrt{\theta^2 + \frac{1}{27\kappa}}$. Together with the preconditioner, the i-th coordinate update rule is

$$\theta_{t+1,i} = \left(\frac{\chi + \frac{2\hat{\theta}_{t,i}\alpha_t\lambda}{C_{t,i}+\delta I}}{8\kappa} \right)^{\frac{1}{3}} - \left(\frac{\chi - \frac{2\hat{\theta}_{t,i}\alpha_t\lambda}{C_{t,i}+\delta I}}{8\kappa} \right)^{\frac{1}{3}}, \tag{36}$$

where $\chi = \sqrt{\hat{\theta}_{t,i}^2 + \frac{1}{27\kappa}}$.

A.12 L_∞ Regularization

For the infinity norm, the solution is constrained by the maximum absolute value of the coefficients. As detailed in (Beck, 2017), the projection onto the L_1 ball encourages a broader range of solutions, allowing for greater flexibility in modeling complex data (Quirin et al., 2015). Following (Parikh & Boyd, 2014), the proximal mapping is given as

$$\text{prox}_{\lambda\mathcal{R}(\cdot)}(\theta) = \theta - \lambda P_\Delta(|\theta|/\lambda) * \text{sign}(\theta), \tag{37}$$

where P_Δ is the projection inside the unit simplex, $*$ is the element-wise product, and $\text{sign}(\theta)$ is the sign function. Then the proximal operator of the infinity norm can be calculated by a projection on the simplex. With the preconditioner defined earlier, the i-th coordinate update is

$$\theta_{t+1,i} = \begin{cases} \hat{\theta}_{t,i} - \frac{2\alpha_t\lambda}{C_{t,i}+\delta I}, & \hat{\theta}_{t,i} > \frac{2\alpha_t\lambda}{C_{t,i}+\delta}, \\ \hat{\theta}_{t,i} + \frac{2\alpha_t\lambda}{C_{t,i}+\delta}, & \hat{\theta}_{t,i} < \frac{2\alpha_t\lambda}{C_{t,i}+\delta}, \\ 0, & \text{otherwise.} \end{cases} \tag{38}$$

B Analysis of Convergence Guarantee

B.1 Convex Regularizers

For the optimization problem (2), the function $\mathcal{L}(\theta)$ is often assumed to be convex and differentiable. When $\mathcal{R}(\theta)$ is a convex regularization function, the condition for stationary point θ^* in (8) is given by:

$$0 \in \nabla\mathcal{L}(\theta^*) + \lambda\partial\mathcal{R}(\theta^*), \tag{39}$$

where $\partial\mathcal{R}(\theta^*)$ is the subdifferential of $\mathcal{R}$. According to (8), if $\mathcal{L}$ is L-smooth, then

$$\left\| \nabla\mathcal{L}(\theta) - \nabla\mathcal{L}(\theta') \right\| \leq L \left\| \theta - \theta' \right\|. \tag{40}$$

Then

$$\mathbb{E}\left[\mathcal{F}(\theta_t) - \mathcal{F}(\theta^*)\right] \leq \frac{L}{2\alpha_t} \left\|\theta_0 - \theta^*\right\|^2. \tag{41}$$

This illustrates that the convergence is linear under appropriate conditions. If the gradient and Lipschitz continuity are bounded:

$$\mathcal{F}(\theta_{t+1}) \leq \mathcal{F}(\theta_t) - \frac{\alpha_t}{2} \left\|\nabla\mathcal{F}(\theta_t)\right\|^2 + \mathcal{R}(\theta_{t+1}) - \mathcal{R}(\theta_t). \tag{42}$$

Based on the theory of proximal gradient method for the parameters update in (Boyd & Vandenberghe, 2004), the above equation will lead to a conclusion that $\mathcal{F}(\theta_t)$ convergences to the minimum.

B.2 Non-Convex Regularizers

For the non-convex regularizers, the goal for the convergence guarantee is to find an ϵ-stationary point in (8), where ϵ is the required precision. To guarantee the convergence, the Fréchet subdifferential of φ at $\hat{\theta}$ with $\left|\varphi(\hat{\theta})\right| < \infty$ can be defined as (Yun et al., 2020)

$$\hat{\partial}_{\varphi}(\hat{\theta}) \stackrel{\text{def}}{=} \left\{ \theta^* \in \Omega \,\middle|\, \liminf_{\theta \to \hat{\theta}} \frac{\varphi(\theta) - \varphi(\hat{\theta}) - \langle \theta^*, \theta - \hat{\theta} \rangle}{\left\| \theta - \hat{\theta} \right\|} \geq 0 \right\}, \tag{43}$$

where φ is a real-valued function. For the non-convex cases, because they are not differentiable everywhere (especially near zero), the limiting subdifferentials of φ at $\hat{\theta}$ are defined as

$$\partial_{\varphi}(\hat{\theta}) \stackrel{\text{def}}{=} \{u \in \mathbb{R}^d : \exists \theta_k \stackrel{\varphi}{\to} \hat{\theta}, u_k \in \hat{\partial}_{\varphi}(\theta_k), u_k \to u\}, \tag{44}$$

where $\theta_k \stackrel{\varphi}{\to} \hat{\theta}$ means that $\theta_k \to \hat{\theta}$ when $\varphi(\theta_k) \to \varphi(\hat{\theta})$. At the stationary point, the following condition should be satisfied:

$$0 \in \nabla \mathcal{F}(\theta) = \nabla \mathcal{L}(\theta) + \partial_{\varphi} \lambda \mathcal{R}(\theta). \tag{45}$$

Let θ_a denote an iterate chosen uniformly randomly from $\{\theta_1, \theta_2, \cdots, \theta_T\}$. The convergence criterion for L_q penalty functions, as with ProxGEN (Yun et al., 2020), is given as

$$\mathbb{E}_a[\text{dist}(0, \hat{\partial}_{\varphi} \mathcal{F}(\theta_a))^2] \leq \epsilon, \tag{46}$$

where $\hat{\partial}_{\varphi} \mathcal{F}(\theta_a)$ is the limiting subdifferential. The distance measure $\text{dist}(0, \cdot)$ ensures that at stationary points, the gradient of the loss function and the subdifferential of the regularizer are close to zero. Based on Theorem 1 and Corollary 1 in (Yun et al., 2020), with the sample size n and constant minibatch size b_t, a good convergence can be ensured.