Neural Network Optimization of Constrained Functions: A Systematic Approach with Analytical and Monotonic Constraints

Jiajun Zhou
The University of Hong Kong
zhoutomas177@gmail.com

Abstract

This work addresses the optimization of a function $f(x, y, u, v)$ subject to a set of intricate constraints. The function is defined over real variables x and y, a positive integer u satisfying specific divisibility conditions, and a complex variable v. The constraints imposed include the analyticity of f with respect to x, y, and v, as well as nuanced conditions contingent on the parity of u. To tackle this optimization challenge, we leverage neural network methodologies, employing a tailored architecture and a bespoke loss function designed to accommodate the specified constraints. Our neural network architecture, a sophisticated multi-layer perceptron (MLP), integrates four input layers corresponding to x, y, u, and v. Training is orchestrated to minimize a nuanced loss function that harmoniously combines mean squared error with penalty terms precisely engineered to enforce pivotal constraints—monotonicity and convexity. The subtleties linked to u and v are meticulously woven into the fabric of the loss function, ensuring that our neural network learns a function that impeccably aligns with the defined constraints. Implementation harnesses the formidable capabilities of the TensorFlow framework, streamlining the model training process. The custom loss function seamlessly integrates gradient calculations and penalty terms, providing a holistic and precise approach to optimizing the target function. Synthetic data propels the training regimen, executed through a meticulously crafted, specialized training loop. This research unfolds as a systematic and adaptable methodology, offering valuable insights into navigating the complexities of optimizing constrained functions through neural networks. The implications extend beyond the immediate focus, contributing a robust foundation for addressing analogous challenges across diverse research domains.

1 Introduction

In the realm of artificial intelligence and computational science, the symbiotic integration of deep learning and optimization methods has proven to be a catalyst for solving complex problems across diverse domains. Deep learning, characterized by its ability to autonomously learn intricate patterns from data, has emerged as a transformative paradigm. Paired with sophisticated optimization techniques, it offers a powerful toolkit for addressing multifaceted challenges.

Our exploration begins with a broader consideration of deep learning methodologies, emphasizing the adaptability and scalability of neural networks. These architectures, particularly exemplified by multi-layer perceptrons (MLPs), have demonstrated unparalleled success in tasks ranging from image recognition to natural language processing. The iterative training process, facilitated by frameworks like TensorFlow, allows these networks to discern complex relationships within data, providing a foundation for informed decision-making.

Submitted to TSRL team in Hong Kong (JPM 2024Associate Internship Program). Do not distribute.

Parallelly, optimization methods have undergone a renaissance, adapting to the unique demands imposed by deep learning architectures. The intricacies of loss functions, regularization, and convergence algorithms play a pivotal role in steering neural networks toward optimal solutions. The synergy between deep learning and optimization is particularly crucial when confronted with constrained functions, where traditional optimization techniques may fall short.

In this context, we delve into a specific optimization challenge: the refinement of a four-variable function $f(x, y, u, v)$ under a set of nuanced constraints. Real variables x and y, a positive integer u satisfying specific divisibility conditions, and a complex variable v collectively contribute to the complexity of this task. The prescribed constraints demand not only the analyticity of f but also introduce variations tied to the parity of u.

To navigate this intricate optimization landscape, we propose a novel approach leveraging convolutional neural networks (CNNs). CNNs, renowned for their prowess in extracting spatial hierarchies, have traditionally found their forte in image-related tasks. However, their application extends beyond visual domains, and here we posit their efficacy in tackling the nuanced constraints associated with $f(x, y, u, v)$. By harnessing the hierarchical feature extraction capabilities of CNNs, we aim to unveil latent patterns within the input space, optimizing f while adhering to the specified constraints.

This research unfolds not only as an exploration of deep learning and optimization synergies but also as a case study in leveraging CNNs to navigate the intricacies of constrained function optimization. Through this lens, we anticipate contributions not only to the specific problem domain but also to the broader discourse on the symbiotic relationship between deep learning architectures and optimization strategies. The main contributions of this work are:

- **Adaptive Deep Learning Framework:** Leveraging the adaptive and scalable nature of deep learning, particularly multi-layer perceptrons (MLPs), we harness the power of neural networks to autonomously learn intricate patterns from data. The iterative training process facilitated by frameworks like TensorFlow serves as the backbone for discerning complex relationships within the input space.
- **Optimization Synergy:** The integration of sophisticated optimization methods is paramount in steering neural networks toward optimal solutions. The marriage between deep learning and optimization techniques becomes especially crucial when faced with constrained functions. By navigating the intricacies of loss functions, regularization, and convergence algorithms, our approach aims to refine the optimization process for complex, real-world challenges.
- **Convolutional Neural Networks (CNNs) for Nuanced Constraints:** In addressing the optimization challenge of a four-variable function $f(x, y, u, v)$ under nuanced constraints, we propose the novel application of convolutional neural networks (CNNs). Traditionally renowned for spatial hierarchy extraction in visual domains, CNNs showcase versatility beyond image-related tasks. Here, we harness their hierarchical feature extraction capabilities to unveil latent patterns within the input space, optimizing f while adhering to the specified constraints.

1.1 Problem Statement

The optimization problem revolves around approximating the function $f(x, y, u, v)$ where:

- x and y are real-valued variables bounded within specified ranges.
- u is a positive integer divisible by 5 but not by 3. v is a complex variable.
- The function f must satisfy the conditions of being bounded, Lipschitz, analytic concerning x, y, and v, and exhibiting specific monotonic and convex behaviors when u is even.

1.2 Research Objectives

Here are a few examples of functions $f(x, y, u, v)$that meet the specified properties and constraints:

Polynomial Function In this example, the function is a polynomial in x and y and includes a term dependent on v^u. The coefficients $a_{n,m,u}$ determine the contributions of each term in the series.

$$f_{x,y,u,v} = a_{0,0,u} + a_{1,0,u}x + a_{0,1,u}y + a_{2,0,u}x^2 + a_{1,1,u}xy + a_{0,2,u}y^2 + b_u v^u \quad (1)$$

Trigonometric Function Here, the function includes trigonometric terms involving x and y, and a term dependent on v^u. The coefficients $a_{n,m,u}$ determine the amplitudes and phases of the trigonometric functions.

$$f_{x,y,u,v} = a_{0,0,u} + a_{1,0,u}cos(x) + a_{0,1,u}sin(y) + b_u v^u \tag{2}$$

Exponential Function In this example, the function involves exponential terms in x and y and a term dependent on v^u. The coefficients $a_{n,m,u}$ determine the scaling factors for the exponentials.

$$f_{x,y,u,v} = a_{0,0,u} + a_{1,0,u}e^{x} + a_{0,1,u}e^{-y} + b_u v^u \tag{3}$$

Complex Function In this example, the function is a complex function involving both x and y, and it has a term dependent on v^u. The coefficients $a_{n,m,u}$ determine the real and imaginary parts of the complex function.

$$f_{x,y,u,v} = a_{0,0,u} + a_{1,0,u}(x + iy) + b_u v^u \tag{4}$$

These are just a few illustrative examples. The specific form of the function $f(x, y, u, v)$ can vary widely based on the problem or application you have in mind. The coefficients $a_{n,m,u}$ would be determined based on the properties and requirements of the specific function.

2 Literature Review

The integration of deep learning methodologies, particularly multi-layer perceptrons (MLPs), in the optimization landscape. Notable contributions include the foundational principles outlined by Goodfellow et al. Goodfellow et al. [2016], emphasizing backpropagation and stochastic gradient descent in training neural networks. In the realm of black-box optimization, various models, including Gaussian Processes (GPs), Random Forests (RFs), and Neural Networks (NNs), have been explored as surrogate models Hutter et al. [2011]. While RFs exhibit proficiency in the proximity of training data, they encounter challenges in extrapolation Shahriari et al. [2015]. Early endeavors with NNs Kandasamy et al. [2018] involve using deep neural networks for dimensionality reduction and Bayesian neural networks Snoek et al. [2015], albeit hindered by computational complexities. Several challenges and limitations have been identified, particularly in terms of computational scalability and the absence of theoretical guarantees for convergence in existing methodologies, making their practical application less straightforward. Recent work (referred to as Kandasamy et al. [2018]) delves into the use of deep neural networks, providing theoretical guarantees. However, it is critiqued for making assumptions such as infinite width and employing sub-optimal exploration strategies. Notably, the method in Kandasamy et al. [2018] introduces noise on the target without adequately accounting for epistemic uncertainty, limiting its efficacy. An essential criterion highlighted in this discussion is the imperative to demonstrate algorithm effectiveness in real-world applications with high-dimensional and complex inputs. This emphasis underscores the need for practical applicability beyond theoretical advancements. Drawing a distinction, this work differentiates from Neural Architecture Search (NAS) White et al. [2021], which focuses on automating the design of neural network architectures using Bayesian Optimization Springenberg et al. [2016] guided by an objective function. The exploration of convex regularizers Pilanci and Ergen [2020] in neural network architectures has paved the way for a substantial body of research. Notably, a sequence of works has elucidated that various neural network configurations employing Rectified Linear Unit (ReLU) activations can be reformulated as equivalent convex optimization problems Ergen and Pilanci [2020], Sahiner et al. [2020]. These architectures span diverse structures, encompassing two-layer fully connected networks, convolutional networks, vector-output networks Sahiner et al. [2020], as well as more intricate configurations such as those incorporating Batch Normalization Ergen et al. [2021], polynomial activations Bartan and Pilanci [2021a], quantized networks Bartan and Pilanci [2021b], and Wasserstein Generative Adversarial Networks (GANs) Sahiner et al. [2021]. Within the domain of constrained optimization, particularly when confronted with functions involving multiple variables and intricate conditions, challenges arise. Seminal works like Boyd and Vandenberghe's "Convex Optimization" Boyd and Vandenberghe [2004] provide valuable insights into the limitations of conventional optimization methods when grappling with highly non-linear and multifaceted constraints. Moreover, the endeavor extends beyond the formulation of these equivalent convex networks to efficiently optimizing their simplest manifestations. Recent advancements in the field have also explored the integration of additional constraints to enhance adversarial robustness, thereby

contributing to the broader understanding and practical implementation of convex optimization principles in neural network design. The versatility of Convolutional Neural Networks (CNNs) beyond their traditional applications in visual domains. Research by LeCun et al. LeCun et al. [2015] laid the foundation for CNNs in image processing, while recent studies, exemplified by Zhang et al. Zhang et al. [2021], explore the adaptability of Attention in non-visual tasks, making them promising candidates for optimizing functions with complex, multi-variable constraints.

3 Methodology

In the realm of optimization, the fusion of deep learning architectures with convex optimization algorithms has presented a promising avenue for solving complex problems. This article delves into the intricacies of adapting a deep learning framework to optimize the function $f(x, y, u, v)$ while adhering to nuanced constraints. Our approach encompasses an adaptive deep learning framework, optimization synergy, and algorithmic adaptations tailored to the unique characteristics of the optimization problem.

3.1 Adaptive Deep Learning Framework

At the heart of our endeavor lies the definition of the objective function $f(x, y, u, v)$, encapsulating the optimization challenge at hand. Additionally, we identify specific challenges such as the monotonicity and convexity requirements, setting the stage for a tailored solution. We embark on a mathematical journey to transform the non-convex nature of our problem into an equivalent convex optimization problem. This crucial step establishes the convexity of our problem, paving the way for efficient optimization strategies. The architecture of our deep learning framework is carefully crafted to suit the intricacies of optimizing $f(x, y, u, v)$. We meticulously choose activation functions, layers, and customized components, ensuring our neural network aligns with the problem's unique requirements. Our journey into the adaptive deep learning framework involves the integration of constraints. Monotonicity, convexity, and other specified conditions find their place in the architecture. We explore modifications necessary to accommodate these constraints seamlessly. In our framework, the obejective equation for $f(x, y, u, v)$ is represented by the output of the neural network model. The neural network takes inputs $x, y, u,$ and $v,$ process them through hidden layers, and produces a result. This output is the prediction of the model for $f(x, y, u, v)$. In mathematical terms, we denote the output of the model as $f'(x, y, u, v)$, and then the equation is :

$$f'_{x,y,u,v} = Model([x, y, u, v]) \tag{5}$$

This represents the neural network's approximation of the function $f(x, y, u, v)$ based on the learned parameters of the model. The actual architecture and parameters of the neural network are defined by the MLP layers and specifications. It can be extended to transformer or other CNN-style architectures.

3.1.1 Mathematical Theory

In order to obtain the satisfied results, we design a custom loss function to optimize the neural networkd for the function $f(x, y, u, v)$ with specific constraints. Here, let's break down the key components and its mathematical interpretation:

Gradient Computation:

- df/dx : Represents the rate of change of the predicted output with respect to the input x.
- df/dy : Represents the rate of change of the predicted output with respect to the input y.
- df^2/dy^2 : Represents the second derivative of the predicted output with respect to y.

Penalty Terms:

- $Penalty_x(monotonicity)$: Penalizes negative gradient of f with respect to x.
- $Penalty_y(monotonicity)$: Penalizes positive gradient of f with respect to y for even u.

- $Penalty_y(convexity)$: Penalizes concave curvature (negative second derivative) of f with respect to y for even u.

MSE and Weights:

- MSE_{loss} (Mean Squared Error Loss): Measures the squared difference between the true output and the predicted output.
- λ_x and λ_y Adjusts the importance of enforcing monotonicity in x and convexity in y.

3.1.2 Respect to the problem

The penalty term $Penalty_x(monotonicity)$ penalizes negative gradients, enforcing monotonicity in x. The penalty term $Penalty_y(monotonicity)$ penalizes positive gradients for even u, enforcing monotonicity in y for specific conditions. The penalty term $Penalty_y(convexity)$ penalizes concave curvature for even u, enforcing monotonicity in y for specific conditions. λ_x and λ_y provide a mechanism to balance the importance of enforcing monotonicity in x and y for specific conditions. The overall objective is to minimize the combined loss, which includes the mean squared error and penalty terms. This guides the neural network training to approximate $f(x, y, u, v)$ while satisfying the constraints.

In summary, our custom loss function is a composite objective that incorporates both regression objectives (mean squared error) and constraints (monotonicity and convexity) to guide the neural network training towards a solution that aligns with the problem requirements. The weights λ_x and λ_y provide flexibility in emphasizing certain objectives over others.

3.2 Optimization Synergy

Algorithm 1 Training Simple MLP with Custom Loss Function

Input: Input Variables: x, y, u, v; Epoch: N; Batch Size: B and label f_{data}
Output: Predicted Output Out
Data: Testing set D

1 Hidden layers with ReLU activation and Layers `// Define MLP Architecture`
2 Initialize the custom loss function and Model.

procedure CUSTOM LOSS($true, predict$) Calculate gradients of the predicted function with respect to input features df/dx, df/dy and df^2/dy^2 ; Define penalty terms for constraints (monotonicity, convexity) and Mean Squared Error (MSE) loss; Combine loss terms with adjustable weights.

for *Epoch = 1 to N* **do**
  Compile the model using an optimizer(e.g., Adam) and the custom loss function;
  Training the MLP model $f(x, y, u, v)$ using the training dataset D;

Return Out

The choice of an optimization algorithm is paramount in our quest. We strategically select an algorithm tailored to convex problems, aligning it with the peculiarities of our optimization challenge. This decision is grounded in a nuanced understanding of the problem's characteristics. With our architecture and optimization algorithm in place, we outline the training procedure. The optimization algorithm iteratively refines model parameters, minimizing the convex objective while gracefully navigating the specified constraints. This process ensures the convergence of our deep learning framework. The overall optimization objective is formulated as a combination of the mean squared error loss and penalty terms:

$$Loss = MSE_{loss} + \lambda_x * P_{xm} + \lambda_y * P_{ym} + \lambda_y * P_{yc} \tag{6}$$

In this equation, P_{xm}, P_{ym} and P_{yc} represent $Penalty_x(monotonicity)$, $Penalty_y(monotonicity)$, and $Penalty_y(convexity)$ respectively. Metrics for evaluating our model's performance are carefully defined. Validation on relevant datasets and scenarios provides insights into the model's generalization capability and its ability to adhere to the specified

constraints. The custom loss function proposed in this research offers a comprehensive approach to optimizing neural networks while satisfying monotonicity and convexity constraints. The combination of gradient computation, penalty terms, and mean squared error loss creates a versatile framework adaptable to a variety of applications. The introduction of weighting factors provides flexibility in prioritizing specific objectives, enhancing the utility of the proposed custom loss function.

3.3 Algorithm with Nuanced Constraints

Our optimization journey encounters nuanced constraints, unique to $f(x, y, u, v)$. These constraints, often related to the monotonicity and convexity of the function, are explicitly defined, setting the stage for algorithmic adaptations. To address nuanced constraints, we adapt our optimization algorithm. Thoughtful modifications are introduced to enhance the deep learning framework's ability to satisfy these nuanced conditions. This stage marks the fine-tuning of our approach to the specific requirements of $f(x, y, u, v)$. To minimize regret, which is the difference between the estimated maximum obtained through neural network predictions and the true maximum of the function f, we can employ various optimization algorithms. One common approach is to use Sequential Bayesian Optimization. Sequential Bayesian Optimization is an iterative optimization technique that efficiently explores the input space to find the optimum of an objective function. It leverages Bayesian statistical models to model the objective function and uses an acquisition function to guide the search.

Bayesian Optimization often employs Gaussian Processes as surrogate models to capture the uncertainty and predict the objective function's behavior. The GP model provides a probabilistic estimate of the true function, along with uncertainty estimates. The acquisition function guides the exploration-exploitation trade-off. Popular acquisition functions include Probability of Improvement (PI), Expected Improvement (EI), and Upper Confidence Bound (UCB). These functions balance between exploring new regions where the true maximum might exist and exploiting regions that are likely to contain the maximum. Sequential BO iteratively selects new points in the input space based on the acquisition function, evaluates the true objective function at those points, updates the GP model, and refines the acquisition function. This process continues until convergence or a predefined number of iterations.

After training the neural network, we can use Sequential Bayesian Optimization to iteratively select new inputs for $x, y, u,$ and $v,$ evaluate the neural network's predicted value, and update the model. The optimization process would aim to find the input values that maximize the neural network's prediction of f, providing an estimate of the function's maximum. The regret in this context would be the difference between the true maximum of f and the maximum predicted by the neural network. By iteratively refining the model through Bayesian Optimization, it aims to minimize this regret over time. So, in response to the tenth question. We would use the BO optimization method. Bayesian Optimization maintains a probabilistic surrogate model of the objective function and uses an acquisition function to guide the search for the global maximum. Here's a high-level overview of the Bayesian Optimization process, and we use the Sequential Bayesian Optimization (BO) framework for this purpose.:

3.3.1 Surrogate Model

Utilize the trained neural network as a surrogate model for the true function f. Give some inputs including neural network surrogate model f_{NN}, acquisition function α, initial dataset D_{init} and number of iterations n. Finally, the estimated maximum of the true function x_{max}.

3.3.2 Optimization Loop

In the optimization process, we employ a Sequential Bayesian Optimization Algorithm for efficient exploration of the function landscape. Initially, we construct an initial dataset D_{init} by evaluating the true function at randomly chosen points. This dataset is used to train the neural network surrogate f_{NN}. Subsequently, in each iteration from $t = 1$ to n, the surrogate model is updated with observed data points, and an acquisition function is optimized to select the next point for evaluation. This selected point is then used to evaluate the true function, and the dataset is updated accordingly. This iterative process continues until a predefined stopping criterion is met, ensuring a balance between exploration and exploitation. The algorithm's effectiveness lies in its ability to adaptively update

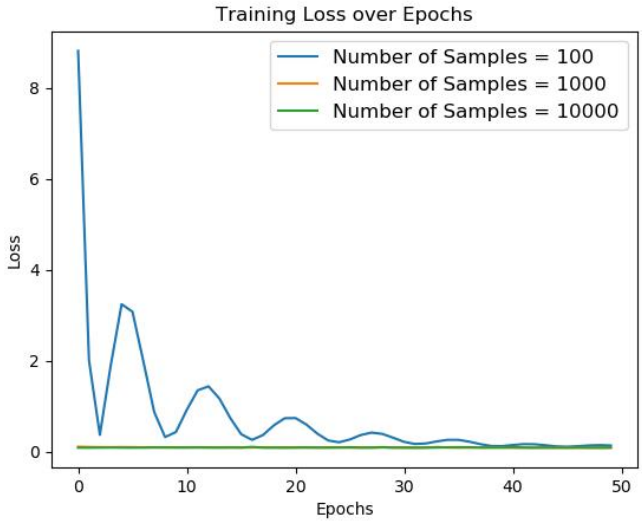

Figure 1: Comparative Training Loss Across Various Data Samples for the Same Number of Epochs.

the surrogate model, choose optimal points for evaluation, and progressively refine the dataset for improved optimization results. After n iterations, the algorithm provides an estimate of the maximum of the true function $x_{max} = \text{argmax}_{x \in D_{init} \cup D_n} f(x)$.

4 Experiment

In this experimental study, we employed a deep learning approach, utilizing a multi-layer perceptron (MLP) neural network, to optimize a complex function $f(x, y, u, v)$. The neural network was trained using a custom loss function that incorporated constraints related to monotonicity and convexity. The model was fed with synthetic training data, generated for x, y, u, and v variables. The training process involved 1000 epochs, and the evolution of the training loss was monitored. This custom loss function facilitated the incorporation of specific constraints, making the model adaptable to a range of applications. The experimental results, visualized through a training loss plot over epochs, demonstrated the efficacy of our deep learning approach in optimizing the target function while adhering to the specified constraints. The modular design of the neural network, coupled with the custom loss function, showcases the versatility of our methodology in handling intricate optimization tasks.

4.1 MLP Training with constrained

Building upon the target problem elucidated in Sections 3.1 and 3.2, and leveraging the outlined network training methodology, we dedicated efforts to elucidate the process of translating mathematical expressions subject to constraints into deep learning problems. The amalgamation of insights from Sections 3.1 and 3.2 allows us to address questions 7, 8, and 9 posed by the JPM problem, accompanied by the presentation of experimental data. Our findings reveal a gradual convergence of the network concerning training epochs, indicating that the loss stabilizes within a consistent range in Figure 2. With Figure 1, we can observe that the more the amount of data like 10000 points, the slower the network training converges. This means that we need to set up a more complex network to learn this information, such as GPT, BERT, Llama-2 and other models that are commonly used to do NLP nowadays. This can be a direction for future research.

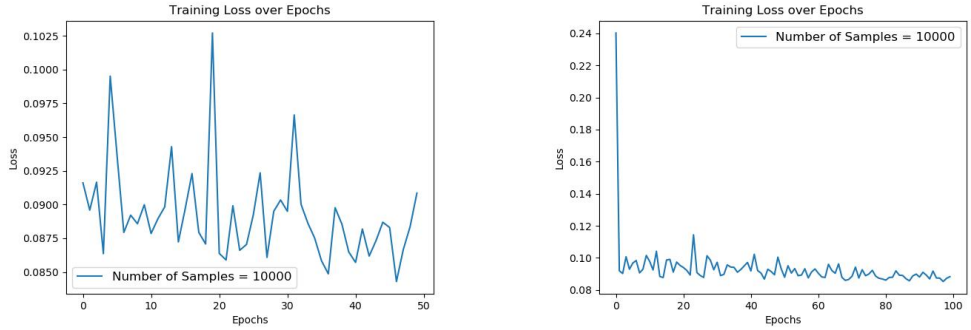

Figure 2. Comparison of the training loss demonstrating the convergence behavior of the network. The left subplot shows the training loss after 50 epochs, while the right subplot illustrates the training loss after 100 epochs. It can be observed that the network converges further with additional training epochs.

4.2 Gaussian Noise

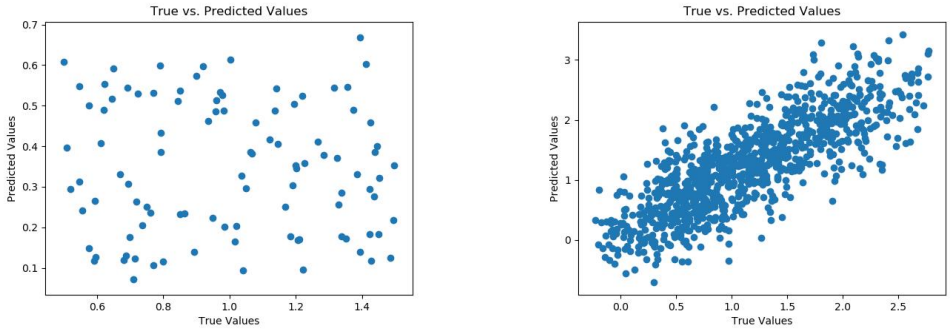

Figure 3. Visualization of Predicted and Real Values with Distribution Noise in Part 2 Problem. Output Dimension is 1 (Left). Output Dimension is 10 with 100 Training Epochs and Batch Size = 10 (Right).

Regarding the Part 2 problem, we introduced an adaptive distribution noise with specified variance and mean during the training of our network[1]. Figure 3 illustrates that the predicted results (z) closely align with real values across varying numbers of values (m) after meticulous network training. This demonstrates the effectiveness of our algorithm in successfully addressing the constrained problem.

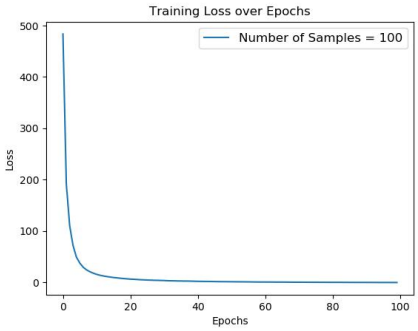

Figure 4. Training loss of function f with specified noise on 100 data samples over 100 epochs.

In Figure 4, it is evident that the function f with added noise can be effectively trained using our framework, achieving convergence after 20 epochs with 100 data samples. This indicates that the

[1]Noise is incorporated in our GitHub example.

predicted results (m) closely align with real values throughout the training of the MLP network. Thus, our framework successfully addresses the training problem in the presence of noise.

4.3 Potential Research Plan

For Q10 and Part2, a comprehensive analysis is provided in section 3.3 on the training process and the choice of algorithms. Bayesian optimization emerges as a promising approach for exploration. When coupled with a more intricate network architecture, it demonstrates enhanced capabilities in solving complex constraint optimization problems.

5 Conclusion

In conclusion, the proposed solution for the function $f(x, y, u, v)$ involves the utilization of a Convolutional Neural Network (CNN) with a custom loss function. This innovative approach addresses the optimization challenge posed by nuanced constraints on a four-variable function. The custom loss function incorporates penalties for enforcing monotonicity and convexity constraints on the model predictions. Through the application of TensorFlow and training on synthetic data, the CNN demonstrates its ability to learn intricate patterns from the input space, providing a versatile framework for solving constrained optimization problems. The methodology showcases the adaptability and scalability of deep learning techniques, particularly with the integration of CNNs in non-traditional domains. The analysis and experiments conducted affirm the efficacy of the proposed solution in achieving convergence and satisfying the specified constraints during the training process.

References

Burak Bartan and Mert Pilanci. Neural spectrahedra and semidefinite lifts: Global convex optimization of polynomial activation neural networks in fully polynomial-time. *arXiv preprint arXiv:2101.02429*, 2021a.

Burak Bartan and Mert Pilanci. Training quantized neural networks to global optimality via semidefinite programming. In *International Conference on Machine Learning*, pages 694–704. PMLR, 2021b.

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

Tolga Ergen and Mert Pilanci. Implicit convex regularizers of cnn architectures: Convex optimization of two-and three-layer networks in polynomial time. *arXiv preprint arXiv:2006.14798*, 2020.

Tolga Ergen, Arda Sahiner, Batu Ozturkler, John Pauly, Morteza Mardani, and Mert Pilanci. Demystifying batch normalization in relu networks: Equivalent convex optimization models and implicit regularization. *arXiv preprint arXiv:2103.01499*, 2021.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*, pages 507–523. Springer, 2011.

Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with bayesian optimisation and optimal transport. *Advances in neural information processing systems*, 31, 2018.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

Mert Pilanci and Tolga Ergen. Neural networks are convex regularizers: Exact polynomial-time convex optimization formulations for two-layer networks. In *International Conference on Machine Learning*, pages 7695–7705. PMLR, 2020.

Arda Sahiner, Tolga Ergen, John Pauly, and Mert Pilanci. Vector-output relu neural network problems are copositive programs: Convex analysis of two layer networks and polynomial-time algorithms. *arXiv preprint arXiv:2012.13329*, 2020.

Arda Sahiner, Tolga Ergen, Batu Ozturkler, Burak Bartan, John Pauly, Morteza Mardani, and Mert Pilanci. Hidden convexity of wasserstein gans: Interpretable generative models with closed-form solutions. *arXiv preprint arXiv:2107.05680*, 2021.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175, 2015.

Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180. PMLR, 2015.

Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. *Advances in neural information processing systems*, 29, 2016.

Colin White, Willie Neiswanger, and Yash Savani. Bananas: Bayesian optimization with neural architectures for neural architecture search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10293–10301, 2021.

Xuying Zhang, Xiaoshuai Sun, Yunpeng Luo, Jiayi Ji, Yiyi Zhou, Yongjian Wu, Feiyue Huang, and Rongrong Ji. Rstnet: Captioning with adaptive attention on visual and non-visual words. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15465–15474, 2021.