"Frank-Wolfe Optimization with L1 Ball Constraints" A Comparative Analysis with Stochastic Gradient Descent

Bahador Mirzazadeh[†], Amirhosein Kalhor[‡]

***Abstract*—This study delves into the application of the Frank-Wolfe optimization algorithm in the context of deep neural networks (DNNs) with a focus on its utilization for training on the Dry Bean dataset. Notably, the investigation incorporates the imposition of constraints through the L1 ball during the optimization process. The primary objective involves the implementation and empirical assessment of the Frank-Wolfe algorithm, accompanied by a meticulous performance evaluation through comparison with the widely adopted Stochastic Gradient Descent (SGD) methodology. Extending the scope, the research entails the application of the Frank-Wolfe optimization algorithm to the training of a DenseNet121 architecture on the CIFAR-10 dataset. This segment of the study provides a comprehensive analysis of the algorithm's versatility and efficacy across distinct neural network architectures. Through rigorous empirical exploration and quantitative evaluation, the report aims to contribute insights into the intrinsic capabilities of the Frank-Wolfe optimization method and its relative performance vis-à-vis SGD in the intricate domain of deep learning.**

***Index Terms*—Frank-Wolfe Optimization Algorithm, L1 Ball Constraints, Deep Neural Networks, Deep Learning Optimization, SGD**

I. Introduction

In the ever-evolving landscape of machine learning, the optimization of neural networks stands as a cornerstone, playing a pivotal role in elevating model performance and driving breakthroughs in artificial intelligence. Amidst traditional gradient-based methodologies, the Frank-Wolfe optimization algorithm has surfaced as a compelling alternative, boasting unique advantages, particularly in contexts characterized by large-scale, sparse, or structured datasets. This research embarks on a comprehensive exploration of the application of the Frank-Wolfe algorithm within the realm of deep neural networks (DNNs), leveraging TensorFlow as the primary framework for implementation. An innovative dimension of this study involves the integration of the L1 ball as a feasible region during the optimization process, introducing constraints that fortify the adaptability of the Frank-Wolfe algorithm. As we delve into the intricacies of this novel approach, the research not only contributes to the expanding body of knowledge surrounding optimization algorithms but also sheds light on the algorithm's performance when confronted with real-world datasets.

[†] bahador.mirzazdeh@studenti.unipd.it
[‡] amirhosein.kalhor@studenti.unipd.it

The primary objective of this investigation is to comprehensively implement and assess the Frank-Wolfe algorithm, leveraging its distinctive characteristics, such as sparsity-inducing properties, and compare its performance against the widely utilized Stochastic Gradient Descent (SGD) method. Through meticulous empirical analysis, we aim to discern the algorithm's strengths and limitations in the context of DNN optimization, shedding light on its convergence properties, computational efficiency, and generalization capabilities. Expanding the purview of our exploration, this research extends its reach to the training of a DenseNet121 architecture on the CIFAR-10 dataset. This secondary task not only broadens the applicability of the Frank-Wolfe algorithm but also facilitates a comparative assessment of its efficacy across different neural network architectures. By undertaking this comprehensive investigation, we aspire to contribute valuable insights into the role of the Frank-Wolfe optimization method in deep learning contexts, thereby advancing our understanding of its potential as an optimization tool in contemporary machine learning research.

II. Related Work

Prior research has explored the application of the Frank-Wolfe algorithm in various optimization tasks, with notable success in convex optimization and sparse learning scenarios. The algorithm's capability to handle convex-concave saddle point problems has been a subject of interest, and its effectiveness in achieving convergence with reduced computational demands has garnered attention in the optimization community.

In the context of machine learning, recent studies have applied the Frank-Wolfe algorithm to tasks such as linear classification, matrix factorization, and dictionary learning, demonstrating its versatility across different domains. However, its application to the training of deep neural networks remains relatively underexplored, warranting a more in-depth investigation.

While SGD has been the de facto optimization method for deep learning tasks, recent comparative studies have indicated that alternative algorithms, like Frank-Wolfe, exhibit promising results, especially in scenarios with specific data characteristics. This research aims to contribute to this evolving body of knowledge by systematically evaluating the Frank-Wolfe algorithm in the realm of DNN optimization, with a

particular emphasis on its performance when confronted with the complexities of real-world datasets, such as the Dry Bean dataset and CIFAR-10 dataset.

III. Processing Pipeline

The execution of the research involves a well-defined processing pipeline encompassing data preprocessing, algorithm implementation, training, evaluation, and comparative analysis. The pipeline is tailored to address the specific objectives of applying the Frank-Wolfe optimization algorithm to deep neural network training on the Dry Bean and CIFAR-10 datasets.

Data Preprocessing: Perform preprocessing steps specific to image datasets, including normalization, augmentation, and reshaping, to prepare both datasets for model training to output directly. This makes it easier for the network to train and reduces the risk of overfitting.

- **Dry Bean Dataset:** During data analysis and preprocessing of the dataset, we observed no missing values and we just scaled features and ensured data compatibility with the chosen optimization algorithm.
- **CIFAR-10 Dataset:** Perform preprocessing steps specific to image datasets, including normalization, augmentation, and reshaping, to prepare the CIFAR-10 dataset for model training.

Algorithm Implementation:

- We have Implemented the Frank-Wolfe optimization algorithm with adaptations for deep neural network training. Incorporate the L1 ball constraints during optimization, ensuring adherence to the feasible region.
- Using Stochastic Gradient Descent (which is already implemented in TensorFlow) as a benchmark for comparative analysis.

Model Training:

- Training the DNN and DenseNet-121 on the Dry Bean and CIFAR-10 dataset using the Frank-Wolfe algorithm and comparing the results with SGD.

Evaluation Metrics:

- The evaluation metrics include accuracy and loss for overall model performance, while precision, recall, and F1 score provide insights into the Frank-Wolfe algorithm's ability to correctly identify positive instances and balance between precision and recall, facilitating a comprehensive comparison with SGD.

Results and Conclusions:

- Interpreting and discussing the results obtained from the experiments, drawing insights into the strengths and limitations of the Frank-Wolfe algorithm in the context of deep neural network optimization.

IV. Preliminaries

Our primary purpose was to evaluate the Stochastic Frank-Wolfe algorithm in a constrained setting with the SGD optimization algorithm as the benchmark. In "Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks" several different learning schedules of types decaying and cyclic were suggested. Due to constraints imposed by lack of proper resources we used a few examples to demonstrate their efficacy and differences.

Algorithm 1 Stochastic Frank–Wolfe (SFW)

Input: Initial parameters $\theta_0 \in \mathcal{C}$, learning rate $\alpha_t \in [0,1]$, momentum $\rho_t \in [0,1]$, batch size $b_t \in [\![1,m]\!]$, number of steps T.

$m_0 \leftarrow 0$
for $t = 0$ **to** $T-1$ **do**
  uniformly sample i.i.d. $i_1, \ldots, i_{b_t}$ from $[\![1,m]\!]$
  $\tilde{\nabla} L(\theta_t) \leftarrow \frac{1}{b_t} \sum_{j=1}^{b_t} \nabla \ell_{i_j}(\theta_t)$
  $m_t \leftarrow (1-\rho_t)\, m_{t-1} + \rho_t\, \tilde{\nabla} L(\theta_t)$
  $v_t \leftarrow \arg\min_{v \in \mathcal{C}} \langle m_t, v \rangle$
  $\theta_{t+1} \leftarrow \theta_t + \alpha_t (v_t - \theta_t)$
end for

Fig. 1. Stochastic Frank-Wolfe Algorithm

TABLE I
Definition of the 13 Learning Rate Functions

Learning Rates		abbr.	k_0	k_1	$g(t)$	Schedule	#Param
Fixed		FIX	k_0	k_0	N/A	fixed	1
Decaying LR	fixed step size	STEP	k_0	0	$\gamma^{floor(t/l)}$	t, l	3
	variable step sizes	NSTEP	k_0	0	Given n step sizes: $l_0, l_1, l_2, ..., l_{n-1}$, $\gamma^i,\ i \in \mathbb{N}\ s.t.\ l_{i-1} \leq t < l_i$	t, l_i	$n+2$
	exponential function	EXP	k_0	0	γ^t	t	2
	inverse time	INV	k_0	0	$\frac{1}{(1+\gamma)^F}$	t	3
	polynomial function	POLY	k_0	0	$(1-\frac{t}{l})^p$	t, l $(l = max\_iter)$	4 (3)
CLR	triangular	TRI	k_0	k_1	$TRI(t) = \frac{2}{\pi} \lvert arcsin(sin(\frac{\pi t}{2l})) \rvert$	t, l	3
	triangular2	TRI2	k_0	k_1	$\frac{1}{2^{floor(\frac{t}{2l})}} TRI(t)$	t, l	3
	triangular_exp	TRIEXP	k_0	k_1	$\gamma^t TRI(t)$	t, l	4
	sin	SIN	k_0	k_1	$SIN(t) = \lvert sin(\pi \frac{t}{2l}) \rvert$	t, l	3
	sin2	SIN2	k_0	k_1	$\frac{1}{2^{floor(\frac{t}{2l})}} SIN(t)$	t, l	3
	sin_exp	SINEXP	k_0	k_1	$\gamma^t SIN(t)$	t, l	4
	cosine	COS	k_0	k_1	$COS(t) = \frac{1}{2}(1 + cos(\pi \frac{2t}{T}))$	t, l	3

Fig. 2. LR Schedules Formulas

V. Model Architectures, Datasets, and Evaluation Metrics

We utilized two different neural network architectures. A simple DNN model with 2 hidden layers, the first one composed of 16 neurons and the second one 8, 471 trainable parameters used on the DryBeans dataset which has 7 different classes, hence 7 output neurons. Softmax Activation function was used for the output layer, with ReLu being utilized by the hidden layers. Next, we trained the famous DenseNet-121 based on Convolutional Neural Networks which was one of the models used in "Deep Neural Network Training with Frank–Wolfe" on Cifar-10, also a dataset used in the same paper to demonstrate a better performance comparison. As the name suggests Cifar-10 has 10 classes, as a result, we also modified the output of Densenet to compute 10 different classes. the following are the details of our datasets:

Dry Bean Dataset Description: Number of Instances: 13,611 Number of Features: 16 Feature Types: Integer, Real Subject Area: Biology Associated Tasks: Classification Data Type: Multivariate Dataset Information This dataset consists of images capturing 13,611 grains of seven different registered Dry Bean, taken with a high-resolution camera. The primary purpose of the dataset is to support classification tasks, specifically to distinguish between seven varieties of Dry Bean based on various features related to form, shape, type, and structure.

CIFAR-10 Dataset Description: Number of Instances: 60,000 (50,000 for training, 10,000 for testing) Number of Classes: 10 Data Type: Multivariate Subject Area: Computer

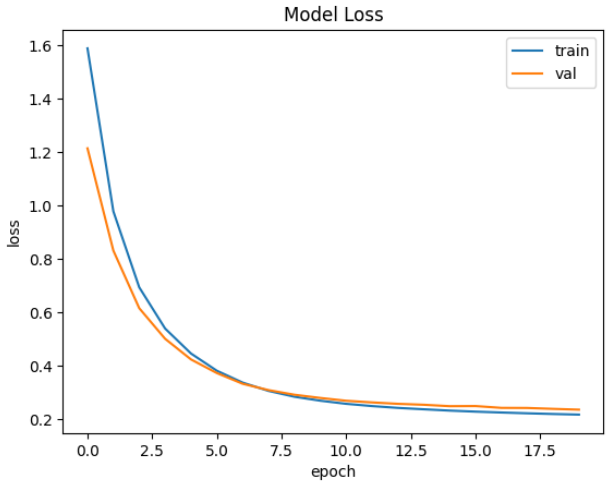

Fig. 3. SGD Loss Curve For Dry Bean Dataset

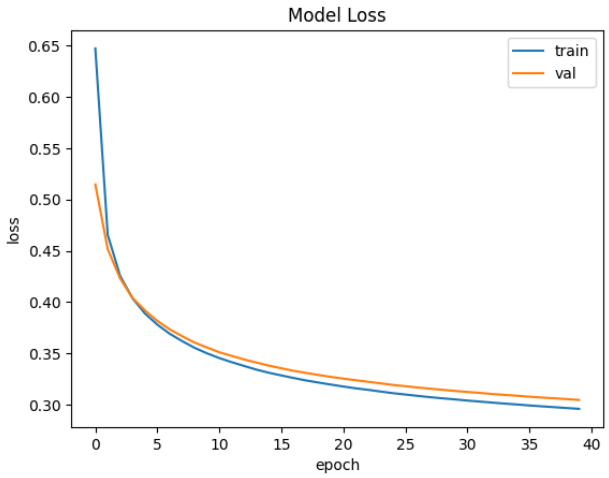

Fig. 4. Frank-Wolfe FIX Loss Curve For Dry Bean Dataset

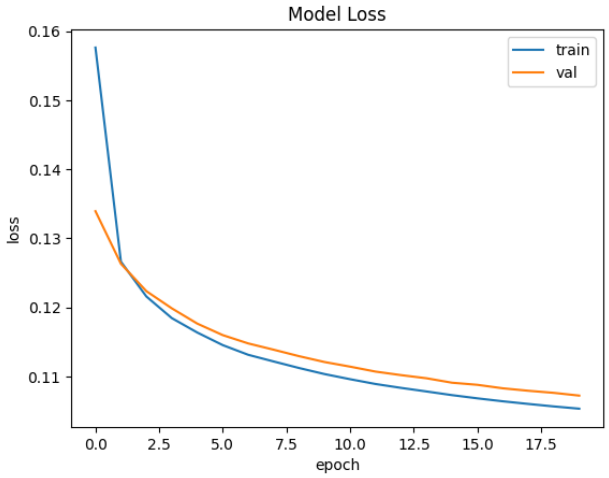

Fig. 5. Frank-Wolfe STEP Loss Curve For Dry Bean Dataset

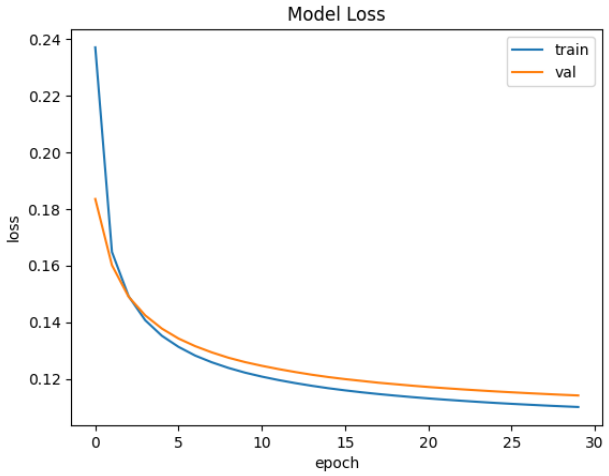

Fig. 6. Frank-Wolfe SINEXP Loss Curve For Dry Bean Dataset

Vision Associated Tasks: Image Classification Image Dimensions: 32x32 pixels with 3 color channels (RGB) Dataset Information The CIFAR-10 dataset is a collection of 60,000 32x32 color images in 10 different classes, with 6,000 images per class. The dataset is split into a training set of 50,000 images and a test set of 10,000 images. Each class represents a distinct object or animal category. For the sake of the evaluation, we used both precision and recall but also the F-1 score as an overall model performance metric.

VI. Results and Discussion

Dry Bean Dataset Results: Due to the volume of the models trained and for the sake of brevity here we will only focus on a few samples that showcase our process. All the rest of our models with details of their performances can be found on our Jupyter Notebook. The following will be four models of each dataset/architecture. A model trained using SGD, which is our benchmark as mentioned before, another using Frank-Wolfe with a fixed learning schedule, and two others will be Frank-Wolfe optimized using a decaying and a cyclic Lr schedule to compare the efficacy and effects of learning rate. The following Four will be of our custom DNN model on the Dry Bean dataset.

CIFAR-10 Dataset Results: Next, we will focus on a more sophisticated model and dataset namely CIFAR-10 and DenseNet121. Considering the size of our dataset and the number of trainable parameters in our model, the training process was rather constrained by the lack of proper resources. As a result the model generally performs worse than the previous one. nevertheless for comparison, whether between SGD and Frank-Wolfe or our LR schedules, we managed to train them sufficiently. We should be careful however to assume these results to be the maximum capabilities of our optimization algorithms.

Results: As it's demonstrated in our training the performance-constrained Frank-Wolfe with l1 ball constrained is on par with the Benchmark SGD as it was suggested in the "Deep Neural Network Training with Frank–Wolfe" paper, concerning L-2 and L-∞ constraints. Also, the cyclic

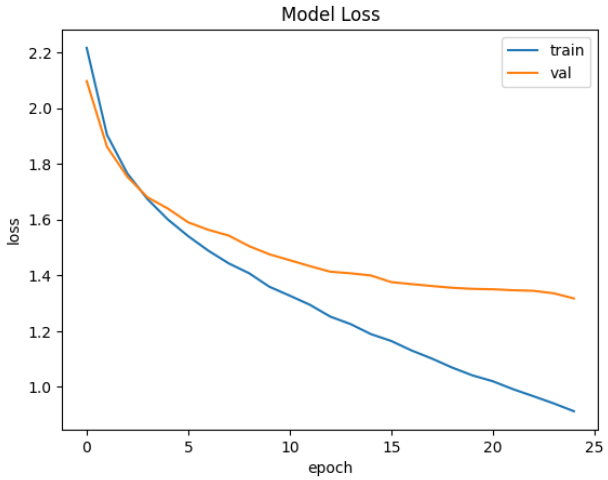

Fig. 7. SGD Loss Curve For CIFAR-10 Dataset

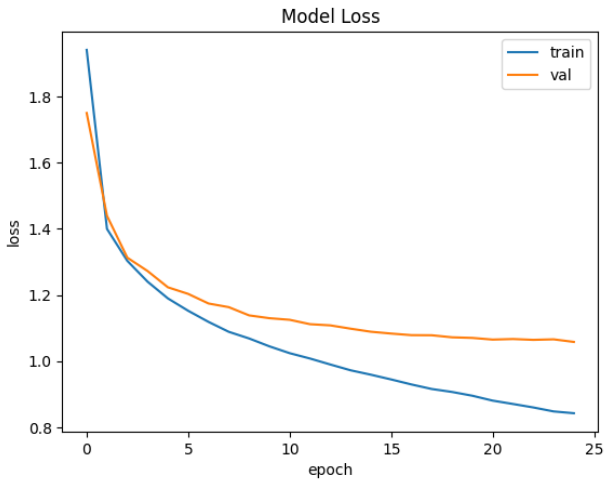

Fig. 8. Frank-Wolfe FIX Loss Curve For CIFAR-10 Dataset

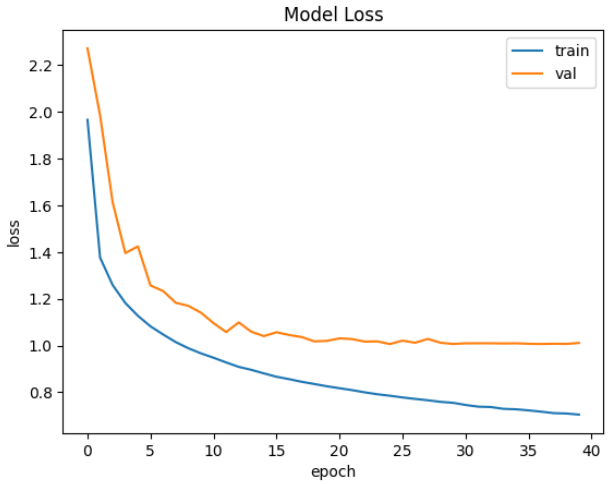

Fig. 9. Frank-Wolfe INV Loss Curve For CIFAR-10 Dataset

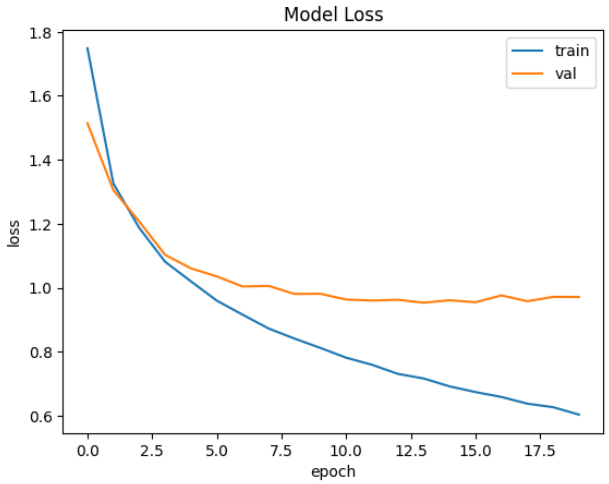

Fig. 10. Frank-Wolfe TRI Loss Curve For CIFAR-10 Dataset

Optimizer	Metric	Outcome
SGD	Testset Loss	0.21987
	Testset Accuracy	0.92
	Testset Precision	0.93
	Testset Recall	0.92
	Testset F1 Score	0.92
Frank-Wolfe FIX	Testset Loss	0.29729
	Testset Accuracy	0.91
	Testset Precision	0.92
	Testset Recall	0.89
	Testset F1 Score	0.90
Frank-Wolfe STEP	Testset Loss	0.10546
	Testset Accuracy	0.89
	Testset Precision	0.90
	Testset Recall	0.87
	Testset F1 Score	0.88
Frank-Wolfe SINEXP	Testset Loss	0.11096
	Testset Accuracy	0.88
	Testset Precision	0.90
	Testset Recall	0.83
	Testset F1 Score	0.86

TABLE I
COMPARISON OF TEST RESULTS FOR DRY BEAN DATASET AND DNN MODEL

Optimizer	Metric	Outcome
SGD	Testset Loss	1.32208
	Testset Accuracy	0.53
	Testset Precision	0.66
	Testset Recall	0.40
	Testset F1 Score	0.50
Frank-Wolfe FIX	Testset Loss	1.06319
	Testset Accuracy	0.63
	Testset Precision	0.73
	Testset Recall	0.53
	Testset F1 Score	0.62
Frank-Wolfe INV	Testset Loss	1.02133
	Testset Accuracy	0.65
	Testset Precision	0.73
	Testset Recall	0.58
	Testset F1 Score	0.64
Frank-Wolfe TRI	Testset Loss	0.98931
	Testset Accuracy	0.68
	Testset Precision	0.74
	Testset Recall	0.62
	Testset F1 Score	0.67

TABLE II
COMPARISON OF TEST RESULTS FOR CIFAR-10 DATASET AND DENSENET MODEL

lr schedules seem to improve the speed of convergence and at times the overall performance of the model as suggested in the ”Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks” paper, although the latter would require further investigation. If you wish to see the results of other LR schedules, they are available on GitHub.