A Appendix

A.1 Training Details and Reproducibility

The models are implemented using the Python programming language, with tensorflow-2.12.0 API and Keras library used to model the training process. Initial hyperparameter tuning is done using optuna-3.1.1 library. For solving the bilevel direction problem, gurobipy-10.0.1 commercial solver is used with academic license access.

In all the experiments, the sparse categorical cross-entropy loss function is used for classification problems. The bounding phase method is used in line search with an initial guess of 0 and $\triangle = 0.1$, where $\triangle$ is the increment parameter. The golden section search is implemented after obtaining the brackets with the bounding phase method. During implementation of the linear program in Gurobi, we retain the default parameter values of the solver. The equality constraints in linear problem (6) are converted to inequalities with a constant tolerance $\delta = 10^{-4}$. Other parameter details of the experimental settings are provided below.

MNIST Dataset: During OPTUNA training, the Adam optimizer with a learning rate of 0.001 is employed. For models with 1,000 and 5,000 samples, we use a batch size of 128 and train for 100 epochs for each OPTUNA trial. Hyperparameters suggested by OPTUNA are drawn from the interval $[10^{-6}, 10^{-1}]$. The number of trials is set to 20, 100, and 200 for the 1HP, 2HP, and 4HP settings, respectively.

CIFAR-10 Dataset: During OPTUNA training, we utilized Adam optimizer with a learning rate of 0.001 and 100 epochs for the CNN architecture, while for the pre-trained ResNet50 model, we set the learning rate to 0.01 and used only 10 epochs. The hyperparameters suggested by OPTUNA were sampled from the range of $[10^{-6}, 10^{-1}]$, and the number of trials for both models was set to 10.

To compute the Hessian matrix $\nabla^2_{(\lambda_c,w)} f(\lambda_c^\circ, w^\circ; S^T)$, we employed the SR1 approximation method. In cases where applicable, we only utilized gradient information from the latest 50 epochs.

Computing Platform: Experiments were conducted on the Google Colab platform utilizing the pro plus subscription. The allocated CPU model was an Intel(R) Xeon(R) CPU @ 2.20GHz with 12 logical processors and 83.5 GB of system RAM. The GPU employed was an Nvidia A100 with 40.0 GB of GPU RAM.

A.2 Additional Results

A.3 Tables

Table 7 indicates the error bars for hyper local tuning over CIFAR-10 (2HP) dataset with ResNet50 model. The table entries with hyper local search are mean and standard deviation for 10 independent hyper local tuning runs.

Table 7: Results for hyper local tuning over grid search, random search, TPESampler, and QMCSampler. The experiments were run on CIFAR-10 (2HP) dataset with ResNet50. The "+" sign indicates additional computational time required for hyper local search.

	ResNet50: CIFAR-10(2HP)							
	Without hyper local search				**With hyper local search**			
Losses	Grid	Random	TPE	QMC	Grid	Random	TPE	QMC
Training Loss	1.085	1.087	1.147	1.123	1.056 ± 1.16^{-4}	0.988 ± 3.08^{-4}	1.147 ± 8.76^{-6}	1.008 ± 1.36^{-4}
Validation Loss	1.769	1.775	1.882	1.846	1.724 ± 1.91^{-4}	1.623 ± 3.83^{-4}	1.882 ± 1.06^{-4}	1.663 ± 1.96^{-4}
Testing Loss	1.904	1.953	1.964	1.968	1.859 ± 1.78^{-4}	1.786 ± 4.52^{-4}	1.964 ± 1.6^{-4}	1.787 ± 2.12^{-4}
Training Accuracy	0.878	0.867	0.867	0.862	0.885 ± 3.73^{-5}	0.889 ± 1.02^{-4}	0.867 ± 3.73^{-5}	0.881 ± 1.37^{-4}
Validation Accuracy	0.843	0.832	0.832	0.826	0.850 ± 4.47^{-5}	0.852 ± 7.42^{-5}	0.832 ± 6.12^{-5}	0.846 ± 9.35^{-5}
Testing Accuracy	0.822	0.810	0.811	0.806	0.828 ± 4.34^{-5}	0.831 ± 1.94^{-4}	0.811 ± 1.82^{-5}	0.825 ± 1.11^{-4}
Runtime (sec)	6663.919	7843.033	6526.534	6782.736	$+216.492 \pm 1.68$	$+239.535 \pm 1.15$	$+172.038 \pm 0.93$	$+226.627 \pm 0.21$

A.4 Plots

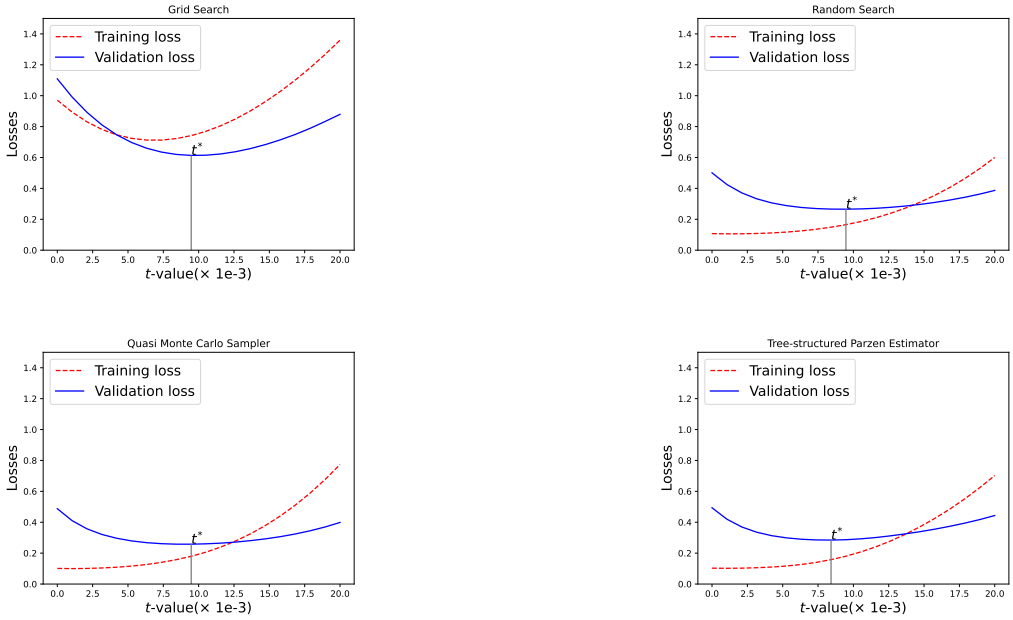

Figure 3: Hyper local search for MNIST (2HP) with 1000 sample points on models found from grid search, random search, TPESampler and QMCSampler.

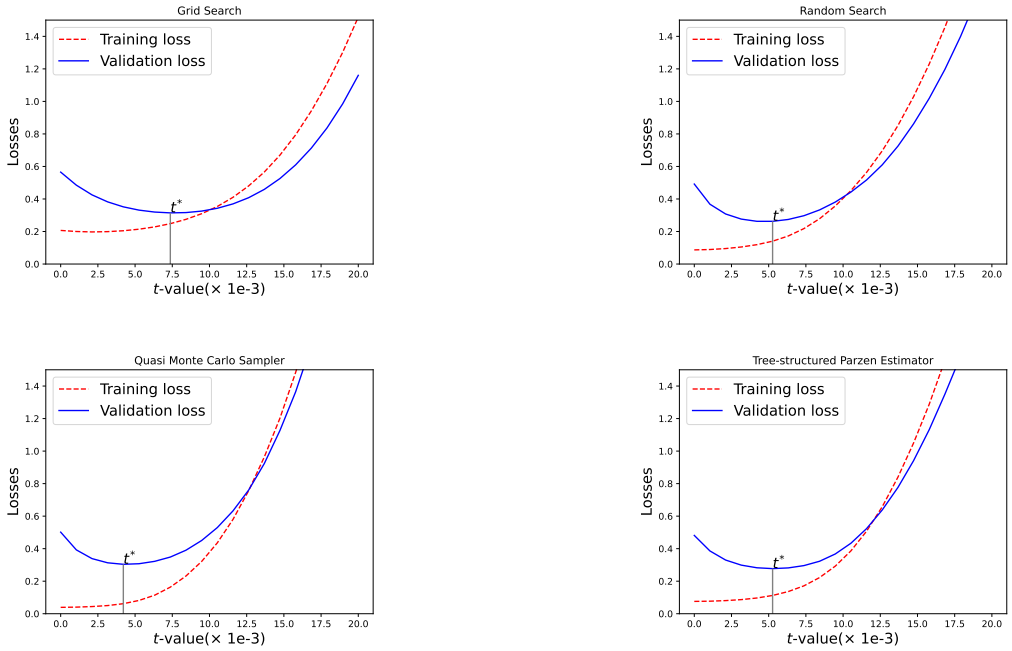

Figure 4: Hyper local search for MNIST (4HP) with 1000 sample points on models found from grid search, random search, TPESampler and QMCSampler.

Note: For all the CIFAR-10 dataset plots, we stop the search process for optimal t after the validation loss stops improving.

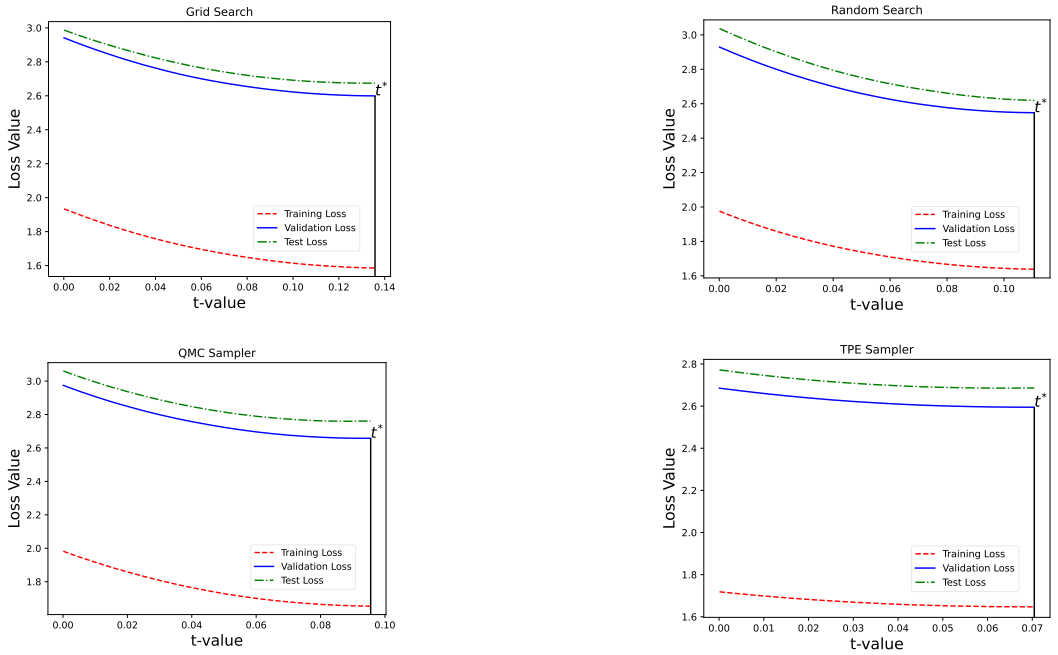

Figure 5: Hyper local search for CIFAR-10 (3HP) on models found from grid search, random search, TPESampler and QMCSampler.

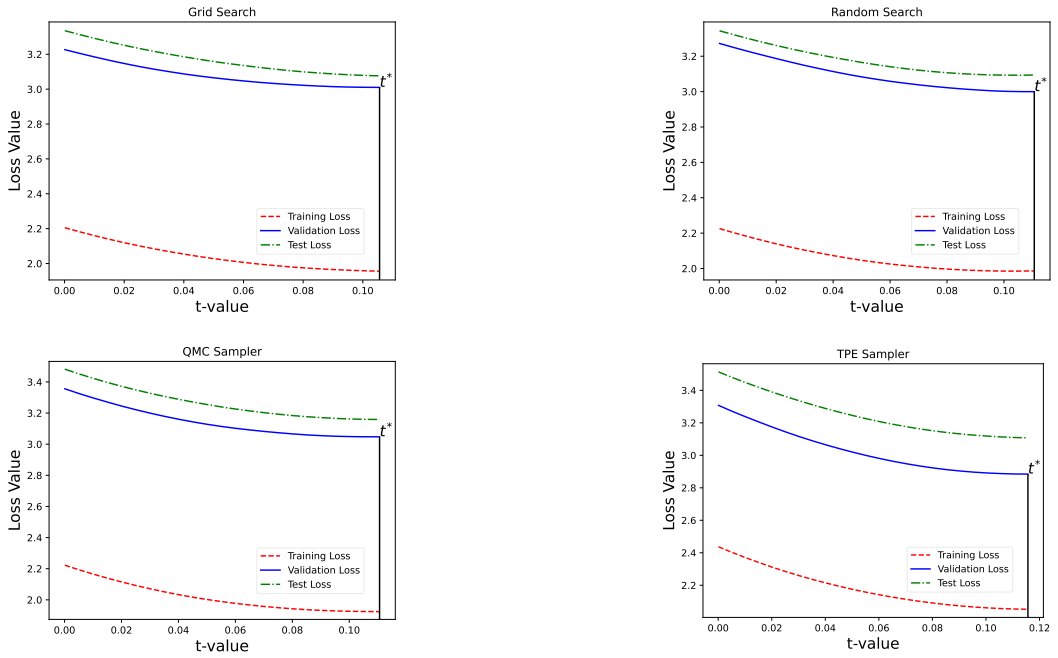

Figure 6: Hyper local search for CIFAR-10 (5HP) on models found from grid search, random search, TPESampler and QMCSampler.

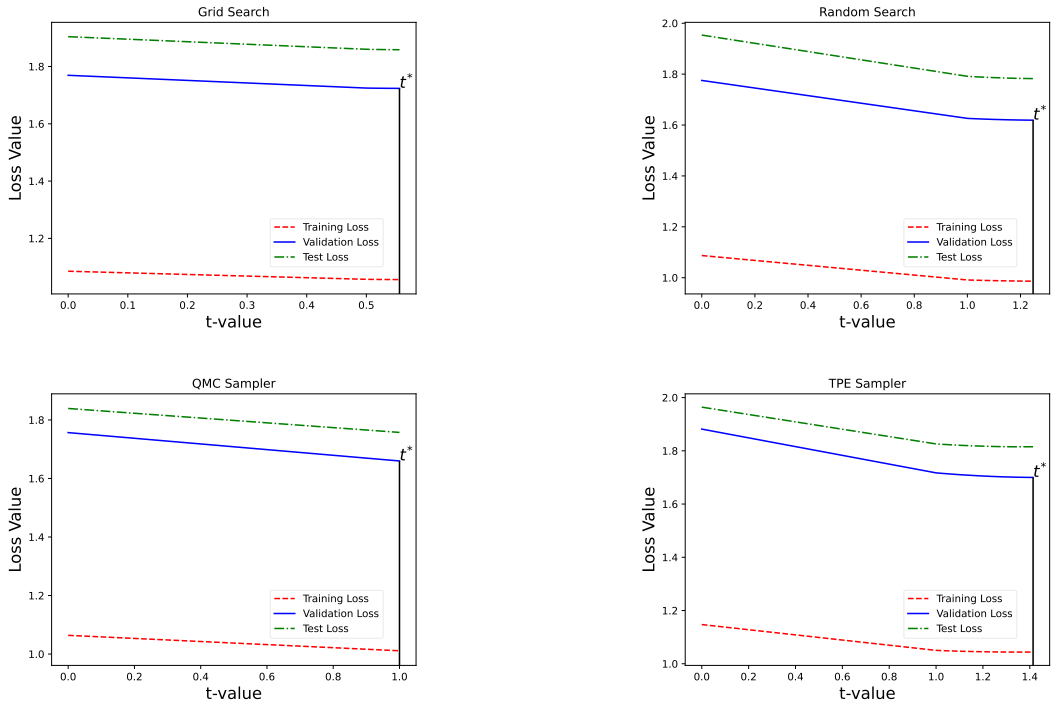

Figure 7: Hyper local search for CIFAR-10 (2HP), with pre-trained ResNet50, on models found from grid search, random search, TPESampler and QMCSampler.