Optimization & Uncertainty in Learning Report

[1st] Nha Van Thanh Do
College of Computing
Georgia Institute of Technology
Atlanta, Georgia
ndo46@gatech.edu

***Abstract*—This report builds on top of previous Supervised Learning report by investigating further on how optimization and regularization choices affect training dynamics, stability, runtime, and multiclass generalization. This study is conducted by applying randomized optimization (RHC, SA, GA), Adam-family optimizer ablations and regularization under a fixed, standard Adam to the same PyTorch MLPs developed last time.**

I. Introduction

We reuse the same Covertype sample, preprocessing, split strategy, and PyTorch neural-network backbone from the SL Report and explain how specific training choices affect validation loss, Macro-F1, balanced accuracy, class-level behavior, and compute cost under controlled budgets.

The SL Report's GridSearchCV selected an MLP of $54 \rightarrow 64 \rightarrow 64 \rightarrow 7$ with ReLU activations. We keep this architecture fixed throughout. The 80/20 split with seed of 85 will be used as well so every number here compares back to SL on an identical test set. $StandardScaler$ is fit on the training split alone, and the 44 binary indicators pass through unscaled.

For gradient methods, one mini-batch forward and backward pass (batch size 256) counts as one gradient evaluation. For RO methods, one validation objective call counts as one function evaluation. Budgets are fixed in advance: 3000 gradient evaluations for part 2-4, 5000 function evaluations per RO algorithm in part 1, and 2000 per cell for the part 2 heatmaps. All runs use seed of 85,86,87. Wall-clock time is also reported.

This report is organized into 4 experimental parts. In part 1, Randomized Optimization algorithms will be applied to the network's final layer. The required algorithms are randomized hill climbing (RHC), simulated annealing (SA), and genetic algorithm (GA) and the total number of trainable parameters is $\leq$ 50,000. Part 2 ablates 7 optimizers on the full network. Part 3 studies 5 regularization techniques under a fixed Adam. And part 4 integrates the best of each.

II. Hypothesis

I believe the choice of optimizer will have a larger effect on neural-network performance than explicit regularization or randomized optimization methods on this Cover Type dataset and current model architecture. This hypothesis is based on the earlier result that the previous model with plain SGD produced relatively low Macro-F1 and balanced accuracy, suggesting that optimization might be the reason of limited performance. Cover Type dataset has both standardized continuous variables and many binary indicator variables which can create different gradient scales across features. Momentum can reduce unstable zig-zag updates while adaptive methods can adjust learning rates separately for each parameter. In [1], the authors also specified that Adam is appropriate for non-stationary objectives and problems with very noisy or sparse gradients. Therefore, I expect momentum-based and adaptive optimizers, specifically Adam-family will reach a target validation loss in fewer epochs and improve Macro-F1 relative to the previous SGD-only baseline. In addition, the earlier SGD model appeared more optimization limited than strongly overfit. LaGrow notes in [2] that AdamW's decoupled weight decay can improve stability and test performance, but that adaptive optimizers do not always generalize better than SGD with momentum. Hence, I think regularization to mainly reduce the training-validation gap rather than significantly improve model performance. For RO algorithms, I think RHC, SA and GA can only provide limited benefit compared to gradient-based optimization. RHC might get stuck in local minima while others might escape local regions but they require many objective-function evaluations. LaGrow also describes that gradient-based optimization is the main training mechanisms for modern Neural Networks [2] that makes me strongly believe that RO algorithms are less efficient and will require more function evaluations and wall-clock time.

III. Part 1 - Randomized Optimization

A. Setup

Following the assignment's freeze-the-backbone protocol, we train the SGD baseline, freeze hidden layer, and re-initialize only the final linear layer. Randomized optimization will search exactly 455 trainable parameters ($64*7+7$), which is far below the threshold of $50,000$. The optimization objective is validation cross entropy, and one function evaluation is one loss computation over the complete validation set. RHC, SA, and GA each received the same budget of 5,000 function evaluations. For GA, each evaluated individual counted as one function evaluation, including individuals in the initial population and later generations. For a controlled comparison, RHC, SA, GA, and gradient-based fine-tuning began from the same re-initialized final-layer parameters within each seed. Results are reported as the median over 3 seeds (85,86,87). Final accuracy, Macro-F1, balanced accuracy, and class-level

metrics are evaluated on the complete test set. The gradient-based reference that fine-tuned the identical final layer with Adam using learning rate 10^{-3} is also included to provide a practical compute efficiency reference.

All RO methods optimize the validation cross-entropy loss on the same fixed validation set and the network is evaluated with gradient computation disabled. Also, the architecture contains neither dropout nor batch normalization, the same weight vector always produces the same validation loss. All algorithms also start from the same reinitialized final-layer weights within each seed, ensuring that performance differences are attributable to the optimization algorithm rather than random initialization.

Table I shows the setting for this experiment.

TABLE I: Part 1 RO Operator Settings.

Algorithm	Parameter	Value
RHC	Step size σ (fixed)	0.1
	Restarts	10
	Plateau patience	200
SA	Initial temperature T_0	1.0
	Cooling (geometric)	$T \leftarrow 0.997\,T$
	Minimum temperature	10^{-3}
	Step size σ (no cooling)	0.1
GA	Population	50
	Selection	tournament, $k = 3$
	Crossover	uniform, 0.7
	Mutation (per-gene Gaussian)	rate 0.1, $\sigma = 0.1$
	Elitism	2
Grad. ref.	Same re-init layer	Adam 10^{-3}, 1500 g

B. Result and Discussion

Fig 1 shows the convergence curves with the dashed line is the SL baseline, the anchor that every RO staircase is trying to reach.

RHC is clearly the most function evaluation efficient method in early search. This curve drops quickly within the first few hundred evaluations. This indicates that the frozen pretrained backbone already supplies well-separated features. The frozen pretrained backbone leaves only the 455 parameter final layer to optimize, substantially reducing the search space, therefore, RHC can improve validation loss rapidly despite using randomized search. However, it then plateaus early and shows no improvement implying that it gets stuck in the local minima and cannot escape from it. Thus, RHC was efficient for rapid local refinement but not for continued improvement under a larger evaluation budget.

SA is the weakest method by validation loss in this case. The validation loss remains the same for the first 1000 evaluations then improves gradually and even though it ends up at 0.849, it's still worse than RHC and GA. This behavior matches the trade off of this algorithm. It's only useful when we need to escape poor local minima but the pretrained solution is already fairly good, accepting worse candidate solutions is less valuable. SA provides more exploration but does not end up into a better result.

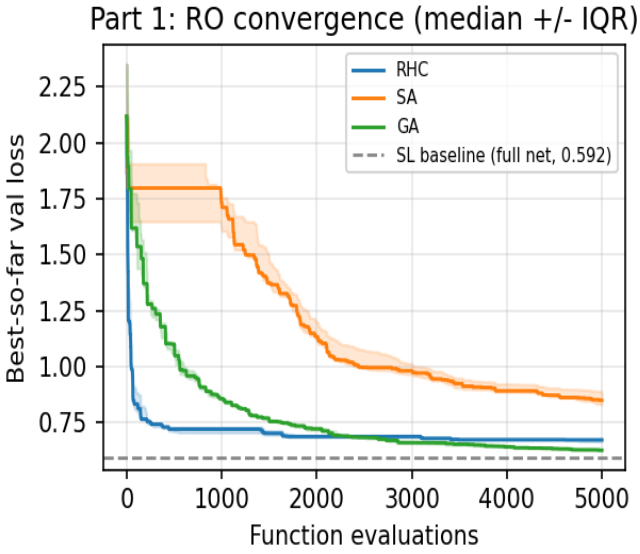

Fig. 1: Best-so-far Validation Loss vs Function Evaluations

GA begins slowly but after around 2500 evaluations, it overtakes RHC. This is consistent with GA algorithm when GA maintains multiple candidate solutions and combines useful parameters through selection, uniform crossover, mutation, and elitism making it more capable of exploring multiple regions of the final layer and escaping the local minima that traps the RHC algorithm. GA therefore returns the best validation loss at 0.626 which is closest to the baseline model.

TABLE II: Part 1 Results (median over 3 seeds)

Method	Val Loss	Test Acc	Macro F1	Bal Acc	Wall (s)	Evals	sd (val)
SL baseline	0.592	0.753	0.518	0.495	0.51	3000	0.0011
RHC	0.672	0.711	0.435	0.437	1.01	5000	0.0112
SA	0.849	0.696	0.474	0.474	1.01	5000	0.0603
GA	0.626	0.743	0.439	0.443	1.05	5000	0.0037
Grad. FT	**0.599**	**0.754**	0.495	0.480	**0.21**	1500	0.0013

Table II summarizes the result across RO algorithms with gradient fine-tune reference. None of the randomized algorithms outperformed gradient fine-tuning of the same final layer. Adam fine-tuning has the best metrics with only 0.21(s) and 1500 evaluations while other RO methods consume all 5000 evaluations. RHC, SA, and GA take much more time and still finish worse. GA and RHC come closest in validation loss and accuracy but its Macro-F1 and balanced accuracy are still low meaning they only do well on the majority classes. Therefore, RO does not improve final performance or stability in this setting. It mainly consumes extra compute.

In addition, comparing between test accuracy and macro-f1 of both RO and gradient fine-tuning, the gap indicates that they work well only with the majority classes while they still collapse on the rare categories.

Looking at the standard deviation across 3 seeds. The seed spread is very small and it confirms that the result is reliable and not seed luck. The Adam fine-tune reference is the most stable of all, which supports our conclusion that gradient fine-tuning is both better and more repeatable than RO algorithms.

IV. Part 2 - Adam Family Optimizer Ablations

A. Setup

This part trains the full network with 8,135 parameters without frozen layers and compares 7 optimizers under one matched budget of 3,000 gradient evaluations where one gradient evaluation is a single mini batch forward and backward pass at batch size of 256. Every optimizer starts from the same cold initialization per seed and uses the same seeds of 85,86,87. To compare convergence speed, we define a time to threshold metric. The number of gradient evaluations an optimizer needs to first reach a validation loss of $l = 0.545$, which is 5% above the lowest validation loss any optimizer reached. The configurations are shown in Table III. The first 3 are SGD-family optimizers and the next 4 are Adam variants. To keep the comparison fair, all SGD-family methods use $\text{lr} = 0.1$ and all Adam-family methods use $\text{lr} = 10^{-3}$.

TABLE III: Part 2 Optimizer Configurations.

Optimizer	LR	Key hyperparameters
SGD	0.1	momentum $= 0$
SGD + Momentum	0.1	momentum $= 0.9$
Nesterov	0.1	momentum $= 0.9$, Nesterov
Adam	10^{-3}	$\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$
Adam (no BC)	10^{-3}	$\beta = (0.9, 0.999)$, no bias corr.
Adam (β_1=0)	10^{-3}	$\beta = (0.0, 0.999)$, RMSProp-like
AdamW	10^{-3}	$\beta = (0.9, 0.999)$, decoupled wd 10^{-2}

B. Results and Discussion

As shown in Fig 2 and table IV, momentum is the most valuable factor on Cover Type dataset and training budget. Plain SGD remains at validation loss 0.591 and never reaches the threshold $l = 0.545$. Adding momentum helps to reduce validation loss to 0.525 and increase Macro-F1 to 0.636. Furthermore, it reaches l in only 1500 evaluations, down from more than 3000. Nesterov is the best in this pool with lowest validation loss at 0.519 and highest Macro-F1 at 0.679 in 1325 evaluations. It's showing that the faster optimization translated into better minority class performance than only better training fit. That is because Nesterov evaluating the gradient after looking ahead, allowing it to correct the update earlier.

The standard deviation across 3 seeds for validation loss and Macro-F1 show that the optimizer ranking is stable rather than driven by a favorable initialization. Nesterov remains the strongest optimizer while also exhibiting one of the smallest dispersions (validation loss sd 0.0012, Macro-F1 sd 0.0083), whereas SGD+Momentum is the least stable. No optimizer produced divergent or failed runs, and every run completed with finite validation loss and test metrics.

The class level analysis in table V explains why optimizer choice affects Macro-F1 much more than overall accuracy.

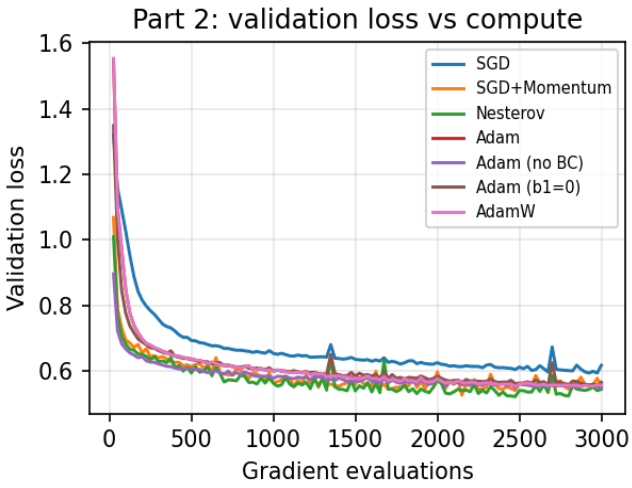

Fig. 2: Validation Loss vs Gradient

TABLE IV: Part 2 Results (Median over 3 Seeds) with SD

Optimizer	Val loss	Test Acc	Macro F1	Bal. Acc	Gen gap	TTT (g)	sd (val)	sd (M-F1)
SGD	0.591	0.752	0.518	0.494	0.017	3000*	0.0017	0.0092
SGD+Mom.	0.525	0.794	0.636	0.608	0.066	1500	0.0094	0.0365
Nesterov	**0.519**	**0.797**	**0.679**	**0.678**	0.069	**1325**	**0.0012**	**0.0083**
Adam	0.551	0.771	0.612	0.575	0.043	3000*	0.0019	0.0050
Adam (no BC)	0.547	0.781	0.637	0.610	0.044	3000*	0.0064	0.0157
Adam (β_1=0)	0.554	0.764	0.613	0.581	0.038	3000*	0.0021	0.0033
AdamW	0.548	0.771	0.608	0.575	0.042	3000*	0.0014	0.0141

Most optimizers perform similarly on the majority classes, but they differ substantially on the minority classes that dominate the Macro-F1 score. Momentum-based methods carry gradient information across updates, allowing those infrequent signals to accumulate instead of being forgotten, which explains why Nesterov achieves the highest Macro-F1 and balanced accuracy. In contrast, adaptive learning rates normalize each parameter's update, reducing the impact of the large, infrequent gradients that rare classes rely on. This impact is shown on Aspen, where Adam trails Nesterov by more than 0.20 F1 despite similar performance on the majority classes. Removing bias correction partially recovers this behavior by allowing larger early updates. This indicates that across the optimizer families, carrying momentum rather than adaptive scaling alone is the key ingredient for learning infrequent classes within a fixed optimization budget.

From the validation loss curves, Adam family is fast but the final results are weak. They flatten around validation loss of 0.55 and never reach the threshold within the budget of maximum 3000 evaluations. However, it does not mean Adam-family is poor because the curves show a sharply drop at the beginning implying Adam found a useful region quickly but under the selected learning rate and compute budget, it did not reach the same final region as Nesterov.

This can be explained by looking at heatmaps in Fig 3 and 4. They are showing that learning rate is the dominant Adam

TABLE V: Part 2 per-class test F1 by optimizer (median over 3 seeds).

Class	SGD	SGD+Mom.	Nesterov	Adam	Adam (no BC)	Adam (β_1=0)	AdamW
Spruce/Fir	0.737	0.788	0.797	0.764	0.770	0.765	0.766
Lodgepole Pine	0.798	0.831	0.837	0.812	0.818	0.810	0.814
Ponderosa Pine	0.728	0.772	0.754	0.750	0.757	0.754	0.750
Cottonwood/Willow	0.095	0.320	0.558	0.516	0.562	0.516	0.516
Aspen	0.192	0.487	0.466	0.265	0.341	0.282	0.289
Douglas-fir	0.302	0.446	0.476	0.403	0.410	0.337	0.384
Krummholz	0.758	0.770	0.762	0.769	0.765	0.762	0.785

TABLE VI: Part 2 Wall-Clock Cost

Optimizer	ms / step	Time to ℓ (s)
SGD	0.18	—
SGD+Momentum	0.18	0.28
Nesterov	0.18	**0.26**
Adam	0.24	—
Adam (no BC)	0.22	—
Adam (β_1=0)	0.24	—
AdamW	0.25	—

hyperparameter. For both (α, β_1) and (α, β_2) grids, validation loss improves consistently as the learning rate increases from 10^{-4} to 3×10^{-3}. The best result occurs at $alpha = 3 \times 10^{-3}$ where validation loss drops below the threshold $l = 0.545$. Across the grid, changing β shifts validation loss far less than changing the learning rate. The heatmaps strongly confirm that Adam's plateau in the main comparison is primarily a learning rate issue rather than indicating it's an ineffective method.

Table VI shows the wall-clock time running on my Macbook Pro M4. Only SGD with momentum and Nesterov are reported with time to l because others never reach this threshold. Furthermore, SGD-family is also cheaper in each step compared to Adam. That makes sense because Adam-family carries extra per parameter moment buffers, square root, and bias-correction terms.

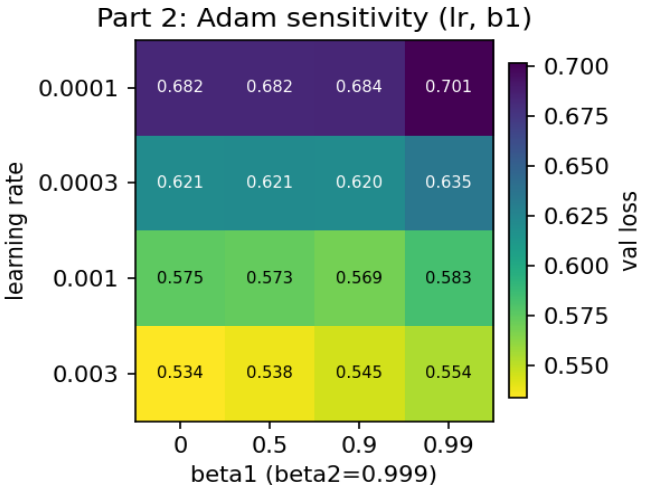

Fig. 3: Sensitivity Heatmap (α, β_1)

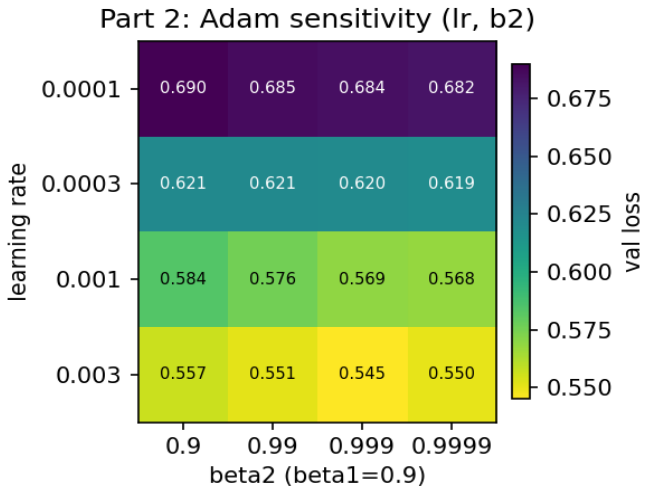

Fig. 4: Sensitivity Heatmap (α, β_2)

The seed stability plot in Fig 5 indicates that the plotted optimizers are insensitive to initialization and mini batch randomness. It's a strong evidence that the algorithms are consistent across seeds and not driven by one lucky run. Adam and AdamW are also reasonably stable and have nearly overlapping bands. Only SGD is visibly noisy.

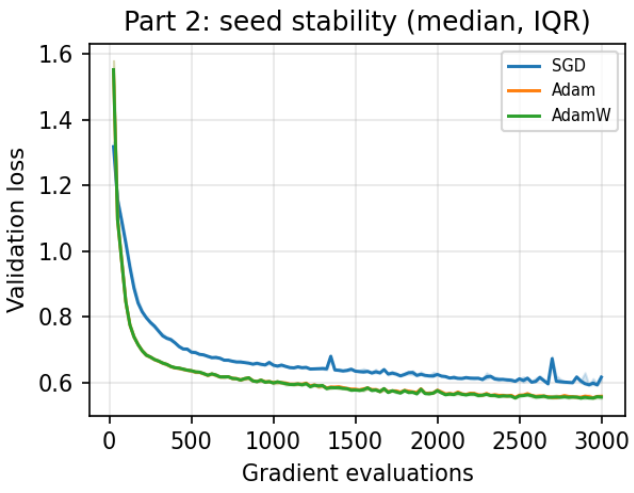

Fig. 5: Seed Stability

For generalization,the gap (train minus validation accuracy) mirrors speed: the fastest optimizers fit hardest (Nesterov 0.069, momentum 0.066), while SGD has the smallest gap (0.017) only because it under-fits within budget. The important point is that Nesterov's larger gap did not hurt its test score. It still has the best test Macro-F1 (0.679). So on this dataset the extra fitting is harmless, and the optimizer, not regularization, is what moved generalization.

Comparing the Adam variants shows which optimizer components mattered most on Cover Type. The most important component was second moment adaptive scaling. When the first moment term was removed by setting β_1 to 0, creating an RMSProp-like version of Adam, performance was nearly unchanged: Macro-F1 was 0.613 compared with 0.612 for standard Adam. This suggests that Adam's per-parameter learning rate adaptation contributed more than its momentum term in this experiment. Bias correction also did not appear essential under the fixed training budget. Removing it did not reduce performance and slightly improved Macro-F1 from 0.612 to 0.637. This is reasonable because bias correction has its greatest effect during the earliest updates and becomes much less important later in training [1], [2]. AdamW tested whether separating the regularization term from Adam's adaptive update would help. AdamW applies weight decay separately from Adam's adaptive gradient update, rather than allowing the adaptive scaling to alter the regularization effect [3]. In this experiment, however, the smaller gap did not produce better test performance: AdamW had slightly lower test Macro-F1 than Adam (0.608 vs 0.612). This pattern suggests that the selected weight-decay value reduced fitting slightly more than necessary for this network and training budget.

V. Part 3 - Regularization Study

A. Setup

For this experiment, we hold standard Adam at Part 2 with $\alpha = 3 \times 10^{-3}$ ($\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$) and change only regularization, so every difference isolates the regularizer rather than the optimizer. All runs use the same full MLP, the same 3000 gradient evaluation budget at batch size 256, and the same 3 seeds (85, 86, 87), reporting the median. As shown in table VII, we sweep the four required regularizers plus label smoothing, documenting the placement details the assignment asks for: dropout is applied after each ReLU, early stopping monitors validation loss and restores the best checkpoint, $L2$ uses Adam's coupled penalty (standard Adam, not decoupled), input noise is zero-mean Gaussian added to training batches only, and label smoothing softens the one-hot targets. We then make the three required comparisons: the no-regularization Adam baseline, the best single regularizer, and the best combination, all under that one fixed budget.

B. Individual Regularizer Sweeps

Table VIII compares the strongest setting of each technique by validation Macro-F1 against the no regularization baseline. From the result, there is no regularizer that beats the baseline on validation Macro-F1 (0.681). Input noise $\sigma = 0.05$ is the strongest individual method. It has the highest regularized validation Macro-F1 (0.673), the lowest validation loss (0.516) while also reducing the train-val gap from 0.059 to 0.050, and is the only technique whose test Macro-F1 gain is corroborated on the validation set. Light dropout (0.1) and small L2 (10^{-5}) raise Macro-F1 to 0.671 and 0.680, but those gains are within seed noise (std $\approx$ 0.001-0.020) and absent on validation, so they are not over claimed. Early stopping at patience 20 reaches baseline level metrics in 2650 evaluations. With the val-loss scale now fixed (hard labels), label smoothing is simply weak: its best setting (ϵ=0.1) trails the baseline on every class-balanced metric. The aggressive ends of the sweeps are harmful and not carried forward: L2 at 10^{-2} collapses the model (Macro-F1 0.394) and dropout at 0.5 drops it to 0.629.

Fig 6 and 7 show the full picture across all individual and combination strategies of Macro-F1 and the generalization gap, respectively

TABLE VII: Part 3 Regularizer Sweeps.

Regularizer	Values swept	Notes
L2 weight decay	$\{10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}\}$	coupled
Early stopping	patience $\{3, 5, 10, 20\}$	restore best
Dropout	$\{0.1, 0.2, 0.3, 0.5\}$	after each ReLU
Input noise (Gauss.)	$\{0.01, 0.05, 0.1\}$	train only
Label smoothing	$\{0.05, 0.1, 0.2\}$	soft targets

TABLE VIII: Best individual regularizer vs. baseline

Technique	Val loss	Val M-F1	Test Acc	Test M-F1	Gap	Grad
No Reg	0.542	**0.681**	0.788	0.670	0.059	3000
L2 (10^{-5})	0.537	0.658	0.784	0.680	0.064	3000
ES (p=20)	0.544	0.650	0.785	0.659	0.054	2650
Dropout (0.1)	0.528	0.668	0.789	0.671	0.051	3000
LS (ϵ=0.1)	0.567	0.628	0.795	0.654	0.062	3000
Noise (σ=0.05)	**0.516**	0.673	0.788	**0.672**	0.050	3000

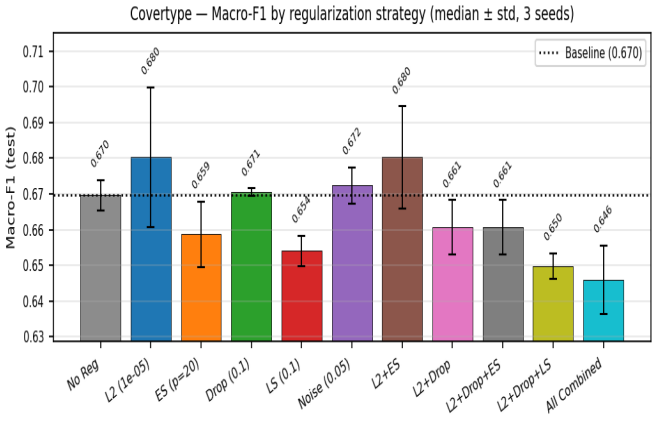

Fig. 6: Test Macro-F1 across all regularization strategies

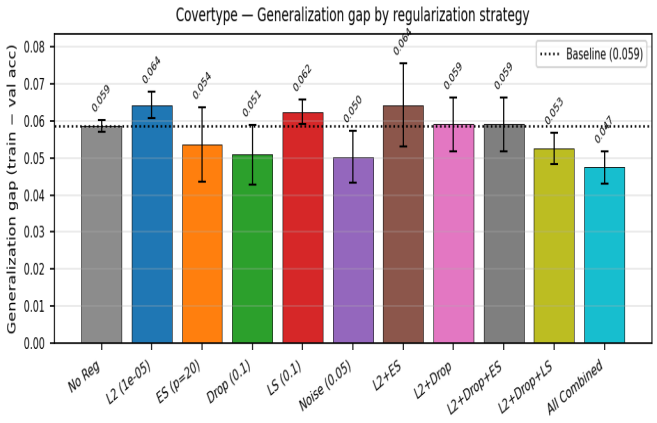

Fig. 7: Generalization Gap across all regularization strategies

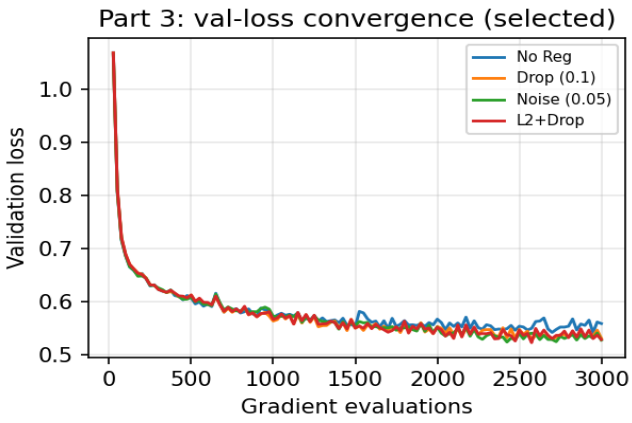

Fig. 8: Validation-loss convergence for selected regularization

C. Regularizer Combinations

Table IX evaluates combinations of the strongest individual regularizer settings. The results show that the regularizers do not combine additively. Although some combinations reduce validation loss or the train-validation gap, none improves validation Macro-F1 beyond the no regularization baseline.

TABLE IX: Regularizer combination results

Combination	Val loss	Val M-F1	Test Acc	Test M-F1	Gap	Grad
L2+Drop	**0.523**	**0.661**	0.787	0.661	0.059	3000
L2+ES	0.537	0.658	0.781	0.680	0.064	2900
L2+Drop+ES	0.523	0.661	0.787	0.661	0.059	3000
L2+Drop+LS	0.570	0.636	0.797	0.650	0.053	3000
All Combined	0.568	0.620	0.795	0.646	0.047	3000

Using validation Macro-F1 as the class-balanced selection criterion, the best combination is the mild $L2 = 10^{-5}$ + Dropout = 0.1 configuration. It achieves the highest validation Macro-F1 among combinations (0.661) and also the lowest validation loss (0.523), so both validation criteria select the same configuration. This agreement supports using L2+Dropout as the selected combination for the next stage. However, its test Macro-F1 is 0.661, below the no-regularization baseline, showing that the best regularized combination is still not better than using no explicit regularization.

Adding early stopping to L2+Dropout has no measurable effect: L2+Drop+ES produces the same validation loss, validation Macro-F1, test accuracy, and test Macro-F1 as L2+Drop. With dropout active, validation loss continues to improve through the 3,000-gradient budget, so patience 20 does not trigger. Adding further regularizers mainly trades class-balanced performance for a smaller generalization gap. The All Combined configuration has the smallest gap (0.047), but also the lowest validation Macro-F1 (0.620) and lowest test Macro-F1 (0.646). This pattern is consistent with over-regularization: the model fits the training data less strongly, but loses useful predictive capacity rather than improving test performance.

Fig 8 shows that no regularization, dropout, input noise, and L2+Dropout have nearly identical validation-loss trajectories. Thus, regularization does not materially change early convergence. It affects only final loss and the train-validation gap. Input noise reaches the lowest final validation loss without reducing test Macro-F1, whereas the other regularizers lower loss or the gap without improving class-balanced test performance. This shows why validation loss alone is insufficient for selection.

D. Discussion

Regularization produced only small gains over the fixed Adam baseline. Input noise (σ=0.05) was the best single technique: it lowered validation loss (0.542 to 0.516), reduced the train-validation gap (0.059 to 0.050), and slightly improved test Macro-F1 (0.670 to 0.672). Most other individual regularizers, and every combination, reduced Macro-F1. Dropout-based strategies produced the smallest gaps, reaching 0.047 for All Combined, but this came with lower class-balanced performance and is consistent with over-regularization rather than improved generalization. Thus, regularization mainly improved the gap and stability, not test Macro-F1 or balanced accuracy.

The best combination by validation Macro-F1 was the mild L2+Dropout configuration, but it still reduced test Macro-F1 by 0.009 and balanced accuracy by 0.024 relative to Adam as shown in table X. It improved several larger classes but reduced F1 for Aspen (-0.038) and Douglas-fir (-0.043). Raw accuracy therefore changed little (0.788 to 0.787), while Macro-F1 and balanced accuracy exposed the minority-class cost. For this imbalanced dataset, the practical choice is input noise, while L2+Dropout is reported as the best combination and carried into Part 4 only because the integration protocol requires it. Overall, lower validation loss or a smaller train-validation gap did not reliably imply better class-balanced test performance.

Comparing the regularization study with the Adam ablations shows that optimizer choice had a far larger impact than

TABLE X: Per-class F1: Adam baseline vs. L2+Dropout.

Class	Adam base	L2+Drop	Δ
Spruce/Fir	0.772	0.787	+0.015
Lodgepole Pine	0.816	0.822	+0.005
Ponderosa Pine	0.730	0.755	+0.025
Cottonwood/Willow	0.545	0.533	−0.012
Aspen	**0.474**	**0.436**	**−0.038**
Douglas-fir	**0.576**	**0.533**	**−0.043**
Krummholz	0.773	0.759	−0.014

regularization because the network was optimization-limited rather than overfitting. Switching from SGD to Nesterov improved Macro-F1 by 0.161, while the best regularizer improved Macro-F1 by only 0.002. Regularization also failed to improve validation loss, balanced accuracy, or minority class behavior consistently: L2, dropout, and label smoothing generally weakened the rare classes because they shrink or soften the already infrequent gradient updates those classes depend on, while early stopping simply ended training before those classes had converged.

VI. Integrated Best Combination (Extra Credit)

A. Setup

Part 4 combines the strongest components from the earlier experiments without introducing a new optimizer or a broad new search. Stage 1 trains the full MLP using standard Adam with the best Part 2 setting $lr = 3 \times 10^{-3}$, and the best Part 3 regularization combination: coupled L2 weight decay $= 10^{-5}$ and dropout $= 0.1$ after each ReLU. This stage uses the same batch size, data splits, seeds, and 3000 gradient evaluation budget as the earlier parts. Stage 2 freezes every layer except the final classification layer and applies the best Part 1 randomized optimization method, genetic algorithm (GA). GA begins from the Stage 1 Adam solution rather than a random initialization. The Adam solution is included in the initial population and retained as the best so far candidate unless GA finds a lower validation loss solution. To follow the Part 4 compute limit, GA uses at most 500 function evaluations, equal to 10% of the Part 1 RO budget. We evaluate one no RO control and three small mutation scales, for four trials total. Results are reported as the median over the same three seeds (85, 86, and 87).

TABLE XI: Part 4 trials (GA mutation scale σ).

Trial	Mutation σ	Func evals
No RO (control)	—	0
GA $\sigma = 0.05$	0.05	500
GA $\sigma = 0.01$	0.01	500
GA $\sigma = 0.10$	0.10	500

B. Results and compute trade-off.

The smallest GA mutation scale, $\sigma = 0.01$, is the only setting that improves the regularized Adam control. It increases Macro-F1 from 0.661 to 0.673, balanced accuracy from 0.635 to 0.656, and test accuracy from 0.787 to 0.796. In contrast, $\sigma = 0.05$ lowers Macro-F1 to 0.656, while $\sigma = 0.10$ produces no Macro-F1 improvement over the control. This suggests that after Adam has trained the network, only very small changes to the final layer are useful; larger GA mutations disrupt the learned decision boundary.

TABLE XII: Part 4 integrated trials (median over 3 seeds)

Trial	Val loss	Test Acc	Macro F1	Bal. Acc	Acc. SD	Func evals
No RO (control)	0.523	0.787	0.661	0.635	0.0060	0
GA σ=0.01	**0.515**	**0.796**	**0.673**	**0.656**	0.0076	500
GA σ=0.05	0.516	0.788	0.656	0.638	0.0061	500
GA σ=0.10	0.521	0.793	0.661	0.639	0.0084	500

The integrated method uses 3000 gradient evaluations in Stage 1 plus 500 final-layer validation evaluations in Stage 2. On the same CPU, Stage 1 required approximately 0.81 seconds and GA fine-tuning required approximately 0.10 seconds, for a total of approximately 0.91 seconds. Thus, the best GA result improves Macro-F1 by 0.012 over the regularized Adam control at the cost of 500 additional function evaluations and approximately 12% more wall-clock time.

The stability result is mixed. The test accuracy standard deviation is 0.0076 for the best GA setting, compared with 0.0060 for the no-RO control. Therefore, GA improves the median result but does not reduce variation across seeds.

The integrated GA result is better than its Adam+regularization control, but it does not beat the best standalone model from Parts 1-3. The best integrated configuration reaches Macro-F1 0.673, while Nesterov reaches Macro-F1 0.679. Therefore, GA fine-tuning provides a small local improvement after Adam training, but it is not the most compute-effective path to the best overall result.

C. Hypothesis resolution.

The full study supports the main hypothesis that optimizer choice was the strongest driver of performance on Covertype. Momentum-based methods, especially Nesterov, reached the target validation loss faster than plain SGD and produced the strongest overall Macro-F1. Changing the optimizer increased Macro-F1 from 0.518 to 0.679, while the best regularizer improved it by only 0.002 and stronger regularization often reduced minority-class performance.

Randomized optimization provided limited value. Standalone RO methods performed worse than gradient-based fine-tuning, while GA fine-tuning in Part 4 improved its Adam+regularization control by 0.012 Macro-F1 but required 500 additional function evaluations and did not improve seed stability. The final integrated model remained below Nesterov (0.673 versus 0.679 Macro-F1). Thus, RO can be used as a small final-layer refinement step, but the evidence does not support it as the best overall strategy for this task.

TABLE XIII: Master summary across all parts

Stage (best of part)	Val. loss	Test Acc	Macro-F1	Bal. Acc	Compute
SL SGD baseline	0.592	0.753	0.518	0.495	3,000 grad
P1 best RO (GA)	0.626	0.743	0.439	0.443	5,000 func
P1 gradient FT (reference)	0.599	0.754	0.495	0.480	1,500 grad
P2 best optimizer (Nesterov)	**0.519**	**0.797**	**0.679**	**0.678**	3,000 grad
P3 Adam baseline (no reg)	0.542	0.788	0.670	0.659	3,000 grad
P3 best single (Noise σ=0.05)	0.516	0.788	0.672	0.659	3,000 grad
P3 best combo (L2 + Dropout)	0.523	0.787	0.661	0.635	3,000 grad
P4 integrated (GA σ=0.01)	0.515	0.796	0.673	0.656	3,000 grad + 500 func

VII. Discussion

A. Summary

Table XIII summarizes the best result from each part under the same 3 seed evaluation. The clearest finding is that the optimizer was the dominant lever. Replacing the SL SGD baseline with Nesterov momentum increased test Macro-F1 from 0.518 to 0.679, the largest improvement in the study. Nesterov achieved the best test accuracy, balanced accuracy, and Macro-F1. Its validation loss was edged out only by the more heavily tuned part 3 and 4, which did not translate into better class balanced test performance. In contrast, regularization and randomized optimization produced only small changes after a strong gradient-based optimizer was already selected.

B. Optimization, generalization, and efficiency.

Momentum-based methods improved both convergence and class-balanced test performance. Nesterov and momentum SGD reached the validation-loss threshold in approximately 1300 to 1500 gradient evaluations, while plain SGD did not reach it within the budget. The optimizer change also improved minority class performance substantially, which explains why Macro-F1 rose much more than raw accuracy. Randomized optimization was not competitive as a training method: the best RO result by validation objective, GA, reached Macro-F1 0.439 after 5000 function evaluations, below the 0.495 achieved by a 1500 evaluation gradient fine-tune of the same final layer. Thus, gradient-based optimization was both more accurate and more compute-efficient.

C. Regularization and class-level trade-offs.

Regularization mainly controlled the train-validation gap rather than improving test performance. Input noise was the best individual regularizer, increasing Adam Macro-F1 only slightly from 0.670 to 0.672. Heavier combinations reduced the gap but harmed balanced performance: L2+Dropout reduced Macro-F1 to 0.661 and lowered balanced accuracy to 0.635. The loss came mainly from minority classes, especially Aspen and Douglas-fir. Therefore, lower validation loss or a smaller generalization gap did not always indicate better class-balanced generalization. For this imbalanced dataset, Macro-F1, balanced accuracy, and per-class F1 are more reliable selection criteria than accuracy alone.

D. Integrated result and limitations.

The Part 4 GA refinement improved its regularized-Adam control from Macro-F1 0.661 to 0.673 using 500 additional function evaluations, but it did not reduce variation across seeds and still did not exceed Nesterov's 0.679. This shows that RO can make a small local improvement after gradient training, but it does not justify its additional compute as the main optimization strategy. These conclusions are limited to the fixed MLP, the approximately 20000-instance sample, and 3 seeds. Minority-class estimates are especially noisy because some test classes are small.

VIII. Conclusion

The hypothesis is supported that on this Covertype MLP, the optimizer was the most important factor, regularization was a secondary tool for controlling overfitting, and randomized optimization was not an efficient replacement for gradient training. Nesterov momentum was the best overall method, improving Macro-F1 from 0.518 for the SGD baseline to 0.679. Input noise provided the best regularization result but improved Macro-F1 by only $+0.002$, while the integrated Adam, regularization and GA recipe reached 0.673 and remained below Nesterov.

The practical recommendation is to tune the optimizer first, then apply only light regularization while monitoring Macro-F1 and minority-class F1. Randomized optimization should be reserved, if used at all, for a small final-layer refinement rather than full model training.

AI USE STATEMENT

I used ChatGPT, Claude and Windsurf IDE to brainstorm the project plan, generate/ debug/ refactor code and OverLeaf AI suggestions for Latex syntax. I also used ChatGPT to learn more about the concepts and Python libraries. The final design, workflow and analysis are my own. I reviewed, verified, and understand all assisted content.

References

[1] D. Kingma, J. Ba, "Adam: A Method for Stochastic Optimization" ICLR, 2015.

[2] T. LaGrow "From SGD to AdamW: History, Math, and Mechanisms" Georgia Institute of Technology, Sep 22, 2025.

[3] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," ICLR, 2019.