APPENDIX - HUB: ENHANCING LEARNED OPTIMIZERS VIA HYBRID UPDATE-BASED STRATEGY

Anonymous authors
Paper under double-blind review

A EXPERIMENTS

A.1 EXPERIMENTAL SETUPS FOR SECTION 4

We present the experimental setup and extra corresponding results (Figure 10, Figure 11, and Figure 12) for the tasks outlined in Tables 1, 2, 3,4, and 5 in Section 4 of our paper. All experiments were conducted using Haiku, a JAX-based framework.

MLP The Sinusoidal Representation Networks (Siren)(Strümpler et al., 2021) is a multi-layer perceptron model that utilizes the Implicit Neural Representation (INR) method. Siren takes in the corresponding coordinate values (x, y, and z) of the target image as input and outputs its RGB/grayscale value at that specific coordinate. Another well-known INR-based model for new view synthesis is Neural Radiance Fields (NeRF)(Mildenhall et al., 2020).

The Siren network we employ consists of 7 layers, with a frequency rate (fr) of 2.2 and a sinusoidal frequency hyperparameter (w0) set to 20. We utilize the Adam and Adamax optimizers with an initial learning rate of 0.001, beta1 = 0.9, beta2 = 0.999, epsilon = 1e-8, employing cosine learning rate decay without warmup - consistent with our reference codebase(Yang et al., 2022). For VeLO, we use default settings as it is hyperparameter-free; for HUB, we hybridize Adamax (using the same setting as when using Adamax alone) with VeLO. The batch size we use is 100k coordinates. We utilized the HiP-CT dataset(Walsh et al., 2021) as our primary image source, which offers cellular-level imaging of various organisms across multiple anatomical planes. Specifically, we focused on four available organs in this dataset (Lung, Heart, Kidney, and Brain). For the purpose of a standardized and equitable comparison, we have partitioned the data into 64^3, 256^3 and 512^3 sizes, and the raw data was utilized without pre-processing, but the coordinates were normalized to $[-1, 1]^3$ in INR-based methods. We run the experiment on a Geforce RTX3090Ti GPU and the PSNR values presented in Table4 represent the average of 10 trials for each size, with a compression ratio of x256.

Resnet We use the Resnet-50(He et al., 2015) model in this experiment, the blocks per group are 3, 4, 6, and 3 and the channels per group are 256, 512, 1024, and 2048. The bottleneck is adopted and the stride is (1, 2, 2, 2). For data augmentation, we first resized the training image using the BICUBIC method to (384, 384) and applied Auto Augment(Cubuk et al., 2019), random horizontal/vertical flip and random crop to get the final resolution (224, 224). The batch size we choose is 128 and in total 100 epochs of training. Adam and Adamax optimizer used an initial learning rate of 7.5e-4, with a 10 epochs warmup followed by cosine decay in learning rate. The rationality behind this learning rate is shown in Figure 1. We ran the experiment on an A100 GPU.

RNN and Neural ODE Our experiments focused on the application of RNN and Neural ODE in two specific tasks: trajectory prediction and lane-keeping. For these tasks, we employed three different models. Firstly, we used a long short-term memory (LSTM)(Hochreiter & Schmidhuber, 1997) network with 64 hidden units for both trajectory prediction and lane-keeping tasks. Secondly, we utilized an 8-cell liquid time-constant (LTC)(Hasani et al., 2020) network for the trajectory prediction task, and finally, a 19-cell LTC for the lane-keeping task. LTC models are wired with neural circuit policies (NCP)(Lechner et al., 2020) with default 75% sparsity. These model choices were based on the experiment in (Hasani et al., 2020; Lechner et al., 2020).

LTC represents a novel class of time-continuous recurrent neural network models. Rather than defining a learning system's dynamics through implicit nonlinearities, LTC constructs networks of linear first-order dynamical systems modulated by nonlinear interlinked gates, drawing inspiration from

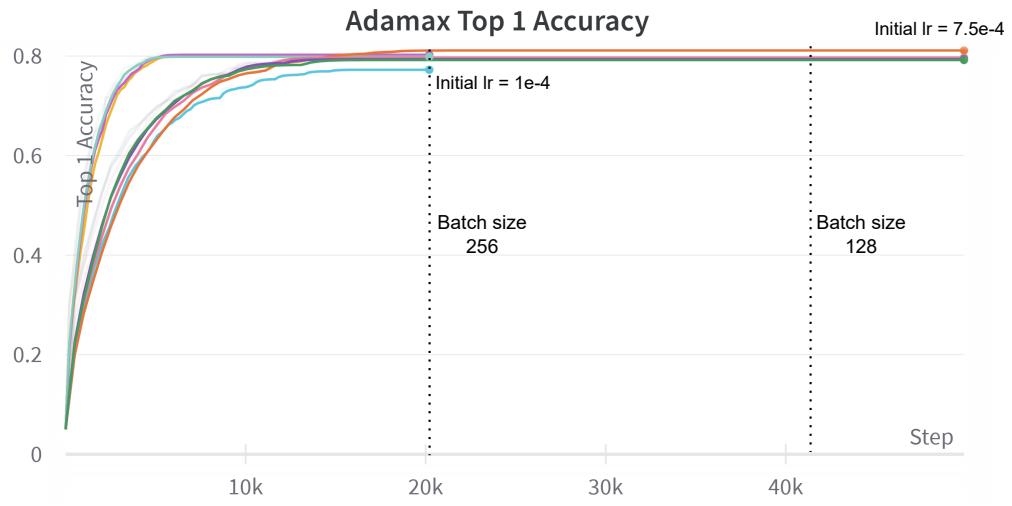

Figure 1: Best top 1 Accuracy for AdamW. This figure demonstrates the tuning process for AdamW, the best top 1 accuracy is the average on CIFAR10 and CIFAR100

principles of brain-neural computation. Specifically, the neural dynamics of a single LTC cell are governed by continuous-time ordinary differential equations (ODEs) originally developed to capture the dynamics of the nervous system in small organisms like C. elegans. The ODE dynamics of the LTC cell are then integrated into a recurrent process to establish a temporal dimension. This design significantly enhances the expressive power of an LTC cell while increasing its interpretability. The amplification of a single cell suggests that complex tasks can be accomplished using a much smaller LTC model compared to other modern RNN models. Furthermore, the connectivity of LTC cells is defined by NCP, which draws inspiration from biological systems. NCP incorporates a four-layer hierarchical network topology comprising sensory, inter-neuron, command, and motor layers, along with polarity (inhibitory and excitatory) connections. This NCP design allows LTC models to achieve a connection sparsity of up to 90% without sacrificing their expressive power.

For the trajectory prediction task, our objective is to predict a sine curve signal wave. During training, three sine curves with different frequencies are used as inputs, and the output is a periodical function curve. This task represents a classical Neural ODE scenario that parameterizes the Fourier transform process. Due to the relative simplicity of the task, we set the learning rates for Adam to 0.01, with beta1 = 0.9, beta2 = 0.999, epsilon = 1e-8, and momentum = 0.9. As for the HUB and LGL2O optimizer, we combine Adam and VeLO without modifying the default hyperparameters.

For the lane-keeping task, we utilize the DeepPiCar dataset, which consists of high-quality indoor close environment images and corresponding turning angles captured by a robotic car. We split the dataset into training, validation, and test sets in a ratio of 75:10:15. Before feeding the images into the LSTM/LTC prediction head, we extract kernel features using a CNN with the structure present in Table 1. Adam is empirically tuned with an initial learning rate of 0.01, which decays in each of the 300 training steps with gamma = 0.3. Similar to before, we combine Adam and VeLO for the HUB and LGL2O optimizer, maintaining the tuned hyperparameters.

The aforementioned task design, data preprocessing, and optimizer setup are based on (Hasani et al., 2020; Lechner et al., 2020), we simply shrink the CNN scale to match up a smaller dataset we use for the lane-keeping experiment. The results in Table 2 are the average of 10 trials. We ran the experiment on a Geforce RTX3090Ti GPU.

Vision Transformer Training Vision Transformer(Dosovitskiy et al., 2020) (ViT) on small datasets can be challenging, as the model tends to strongly overfit the training data without careful selection of hyperparameters and model size, leading to underperformance. In this case, we

Table 1: CNN block structure. This CNN block is for feature extraction, so the size is restricted. The CNN structure is similar to (Hasani et al., 2020; Lechner et al., 2020)

CNN block structure		
Layer(type)	Output Shape	Parameter Count
conv2d 1 (Conv2D)	(None, 31, 98, 24)	1824
conv2d 2 (Conv2D)	(None, 14, 47, 36)	21636
conv2d 3 (Conv2D)	(None, 5, 22, 48)	43248
conv2d 4 (Conv2D)	(None, 3, 20, 64)	27712
dropout 1 (Dropout)	(None, 3, 20, 64)	0
conv2d 5 (Conv2D)	(None, 1, 18, 64)	36928
flatten 1 (Flatten)	(None, 1152)	0
dropout 2 (Dropout)	(None, 1152)	0
dense 1 (Dense)	(None, 100)	115300
dense 2 (Dense)	(None, 50)	5050
Total Parameter Count: 251,698		

adopt the setups discussed in (Lee et al., 2021), which provide detailed guidelines for training ViT on small datasets.

Regarding the model size, we configure the ViT with a depth of 9, a hidden dimension of 192, and 12 attention heads. The patch size for the patch embedding layer is set to 8. For image preprocessing, we employ techniques such as CutMix(Yun et al., 2019), Auto Augment(Cubuk et al., 2019), random horizontal/vertical flip, and random crop.

To optimize the model, we utilize AdamW(Kingma & Ba, 2014) as the tuned optimizer. The initial learning rate is set to 0.003, with a warmup period of 1/10 of the total epochs, followed by cosine decay in the learning rate. Adam share the same hyperparameter settings as AdamW and are used as baselines. Additionally, we hybridize AdamW with VeLO, incorporating tuned hyperparameters. A weight decay of 0.05 and a batch size of 128 are employed. The training is conducted for a total of 100 epochs on an A100 GPU.

As a validation for scaling up, we followed the instructions outlined in [1] to train a large vision transformer with 16 as the patch size on the Imagenet dataset. The initial learning rate for both AdamW and Adam was set at 0.01, with a total of 20k training steps. Additionally, we implemented a warmup period of 1/10 and cosine decay. For HUB and LGL2O, we utilized hybrid AdamW with VeLO. Other procedures, such as image preprocessing, remain consistent with the aforementioned setups.

Xception The Xception(Chollet, 2016) we employ has been pre-trained on the Imagenet1k dataset(Deng et al., 2009a). For our downstream tasks, we selected CIFAR10, CIFAR100, and Tiny-imagenet datasets. Following the methodology outlined in our reference source [2], we conducted full fine-tuning experiments.

Regarding image preprocessing, we initially resized the training images using the BICUBIC method to a resolution of (384, 384). We then applied Auto Augment(Cubuk et al., 2019), random horizontal/vertical flip, and random crop to obtain a final resolution of (299, 299). The batch size utilized during training was set to 128, and we fine-tuned the model for a total of 100 epochs on an A100 GPU.

In this experiment, the hyperparameters for AdamW were carefully tuned. Since it is a fine-tuning task, we empirically determined that a relatively small learning rate is desirable. We found that learning rates below 5e-4 generally yielded better results (see Figure 2). Therefore, we selected an initial learning rate of 1e-4. We employed a warmup period of 1/10 of the total epochs, followed by cosine decay, which is a standard setup.

[1] `https://github.com/google-research/vision_transformer`

[2] Codebase: `https://github.com/abarcel/haikumodels`

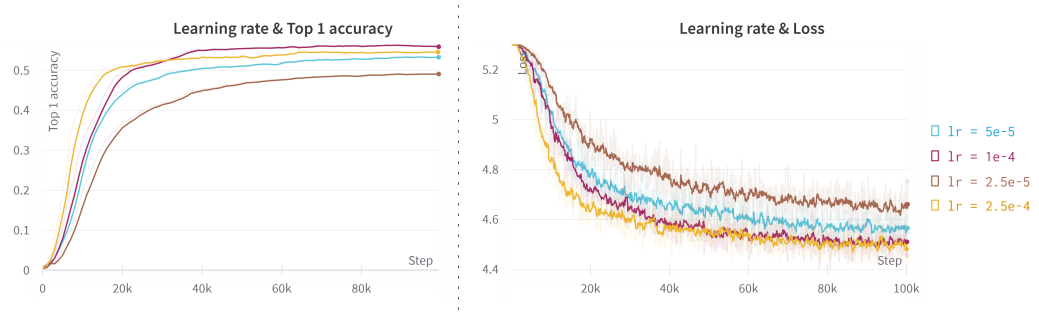

Figure 2: The loss curve and top 1 accuracy (here demonstrate Tiny-imagenet dataset) are closely tied to the initial learning rate. As depicted in the figure, a higher initial learning rate (2.5e-4) leads to faster convergence but results in an early plateau, while lower initial learning rates (5e-5 and 2.5e-5) converge slowly but at relatively suboptimal points. Therefore, an optimal initial learning rate of 1e-4 is recommended.

As mentioned in Section **??** of our paper, VeLO performs poorly in fine-tuning tasks. In Section 3.1, we briefly discussed the reasons behind invert weighting HUB. In this particular case, although we chose to hybridize AdamW with VeLO, the HUB strategy we employed differed from that in train-from-scratch tasks. We will provide detailed explanations of the variations of HUB in the upcoming Section C.

A.2 Additional Experiments

Compare HUB to reinitialize extend training method Although larger optimizers may achieve satisfactory performance with fewer iterations, it is crucial to consider the potential increase in computational overhead per step. This overhead can result in issues where the gradient and loss values become unstable, thereby impeding the training process. To tackle this challenge, two reinitialization training strategies have been previously proposed in (Metz et al., 2022) and will be compared to the HUB method in this analysis.

The experiment employed in this study is the LTC trajectory prediction task. In our paper, we addressed the issue of NAN when using VeLO as an optimizer. (see section 4.1 RNN and Neural ODE paragraph)

- Increase Steps: Continue from the final optimizer state of the previous run but increase the number of steps.
- Min-Loss Reinit: Continue from the min-Loss optimizer state of the previous run.

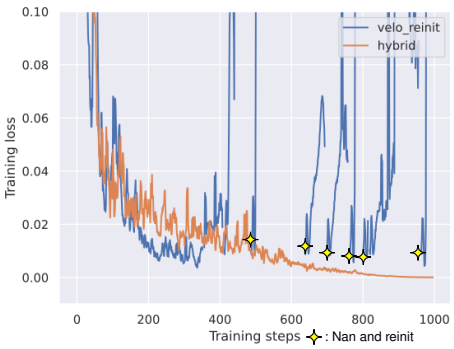

Figure 3: Min-Loss Reinit Solution can help solve NAN Problem but show poor performance

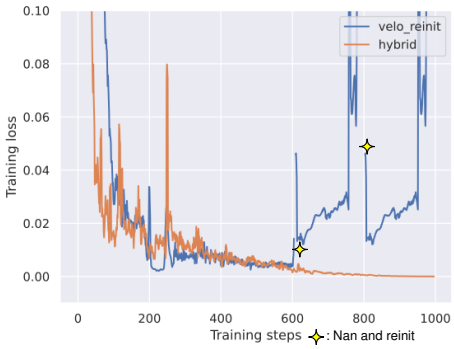

Figure 4: Increase Steps Solution also can help solve NAN Problem but built-in cycle in velo leads to this just repeating the poor performance

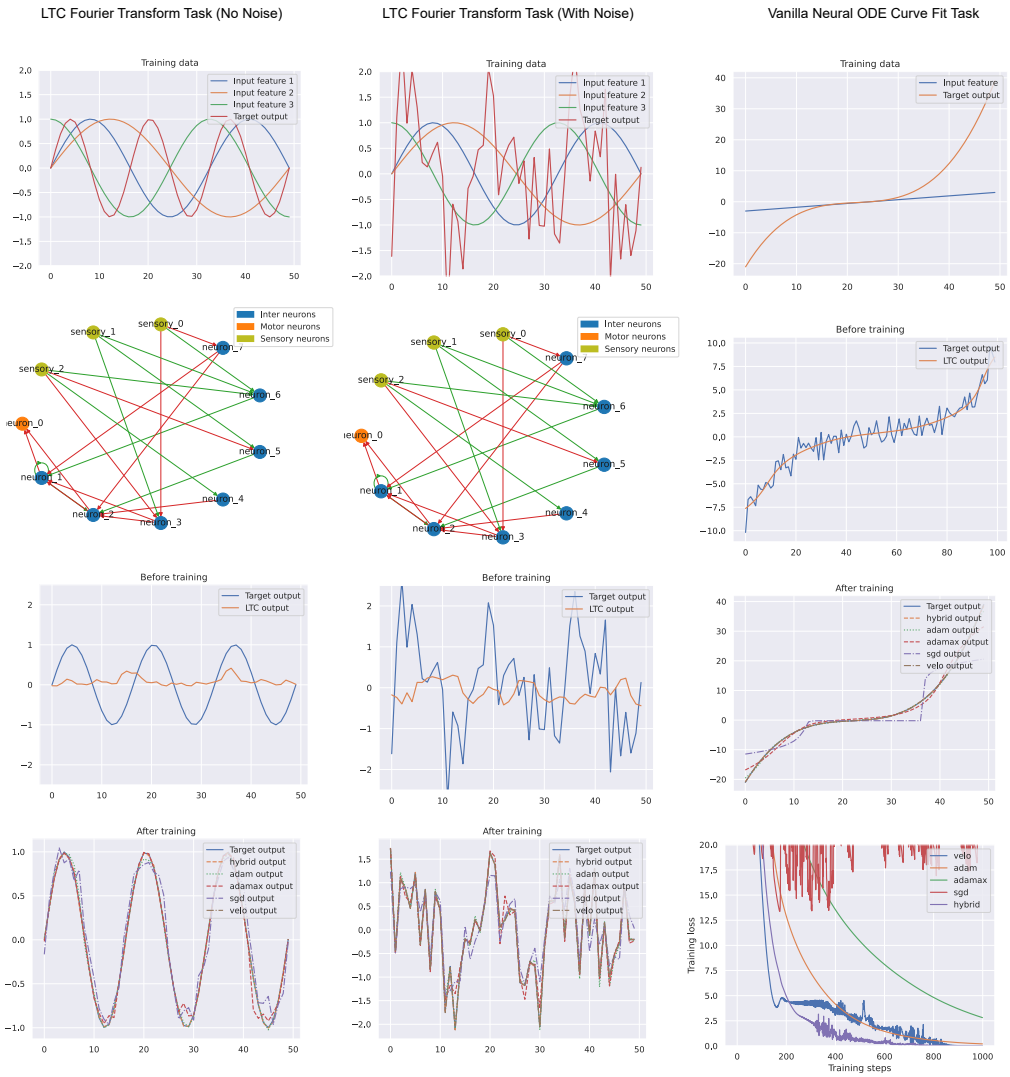

Figure 5: LTC and Neural ODE related trajectory predicting tasks.

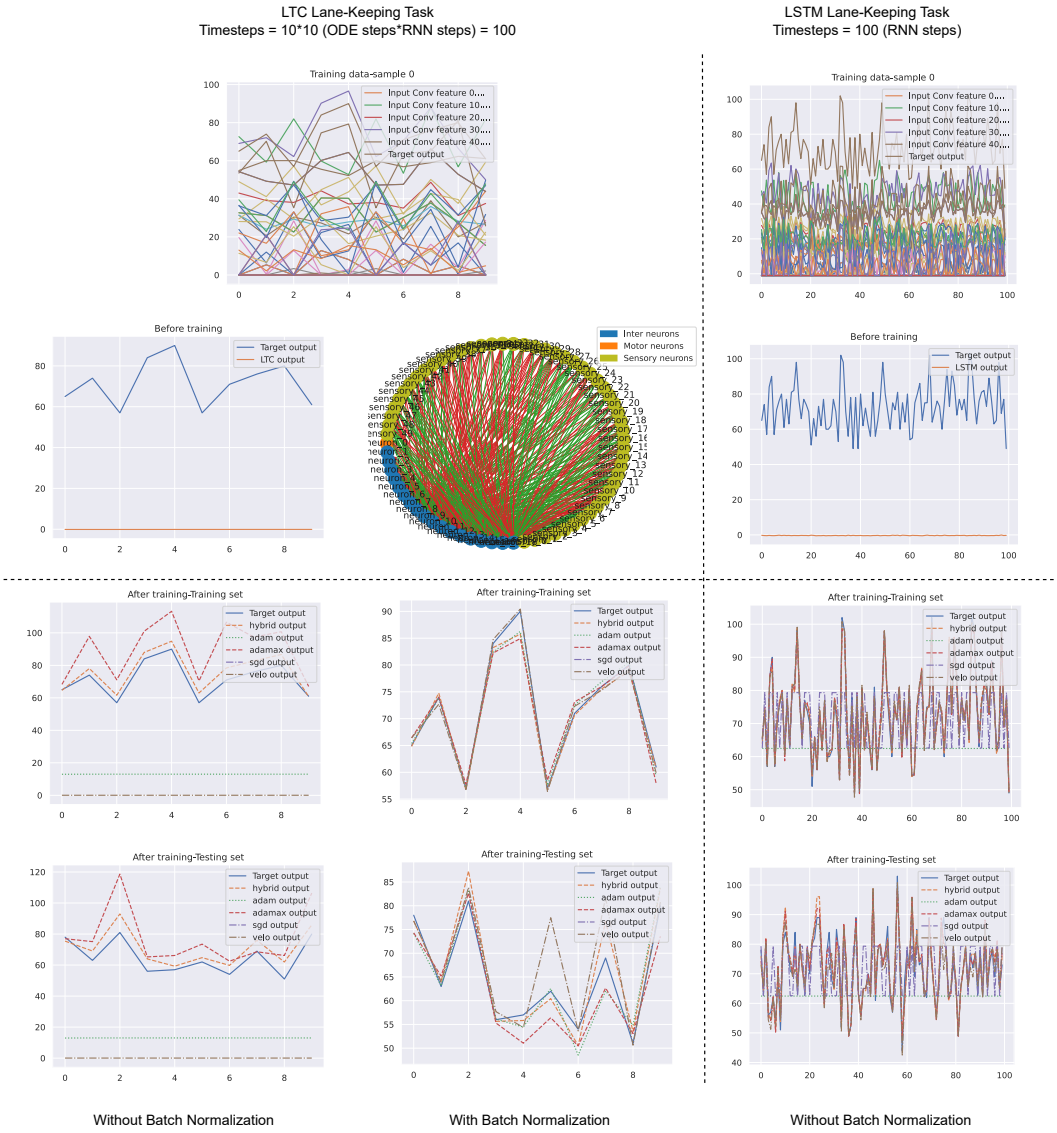

Figure 6: LTC and LSTM lane-keeping tasks.

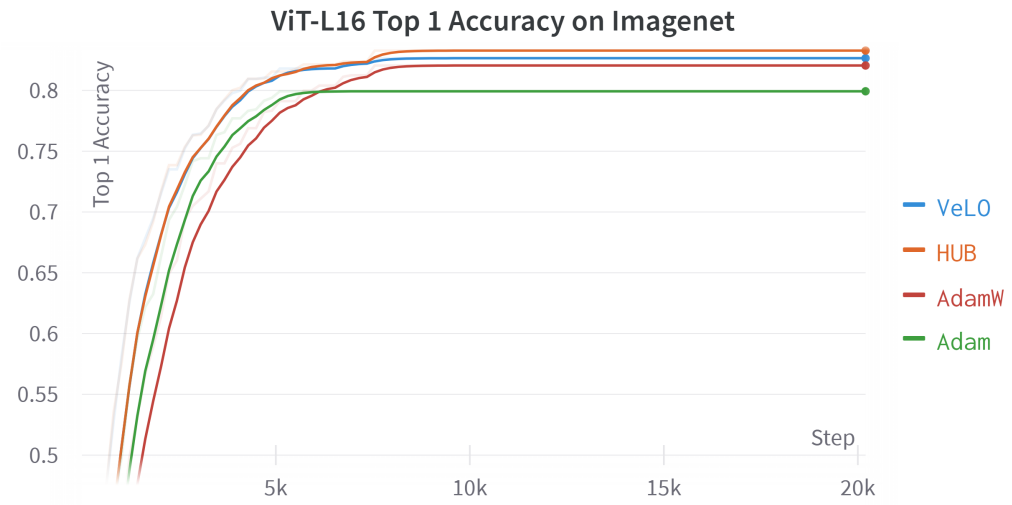

Figure 7: Vision transformer large top 1 Accuracy on Imagenet

B THEOREMS AND PROOFS FOR SECTION 3

B.1 REPRESENTATION FORMAT FOR ADAM AND ADAMAX

With $S^{(t)}, M^{(t)}$ and $n^{(t)}$ in the second row representing the hidden layers of Adam and Adamax at time t, we have the following RNN-form representations:

$$\begin{aligned} U_{Adam}(g^{(t)}, \beta_1, \beta_2, \alpha) &= \frac{\alpha}{\sqrt{\frac{M^{(t)}}{1-\beta_2^t}} + \epsilon} \odot \frac{S^{(t)}}{1-\beta_1^t} \\ \text{Adam: } S^{(t)} = \beta_1 S^{(t-1)} + (1-\beta_1) g^{(t)}; M^{(t)} &= \beta_2 M^{(t-1)} + (1-\beta_2) g^{(t)} \odot g^{(t)} \\ U_{Adamax}(g^{(t)}, \beta_1, \beta_2, \alpha) &= \frac{\alpha}{n^{(t)} + \epsilon} \odot \frac{S^{(t)}}{1-\beta_1^t} \\ \text{Adamax: } S^{(t)} = \beta_1 S^{(t-1)} + (1-\beta_1) g^{(t)}; n^{(t)} &= \max(\beta_2 * n^{(t-1)}, |g^{(t)}|) \end{aligned} \tag{1}$$

B.2 PROOF OF ADAM AND ADAMAX STABILITY

If $\lim_{t\to\infty} g^{(t)} = 0$, assume for any $\epsilon > 0$, there exist time t > T we have $\|g^{(t)}\| < \epsilon$, then:

$$\|S^{(t)}\| = \|\beta_1^{t-T} S^{(T)} + (1-\beta_1) \sum_{i=T+1}^{t} \beta_1^{t-i} g^{(i)}\| \leq \beta_1^{t-T} \|S^{(T)}\| + (1-\beta_1^{t-T})\epsilon \tag{2}$$

So $\lim_{t\to\infty} \left\|S^{(t)}\right\| = 0$. Similarly $\lim_{t\to\infty} \left\|M^{(t)}\right\| = 0$. Thus $\lim_{t\to\infty} U_{Adam}\left(g^{(t)}, \theta^{(t)}\right) = 0$

When the assumption in equation (7) holds true, Adam and Adamax exhibit good stability properties for strictly convex problems.

B.3 THE ANALYSIS OF MAIN DEPENDENCIES IN HUB

With the definition of HUB provided in Section 3.1, we can demonstrate its main dependence as follows:

$$\begin{aligned} \sigma(g^{(t)})_i &= \frac{exp(|g_i^{(t)}|)}{\sum_{j\in layer\ l} exp(|g_j^{(t)}|)}, \text{ if } |g_i^{(t)}| \gg |g_{k\neq i\in layer\ l}^{(t)}| : \\ exp(|g_i^{(t)}|) &\approx \sum_{j\in layer\ l} exp(|g_j^{(t)}|) \Rightarrow \sigma(g^{(t)})_i \approx 1. \end{aligned} \tag{3}$$

$$\sigma(g^{(t)})_{k\neq i\in layer\ l} \approx 0 \Rightarrow \text{Majority of parameters rely on } U_L(g^{(t)}, \theta_L^{(t)}.)$$

Equation (4) indicates that the HUB strategy exhibits a stronger reliance on the learned optimizer. This trend increases as the layer size grows:

$$\begin{aligned} &\text{If } |g_m^{(t)}| = |g_n^{(t)}| \text{ holds for arbitrary m and n in layer } l \text{ with K parameters,} \\ &\sigma(g^{(t)})_i = \frac{exp(|g_i^{(t)}|)}{\sum_{j\in layer\ l} exp(|g_j^{(t)}|)} = \frac{1}{K}. \end{aligned} \tag{4}$$

B.4 RATIONALE BEHIND HUB APPROACH

Assumption 1: Consider an optimization scenario where x^* is a local optimum of the continuous loss function $f(x)$. We assume that $f(x)$ is strictly convex and L-smooth. We have $\nabla f(x^*) = 0$, because the gradient at the local optimum point x^* vanishes according to this objective function.

Assumption 2: Section B.2 demonstrates the stability of hand-designed optimizers like Adam and Adamax in converging to local optima in the Assumption 1 scenario.

Considering the black-box nature of the learned optimizer, we discuss its behavior based on two cases:

- Case 1: When almost all the gradients are small and approaching convergence near x^*. This scenario resembles the situation depicted in Section B.3, where approximately $\frac{1}{K}$ of the weight for each parameter would rely on a hand-designed optimizer. Here, K represents the number of parameters in that layer (typically large, leading to the dominant role of the learned optimizer).
  - Sub-Case 1: If the learned optimizer is stable: Both optimizers progressively approach zero distance between the current location and x^* as the iteration number t increases.
  - Sub-Case 2: If the learned optimizer is unstable: In the following iterations, adverse optimization along the m dimensions emerges, causing these parameters to deviate from the optimal point x^*. This phenomenon subsequently gives rise to Case 2.
- Case 2: When gradients are small and near convergence for some parameters, while others (m in Sub-Case 2) are relatively large: By utilizing Softmax, the hand-designed optimizer's weight becomes dominant for the m parameters with relatively large gradients. This dominance guides the descent path toward x^*, effectively reverting to Case 1.

B.5 Proof for Gradient Vanishing and Exploding Properties for The Continuous Function in Section 3.3

Recall the continuous function is represented as:

$$f(x) = \begin{cases} 0 & x = 0 \\ \|x\|_1(1 + \lambda\|x\|_1 + \cos\frac{1}{\|x\|_1}) & etc. \end{cases} \tag{5}$$

In the experiment, we use $\lambda = 0.01, d = 1000$. Since we can consider this function as a variation of $\|x\|_1$, so within in 1D:

$$\begin{aligned} &\textbf{Case 1: } f'(\frac{2}{(4k+1)\pi}) = 1 + \frac{4\lambda}{(4k+1)\pi} + \frac{(4k+1)\pi}{2} \underset{k\to+\infty}{\to} +\infty \\ &\textbf{Case 2: } f'(\frac{1}{(2k+1)\pi}) = \frac{2\lambda}{(2k+1)\pi} \underset{k\to+\infty}{\to} 0,\ f(\frac{1}{(2k+1)\pi}) = \frac{\lambda}{(2k+1)^2\pi^2} \end{aligned} \tag{6}$$

In equation (9), we demonstrate that gradient explosion occurs in **Case 1**, while gradient vanishing arises in **Case 2**. Additionally, we illustrate the existence of numerous local minima with low function values surrounding the global minimum of zero.

C Exploration of Variations in HUB Strategy

In this section, we will delve deeper into discussing variations of the HUB strategy, which can prove useful when dealing with more complex real-world tasks.

C.1 Invert weighting HUB

As mentioned earlier, VeLO's effectiveness is limited in fine-tuning tasks. Consequently, it becomes unreasonable to allocate the majority of weighting to VeLO. In the case of Inverted Weighting HUB, the following formulation is employed:

$$U_{HUB}(g^{(t)}, \theta_H, \theta_L) = (1 - \sigma(g_l^{(t)})) \odot U_H(g^{(t)}, \theta_H) + \sigma(g_l^{(t)}) \odot U_L(g^{(t)}, \theta_L) \tag{7}$$

From format (5) we can see that this variation simply involves inverting the weighting matrix. This simple modification can result in a significant alteration to the behaviour of HUB, allowing for a majority of weight allocation towards the hand-designed optimizer. This approach is also preferable as it adheres to the concept of fine-tuning, where only a small number of parameters require significant adjustment to bridge the domain gap between pre-trained knowledge and downstream knowledge.

C.2 Block-wise HUB

Another useful variation of HUB is to employ different weighting strategies in different blocks of the neural network. For instance, in many pre-trained models, a new MLP head is added to adapt to

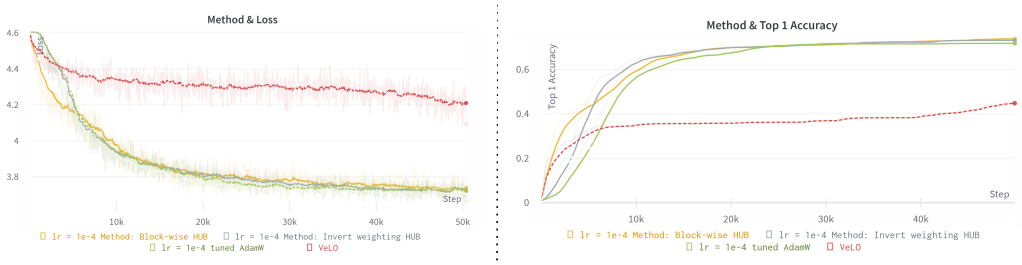

Figure 8: Using Block-wise HUB compared to Invert Weighting HUB, we can observe from the figure that both variations of HUB exhibit faster and superior performance than relying solely on one of their hybrid sources (tuned AdamW and VeLO). However, Block-wise HUB yields a slightly higher top-1 accuracy and demonstrates an evident two-stage convergence curve.

the downstream dataset, and this MLP head is trained from scratch. In such cases, it is reasonable to rely more on VeLO for fine-tuning this specific block. Therefore, we can perform a multi-stage optimization using HUB with different weighting strategies (Figure8). Specifically, for the blocks with pre-trained weights, we employ:

$$U_{HUB}(g^{(t)}, \theta_H, \theta_L) = \sigma(g_l^{(t)}) \odot U_H(g^{(t)}, \theta_H) + (1 - \sigma(g_l^{(t)})) \odot U_L(g^{(t)}, \theta_L) \tag{8}$$

Otherwise, we employ:

$$U_{HUB}(g^{(t)}, \theta_H, \theta_L) = (1 - \sigma(g_l^{(t)})) \odot U_H(g^{(t)}, \theta_H) + \sigma(g_l^{(t)}) \odot U_L(g^{(t)}, \theta_L) \tag{9}$$

C.3 Clipping HUB

Clipping HUB can be represented with the following format:

$$U_{HUB}(g^{(t)}, \theta_H, \theta_L) = \sigma(g_l^{(t)}) \odot U_H(g^{(t)}, \theta_H) + (1 - \sigma(g_l^{(t)})) \odot U_L(g^{(t)}, \theta_L) \tag{10}$$

With threshold maximum = A and minimum = B

$$\text{The i-th parameter in layer l } \sigma(g_l^{(t)})_i = \begin{cases} A & \frac{exp(|g_i^{(t)}|)}{\sum_{j \in layer\ l} exp(|g_j^{(t)}|)} > A \\ B & \frac{exp(|g_i^{(t)}|)}{\sum_{j \in layer\ l} exp(|g_j^{(t)}|)} < B \\ \frac{exp(|g_i^{(t)}|)}{\sum_{j \in layer\ l} exp(|g_j^{(t)}|)} & \text{Otherwise} \end{cases} \tag{11}$$

This method proves to be particularly useful when dealing with an extremely wide network structure. As mentioned in equation (4) in section 3.2 of our paper, when the value of K approaches infinity, the HUB strategy would solely rely on VeLO, which deviates from our original intention. To address this issue, we can employ clipping HUB to prevent such a situation from occurring.

References

Marcin Andrychowicz, Misha Denil, Sergio Gomez Colmenarejo, Matthew W. Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. Learning to learn by gradient descent by gradient descent. In *NIPS*, 2016.

Sungyong Baik, Myungsub Choi, Janghoon Choi, Heewon Kim, and Kyoung Mu Lee. Meta-learning with adaptive hyperparameters. *Advances in neural information processing systems*, 33:20755–20765, 2020.

Sungyong Baik, Myungsub Choi, Janghoon Choi, Heewon Kim, and Kyoung Mu Lee. Learning to learn task-adaptive hyperparameters for few-shot learning. *IEEE transactions on pattern analysis and machine intelligence*, PP, 2023.

Michael G. Bechtel, Elise McEllhiney, Minje Kim, and Heechul Yun. Deeppicar: A low-cost deep neural network-based autonomous car, 2018.

Tian Qi Chen, Yulia Rubanova, Jesse Bettencourt, and David Kristjanson Duvenaud. Neural ordinary differential equations. In *Neural Information Processing Systems*, 2018.

Yutian Chen, Matthew W. Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Timothy P. Lillicrap, Matthew M. Botvinick, and Nando de Freitas. Learning to learn without gradient descent by gradient descent. In *International Conference on Machine Learning*, 2016.

Yutian Chen, Matthew W. Hoffman, Sergio Gómez Colmenarejo, Misha Denil, Timothy P. Lillicrap, Matt Botvinick, and Nando de Freitas. Learning to learn without gradient descent by gradient descent. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 748–756. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/chen17e.html.

François Chollet. Xception: Deep learning with depthwise separable convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2016.

Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation strategies from data. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 113–123, 2019.

Yann Dauphin, Harm de Vries, Junyoung Chung, and Yoshua Bengio. Rmsprop and equilibrated adaptive learning rates for non-convex optimization. *arXiv: Learning*, 2015.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009a. doi: 10.1109/CVPR.2009.5206848.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. pp. 248–255, 2009b.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ArXiv*, abs/2010.11929, 2020.

Timothy Dozat. Incorporating nesterov momentum into adam. 2016.

Simon Shaolei Du. *Gradient descent for non-convex problems in modern machine learning*. PhD thesis, Department of Energy award DEAR0000596, Department of the Interior award . . . , 2019.

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

Chelsea Finn, P. Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *ArXiv*, abs/1703.03400, 2017a.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135. PMLR, 06–11 Aug 2017b. URL https://proceedings.mlr.press/v70/finn17a.html.

Rinon Gal, Yuval Alaluf, Yuval Atzmon, Or Patashnik, Amit H. Bermano, Gal Chechik, and Daniel Cohen-Or. An image is worth one word: Personalizing text-to-image generation using textual inversion. *ArXiv*, abs/2208.01618, 2022.

Robert Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtárik. Sgd: General analysis and improved rates. In *International conference on machine learning*, pp. 5200–5209. PMLR, 2019.

James Harrison, Luke Metz, and Jascha Narain Sohl-Dickstein. A closer look at learned optimization: Stability, robustness, and inductive biases. *ArXiv*, abs/2209.11208, 2022.

Ramin M. Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. Liquid time-constant networks. In *AAAI Conference on Artificial Intelligence*, 2020.

Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

Howard Heaton, Xiaohan Chen, Zhangyang Wang, and Wotao Yin. Safeguarded learned convex optimization. In *AAAI Conference on Artificial Intelligence*, 2020. URL `https://api.semanticscholar.org/CorpusID:209485857`.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

Takashi Ishida, Ikko Yamane, Tomoya Sakai, Gang Niu, and Masashi Sugiyama. Do we need zero training loss after achieving zero training error? *ArXiv*, abs/2002.08709, 2020.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Mathias Lechner, Ramin M. Hasani, Alexander Amini, Thomas A. Henzinger, Daniela Rus, and Radu Grosu. Neural circuit policies enabling auditable autonomy. *Nature Machine Intelligence*, 2:642–652, 2020.

Seung Hoon Lee, Seunghyun Lee, and Byung Cheol Song. Vision transformer for small-size datasets. *ArXiv*, abs/2112.13492, 2021.

Ke Li and Jitendra Malik. Learning to optimize neural nets. *ArXiv*, abs/1703.00441, 2017.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, abs/2101.00190, 2021.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Annual Meeting of the Association for Computational Linguistics*, 2022.

Sachit Menon and Carl Vondrick. Visual classification via description from large language models. *ArXiv*, abs/2210.07183, 2022.

Luke Metz, Niru Maheswaranathan, C. Daniel Freeman, Ben Poole, and Jascha Narain Sohl-Dickstein. Tasks, stability, architecture, and compute: Training more effective learned optimizers, and using them to train themselves. *ArXiv*, abs/2009.11243, 2020.

Luke Metz, C Daniel Freeman, Samuel S Schoenholz, and Tal Kachman. Gradients are not all you need. *arXiv preprint arXiv:2111.05803*, 2021.

Luke Metz, James Harrison, C. Daniel Freeman, Amil Merchant, Lucas Beyer, James Bradbury, Naman Agrawal, Ben Poole, Igor Mordatch, Adam Roberts, and Jascha Narain Sohl-Dickstein. Velo: Training versatile learned optimizers by scaling up. *ArXiv*, abs/2211.09760, 2022.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *ArXiv*, abs/2003.08934, 2020.

Yurii Nesterov and Vladimir G. Spokoiny. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics*, 17:527–566, 2017.

Isabeau Pr'emont-Schwarz, Jaroslav V'itkru, and Jan Feyereisl. A simple guard for learned optimizers. *ArXiv*, abs/2201.12426, 2022. URL https://api.semanticscholar.org/CorpusID:246430352.

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond, 2019.

Tim Salimans, Jonathan Ho, Xi Chen, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *ArXiv*, abs/1703.03864, 2017.

Jiayi Shen, Xiaohan Chen, Howard Heaton, Tianlong Chen, Jialin Liu, Wotao Yin, and Zhangyang Wang. Learning a minimax optimizer: A pilot study. In *International Conference on Learning Representations*, 2021.

Vincent Sitzmann, Julien N. P. Martel, Alexander W. Bergman, David B. Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *ArXiv*, abs/2006.09661, 2020.

Yannick Strümpler, Janis Postels, Ren Yang, Luc Van Gool, and Federico Tombari. Implicit neural representations for image compression. In *European Conference on Computer Vision*, 2021.

Walter Vent. Rechenberg, ingo, evolutionsstrategie — optimierung technischer systeme nach prinzipien der biologischen evolution. 170 s. mit 36 abb. frommann-holzboog-verlag. stuttgart 1973. broschiert. *Feddes Repertorium*, 86:337–337, 1975.

CL Walsh, P Tafforeau, WL Wagner, DJ Jafree, A Bellier, C Werlein, MP Kühnel, E Boller, S Walker-Samuel, JL Robertus, et al. Imaging intact human organs with local resolution of cellular structures using hierarchical phase-contrast tomography. *Nature methods*, 18(12):1532–1541, 2021.

Yuxin Wen, Neel Jain, John Kirchenbauer, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Hard prompts made easy: Gradient-based discrete optimization for prompt tuning and discovery. *ArXiv*, abs/2302.03668, 2023.

Runzhao Yang, Tingxiong Xiao, Yu-Shen Cheng, Qi Cao, Jinyuan Qu, Jinli Suo, and Qionghai Dai. Sci: A spectrum concentrated implicit neural compression for biomedical data. *ArXiv*, abs/2209.15180, 2022.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Young Joon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6022–6031, 2019.

Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *ArXiv*, abs/1611.01578, 2016.