

384 **Checklist**

385 The checklist follows the references. Please read the checklist guidelines carefully for information on  
386 how to answer these questions. For each question, change the default [TODO] to [Yes] , [No] , or  
387 [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing  
388 the appropriate section of your paper or providing a brief inline description. For example:

- 389 • Did you include the license to the code and datasets? [Yes] See Section ??.
- 390 • Did you include the license to the code and datasets? [No] The code and the data are  
391 proprietary.
- 392 • Did you include the license to the code and datasets? [N/A]

393 Please do not modify the questions and only use the provided macros for your answers. Note that the  
394 Checklist section does not count towards the page limit. In your paper, please delete this instructions  
395 block and only keep the Checklist section heading above along with the questions/answers below.

396 1. For all authors...

397 (a) Do the main claims made in the abstract and introduction accurately reflect the paper's  
398 contributions and scope? [Yes] Where to find mentioned contributions of the abstract  
399 in the paper:

- 400 • **The paper establishes a theoretical connection between curvature and locally  
401 chaotic dynamics:**

402 We show this in Theorem 2.1 in section 2.4

- 403 • **Locally chaotic dynamics are mainly determined by the negative eigenvalue  
404 spectrum of the Hessian:**

405 Applying Theorem 2.1 to the results of logging the maximal Hessian eigenvalue  $\lambda_1$   
406 (see Appendix B.2) verifies that local chaos is rarely induced by the top eigenspectrum  
407 of the Hessian as  $\lambda_1 < 2/\gamma$  for most of the training.

- 408 • **The directions of local chaos cannot be removed from SGD without hurting  
409 the training performance:**

410 See subsection 4.1 and Figure 1

- 411 • **We challenge the widespread understanding of convergence to a confined  
412 region in parameter space and show models diverge at a slow pace even after  
413 reaching top training performance:**

414 See subsection 4.2 and Figure 3

- 415 • **Their divergence can be modelled through a composition of a random walk  
416 and a linear divergence:**

417 See equation 15 and Figure 3

- 418 • **The tools and insights developed as part of our work contribute to improving  
419 the understanding of neural network training dynamics and provide a basis  
420 for future improvements of optimization:**

421 See subsection 4.2 § Theoretical distance evolution as well as section 5

422 (b) Did you describe the limitations of your work? [Yes] See discussion section 5

423 (c) Did you discuss any potential negative societal impacts of your work? [No] No, the  
424 scope of this paper is to improve understanding of SGD and we do not expect any  
425 immediate negative societal impacts from this.

426 (d) Have you read the ethics review guidelines and ensured that your paper conforms to  
427 them? [Yes]

428 2. If you are including theoretical results...

429 (a) Did you state the full set of assumptions of all theoretical results? [Yes]

430 (b) Did you include complete proofs of all theoretical results? [Yes] All proofs can be  
431 found in the proof section in Appendix A except the equations from the preliminaries  
432 section 2 which are taken from Skokos, 2009.

433 3. If you ran experiments...

434 (a) Did you include the code, data, and instructions needed to reproduce the main experi-  
435 mental results (either in the supplemental material or as a URL)? [No] Not yet, these  
436 will be included in the final submission including supplementary material by 26th May.

- 437 (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were  
438 chosen)? **[Yes]** Aside from logging specifics, all training details required to reproduce  
439 our experiments given the code base are specified. More detailed explanations for  
440 reproducing the experiments will be provided with the code in the supplementary  
441 material.
- 442 (c) Did you report error bars (e.g., with respect to the random seed after running exper-  
443 iments multiple times)? **[No]** No, our work does not aim to beat the performance of  
444 existing models on certain tasks but merely to improve the understanding of how SGD  
445 works. Errors bars would not be purposeful here.
- 446 (d) Did you include the total amount of compute and the type of resources used (e.g., type  
447 of GPUs, internal cluster, or cloud provider)? **[Yes]** Yes, since the computations we run  
448 involving O2Grad require GPUs with state-of-the-art VRAM size and can thus only be  
449 reproduced by people with access to comparably powerful systems.
- 450 4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...  
451 (a) If your work uses existing assets, did you cite the creators? **[Yes]** We cite the creators  
452 of the main assets used, i.e. Pytorch, USPS and FashionMNIST.  
453 (b) Did you mention the license of the assets? **[No]** No, the datasets (USPS and Fash-  
454 ionMNIST) are released under no license and under an open source license (MIT),  
455 respectively, and commonly used in the ML community. As such, mentioning the  
456 license in the paper is superfluous. We declare the Python packages used for our  
457 experiments as part of the code in the supplementary material.  
458 (c) Did you include any new assets either in the supplemental material or as a URL? **[Yes]**  
459 We include a self-developed asset (O2Grad) in the supplementary material.  
460 (d) Did you discuss whether and how consent was obtained from people whose data you're  
461 using/curating? **[No]** No, the datasets used are publicly available under open source  
462 licenses, explicit consent by the creators is not necessary.  
463 (e) Did you discuss whether the data you are using/curating contains personally identifiable  
464 information or offensive content? **[No]** No, neither should apply to the USPS or to the  
465 FashionMNIST dataset.
- 466 5. If you used crowdsourcing or conducted research with human subjects...  
467 (a) Did you include the full text of instructions given to participants and screenshots, if  
468 applicable? **[N/A]**  
469 (b) Did you describe any potential participant risks, with links to Institutional Review  
470 Board (IRB) approvals, if applicable? **[N/A]**  
471 (c) Did you include the estimated hourly wage paid to participants and the total amount  
472 spent on participant compensation? **[N/A]**

473 **A Proofs**

474 **Tangent map of SGD**

475 **Theorem A.1.** Consider a neural network with  $N$  parameters  $\theta^{(t)} \in \mathbb{R}^N$  at time step  $t$  and training  
476 with SGD according to

$$\theta^{(t+1)} = \mathbf{f}(\theta^{(t)}) = \theta^{(t)} - \gamma \mathbf{g}(\theta^{(t)}, \mathbf{z}^{(t)}), \quad (16)$$

477 where  $\gamma$  is the learning rate and  $\mathbf{z}^{(t)}$  is a data batch sampled at time step  $t$ . Then, the tangent map of  
478 the corresponding dynamical system can be iteratively calculated as

$$\mathbf{Y}^{(t+1)} = (\mathbf{I} - \gamma \mathbf{H}^{(t)}) \mathbf{Y}^{(t)}. \quad (17)$$

479

480 *Proof.* The tangent map of a dynamical system at time step which only depends on the previous time  
481 step can be obtained as

$$\mathbf{Y}^{(t+1)} = \mathbf{J}_f(\theta^{(t)}) \mathbf{Y}^{(t)} = \frac{\partial \theta^{(t+1)}}{\partial \theta^{(t)}} \mathbf{Y}^{(t)}, \quad (18)$$

482 where  $\mathbf{J}_f(\theta^{(t)})$  is the Jacobian of the evolution function  $\mathbf{f}$  of the dynamical system. In this case, this  
483 is simply

$$\frac{\partial \theta_i^{(t+1)}}{\partial \theta_j^{(t)}} = \frac{\partial \theta_i^{(t)}}{\partial \theta_j^{(t)}} - \gamma \frac{\partial}{\partial \theta_j^{(t)}} g_i(\theta^{(t)}) = \delta_{ij} - \gamma H_{ij}(\theta^{(t)}), \quad (19)$$

484 where  $\delta_{ij}$  is the Kronecker delta and  $\mathbf{H}$  denotes the Hessian of the loss w.r.t. network parameters and  
485 we are done.  $\square$

486 **Tangent map of SGD with momentum**

487 **Theorem A.2.** Consider a neural network evolving according to SGD with learning rate  $\gamma$  and  
488 momentum  $\beta$  as

$$\theta^{(t+1)} = \theta^{(t)} + \mathbf{v}^{(t)}, \quad \mathbf{v}^{(t)} = -\gamma \mathbf{g}(\theta^{(t)}; \mathbf{z}^{(t)}) + \beta \mathbf{v}^{(t-1)}, \quad (20)$$

489 The parameters at time step  $t+1$  depend on the gradients at all previous time steps and the dependency  
490 can be written as

$$\theta^{(t+1)} = \theta^{(t)} + \mathbf{v}^{(t)}, \quad \mathbf{v}^{(t)} = -\gamma \sum_{s=0}^t \beta^{(t-s)} \mathbf{g}(\theta^{(s)}, \mathbf{z}^{(s)}). \quad (21)$$

491

492 *Proof.* For the sake of convenience, we omit writing down the dependency on the batch  $\mathbf{z}(t)$ . Induc-  
493 tion start for  $t = 0$ , where we assume  $v^{(0)} = 0$ :

$$\theta^{(1)} = \theta^{(0)} - \gamma \mathbf{g}(\theta^{(0)}) = \theta^{(0)} - \gamma \beta^0 \mathbf{g}(\theta^{(0)}) \quad (22)$$

494 Induction step assuming equation holds true for time step  $t$ :

$$\theta^{(t+1)} = \theta^{(t)} + \mathbf{v}^{(t)} \quad (23)$$

$$= \theta^{(t)} - \gamma \mathbf{g}(\theta^{(t)}) + \beta \mathbf{v}(\theta^{(t-1)}) \quad (24)$$

$$= \theta^{(t)} - \gamma \mathbf{g}(\theta^{(t)}) + \beta \cdot (-\gamma) \sum_{s=0}^{t-1} \beta^{(t-1-s)} \mathbf{g}(\theta^{(s)}) \quad (25)$$

$$= \theta^{(t)} - \gamma \mathbf{g}(\theta^{(t)}) - \gamma \sum_{s=0}^{t-1} \beta^{(t-s)} \mathbf{g}(\theta^{(s)}) \quad (26)$$

$$= \theta^{(t)} - \gamma \sum_{s=0}^t \beta^{(t-s)} \mathbf{g}(\theta^{(s)}) \quad (27)$$

495

$\square$

496 **Theorem A.3.** Consider a neural network evolving according to SGD with learning rate  $\gamma$  and  
497 momentum  $\beta$  as

$$\theta^{(t+1)} = \theta^{(t)} + v^{(t)}, \quad v^{(t)} = -\gamma g(\theta^{(t)}; z^{(t)}) + \beta v^{(t-1)}, \quad (28)$$

498 where  $g$  is the gradient of the loss w.r.t. parameters  $\theta^{(t)}$  at time point  $t$ . Then, the tangent map  $\mathbf{Y}^{(t+1)}$   
499 of the dynamical system is given by

$$\mathbf{Y}^{(t+1)} = \mathbf{Y}^{(t)} - \gamma \left( \sum_{s=1}^t \beta^{t-s} \mathbf{H}^{(s)} \mathbf{Y}^{(s)} \right). \quad (29)$$

500

501 *Proof.* At time step  $t$ , the update rule for the gradient depends on all previous time steps according to  
502 Theorem A.2. Thus, the tangent map can be obtained through application of the chain rule:

$$\mathbf{Y}^{(t+1)} = \frac{d\theta^{(t+1)}}{d\theta^{(0)}} = \sum_{s=0}^t \frac{\partial \theta^{(t+1)}}{\partial \theta^{(s)}} \frac{d\theta^{(s)}}{d\theta^{(0)}} = \sum_{s=0}^t \frac{\partial \theta^{(t+1)}}{\partial \theta^{(s)}} \mathbf{Y}^{(s)}. \quad (30)$$

503 Using Theorem A.2 again, the partial derivatives can be resolved as

$$\frac{\partial \theta^{(t+1)}}{\partial \theta^{(s)}} = \begin{cases} \mathbf{I} - \gamma \beta^0 \partial_{\theta^{(t)}} g(\theta^{(t)}) = \mathbf{I} - \gamma \mathbf{H}^{(t)}, & s = t \\ -\gamma \beta^{t-s} \partial_{\theta^{(s)}} g(\theta^{(s)}) = -\gamma \beta^{t-s} \mathbf{H}^{(s)}, & s < t \end{cases} \quad (31)$$

504 And reinserting into equation (30), we obtain the desired result

$$\mathbf{Y}^{(t+1)} = (\mathbf{I} - \gamma \mathbf{H}^{(t)}) \mathbf{Y}^{(t)} - \sum_{s=0}^{t-1} \gamma \beta^{t-s} \mathbf{H}^{(s)} \mathbf{Y}^{(s)} = \mathbf{Y}^{(t)} - \gamma \sum_{s=0}^t \beta^{t-s} \mathbf{H}^{(s)} \mathbf{Y}^{(s)} \quad (32)$$

505  $\square$

### 506 Hessian eigenvalues and Local Chaos

507 **Theorem A.4.** Let  $\mathbf{A} \in \mathbb{R}^{N \times N}$  be a positive defined matrix, and the natural logarithm of a matrix be  
508 defined as the infinite series

$$\ln \mathbf{A} = \sum_{k=1}^{\infty} \frac{(-1)^k}{k} (\mathbf{A} - \mathbf{I})^k \quad (33)$$

509 Then, if  $(\lambda_i, \mathbf{v}_i)_{i=1}^N$  is the set of all eigenpairs of  $\mathbf{A}$ ,  $(\ln \lambda_i, \mathbf{v}_i)_{i=1}^N$  is the set of all eigenpairs of  $\ln \mathbf{A}$ .

510 *Proof.* Since  $\mathbf{A}$  is positive defined, its eigenvalues  $\lambda_i$  are all positive, so  $\ln \lambda_i$  is defined for all  
511 eigenvalues of  $\mathbf{A}$ . Consider an arbitrary eigenvalue of  $\mathbf{A}$  satisfying  $\mathbf{A}\mathbf{v}_i = \lambda_i \mathbf{v}_i$ . Clearly,  $(\mathbf{A} - \mathbf{I})\mathbf{v}_i =$   
512  $\mathbf{A}\mathbf{v}_i - \mathbf{I}\mathbf{v}_i = \lambda_i \mathbf{v}_i - \mathbf{v}_i = (\lambda_i - 1)\mathbf{v}_i$ . It follows immediately that  $(\mathbf{A} - \mathbf{I})^k \mathbf{v}_i = (\lambda_i - 1)^k \mathbf{v}_i$  and  
513 thus:

$$\ln \mathbf{A}\mathbf{v}_i = \sum_{k=1}^{\infty} \frac{(-1)^k}{k} (\mathbf{A} - \mathbf{I})^k \mathbf{v}_i = \sum_{k=1}^{\infty} \frac{(-1)^k}{k} (\lambda_i - 1)^k \mathbf{v}_i = \ln \lambda_i \mathbf{v}_i \quad (34)$$

514 Since  $\ln \mathbf{A} \in \mathbb{R}^{N \times N}$ , the matrix can have at most  $N$  eigenvalues, so we know that these are all  
515 eigenvalues of  $\ln \mathbf{A}$  and we are done.  $\square$

516 **Theorem A.5.** Given a neural network with  $N$  parameters, let  $\mathbf{H}$  be the Hessian of the loss w.r.t.  
517 network parameters, and let  $\{(\lambda_i, \mathbf{v}_i)\}_{i=1}^N$  be the Hessian's eigenpairs. Furthermore, let us assume  
518 that the network is trained using SGD with learning rate  $\gamma$ . Consider the eigenvalues for which  
519  $\lambda_i \neq \frac{1}{\gamma}$ . Then, the network's LLEs indicate locally chaotic training behaviour in the direction of  $\mathbf{v}_i$  if

$$\lambda_i \leq 0 \vee \lambda_i \geq \frac{2}{\gamma} \quad (35)$$

520

521 *Proof.* If  $(\lambda_i, \mathbf{v}_i)$  is an eigenpair of  $\mathbf{H}$ ,  $(1 - \gamma\lambda_i, \mathbf{v}_i)$  is an eigenpair of  $\mathbf{J}_f = (\mathbf{I} - \gamma\mathbf{H})$ , since  
 522  $(\mathbf{I} - \gamma\mathbf{H})\mathbf{v} = (1 - \gamma\lambda_i)\mathbf{v}$ . Due to the identity matrix and the Hessian being symmetric,  $(\mathbf{I} -$   
 523  $\gamma\mathbf{H})^T(\mathbf{I} - \gamma\mathbf{H}) = (\mathbf{I} - \gamma\mathbf{H})^2$ . Furthermore, if  $\mathbf{A}$  is a positive definite matrix and  $(\lambda_i, \mathbf{v}_i)$  is an  
 524 eigenpair of  $\mathbf{A}$ , then  $(\ln \lambda_i, \mathbf{v}_i)$  is an eigenpair of  $\ln \mathbf{A}$  (see Appendix A.4). If all eigenvalues of  
 525  $(\mathbf{I} - \gamma\mathbf{H})^2$  are positive and nonzero due to  $\lambda_i \neq \frac{1}{\gamma}$ , the matrix is positive definite and it follows that  
 526  $\frac{1}{2} \ln(1 - \gamma\lambda_i)^2$  is an eigenvalue of

$$\Lambda_x^{(1)} = \frac{1}{2} \ln(\mathbf{I} - \gamma\mathbf{H})^T(\mathbf{I} - \gamma\mathbf{H})$$

527 for the same eigenvector  $\mathbf{v}_i$  and thus an LLE. To indicate locally chaotic training behaviour, the LLE  
 528 must satisfy

$$\ln(1 - \gamma\lambda_i)^2 \geq 0 \Leftrightarrow (1 - \gamma\lambda_i)^2 \geq 1 \quad (36)$$

529 Solving for  $\lambda_i$  gives the desired result and we are done.  $\square$

530 Now, consider the edge case where some eigenpair  $(\lambda_i, \mathbf{v}_i)$  of the Hessian  $\mathbf{H}$  satisfies  $\lambda_i = \frac{1}{\gamma}$  s.t. for  
 531 an infinitesimally small displacement  $\delta\mathbf{v} \parallel \mathbf{v}_i$

$$\mathbf{J}_f \delta\mathbf{v} = (\mathbf{I} - \gamma\mathbf{H})\delta\mathbf{v} = (1 - \gamma \cdot 1/\gamma)\delta\mathbf{v} = 0. \quad (37)$$

532 That is, the dynamical system exhibits a maximal contraction along the axis of  $\mathbf{v}_i$ , meaning that  
 533 the system does not evolve along this axis. Therefore, these eigenpairs can be excluded from our  
 534 considerations on chaotic behaviour. The respective eigenvalues can still be accounted for in the  
 535 definition of the Local Lyapunov spectrum by either accepting eigenvalues of  $-\infty$ , or by adapting  
 536 the definition of the LLEs to be the singular values of

$$\tilde{\mathbf{U}}\tilde{\Sigma}\tilde{\mathbf{V}}^* = \mathbf{U}\Sigma\mathbf{V}^* = (\mathbf{I} - \gamma\mathbf{H})^2, \quad (38)$$

537 where the first decomposition is the zero-truncated SVD. In this case, the singular pairs  $(\sigma_i, \mathbf{u}_i)$ ,  
 538 where  $\mathbf{u}_i$  is the singular vector corresponding to singular value  $\sigma_i$ , define Local Lyapunov Exponent  
 539 (LLE) and corresponding Lyapunov vector, respectively. Note that since  $(\mathbf{I} - \gamma\mathbf{H})^2$  is always  
 540 positive-semidefinite, the singular value decomposition is identical to the eigenvalue decomposition  
 541  $(\mathbf{I} - \gamma\mathbf{H}) = \mathbf{U}\mathbf{D}\mathbf{U}^{-1}$  provided it exists. Therefore the Lyapunov spectrum can be constrained to  
 542 exclude Hessian eigenpairs where  $\lambda = \frac{1}{\gamma}$  by considering the zero-truncated SVD. This definition is  
 543 also interesting because it allows us to loosely interpret the top Lyapunov vectors as the directions  
 544 of maximum trajectory variance if we performed a PCA on trajectory distances in the limit for an  
 545 infinite number of perturbed trajectories sampled, with the perturbation magnitude and evolution time  
 546 of the trajectories going to 0.

547 **Theorem A.6.** *Given a neural network with  $N$  parameters, let  $\mathbf{H}^{(t)}$  be the Hessian of the loss  
 548 w.r.t. network parameters at time step  $t$ . Furthermore, let  $\lambda_1^{(s)}$  be the maximum LLE of the training  
 549 dynamics at time step  $s$  for  $0 < s \leq t$ . Then, the mLCE of the system evolution can be calculated as*

$$\lambda_1 = \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{s=1}^t \lambda_1^{(s)} \quad (39)$$

550

551 *Proof.* As previously established, the mLCE  $\lambda_1$  is defined as the maximum eigenvalue of  $\Lambda^{(\infty)}$ ,  
 552 which is precisely the limit of the time-normalized spectral norm of the tangent map:

$$\lambda_1 = \lambda_{\max} \lim_{t \rightarrow \infty} \frac{1}{2t} \ln \mathbf{Y}^{(t)T} \mathbf{Y}^{(t)} = \lim_{t \rightarrow \infty} \frac{1}{t} \lambda_{\max} \sqrt{\mathbf{Y}^{(t)T} \mathbf{Y}^{(t)}} = \lim_{t \rightarrow \infty} \frac{1}{t} \|\mathbf{Y}^{(t)}\| \quad (40)$$

553 For SGD without momentum, this can be further rewritten using the chain rule and the rules for the  
 554 logarithm:

$$\lambda_1 = \lim_{t \rightarrow \infty} \frac{1}{t} \ln \left\| \prod_{s=1}^t \frac{\partial \boldsymbol{\theta}^{(s)}}{\partial \boldsymbol{\theta}^{(s-1)}} \right\| = \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{s=1}^t \ln \left\| \frac{\partial \boldsymbol{\theta}^{(s)}}{\partial \boldsymbol{\theta}^{(s-1)}} \right\| \quad (41)$$

555 Using the definition of the spectral norm again, we realize the eigenvalues in the sum are the maximum  
556 LLEs and we end up with

$$\lambda_1 = \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{s=1}^t \ln \lambda_{\max} \sqrt{\mathbf{J}_f(\boldsymbol{\theta}^{(s+1)})^T \mathbf{J}_f(\boldsymbol{\theta}^{(s+1)})} \quad (42)$$

$$= \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{s=1}^t \lambda_{\max} \ln \sqrt{\mathbf{J}_f(\boldsymbol{\theta}^{(s+1)})^T \mathbf{J}_f(\boldsymbol{\theta}^{(s+1)})} = \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{s=0}^t \lambda_1^{(s)}. \quad (43)$$

557  $\square$

558 **Theorem A.7.** *Given a neural network with  $N$  parameters, let  $\mathbf{H}^{(t)}$  be the Hessian of the loss w.r.t.  
559 network parameters at time step  $t$ . If for all time steps  $t$  the Hessian  $\mathbf{H}^{(t)}$  has at least one negative  
560 eigenvalue, then the mLCE of the training satisfies  $\lambda_1 > 0$  and thus the global training dynamics are  
561 chaotic.*

562 *Proof.* If at any time step  $s$ , the Hessian has a negative eigenvalue  $\lambda_-^{(s)} < 0$ , then the Oseledet matrix  
563 at that time step has an eigenvector  $\ln(1 - \gamma \lambda_-^{(s)})^2 > 0$  (see Theorem A.5). Thus, the maximum LLE  
564 at time step  $s$  is lower bounded by  $0 < \lambda_-^{(s)} \leq \lambda_1^{(s)}$ . Using the result of Theorem A.6 we conclude

$$\lambda_1 = \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{s=1}^t \lambda_1^{(s)} \leq \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{s=1}^t 0 = 0, \quad (44)$$

565  $\square$

566 **Theorem A.8.** *Given a neural network with Hessian  $\mathbf{H}^{(t)}$ , eigenpairs  $(\lambda_i^{(t)}, \mathbf{v}_i^{(t)})_{i=1}^n$  of  $\Lambda^{t,t+1}$  and  
567 gradient updates  $\Delta\theta^{(t)}$ , let there be  $k \leq n$  chaotic eigenpairs, i.e.  $\lambda_1 \geq \dots \lambda_k > 0$ . Suppose all  
568 chaotic components are pruned through projection, leading to a modified update vector*

$$\Delta\tilde{\theta}^{(t)} = \mathbf{v}^{(t)} (\mathbf{v}^{(t)})^T \Delta\theta^{(t)}, \quad \mathbf{v}^{(t)} = \sum_{l>k} \mathbf{v}_l^{(t)} \mathbf{v}_l^{(t)T}. \quad (45)$$

569 Then, for SGD training with learning rate of  $\gamma$  and without momentum, the updated training dynamics  
570 are given by

$$\theta^{(t+1)} = \theta^{(t)} - \gamma \mathbf{v}^{(t)} \mathbf{v}^{(t)T} \mathbf{g}(\theta^{(t)}), \quad \mathbf{Y}^{(t+1)} = (\mathbf{I} - \gamma \mathbf{v}^{(t)} \mathbf{v}^{(t)T}) \mathbf{Y}^{(t)} \quad (46)$$

571 and for sufficiently small variations of the initial parameters, the system is guaranteed not to have  
572 exponentially diverging orbits.

573 *Proof.* The updated training dynamics are given by

$$\theta^{(t+1)} = \theta^{(t)} - \Delta\tilde{\theta}^{(t)} = \theta^{(t)} - \mathbf{v}^{(t)} \mathbf{v}^{(t)T} \Delta\theta^{(t)} = \theta^{(t)} - \gamma \mathbf{v}^{(t)} \mathbf{v}^{(t)T} \mathbf{g}(\theta^{(t)}, \mathbf{x}^{(t)}, \mathbf{Y}^{(t)}). \quad (47)$$

574 Thus, we can derivate w.r.t.  $\theta^{(t)}$  to obtain the dynamics Jacobian and get

$$\frac{\partial \theta_i^{(t+1)}}{\partial \theta_j^{(t)}} = \frac{\partial}{\partial \theta_j^{(t)}} \left( \theta_i^{(t)} - \gamma \sum_k \left\{ \mathbf{v}^{(t)} \mathbf{v}^{(t)T} \right\}_{ik} g_k^{(t)}(\theta^{(t)}, \mathbf{x}^{(t)}, \mathbf{Y}^{(t)}) \right) \quad (48)$$

$$= \frac{\partial \theta_i^{(t)}}{\partial \theta_j^{(t)}} - \gamma \sum_k \left\{ \mathbf{v}^{(t)} \mathbf{v}^{(t)T} \right\}_{ik} \frac{\partial g_k^{(t)}(\theta^{(t)}, \mathbf{x}^{(t)}, \mathbf{Y}^{(t)})}{\partial \theta_j^{(t)}} \quad (49)$$

$$= \delta_{ij} - \gamma \sum_k \left\{ \mathbf{v}^{(t)} \mathbf{v}^{(t)T} \right\}_{ik} H_{kj}, \quad (50)$$

575 or more succinctly in matrix notation

$$\frac{\partial \theta^{(t+1)}}{\partial \theta^{(t)}} = \mathbf{I} - \gamma \mathbf{v}^{(t)} \mathbf{v}^{(t)(T)} \mathbf{H}^{(t)}. \quad (51)$$

576 Now, we still need to show that the system is guaranteed not to have exponentially diverging orbits.  
 577 If  $\mathbf{v}_i^{(t)}$  is a pruned Lyapunov vector (i.e.  $i > k$ ), its eigenvalue for the modified dynamics Jacobian  
 578 becomes 1.

$$(\mathbf{I} - \gamma \mathbf{v}^{(t)} \mathbf{v}^{(t)T} \mathbf{H}^{(t)}) \mathbf{v}_i^{(t)} = \mathbf{v}_i^{(t)} - \gamma \mathbf{v}^{(t)} \mathbf{v}^{(t)T} \lambda_i \mathbf{v}_i^{(t)} \quad (52)$$

$$= \mathbf{v}_i^{(t)} - \gamma \lambda_i \underbrace{\mathbf{v}^{(t)} \mathbf{v}^{(t)T} \mathbf{v}_i}_{=0} \quad (53)$$

$$= \mathbf{v}_i^{(t)} \quad (54)$$

579 It follows immediately that the eigenvalue of the local Oseledec matrix (the LLE) is  $\ln(1) = 0$ .  
 580 Otherwise, if  $\mathbf{v}_i^{(t)}$  was a preserved Lyapunov vector, then it satisfies

$$\mathbf{v}^{(t)} \mathbf{v}^{(t)T} \mathbf{v}_i^{(t)} = \mathbf{v}_i^{(t)} \quad (55)$$

581 by design and thus its LLE is preserved. Summing up, the eigenspectrum of the modified dynamics  
 582 Jacobian satisfies the eigenvalue equation

$$\frac{\partial \theta^{(t+1)}}{\partial \theta^{(t)}} \mathbf{v}_i = \begin{cases} 1\mathbf{v}_i, & k \leq i \\ (1 - \lambda_i)\mathbf{v}_i, & 0 \leq i < k \end{cases} \quad (56)$$

583 such that all eigenvalues translate to non-chaotic or edge-chaotic LLEs smaller than (or equal to) 0.  
 584 Applying Theorem A.6 and using the fact that the LLEs, in particular the maximum LLE at any time  
 585 step, are all smaller than 0 yields

$$\lambda_1 = \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{s=0}^t \lambda_1^{(s)} \leq \lim_{t \rightarrow \infty} \frac{1}{t} \sum_{s=0}^t 0 = 0. \quad (57)$$

586 Hence, the trajectories diverge at most polynomially and we are done.  $\square$

### 587 Distance Statements

588 **Theorem A.9.** Two standard initialized 784-20-10 MLPs in Pytorch have a expected L2-distance of  
 589  $\approx 6.38$ .

590 *Proof.* We calculate the distance between single layers first and differentiate between weight and  
 591 bias. Pytorch initializes weights  $\mathbf{w}$  and biases  $\mathbf{b}$  uniformly with

$$\mathbf{w}_i, \mathbf{b}_i \sim \mathcal{U}\left(-\frac{1}{\sqrt{k}}, \frac{1}{\sqrt{k}}\right), \quad k := \# \text{ in\_features} \quad \text{and} \quad m := \# \text{ out\_features}$$

592 1. case: weight

593 So let  $\mathbf{w}_i^{(1)}, \mathbf{w}_i^{(2)} \sim \mathcal{U}\left(-\frac{1}{\sqrt{k}}, \frac{1}{\sqrt{k}}\right)$  for all  $i$ .

$$\begin{aligned} \mathbb{E}_{\mathbf{w}_i^{(1)}, \mathbf{w}_i^{(2)} \sim \mathcal{U}\left(-\frac{1}{\sqrt{k}}, \frac{1}{\sqrt{k}}\right)} [\|\mathbf{w}^{(1)} - \mathbf{w}^{(2)}\|_2^2] &= \mathbb{E}_{\mathbf{w}_i \sim \mathcal{U}\left(-\frac{2}{\sqrt{k}}, \frac{2}{\sqrt{k}}\right)} [\|\mathbf{w}\|_2^2] \\ &= \sum_{i=1}^{km} \mathbb{E}_{\mathbf{w}_i \sim \mathcal{U}\left(-\frac{2}{\sqrt{k}}, \frac{2}{\sqrt{k}}\right)} [w_i^2] = \sum_{i=1}^{km} \text{Var}(\mathbf{w}_i) = km \frac{4^2}{12k} = \frac{4m}{3} \end{aligned}$$

594 Thus  $\mathbb{E}[\|\mathbf{w}^{(1)} - \mathbf{w}^{(2)}\|_2] = \sqrt{\frac{4m}{3}}$ .

595 2. case: bias

596 Analogously for the bias where the dimension of the bias is not  $km$  but  $m$ , we end up with

$$\mathbb{E}[\|\mathbf{b}^{(1)} - \mathbf{b}^{(2)}\|_2] = \sqrt{\frac{4m}{3k}}.$$

598 We now calculate the expected distance over all layers of the two 784-20-10 MLPs  $\theta^{(1)}, \theta^{(2)}$ :

$$\mathbb{E}[\|\theta^{(1)} - \theta^{(2)}\|_2] = \sqrt{\frac{4(m_1 + m_2)}{3} + \frac{4m_1}{k_1} + \frac{4m_2}{k_2}} = \sqrt{\frac{4(30)}{3} + \frac{4(20)}{3(784)} + \frac{4(10)}{3(20)}} \approx 6.38$$

599  $\square$

600 **Theorem A.10.** Two random Gaussian walks  $\mathbf{x}_1(t), \mathbf{x}_2(t) = \mathbf{x}_i(t-1) + \boldsymbol{\eta}_i(t)$  with  $\mathbf{x}_1(0), \mathbf{x}_2(0) =$   
601  $\mathbf{0}$  and  $\boldsymbol{\eta}_1(t), \boldsymbol{\eta}_2(t) \in \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$  have an expected distance of  $\sqrt{2T\bar{D}}\sigma$  until time step  $T$ .

*Proof.*

$$\begin{aligned} \mathbb{E} [\|\mathbf{x}_1(t) - \mathbf{x}_2(t)\|_2] &= \mathbb{E}_{\boldsymbol{\eta}_1(t), \boldsymbol{\eta}_2(t) \in \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})} \left[ \left\| \sum_{t=1}^T \boldsymbol{\eta}_1(t) - \boldsymbol{\eta}_2(t) \right\|_2 \right] \\ &\stackrel{(1)}{=} \mathbb{E}_{\boldsymbol{\eta}'(t) \in \mathcal{N}(0, \sigma^2 \mathbf{I})} \left[ \left\| \sum_{t=1}^{2T} \boldsymbol{\eta}'(t) \right\|_2 \right] \stackrel{(2)}{=} \mathbb{E}_{\boldsymbol{\eta}'(t) \in \mathcal{N}(0, 2T\sigma^2 \mathbf{I})} [\|\tilde{\boldsymbol{\eta}}(t)\|_2] \\ &\stackrel{(3)}{=} \sqrt{2T\bar{D}}\sigma. \end{aligned} \quad (58)$$

602 Equation (1) is valid, because  $\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}) = -\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$  and (2), because variances of the sum two  
603 independent variables add up. For (3), observe that

$$\mathbb{E}_{\mathbf{x} \in \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})} [\|\mathbf{x}\|_2^2] = \sum_{i=1}^D \mathbb{E}_{x_i \in \mathcal{N}(0, \sigma^2)} [x_i^2] = \mathbb{E}_{x' \in \mathcal{N}(0, D\sigma^2)} [x'^2] = D\sigma^2 \quad (59)$$

604  $\square$

605 **Theorem A.11.** Let the gradients of two models  $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^D$  be defined as

$$g_i(t) = \mathbf{w}_i(t) + \boldsymbol{\eta}_i$$

606 where  $\boldsymbol{\eta}_1, \boldsymbol{\eta}_2 \in \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$  are independent and identically distributed, isotropic noise terms and  
607  $\|\sum_{s=1}^t (\mathbf{w}_1(s) - \mathbf{w}_2(s))\|_2 = \alpha t$  diverges linearly over time. Assume both models are initialized  
608 with the same set of parameters  $\boldsymbol{\theta}_1^{(0)} = \boldsymbol{\theta}_2^{(0)}$ . Then distance of the two models evolves as

$$\mathbb{E}_{\boldsymbol{\eta}_1, \boldsymbol{\eta}_2 \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})} [\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2] = \sqrt{\alpha^2 t^2 + \beta t},$$

609 where  $\beta = 2D\sigma^2$ .

*Proof.*

$$\mathbb{E}_{\boldsymbol{\eta}_1, \boldsymbol{\eta}_2 \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})} \left[ \left\| \sum_{s=0}^t (\mathbf{w}_1(s) - \mathbf{w}_2(s)) + \sum_{s=0}^t (\boldsymbol{\eta}_1(s) - \boldsymbol{\eta}_2(s)) \right\|_2 \right] \quad (60)$$

610 We want to calculate the term (60) where  $\sum_{s=0}^t (\mathbf{w}_1(s) - \mathbf{w}_2(s))$  defines the linear divergence term  
611 and  $\sum_{s=0}^t (\boldsymbol{\eta}_1(s) - \boldsymbol{\eta}_2(s))$  is the random Gaussian walk as in equation (15). For simplification,  
612 assume that we take the expected value over  $\boldsymbol{\eta}_1(s), \boldsymbol{\eta}_2(s) \in \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$  in all of the following  
613 expression

$$\begin{aligned} &\mathbb{E} \left[ \left\| \sum_{s=0}^t (\mathbf{w}_1(s) - \mathbf{w}_2(s)) + \sum_{s=0}^t (\boldsymbol{\eta}_1(s) - \boldsymbol{\eta}_2(s)) \right\|_2^2 \right] \\ &= \mathbb{E} \left[ \left\| \sum_{s=0}^t (\mathbf{w}_1(s) - \mathbf{w}_2(s)) \right\|_2^2 + \left\| \sum_{s=0}^t (\boldsymbol{\eta}_1(s) - \boldsymbol{\eta}_2(s)) \right\|_2^2 \right. \\ &\quad \left. + \sum_{s,s'=0}^t \langle (\mathbf{w}_1(s') - \mathbf{w}_2(s')), \boldsymbol{\eta}_1(s) - \boldsymbol{\eta}_2(s) \rangle \right] \\ &= \alpha^2 t^2 + \beta t + \mathbb{E} \left[ \sum_{s,s'=0}^t \langle (\mathbf{w}_1(s') - \mathbf{w}_2(s')), \boldsymbol{\eta}_1(s) - \boldsymbol{\eta}_2(s) \rangle \right] \end{aligned}$$

614 where  $\beta := 2D\sigma^2$ , which follows from Theorem A.10. Now this leaves to show that the last summand  
615 is zero. In fact, observe that for all  $v \in \mathbb{R}^D$

$$\mathbb{E}_{\boldsymbol{\omega} \sim \mathcal{N}(\mathbf{0}, 2\sigma^2 \mathbf{I})} [\langle v, \boldsymbol{\omega} \rangle] = \sum_{i=1}^D v_i \mathbb{E}_{\boldsymbol{\omega}_i \sim \mathcal{N}(\mathbf{0}, 2\sigma^2)} [\omega_i] = 0,$$

616 which can be applied to  $v = \mathbf{w}_1(s) - \mathbf{w}_2(s)$  and  $\boldsymbol{\omega} = \boldsymbol{\eta}_1(s) - \boldsymbol{\eta}_2(s)$  for all  $s$ .  $\square$

617 **Theorem A.12.** *The modified Subspace Similarity defined as*

$$\text{overlap}(V, W) = \frac{\text{Tr}(P_V, P_W)}{\sqrt{\text{Tr}(P_V)\text{Tr}(P_W)}} \sqrt{\frac{d_{\max}}{d_{\min}}} = \frac{1}{d_{\min}} \|\text{flatten}(V^T W)\|_2^2$$

618 where  $P_V, P_W$  are the orthogonal projectors onto the subspaces  $V, W$  fulfills the following properties:  
619

- 620 1. If  $V \perp W$ , then  $\text{overlap}(V, W) = 0$
- 621 2. If  $V \subseteq W$  or  $W \subseteq V$ , then  $\text{overlap}(V, W) = 1$
- 622 3. For arbitrary subspaces  $V, W$ ,  $\text{overlap}(V, W) \in [0, 1]$

623 *Proof.* We first show the equality of the definition. The orthogonal projector is defined as  $P_V :=$   
624  $\mathbf{V}\mathbf{V}^T$  using  $\mathbf{V}$  as a matrix of orthonormal basis vectors (in the columns) and as the space  $V$  to  
625 simplify the following equations. Now the numerator equals

$$\text{Tr}(P_V, P_W) = \text{Tr}(\mathbf{V}\mathbf{V}^T \mathbf{W}\mathbf{W}^T) \stackrel{*}{=} \text{Tr}((\mathbf{W}^T \mathbf{V})(\mathbf{V}^T \mathbf{W})) = \|\text{flatten}(\mathbf{V}^T \mathbf{W})\|_2^2$$

626 where for \* we used the cyclic property of the trace. We utilize this property again to calculate

$$\text{Tr}(P_V) = \text{Tr}(\mathbf{V}\mathbf{V}^T) = \text{Tr}(\mathbf{V}^T \mathbf{V}) = \text{Tr}(\text{Id}_V) = \dim(V)$$

627 The rest of the equation follows by applying the two results.

628 Now statement 1. follows directly from the fact that if  $V \perp W$  then  $\mathbf{V}^T \mathbf{W} = \mathbf{0}$ . For statement 2.  
629 observe that if  $W \subseteq V$  then the orthogonal projection of  $W$  onto  $V$  is the identity:  $P_V|_W = \text{Id}_W$ .  
630 So the similarity is

$$\text{overlap}(V, W) = \frac{1}{\dim(W)} \text{Tr}(\mathbf{V}\mathbf{V}^T \mathbf{W}\mathbf{W}^T) = \frac{\text{Tr}(\mathbf{W}\mathbf{W}^T)}{\dim(W)} = \frac{\text{Tr}(\mathbf{W}^T \mathbf{W})}{\dim(W)} = 1$$

631 Finally for statement 3., we see that  $\|\text{flatten}(\mathbf{V}^T \mathbf{W})\|_2^2 \geq 0$  is clearly positive and what is left to  
632 prove is that it is bounded by 1. Let  $n := \dim(V)$ ,  $m := \dim(W)$  and  $\mathbf{v}_1, \dots, \mathbf{v}_n$  and  $\mathbf{w}_1, \dots, \mathbf{w}_m$  be  
633 an orthonormal basis of  $V, W$ , respectively. Assume with out loss of generality  $m < n$

$$\|\text{flatten}(\mathbf{V}^T \mathbf{W})\|_2^2 = \sum_{i=1}^n \sum_{j=1}^m \langle \mathbf{v}_i, \mathbf{w}_j \rangle^2$$

634 Now extend the orthonormal basis of  $V$  to a orthonormal basis  $B := \mathbf{v}_1, \dots, \mathbf{v}_n, \dots, \mathbf{v}_k$  of the whole  
635 space  $V' \supseteq V, W$ . And write the orthonormal basis of  $W$  in terms of  $B$ :  $\mathbf{w}_j = \sum_{i=1}^k \lambda_{ji} \mathbf{v}_i$ . Then

$$\sum_{i=1}^n \sum_{j=1}^m \langle \mathbf{v}_i, \mathbf{w}_j \rangle^2 = \sum_{i=1}^n \sum_{j=1}^m \lambda_{ji}^2 \langle \mathbf{v}_i, \mathbf{v}_i \rangle^2 = \sum_{i=1}^n \sum_{j=1}^m \lambda_{ji}^2$$

636 Now  $\sum_{i=1}^n \lambda_{ji}^2 \leq 1$ , since

$$1 = \|\mathbf{w}_j\|_2^2 = \left\langle \sum_{i=1}^k \lambda_{ji} \mathbf{v}_i, \sum_{i=1}^k \lambda_{ji} \mathbf{v}_i \right\rangle = \sum_{i=1}^k \lambda_{ji}^2 \geq \sum_{i=1}^n \lambda_{ji}^2$$

637 And we apply this inequality to what we previously ended up with

$$\sum_{i=1}^n \sum_{j=1}^m \lambda_{ji}^2 \leq \sum_{j=1}^m 1 = m$$

638 to arive at what we wanted to show:  $\text{overlap}(V, W) = \frac{1}{m} \|\text{flatten}(V^T W)\|_2^2 \leq \frac{m}{m} = 1$ .  $\square$

639 **A.1 Generalization and long-term chaos**

640 In this subsection, we provide a mathematical argument for why we can assume chaotic dynamics (in  
641 the sense of sensitivity to initial conditions) to generally have an impact on generalization properties  
642 of ANNs trained with SGD. Consider an ANN trained with some procedure  $M$  to yield a parametrized  
643 function  $f_\theta^M$  by the end of the training, where  $\theta := \theta^{(T)}$  are the parameters at the end of the training.  
644 Let  $S$  be the set of all data points that could potentially be used to train a network for its respective  
645 task, and let  $S^{train} \subset S$  be the entirety of data points used to actually train the network. Furthermore,  
646 let  $\mathcal{L}(f_\theta^M; x)$  be the loss from calculating the network outputs, given a sample  $x \in S$ . Let us assume  
647 the input data to be structured s.t.  $\forall x \in S \exists \epsilon > 0$  s.t.  $\forall y \in \mathcal{B}_\epsilon(x) : y \in S$ . A training procedure that  
648 leads to good generalization should be one that minimizes the variance of the loss of the learned  
649 function with respect to data points  $x \in S$ . Thus, the generalization capacity can be metricized by

$$\text{Var}[\mathcal{L}(f_\theta^M)] = \mathbb{E}_{x \in S} \left[ (\mathcal{L}(f_\theta^M; x) - \mathbb{E}_{x \in S} [\mathcal{L}(f_\theta^M; x)])^2 \right] = \mathbb{E}_{x \in S} \left[ (\mathcal{L}(f_\theta^M; x) - \bar{\mathcal{L}})^2 \right] \quad (61)$$

650 In particular, the generalization properties should be good within any finite region  $\tilde{S} \subset S$ . Assuming  
651  $\mathcal{L}(f_\theta^M; x)$  is a piecewise-smooth function with respect to  $x$  and assuming  $x$  is not a border point at a  
652 derivative discontinuity, the loss function may be expressed using a Taylor polynomial:

$$\mathcal{L}(f_\theta^M; \mathbf{x}_s + \delta \mathbf{x}) = \mathcal{L}(f_\theta^M; \mathbf{x}_s) + \delta \mathbf{x}^T \nabla_{\mathcal{L}} f_\theta^M + \frac{1}{2} \delta \mathbf{x}^T \mathbf{H}_{\mathcal{L}}(f_\theta^M; \mathbf{x}_s) \delta \mathbf{x} + \dots \quad (62)$$

653 Let us consider the ability of the function  $f_\theta$  to generalize on points in a vicinity  $\tilde{S}$  of a point  $x_s$   
654 which was part of the training data, and where all other points  $x \in \tilde{S} \setminus \{x_s\}$  were not seen during the  
655 training procedure  $M$ . For the sake of simplicity, let us assume that  $x_s$  was seen exactly once during  
656 the training, namely at time step  $s$ . Then from the chain rule, it follows that

$$\mathcal{L}'(f_\theta^M; \mathbf{x}_s) = \frac{d\mathcal{L}}{d\mathbf{x}}|_{\mathbf{x}=\mathbf{x}_s} = \frac{d\mathcal{L}}{d\theta^{(T)}} \frac{d\theta^{(T)}}{d\theta^{(s+1)}} \frac{d\theta^{(s+1)}}{dx_s} = \frac{d\mathcal{L}}{d\theta^{(T)}} \mathbf{Y}^{(s+1,T)} \frac{d\theta^{(s+1)}}{dx_s} \quad (63)$$

657 When the training procedure  $M$  is SGD (without momentum), the update rule gives us that

$$\frac{d\theta^{(s+1)}}{dx_s} = \frac{d}{d\theta^{(s)}}(\theta^{(s)} - \gamma \mathbf{g}(\theta^{(s)}; \mathbf{x}_s)) = \mathbf{I} - \gamma \frac{\partial^2 \mathcal{L}(\theta^{(s)}; x_s)}{\partial \mathbf{x}_s \partial \theta^{(s)}} = \mathbf{I} - \gamma \mathbf{H}_{x\theta}^{(s)}, \quad (64)$$

658 where the tangent map  $\mathbf{Y}^{(s+1,T)}$  gives us the sensitivity of the parameters at the end of the training  
659 procedure to the parameters at a previous time point  $\theta^{(s+1)}$ . Due to the product rule, the higher-order  
660 terms of the Taylor expansion will also contain some dependency on  $\mathbf{Y}^{(s+1,T)}$ . However, for more  
661 clarity, we restrict ourselves to first-order terms to end up with the expression

$$\text{Var}_{\tilde{S}}[\mathcal{L}(f_\theta^{SGD})] \approx \mathbb{E}_{\mathbf{x} \in \tilde{S}} \left[ \left( \mathcal{L}(f_\theta^{SGD}; \mathbf{x}_s) + \frac{d\mathcal{L}}{d\theta^{(T)}} \mathbf{Y}^{(s+1,T)} (\mathbf{I} - \gamma \mathbf{H}_{x\theta}^{(s)}) \delta x - \bar{\mathcal{L}} \right)^2 \right] \quad (65)$$

662 What this argument tells us is that, in order for the generalization capacity of a trained function  $f_\theta^{SGD}$   
663 to be good at least for unseen samples within some vicinity of previously seen samples  $\mathbf{x}_s$ , training  
664 dynamics should not be too sensitive on initial conditions. The argument can be extended to vicinities  
665 of arbitrary sets of points seen during training by considering the expected values over respective  
666 regions.

667 **B Further Analysis on Locally Chaotic Dynamics**

668 **B.1 Pruning Experiments: Performance Metrics**

669 **USPS**

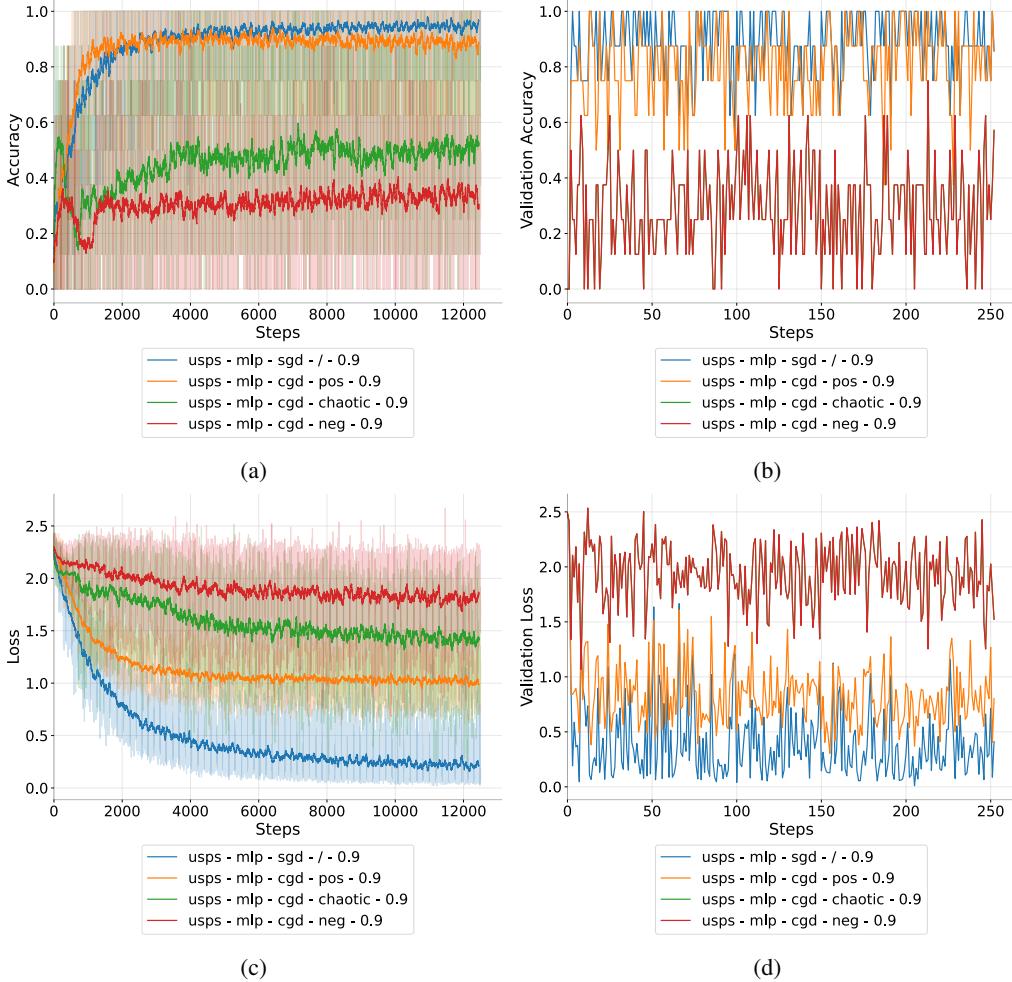


Figure B.1.1: Line plots of the (a) training accuracy and (b) validation accuracy, as well as (c) training loss and (d) validation loss for an MLP (relu activation) trained on the USPS dataset with momentum (0.9). The models trained with CGD use pruning of the full chaotic Hessian spectrum at every time step. The smoothed, solid lines are obtained through local averaging (window size 50) of the respective sequential data. Models trained with pruning of the full chaotic/negative spectrum show much slower convergence than the model trained with full positive spectrum pruning and reach lower performance in the observed time range. All models are initialized with the same random seed.

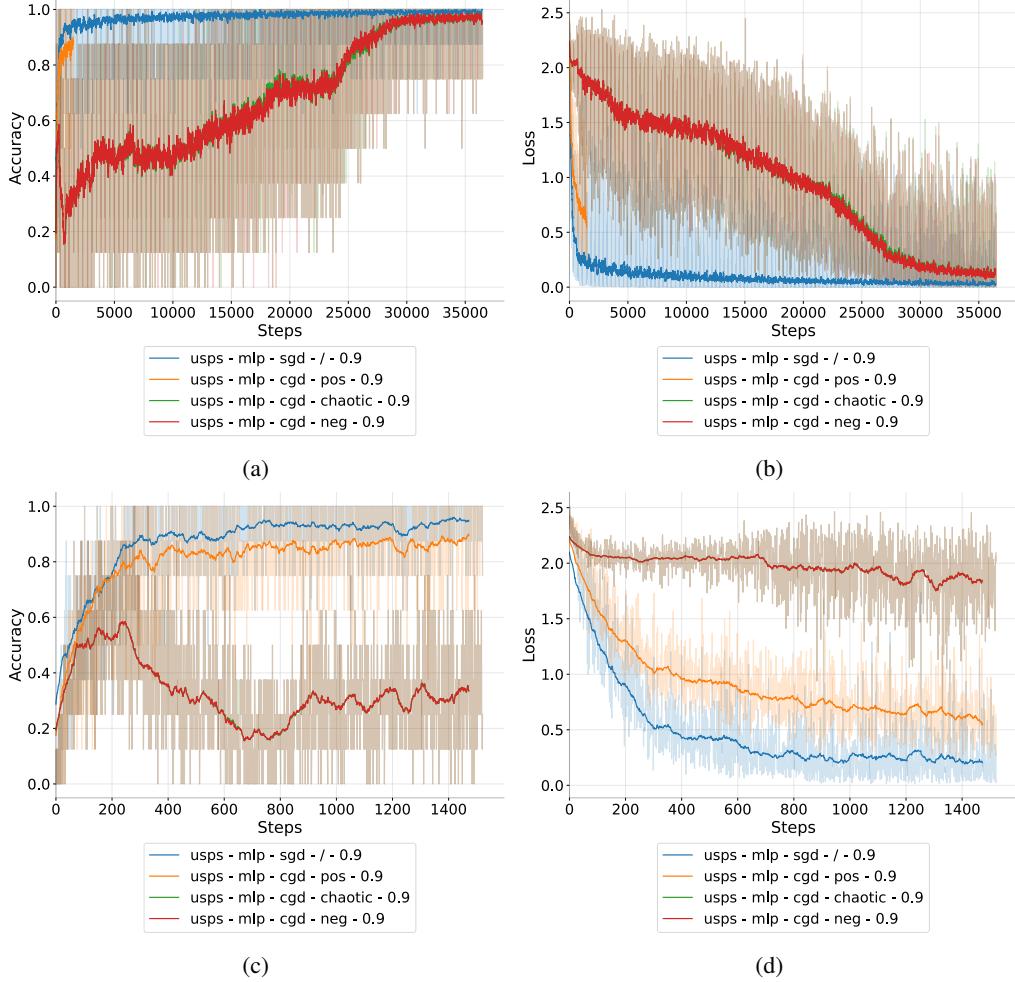


Figure B.1.2: Line plots of the (a)/(c) accuracy and (b)/(d) loss curves for an MLP (relu activation) trained on the USPS dataset with momentum (0.9). The models trained with CGD use pruning of the full chaotic, negative or positive Hessian spectrum at every time step. For greater clarity, the plots in the bottom row show a smaller time window. The smoothed, solid lines are obtained through local averaging (window size 50) of the respective sequential data. Using pruning of the full positive spectrum (yellow), the loss becomes singular, crashing the run. This suggests some information about positive curvature is required to avoid an excessively chaotic evolution of the system. However, the run with positive pruning still reaches high performance on both metrics before crashing. Meanwhile, the runs with full negative/chaotic pruning take much longer to converge, suggesting the negative Hessian eigenpairs are essential to enable fast training by SGD.

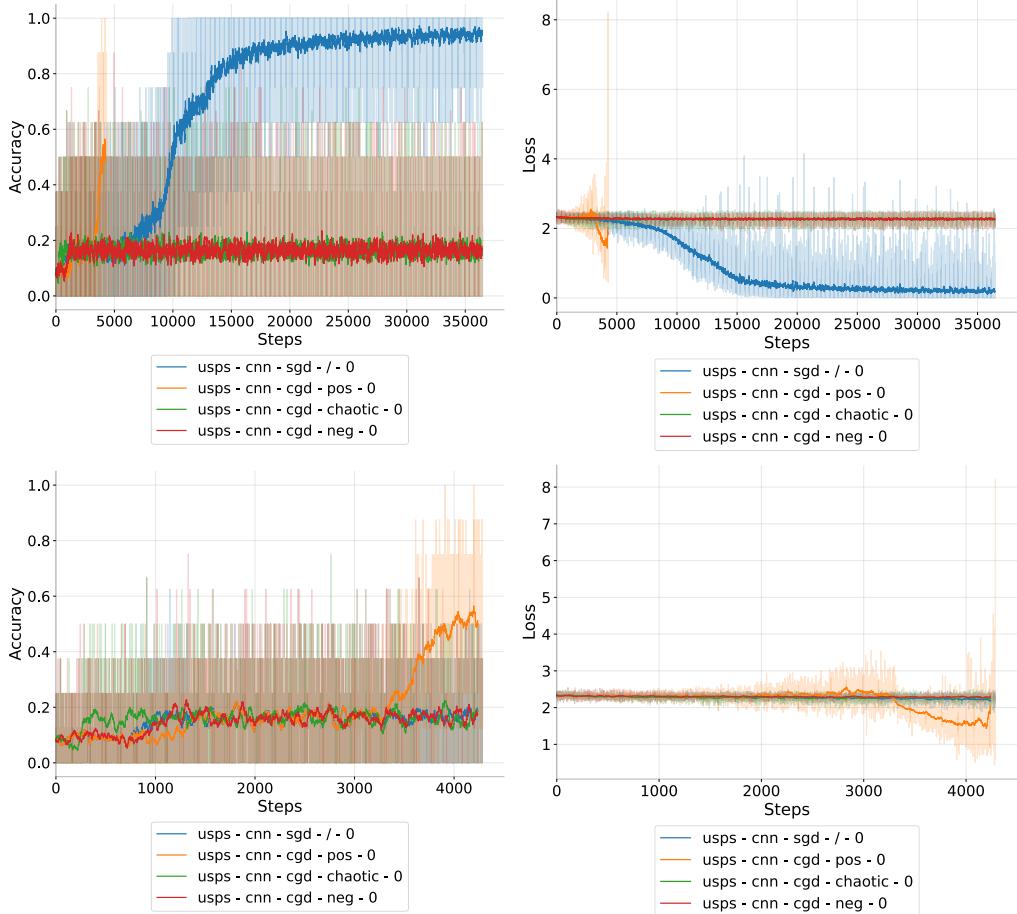


Figure B.1.3: Line plots of the (a)/(c) accuracy and (b)/(d) loss curves for a CNN trained on the USPS dataset without momentum. The models trained with CGD use pruning of the full chaotic, negative or positive Hessian spectrum at every time step. For greater clarity, the plots in the bottom row show a smaller time window. The smoothed, solid lines are obtained through local averaging (window size 50) of the respective sequential data. Using pruning of the full positive spectrum (yellow), the loss becomes singular, crashing the run. This suggests some information about positive curvature is required to avoid an excessively chaotic evolution of the system. However, the run with positive pruning still reaches high performance on both metrics before crashing. Meanwhile, the runs with full negative/chaotic pruning are unable to achieve significant improvements on both metrics.

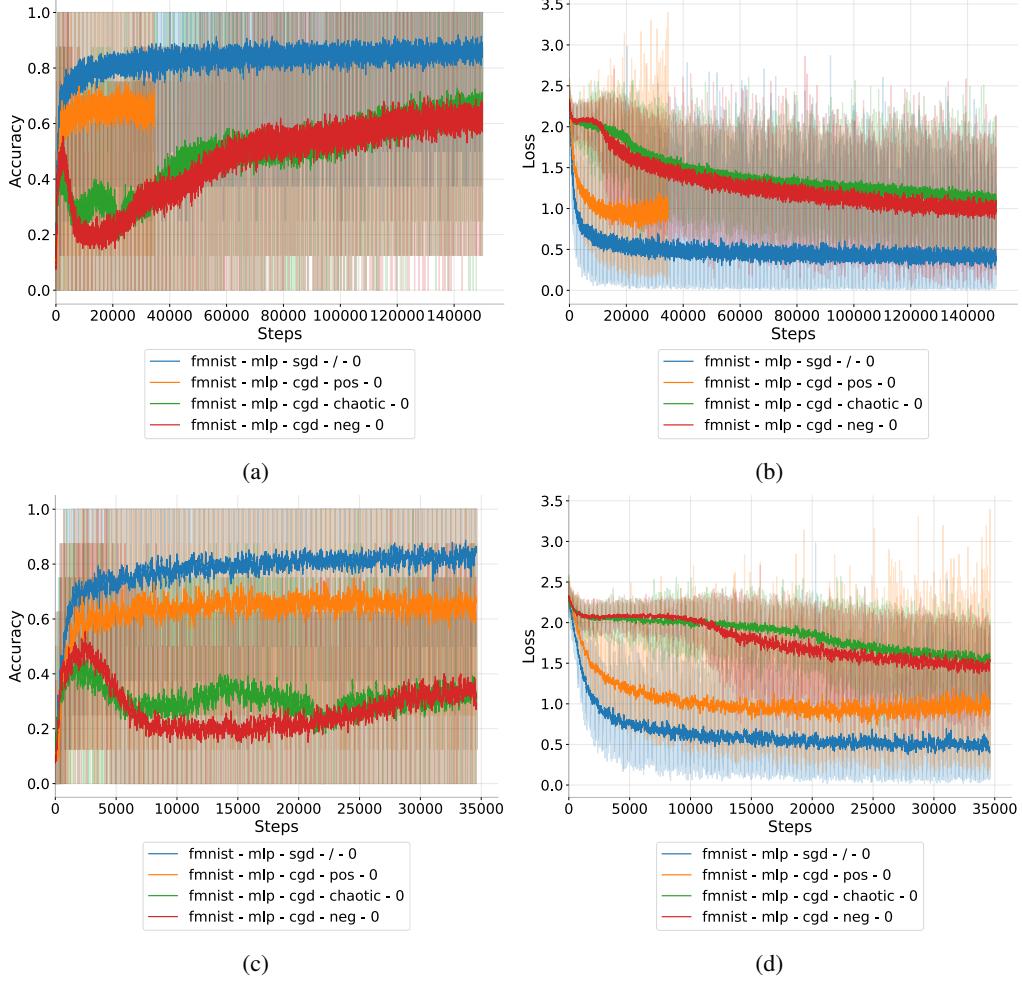


Figure B.1.4: Line plots of the (a)/(c) accuracy and (b)/(d) loss curves for an MLP trained on the FashionMNIST dataset without momentum. The models trained with CGD use pruning of the full chaotic, negative or positive Hessian spectrum at every time step. For greater clarity, the plots in the bottom row show a smaller time window. The smoothed, solid lines are obtained through local averaging (window size 50) of the respective sequential data. Using pruning of the full positive spectrum (yellow), the loss becomes singular, crashing the run. The run with positive pruning quickly reaches high performance on both metrics before crashing. The runs with full negative/chaotic pruning need much longer to reach a comparable performance on the training metrics.

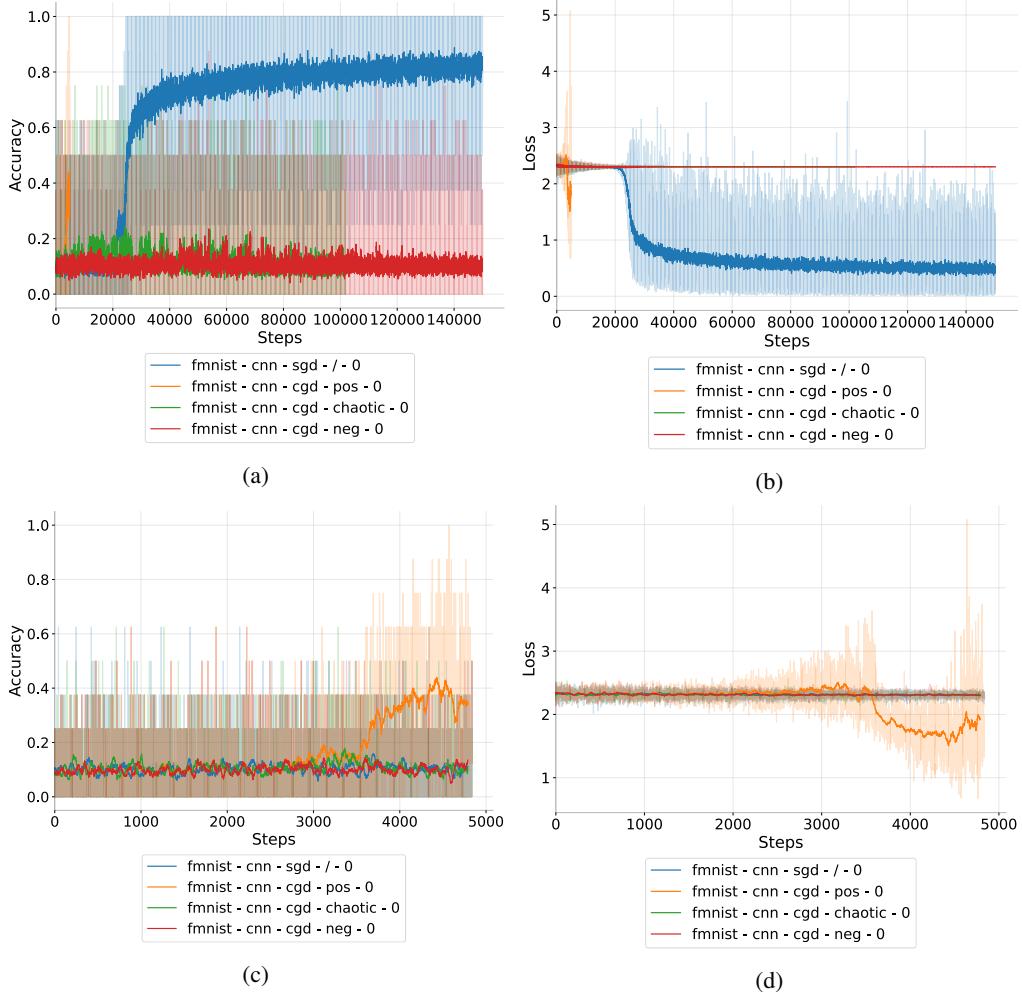


Figure B.1.5: Line plots of the (a)/(c) accuracy and (b)/(d) loss curves for a 2D CNN trained on the FashionMNIST dataset without momentum. The models trained with CGD use pruning of the full chaotic, negative or positive Hessian spectrum at every time step. For greater clarity, the plots in the bottom row show a smaller time window. The smoothed, solid lines are obtained through local averaging (window size 50) of the respective sequential data. Using pruning of the full positive spectrum (yellow), the loss becomes singular, crashing the run. The run with positive pruning quickly reaches high performance on both metrics before crashing. The runs with full negative/chaotic pruning need much longer to reach a comparable performance on the training metrics.

671 **B.2 Pruning Experiments: Eigenspaces**

672 **USPS**

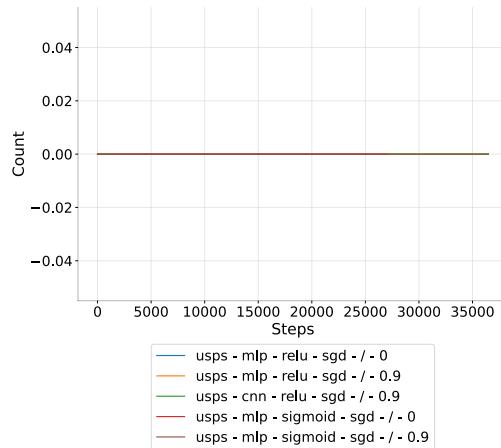


Figure B.2.1: Count of positive eigenvalues of the Hessian that satisfy the chaos criterion  $\lambda_i > \frac{2}{\gamma}$  throughout the training, for several runs. The count is effectively zero.

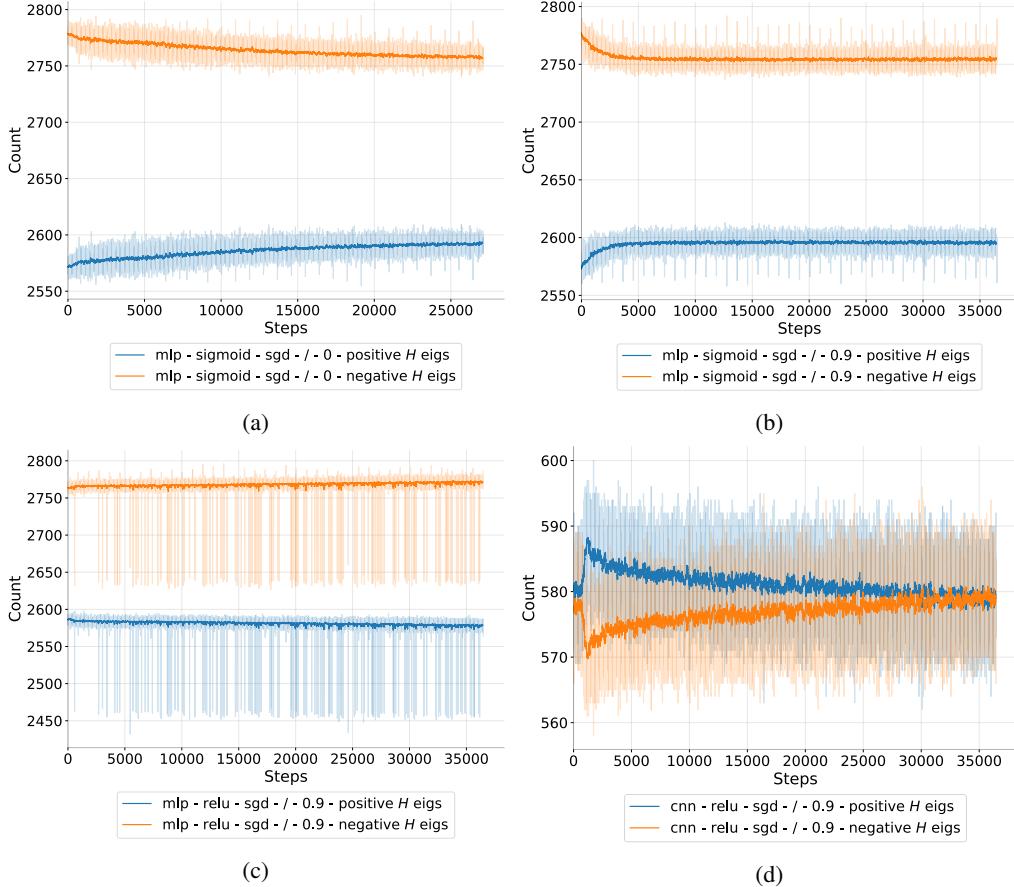


Figure B.2.2: **Number of positive** ( $\lambda_i > 0$ ) and **negative** ( $\lambda_i < 0$ ) eigenvalues of the Hessian for different runs on the USPS dataset. For the MLP models with sigmoid activation (a) and (b), the number of positive eigenvalues increases as training proceeds, while the number of negative eigenvalues decreases (regardless of momentum). With ReLU activation (c), the number of positive and negative eigenvalues is subject to strong fluctuations, although the positive eigenvalue count seems to decrease very slightly, opposed to the sigmoid runs. For a CNN (d), the count of positive eigenvalues increases very quickly, suggesting a very fast exploration phase in agreement with our model, but eventually relaxes back to the initial levels. The evolution of the negative eigenvalue count is complementary.

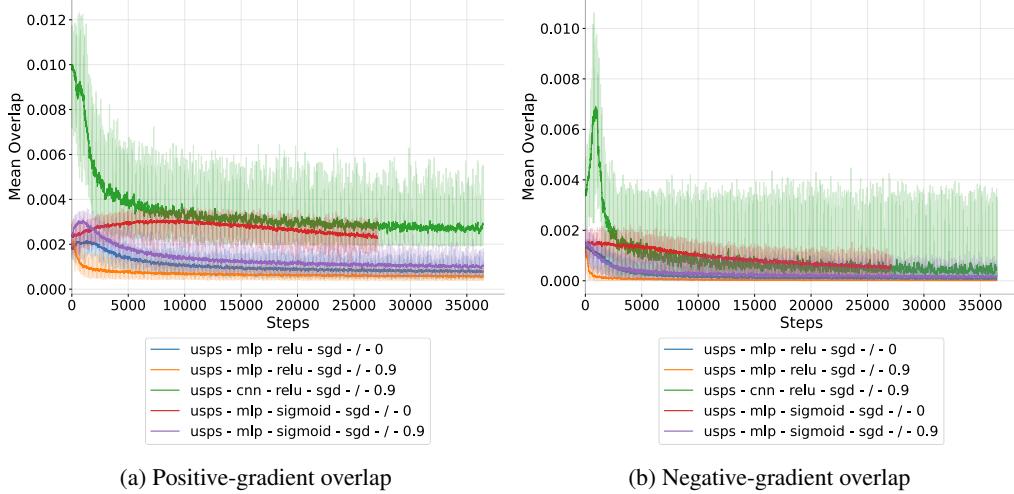


Figure B.2.3: Average overlaps between the normalized (a) positive Hessian eigenvectors and the gradient, and (b) negative Hessian eigenvectors and the gradient, where each curve corresponds to a different run configuration for the USPS dataset. The overlap between two vectors is measured as absolute cosine similarity. For all models, the positive spectra have on average a greater overlap with the gradient than the negative spectra, and the overlaps decrease as training progresses.

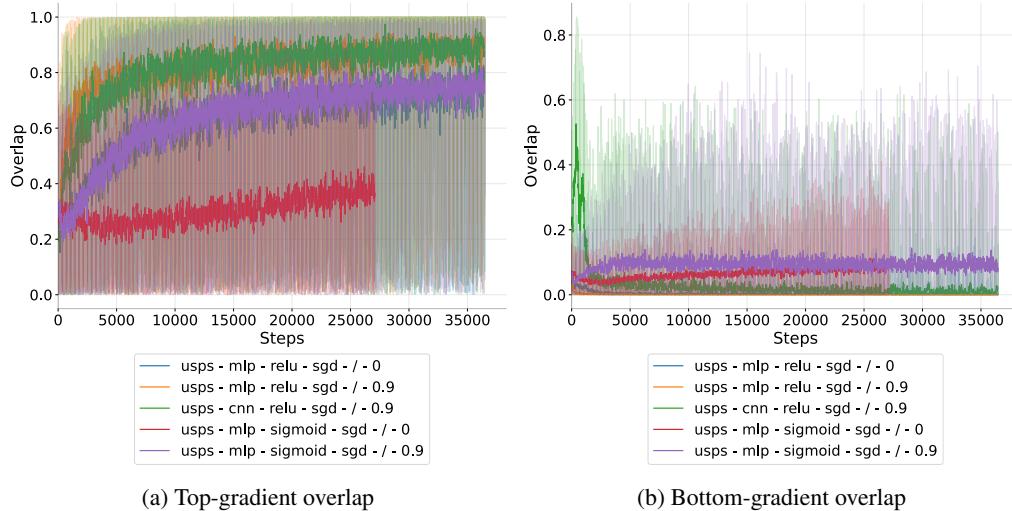


Figure B.2.4: Overlaps between the (normalized) (a) top Hessian eigenvector and the gradient, and (b) the minimum (i.e. most negative) Hessian eigenvector and the gradient, where each curve corresponds to a different run configuration. The overlap between two vectors is measured as absolute cosine similarity. The overlap with the top eigenvector is on average bigger than with the minimum eigenvector, in agreement with the findings of Gur-Ari et al. (2018). Although subject to wild fluctuations, the overlap with the top eigenvector increases on the long run during training, while the average overlap with the minimum eigenvectors decreases quickly at the beginning of the training (after an initial increase for the MLP), suggesting the negative spectrum of the Hessian is more important during an initial **discovery phase**, while the positive spectrum becomes more important during a subsequent **discovery phase**.

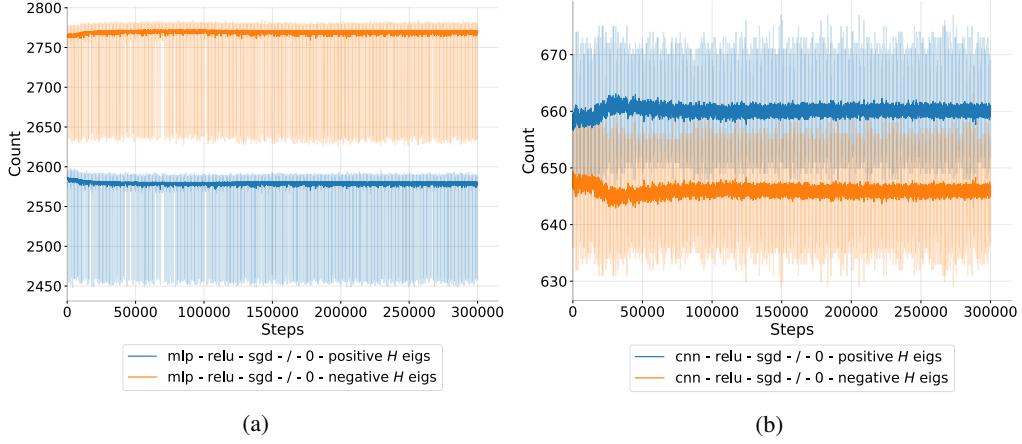


Figure B.2.5: Number of positive ( $\lambda_i > 0$ ) and negative ( $\lambda_i < 0$ ) eigenvalues of the Hessian for different runs on the FashionMNIST. The eigenvalues behave analogously as for the USPS dataset. In the MLP with ReLU, the positive eigenvalue count slightly decreases at the beginning of the training, while for the CNN, the positive eigenvalue count initially increases and then relaxes to lower levels. The evolution of the negative eigenvalue count is complementary.

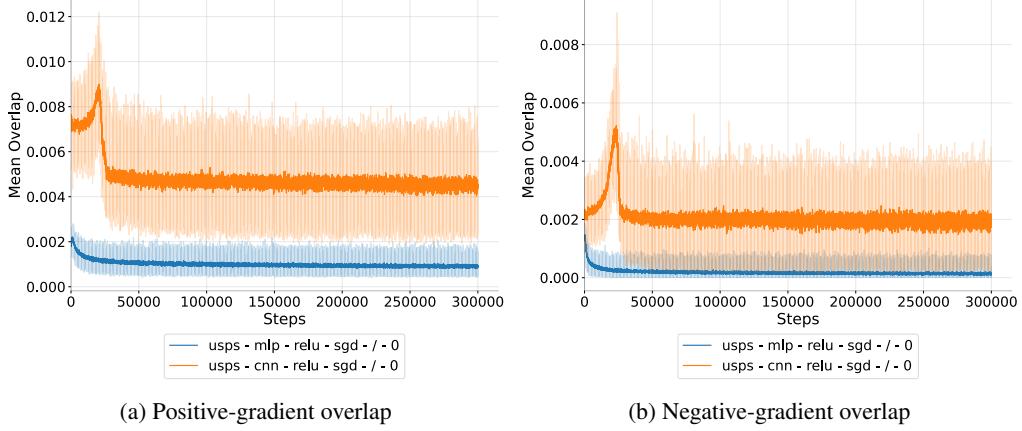


Figure B.2.6: Average overlaps between the (normalized) (a) positive Hessian eigenvectors and the gradient, and (b) negative Hessian eigenvectors and the gradient, where each curve corresponds to a different run configuration for the USPS dataset. The overlap between two vectors is measured as absolute cosine similarity. The situation here is similar to the USPS dataset: For all models, the positive spectra have on average a greater overlap with the gradient than the negative spectra, and the overlaps decrease as training progresses.

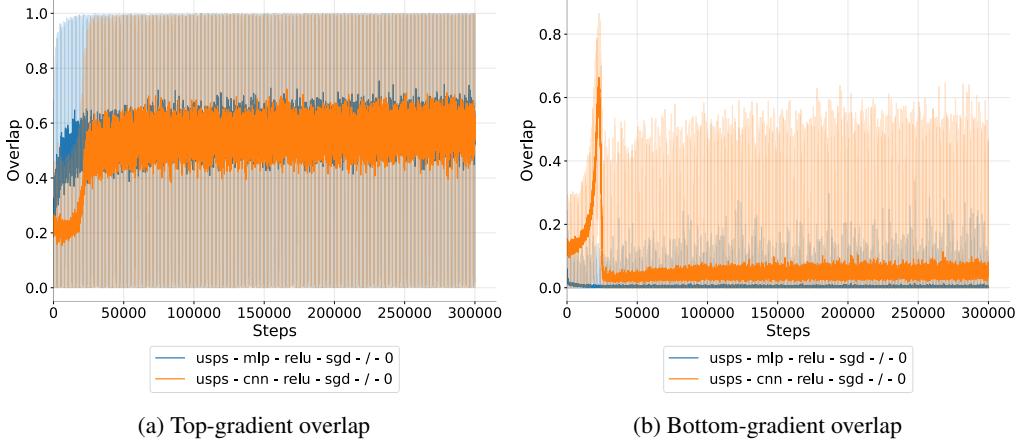


Figure B.2.7: Overlaps between the (normalized) (a) top Hessian eigenvector and the gradient, and (b) the minimum (i.e. most negative) Hessian eigenvector and the gradient, where each curve corresponds to a different run configuration. The overlap between two vectors is measured as absolute cosine similarity. The situation here is similar to the USPS dataset: The overlap with the top eigenvector is on average bigger than with the minimum eigenvector, in agreement with the findings of Gur-Ari et al. [2018]. Although subject to wild fluctuations, the overlap with the top eigenvector increases on the long run during training, while the average overlap with the minimum eigenvectors decreases quickly at the beginning of the training for most runs.

### 674 B.3 Additional

$$z(x, y) = -\cos(x) - 0.05y^2$$

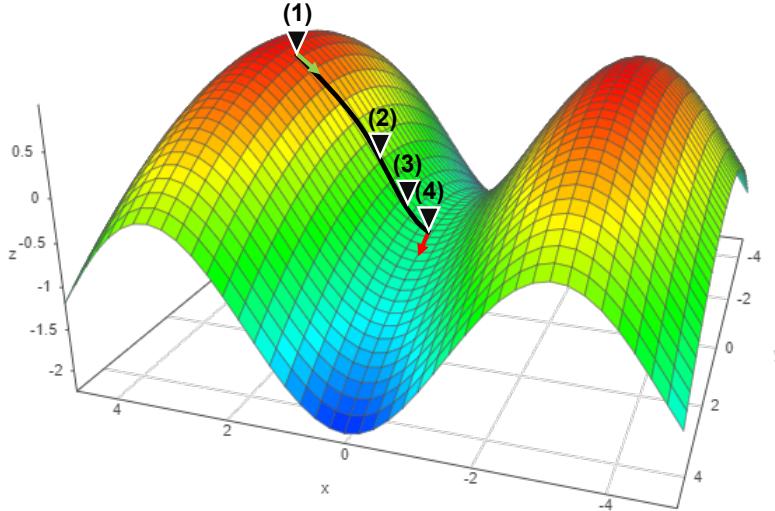


Figure B.3.1: A simplified model of training by SGD illustrating how negative curvature precedes positive curvature: At the beginning of the training, the optimizer is near a high-dimensional local maximum or saddle-point (1), from which the loss can be reduced by updating the parameters in directions of high negative curvature. As the optimizer passes the inflection point (2) along that given axis, the curvature along that axis becomes positive (3), but the potential for loss improvements in this direction is smaller at this point. In the vicinity of the local minimum (4) along this axis (actually a saddlepoint), the optimizer encounters new axes of high negative curvature and start moving in those directions. Image generated using <https://academo.org/demos/3d-surface-plotter/>.

675 **C Further Analysis on Globally Chaotic Dynamics**

676 **C.1 Lyapunov Exponents**

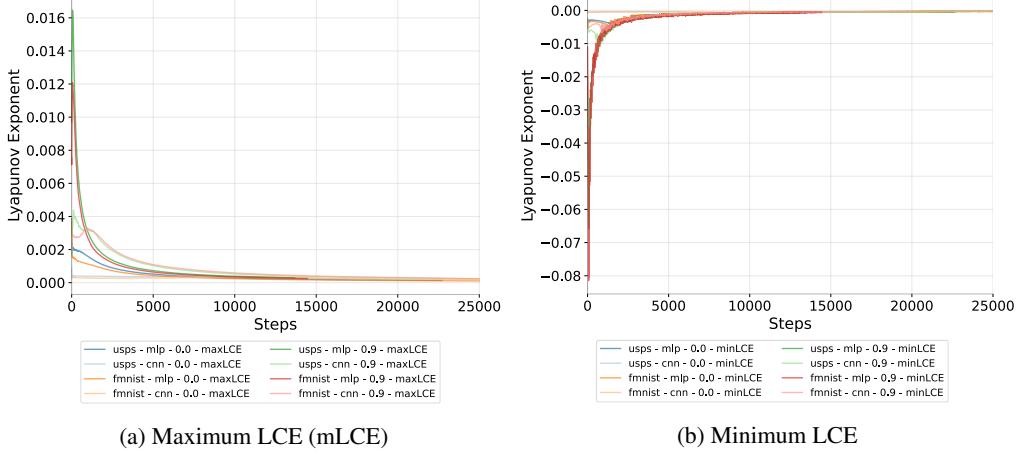


Figure C.1.1: Maximum (a) and minimum (b) Lyapunov characteristic exponents (LCE) of the finite-time Lyapunov matrix at every time step for different runs with SGD. Across all run configurations, we observe a quick drop of the maximum LCEs to 0, suggesting the dynamics start out chaotic, but converge against edge-chaotic as the training advances. Interestingly, the minimum LCEs increase, suggesting an overall transition of the system to edge-chaotic behaviour. Although not visible in this plot, the minimum LCE is less stable than the maximum LCE, since the minimum eigenvalues of the Lyapunov matrix (before applying the natural logarithm) occasionally become 0, meaning the Lyapunov exponent would be  $-\infty$ . The respective points are not visible in the plot.

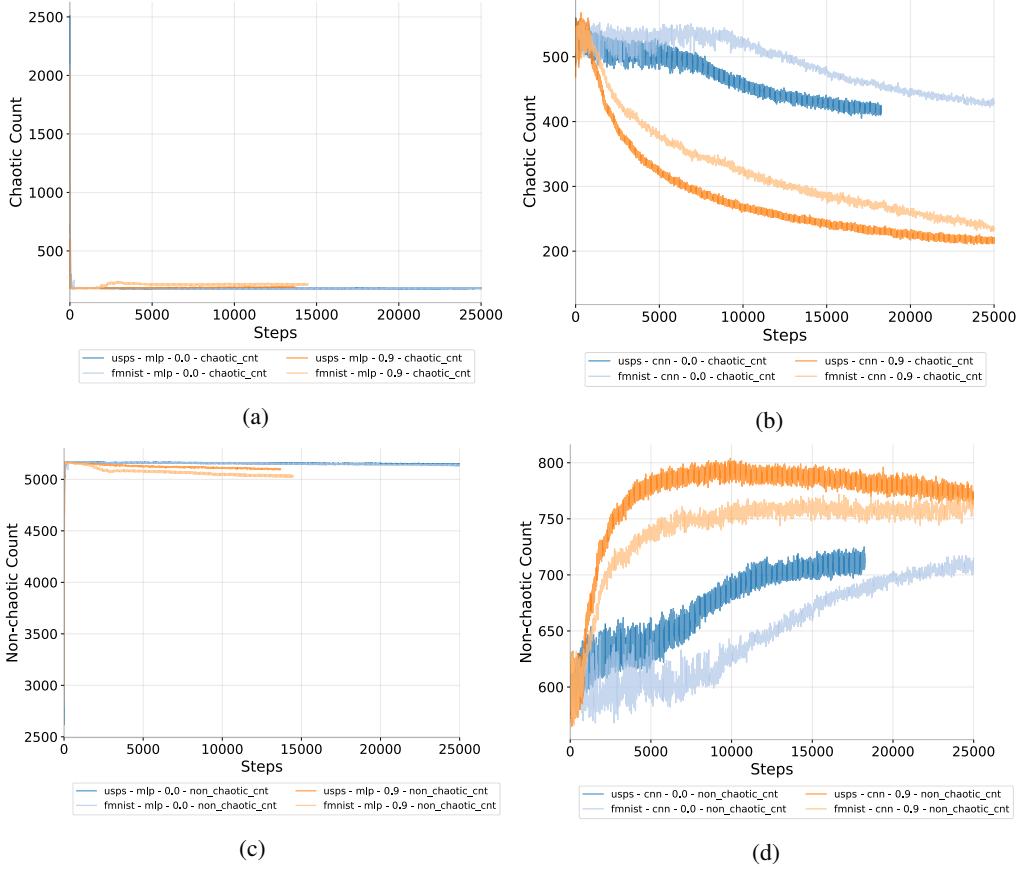


Figure C.1.2: Chaotic (a)/(b) and non-chaotic count (c)/(d) for MLP (a)/(c) and CNN (b)/(d). The counts are defined as  $\dim(V_{\text{chaotic}}(t))$  and  $\dim(V_{\text{non-chaotic}}(t))$ , respectively. The chaotic count for the MLP drops quickly from a high value to a low value and stays almost constant afterwards. For the CNN the chaotic count drops more slowly. All of this can be said for the non-chaotic count vice-versa as the non-chaotic count is  $D - \text{chaotic\_count}$  where  $D$  is the total number of parameters of the model. All different plots back up the thesis that training is initially chaotic but quickly/slowly stabilizes in a less chaotic regime over time.

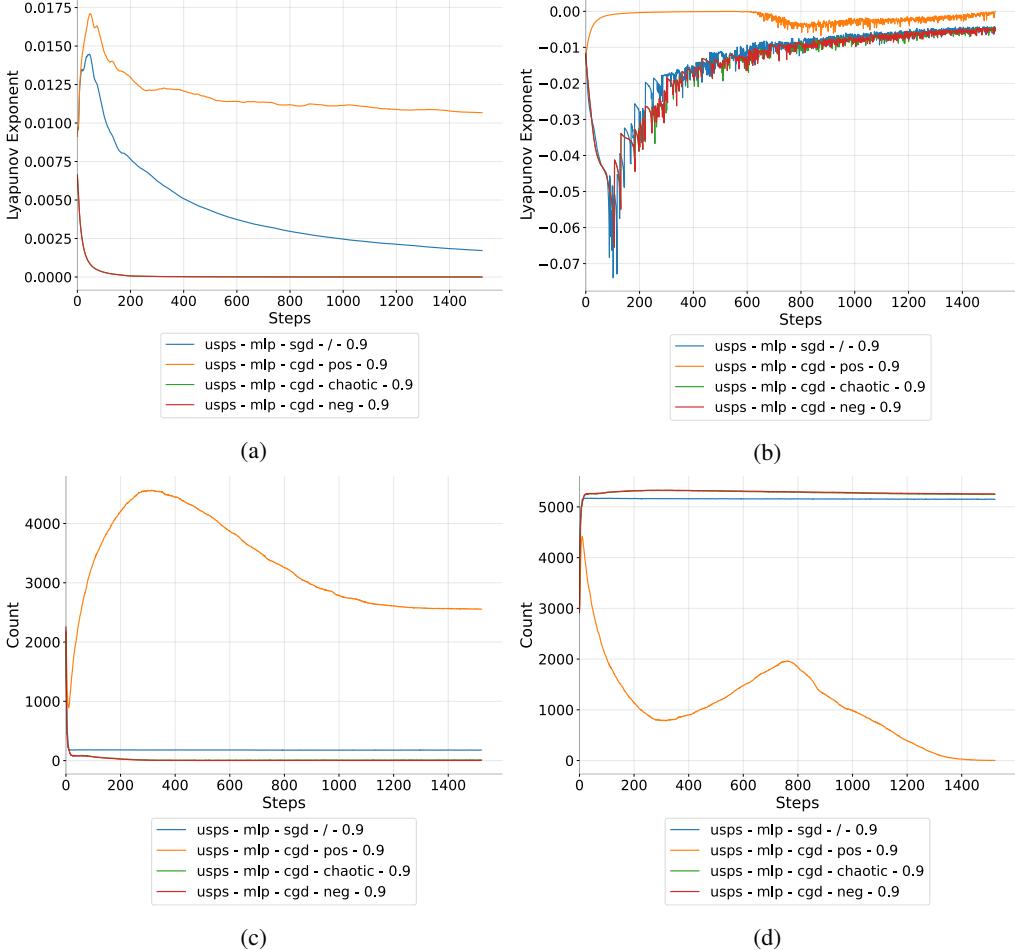


Figure C.1.3: Maximum (a) and minimum (b) Lyapunov exponent for an MLP trained on the USPS dataset with momentum (0.9). Even with momentum, CGD is effective at quickly dampening the maximum Lyapunov exponent to values close to 0 for chaotic and negative pruning. Without pruning, the Lyapunov exponent also converges to 0, but at a slower pace. The minimum Lyapunov exponents converge against values close to 0, indicating the chaotic dynamics overall converge against edge-chaotic behaviour. This is also reflected by count of (c) chaotic and (d) unchaotic eigenvalues. The number of chaotic eigenvalues quickly drops in regular SGD and then stabilizes at a low (but nonzero) level. For positive pruning, the number of nonchaotic eigenvalues drops to 0, causing the training to become unstable.

677 **C.2 Subspace Similarity of Chaotic spaces**

678 In this section we provide an argument for why it is sound to talk about global chaos when we  
 679 approximate the Lyapunov eigenspace with the eigenspaces  $V_{\text{chaotic}}(T_{\text{conv}})$  and  $V_{\text{non-chaotic}}(T_{\text{conv}})$  until  
 680 training convergence  $T_{\text{conv}}$ .

681 Figure C.2.1 shows that at time point  $T_{\text{conv}}$  the chaotic subspace and thus due to orthogonality  
 682  $V_{\text{chaotic}}(t) \perp V_{\text{non-chaotic}}(t)$  the space  $V_{\text{non-chaotic}}(t)$  stabilize over time. Where for fixing early time  
 683 steps  $t_0$ , the similarity stabilizes in a lower regime instead of dropping towards randomness, for  
 684 later time steps  $t_0 = T_{\text{conv}}$  the similarity stays close to one. The rate of change should be lower for  
 685 even later time steps, which is when we saved the eigenspaces  $V_{\text{chaotic}}(T_{\text{end}}), V_{\text{non-chaotic}}(T_{\text{end}})$  for the  
 686 different perturbation strategies. Although for some experiments, e.g. MLP with momentum on  
 687 FashionMNIST, one can argue that longer training is needed as the subspace similarity still drops  
 688 significantly.

689 For comparison, we also estimated the overlap of two random subspaces of the same size as  $V_{\text{chaotic}}(t_0)$   
 690 and  $V_{\text{chaotic}}(t)$  and averaged it over all different runs: Let  $d_{(0,i)} = \dim(V_{\text{chaotic}}(t_0, \theta_i))$  and  $d_{(t,i)} =$   
 691  $\dim(V_{\text{chaotic}}(t, \theta_i))$  for different experiments  $i = 1, \dots, 8 =: N$ . The baseline is defined as

$$\frac{1}{N} \sum_{i=1}^N \mathbb{E}_{V_{(0,i)}, V_{(t,i)}} [\text{overlap}(V_{(0,i)}, V_{(t,i)})] \quad (66)$$

692 where  $\dim(V_{(0,i)}) = d_{(0,i)}$  and  $\dim(V_{(t,i)}) = d_{(t,i)}$ .

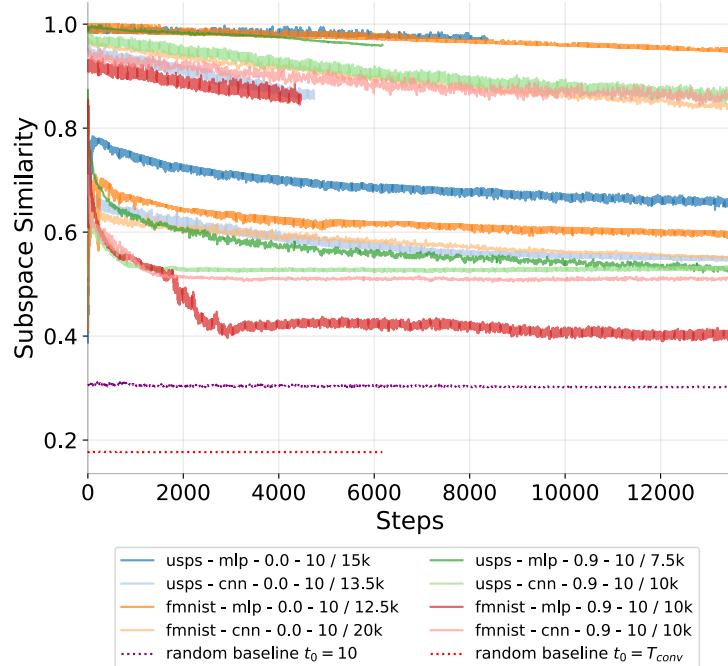


Figure C.2.1: Subspace similarities measured for the same configuration of models (CNN, MLP) and datasets (USPS, FashionMNIST) with and without momentum as we used in our other experiments. We fix a chaotic subspace  $V_{\text{chaotic}}(t_0)$  (see section 4.2) at the start  $t_0 = 10$  and the end of training  $t_0 = T_{\text{conv}}$ , e.g. indicated with  $10 / 10k$ , and measure the overlap between the fixed subspace and following chaotic subspaces  $V_{\text{chaotic}}(t)$  for  $t > t_0$ . The two plots are visualized with the same color for each model/dataset configuration. We modify the calculation of the metric in order to calculate the similarities between subspaces of different dimensionalities (see A.12). We also plot a random baseline for comparison which is defined as the mean of the expected overlap between random subspaces of the same dimensionality (see equation (66)).

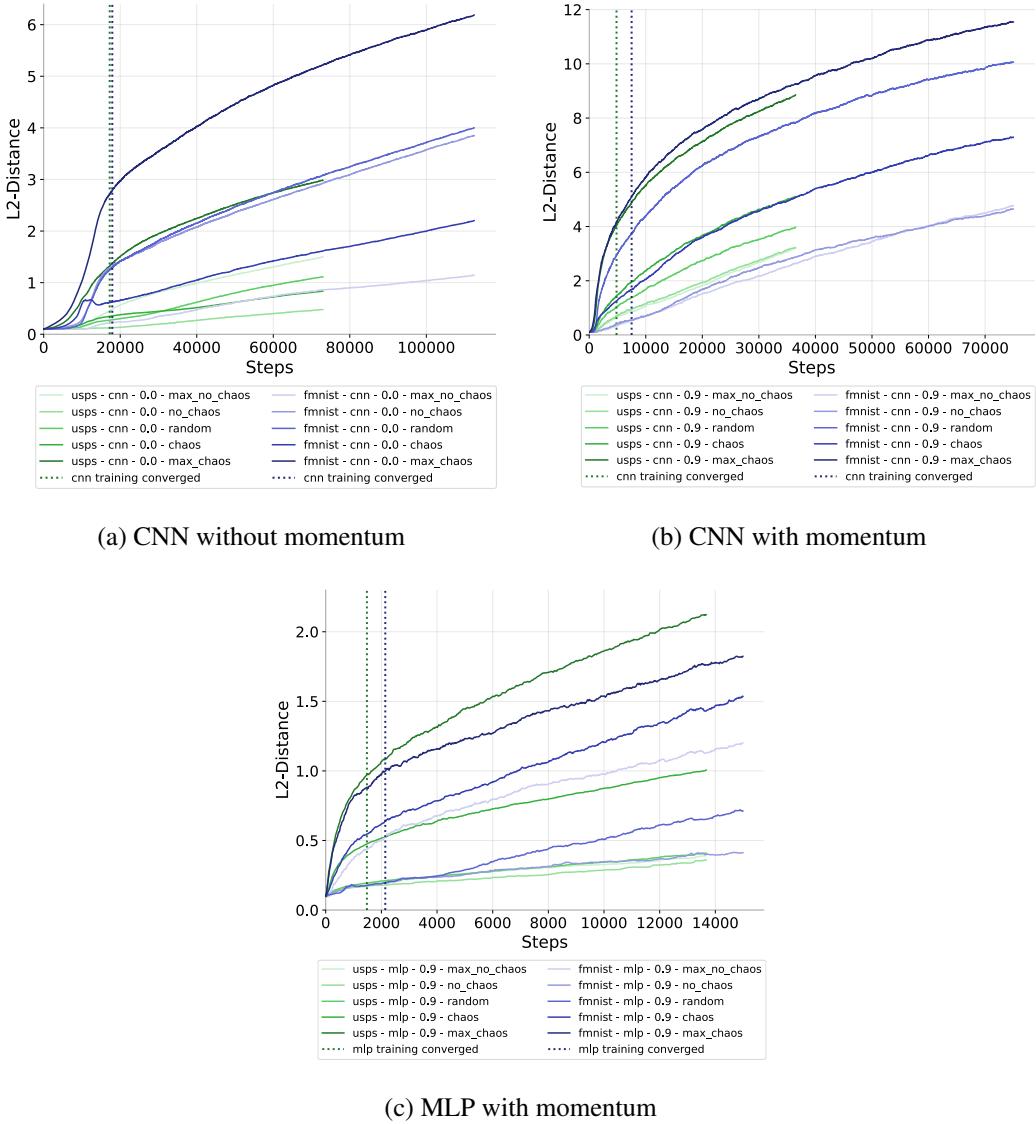


Figure C.3.1: Further distance experiments of Figure 3(a) with a perturbation magnitude of 0.1. We only see minor differences between datasets so it seems that chaotic behavior is more affected by the model choice and general training parameters. In fact, the CNN model exhibits more global chaotic behavior while having less parameters than the MLP, but more experiments have to be conducted to gather a larger sample size to verify this observation. The models trained with momentum have a greater distance divergence than without momentum which could be explained by the process carrying along energy. This could lead to bigger gradients when entering the valley of the local minimum.

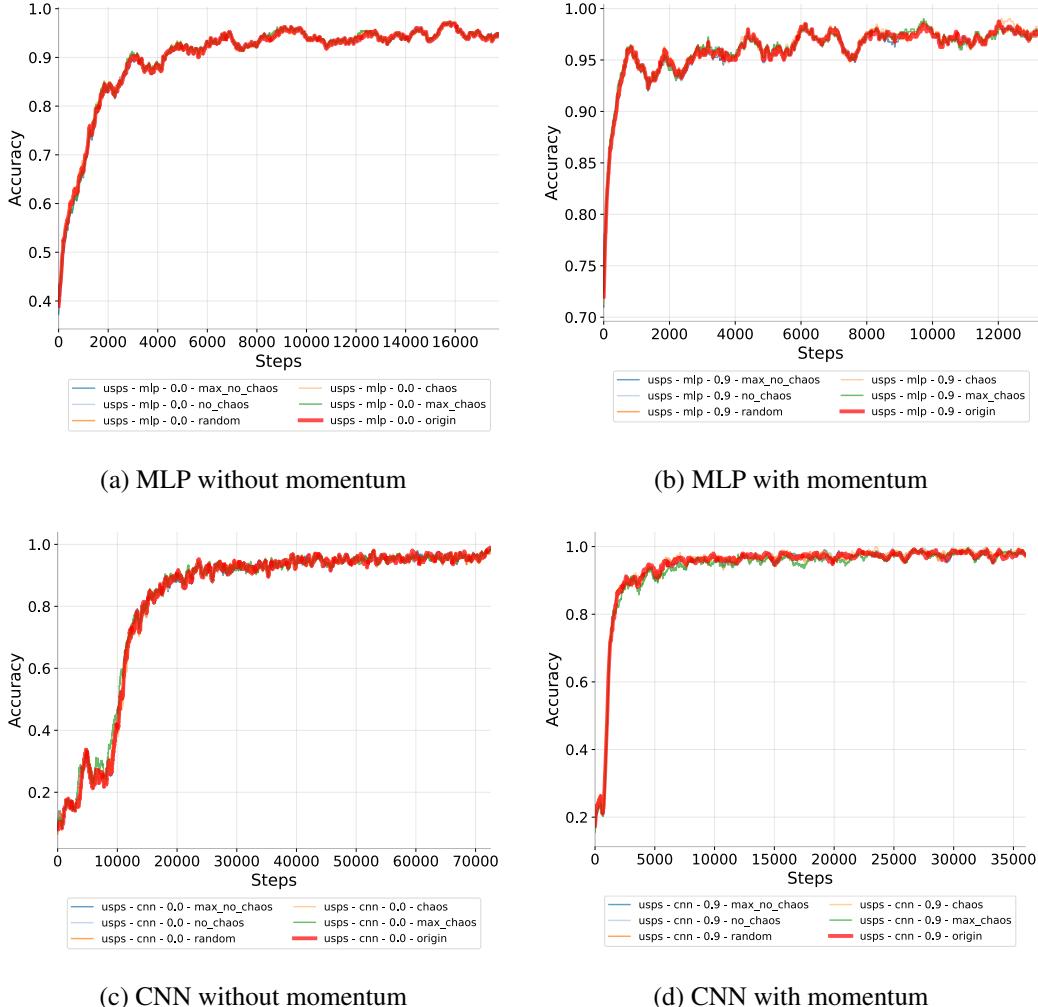


Figure C.3.2: Averaged accuracy (moving average over the last 500 steps) of runs with a perturbation magnitude of  $\varepsilon = 0.1$  over the different strategies. We see no significant difference between the models with different perturbation strategies compared to the original model.

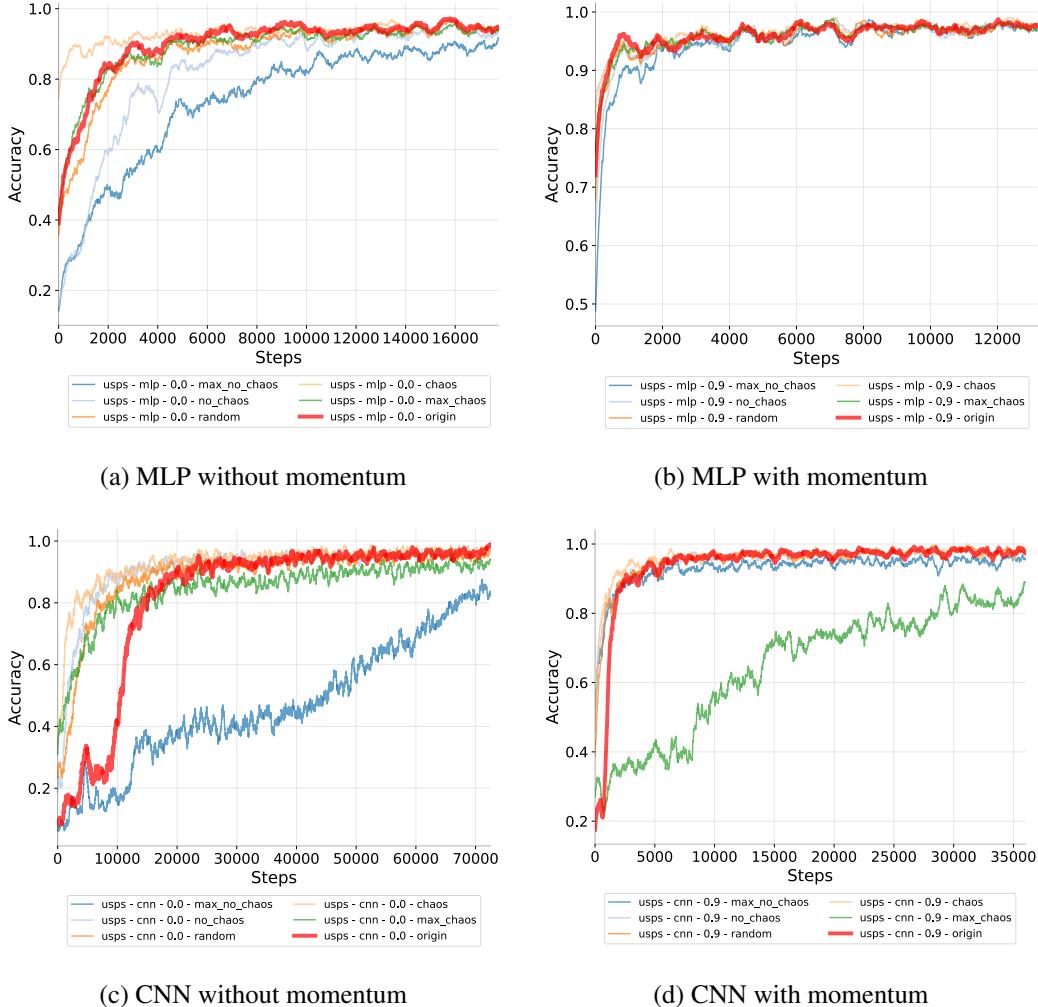


Figure C.3.3: Averaged accuracy (moving average over the last 500 steps) of runs with a perturbation magnitude of  $\varepsilon = 10$  over the different strategies. They generally perform slightly worse with the exception of the chaotically perturbed model. Surprisingly, we see that this model consistently outperforms the original model, particularly in the beginning of training. This could be connected to what we find for the pruning experiments: Chaos seems to be beneficial especially at the start of the training. Also, the maximal non chaotic perturbation always seems to lead to a worse accuracy than all other strategies with the exception of (d). A further analysis has to be provided to better understand both phenomena.

#### 694 C.4 Choice of Perturbation Magnitude

695 We chose to discuss the experiments with magnitude  $\varepsilon = 0.1$ , because it shows the expected behavior  
 696 over all runs we tracked. So this can be said to be cherry picked. Generally, we expect the maximal  
 697 chaotic perturbation to have the largest distance, and then chaotic, random, non-chaotic and maximal  
 698 non-chaotic to have smaller distances in that order. For bigger epsilon, we argue that locality breaks  
 699 and the error of approximation with the Lyapunov matrix gets too high. Surprisingly, we do not find  
 700 to have the expected order of the perturbation strategies for smaller values of  $\varepsilon$ , as well. For these  
 701 cases, we see that the trajectories become more noisy. In fact, this noise seems to be larger than the  
 702 perturbation magnitudes such that the trajectories end up in random locations in the range of what  
 703 we see in Figure [3]. We calculated the absolute mean positional divergence as a measure of order  
 704 for every distance over all experiments and show that smaller and higher choices of  $\varepsilon$  become more  
 705 random the farther away they are from 0.1 (see Figure [C.4.1]). What causes the noise is still not clear

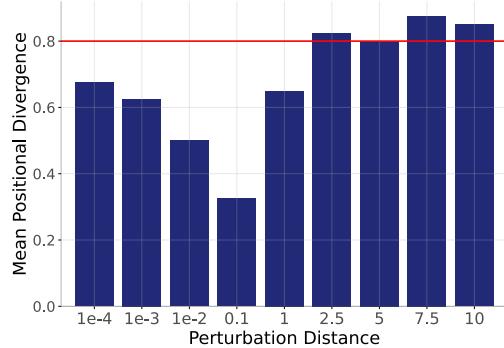


Figure C.4.1: The mean positional divergence:  $1/(2N) \sum_{i=1}^N |\text{pos}_{\text{strat}(i)} - i|$  over all  $N = 5$  different strategies. We expect the strategies to be ordered in the following way: maximal chaotic, chaotic, random, non-chaotic, maximal non-chaotic (from highest divergence to lowest). The red line marks the expected value for a randomly chosen order.

706 to us as it cannot be caused by SGD due to the fixed random seed. It is probably a numerical error  
 707 and to understand this phenomenon additional research is needed.

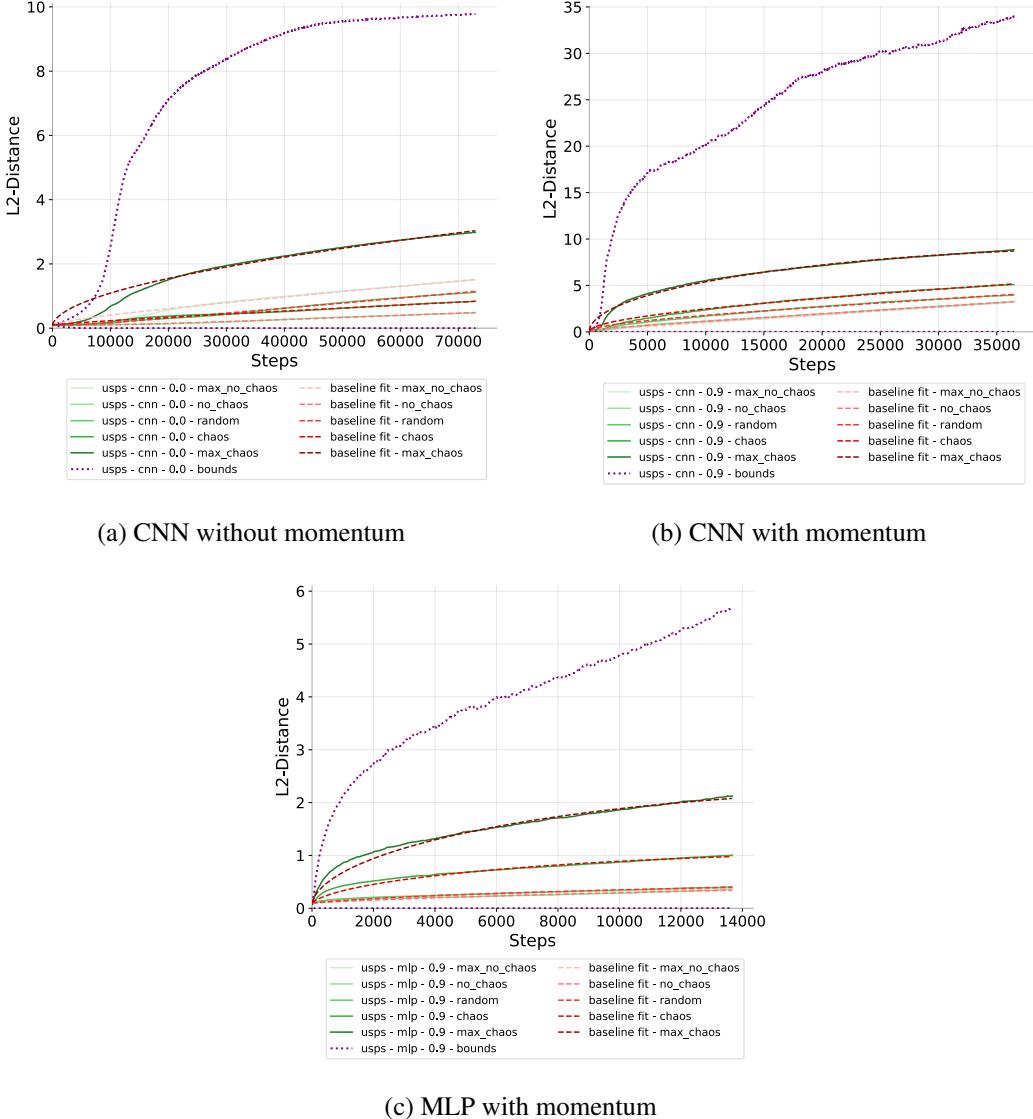


Figure C.5.1: Further theoretical distance experiments of Figure 3(b) with a perturbation magnitude of  $\varepsilon = 0.1$ . This experiment shows that the bounds hold in practise even when they are not tight which leaves two possible options: Either the first order approximation that defines the bound (see equation 5) is not precise for  $\varepsilon = 0.1$  or we were not able to determine the exact direction of maximal/minimal global chaos due to a numerical error (see section C.4).