A Proofs

Tangent map of SGD

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

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

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

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

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

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

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

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

where δ_{ij} is the Kronecker delta and $\boldsymbol{H}$ denotes the Hessian of the loss w.r.t. network parameters and we are done. □

Tangent map of SGD with momentum

Theorem A.2. *Consider a neural network evolving according to SGD with learning rate γ and momentum β as*

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

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

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

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

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

Induction step assuming equation holds true for time step t:

$$\begin{aligned}
\boldsymbol{\theta}^{(t+1)} &= \theta^{(t)} + \boldsymbol{v}^{(t)} && (23)\\
&= \boldsymbol{\theta}^{(t)} - \gamma \boldsymbol{g}(\boldsymbol{\theta}^{(t)}) + \beta \boldsymbol{v}(\boldsymbol{\theta}^{(t-1)}) && (24)\\
&= \boldsymbol{\theta}^{(t)} - \gamma \boldsymbol{g}(\boldsymbol{\theta}^{(t)}) + \beta \cdot (-\gamma) \sum_{s=0}^{t-1} \beta^{(t-1-s)} \boldsymbol{g}(\boldsymbol{\theta}^{(s)}) && (25)\\
&= \boldsymbol{\theta}^{(t)} - \gamma \boldsymbol{g}(\theta^{(t)}) - \gamma \sum_{s=0}^{t-1} \beta^{(t-s)} \boldsymbol{g}(\boldsymbol{\theta}^{(s)}) && (26)\\
&= \boldsymbol{\theta}^{(t)} - \gamma \sum_{s=0}^{t} \beta^{t-s} \boldsymbol{g}(\boldsymbol{\theta}^{(s)}) && (27)
\end{aligned}$$

□

Theorem A.3. *Consider a neural network evolving according to SGD with learning rate γ and momentum β as*

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

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

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

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

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

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

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

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

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

□

Hessian eigenvalues and Local Chaos

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

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

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

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

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

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 eigenvalues of $\ln \mathbf{A}$ and we are done. □

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

$$\lambda_i \leq 0 \vee \lambda_i \geq \frac{2}{\gamma} \tag{35}$$

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

$$\boldsymbol{\Lambda}_x^{(1)} = \frac{1}{2}\ln(\boldsymbol{I} - \gamma\boldsymbol{H})^T(\boldsymbol{I} - \gamma\boldsymbol{H})$$

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

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

Solving for λ_i gives the desired result and we are done. □

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

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

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

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

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

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

$$\lambda_1 \leq \lim_{t\to\infty}\frac{1}{t}\sum_{s=1}^{t}\lambda_1^{(s)} \tag{39}$$

Proof. As previously established, the mLCE λ_1 is defined as the maximum eigenvalue of $\boldsymbol{\Lambda}^{(\infty)}$, which is precisely the limit of the time-normalized spectral norm of the tangent map:

$$\lambda_1 = \lambda_{\max}\lim_{t\to\infty}\frac{1}{2t}\ln\boldsymbol{Y}^{(t)T}\boldsymbol{Y}^{(t)} = \lim_{t\to\infty}\frac{1}{t}\ln\lambda_{\max}\sqrt{\boldsymbol{Y}^{(t)T}\boldsymbol{Y}^{(t)}} = \lim_{t\to\infty}\frac{1}{t}\ln\|\boldsymbol{Y}^{(t)}\| \tag{40}$$

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

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

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

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

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

□

Theorem A.7. *Given a neural network with Hessian $\boldsymbol{H}^{(t)}$, eigenpairs $(\lambda_i^{(t)}, \boldsymbol{v}_i^{(t)})_{i=1}^n$ of $\Lambda^{t,t+1}$ and gradient updates $\Delta\theta^{(t)}$, let there be $k \leq n$ chaotic eigenpairs, i.e. $\lambda_1 \geq ...\lambda_k > 0$. Suppose all chaotic components are pruned through projection, leading to a modified update vector*

$$\Delta\tilde{\boldsymbol{\theta}}^{(t)} = \boldsymbol{v}^{(t)}(\boldsymbol{v}^{(t)})^T \Delta\boldsymbol{\theta}^{(t)}, \qquad \boldsymbol{v}^{(t)} = \sum_{l>k} \boldsymbol{v}_l^{(t)} \boldsymbol{v}_l^{(t)^T}. \tag{44}$$

Then, for SGD training with learning rate of γ and without momentum, the updated training dynamics are given by

$$\theta^{(t+1)} = \boldsymbol{\theta}^{(t)} - \gamma \boldsymbol{v}^{(t)} \boldsymbol{v}^{(t)^T} \boldsymbol{g}(\boldsymbol{\theta}^{(t)}), \qquad \boldsymbol{Y}^{(t+1)} = (\boldsymbol{I} - \gamma \boldsymbol{v}^{(t)} \boldsymbol{v}^{(t)^T}) \boldsymbol{Y}^{(t)} \tag{45}$$

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

Proof. The updated training dynamics are given by

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \Delta\tilde{\boldsymbol{\theta}}^{(t)} = \boldsymbol{\theta}^{(t)} - \boldsymbol{v}^{(t)} \boldsymbol{v}^{(t)^T} \Delta\theta^{(t)} = \boldsymbol{\theta}^{(t)} - \gamma \boldsymbol{v}^{(t)} \boldsymbol{v}^{(t)^T} \boldsymbol{g}(\boldsymbol{\theta}^{(t)}, \mathbf{x}^{(t)}, \boldsymbol{Y}^{(t)}). \tag{46}$$

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

$$\frac{\partial \theta_i^{(t+1)}}{\partial \theta_j^{(t)}} = \frac{\partial}{\partial \theta_j^{(t)}} \left(\theta_i^{(t)} - \gamma \sum_k \left\{ \boldsymbol{v}^{(t)} \boldsymbol{v}^{(t)^T} \right\}_{ik} g_k^{(t)}(\boldsymbol{\theta}^{(t)}, \mathbf{x}^{(t)}, \boldsymbol{Y}^{(t)}) \right) \tag{47}$$

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

$$= \delta_{ij} - \gamma \sum_k \left\{ \boldsymbol{v}^{(t)} \boldsymbol{v}^{(t)^T} \right\}_{ik} H_{kj}, \tag{49}$$

or more succinctly in matrix notation

$$\frac{\partial \theta^{(t+1)}}{\partial \theta^{(t)}} = \boldsymbol{I} - \gamma \boldsymbol{v}^{(t)} \boldsymbol{v}^{(t)^{(T)}} \boldsymbol{H}^{(t)}. \tag{50}$$

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

$$(\boldsymbol{I} - \gamma \boldsymbol{v}^{(t)} \boldsymbol{v}^{(t)^T} \boldsymbol{H}^{(t)}) \boldsymbol{v}_i^{(t)} = \boldsymbol{v}_i^{(t)} - \gamma \boldsymbol{v}^{(t)} \boldsymbol{v}^{(t)^T} \lambda_i \boldsymbol{v}_i^{(t)} \tag{51}$$

$$= \boldsymbol{v}_i^{(t)} - \gamma \lambda_i \underbrace{\boldsymbol{v}^{(t)} \boldsymbol{v}^{(t)^T} \boldsymbol{v}_i}_{=0} \tag{52}$$

$$= \boldsymbol{v}_i^{(t)} \tag{53}$$

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

$$\boldsymbol{v}^{(t)} \boldsymbol{v}^{(t)^T} \boldsymbol{v}_i^{(t)} = \boldsymbol{v}_i^{(t)} \tag{54}$$

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

$$\frac{\partial \boldsymbol{\theta}^{(t+1)}}{\partial \boldsymbol{\theta}^{(t)}} \boldsymbol{v}_i = \begin{cases} 1\boldsymbol{v}_i, & k \leq i \\ (1-\lambda_i)\boldsymbol{v}_i, & 0 \leq i < k \end{cases} \tag{55}$$

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

$$\lambda_1 \leq \lim_{t\to\infty} \frac{1}{t} \sum_{s=0}^{t} \lambda_1^{(s)} \leq \lim_{t\to\infty} \frac{1}{t} \sum_{s=0}^{t} 0 = 0. \tag{56}$$

Hence, the trajectories diverge at most polynomially and we are done. □

Distance Statements

Theorem A.8. *Two standard initialized 784-20-10 MLPs in Pytorch have a expected L2-distance of* ≈ 6.38.

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

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

1. case: weight

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

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

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

2. case: bias

Analogously for the bias where the dimension of the bias is not km but m, we end up with $\mathbb{E}[\|\boldsymbol{b}^{(1)} - \boldsymbol{b}^{(2)}\|_2] = \sqrt{\frac{4m}{3k}}$.

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

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

□

Theorem A.9. *Two random Gaussian walks* $\boldsymbol{x_1}(t), \boldsymbol{x_2}(t) = \boldsymbol{x_i}(t-1) + \boldsymbol{\eta_i}(t)$ *with* $\boldsymbol{x_1}(0), \boldsymbol{x_2}(0) = \boldsymbol{0}$ *and* $\boldsymbol{\eta_1}(t), \boldsymbol{\eta_2}(t) \in \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})$ *have an expected distance of* $\sqrt{2TD}\sigma$ *until time step* T.

Proof.

$$\begin{aligned} \mathbb{E}\left[\|\boldsymbol{x}_1(t) - \boldsymbol{x}_2(t)\|_2\right] &= \mathop{\mathbb{E}}_{\boldsymbol{\eta}_1(t), \boldsymbol{\eta}_2(t) \in \mathcal{N}(\boldsymbol{0}, \sigma^2 \boldsymbol{I})} \left[\| \sum_{t=1}^{T} \boldsymbol{\eta}_1(t) - \boldsymbol{\eta}_2(t) \|_2 \right] \\ &\overset{(1)}{=} \mathop{\mathbb{E}}_{\boldsymbol{\eta}'(t) \in \mathcal{N}(0, \sigma^2 \boldsymbol{I})} \left[\| \sum_{t=1}^{2T} \boldsymbol{\eta}'(t) \|_2 \right] \overset{(2)}{=} \mathop{\mathbb{E}}_{\boldsymbol{\eta}'(t) \in \mathcal{N}(0, 2T\sigma^2 \boldsymbol{I})} [\|\tilde{\boldsymbol{\eta}}(t)\|_2] \\ &\overset{(3)}{=} \sqrt{2TD}\sigma. \end{aligned} \tag{57}$$

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

$$\mathop{\mathbb{E}}_{\boldsymbol{x} \in \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})} [\|\boldsymbol{x}\|_2^2] = \sum_{i=1}^{D} \mathop{\mathbb{E}}_{x_i \in \mathcal{N}(0, \sigma^2)} [x_i^2] = \mathop{\mathbb{E}}_{x' \in \mathcal{N}(0, D\sigma^2)} [x'^2] = D\sigma^2 \tag{58}$$

□

Theorem A.10. *Let the gradients of two models $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \mathbb{R}^D$ be defined as*

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

where $\boldsymbol{\eta}_1, \boldsymbol{\eta}_2 \in \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$ are independent and identically distributed, isotropic noise terms and $\| \sum_{s=1}^{t} (\boldsymbol{w}_1(s) - \boldsymbol{w}_2(s) \|_2 = \alpha t$ diverges linearly over time. Assume both models are initialized with the same set of parameters $\theta_1^{(0)} = \theta_2^{(0)}$. Then distance of the two models evolves as

$$\mathop{\mathbb{E}}_{\eta_1, \eta_2 \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})} [\|\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2\|_2] = \sqrt{\alpha^2 t^2 + \beta t},$$

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

Proof.

$$\mathop{\mathbb{E}}_{\boldsymbol{\eta}_1, \boldsymbol{\eta}_2 \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})} \left[\left\| \sum_{s=0}^{t} (\boldsymbol{w}_1(s) - \boldsymbol{w}_2(s)) + \sum_{s=0}^{t} (\boldsymbol{\eta}_1(s) - \boldsymbol{\eta}_2(s)) \right\|_2 \right] \tag{59}$$

We want to calculate the term (59) where $\sum_{s=0}^{t} (\boldsymbol{w}_1(s) - \boldsymbol{w}_2(s))$ defines the linear divergence term and $\sum_{s=0}^{t} (\boldsymbol{\eta}_1(s) - \boldsymbol{\eta}_2(s))$ is the random Gaussian walk as in equation (15). For simplification, assume that we take the expected value over $\boldsymbol{\eta}_1(s), \boldsymbol{\eta}_2(s) \in \mathcal{N}(\mathbf{0}, \sigma^2 I)$ in all of the following expression

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

where $\beta := 2D\sigma^2$, which follows from Theorem A.9. Now this leaves to show that the last summand is zero. In fact, observe that for all $v \in \mathbb{R}^D$

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

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

Theorem A.11. *The modified Subspace Similarity defined as*

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

where P_V, P_W are the orthogonal profectors onto the subspaces V, W fulfills the following properties:

1. *If* $V \perp W$*, then overlap*$(V, W) = 0$
2. *If* $V \subseteq W$ *or* $W \subseteq V$*, then overlap*$(V, W) = 1$
3. *For arbitrary subspaces* V, W*, overlap*$(V, W) \in [0, 1]$

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

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

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

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

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

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

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

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

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

Now extend the orthonormal basis of V to a orthonormal basis $B := v_1, ..., v_n, ..., v_k$ of the whole space $V' \supseteq V, W$. And write the orthonormal basis of W in terms of B: $w_j = \sum_{i=1}^{k}\lambda_{ji}v_i$. Then

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

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

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

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

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

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

A.1 Generalization and long-term chaos

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

leads to good generalization should be one that minimizes the variance of the loss of the learned function with respect to data points $x \in S$. Thus, the generalization capacity can be metricized by

$$\text{Var}[\mathcal{L}(f_\theta^M)] = \mathbb{E}_{\boldsymbol{x}\in S}\left[\left(\mathcal{L}(f_\theta^M;\boldsymbol{x}) - \mathbb{E}_{\boldsymbol{x}\in S}\left[\mathcal{L}(f_\theta^M;\boldsymbol{x})\right]\right)^2\right] = \mathbb{E}_{\boldsymbol{x}\in S}\left[\left(\mathcal{L}(f_\theta^M;\boldsymbol{x}) - \bar{\mathcal{L}}\right)^2\right] \tag{60}$$

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

$$\mathcal{L}(f_\theta^M;\boldsymbol{x}_s + \boldsymbol{\delta x}) = \mathcal{L}(f_\theta^M;\boldsymbol{x}_s) + \boldsymbol{\delta x}^T \nabla_{\mathcal{L}} f_\theta^M + \frac{1}{2}\boldsymbol{\delta x}^T \boldsymbol{H}_{\mathcal{L}}(f_\theta^M;\boldsymbol{x}_s)\delta\boldsymbol{x} + ... \tag{61}$$

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

$$\mathcal{L}'(f_\theta^M;\boldsymbol{x}_s) = \frac{d\mathcal{L}}{d\boldsymbol{x}}|\boldsymbol{x} = \boldsymbol{x}_s = \frac{d\mathcal{L}}{d\boldsymbol{\theta}^{(T)}}\frac{d\boldsymbol{\theta}^{(T)}}{d\boldsymbol{\theta}^{(s+1)}}\frac{d\boldsymbol{\theta}^{(s+1)}}{d\boldsymbol{x}_s} = \frac{d\mathcal{L}}{d\boldsymbol{\theta}^{(T)}}\boldsymbol{Y}^{(s+1,T)}\frac{d\boldsymbol{\theta}^{(s+1)}}{d\boldsymbol{x}_s} \tag{62}$$

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

$$\frac{d\boldsymbol{\theta}^{(s+1)}}{d\boldsymbol{x}_s} = \frac{d}{d\boldsymbol{\theta}^{(s)}}(\boldsymbol{\theta}^{(s)} - \gamma\boldsymbol{g}(\boldsymbol{\theta}^{(s)};\boldsymbol{x}_s)) = \boldsymbol{I} - \gamma\frac{\partial^2\mathcal{L}(\boldsymbol{\theta}^{(s)};\boldsymbol{x}_s)}{\partial\boldsymbol{x}_s\partial\boldsymbol{\theta}^{(s)}} = \boldsymbol{I} - \gamma\boldsymbol{H}_{x\theta}^{(s)}, \tag{63}$$

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

$$\text{Var}_{\tilde{S}}[\mathcal{L}(f_\theta^{SGD})] \approx \mathbb{E}_{\boldsymbol{x}\in\tilde{S}}\left[\left(\mathcal{L}(f_\theta^{SGD};\boldsymbol{x}_s) + \frac{d\mathcal{L}}{d\boldsymbol{\theta}^{(T)}}\boldsymbol{Y}^{(s+1,T)}(\boldsymbol{I} - \gamma\boldsymbol{H}_{x\theta}^{(s)})\boldsymbol{\delta x} - \bar{\mathcal{L}}\right)^2\right] \tag{64}$$

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

B Further Analysis on Locally Chaotic Dynamics

B.1 Pruning Experiments: Performance Metrics

USPS

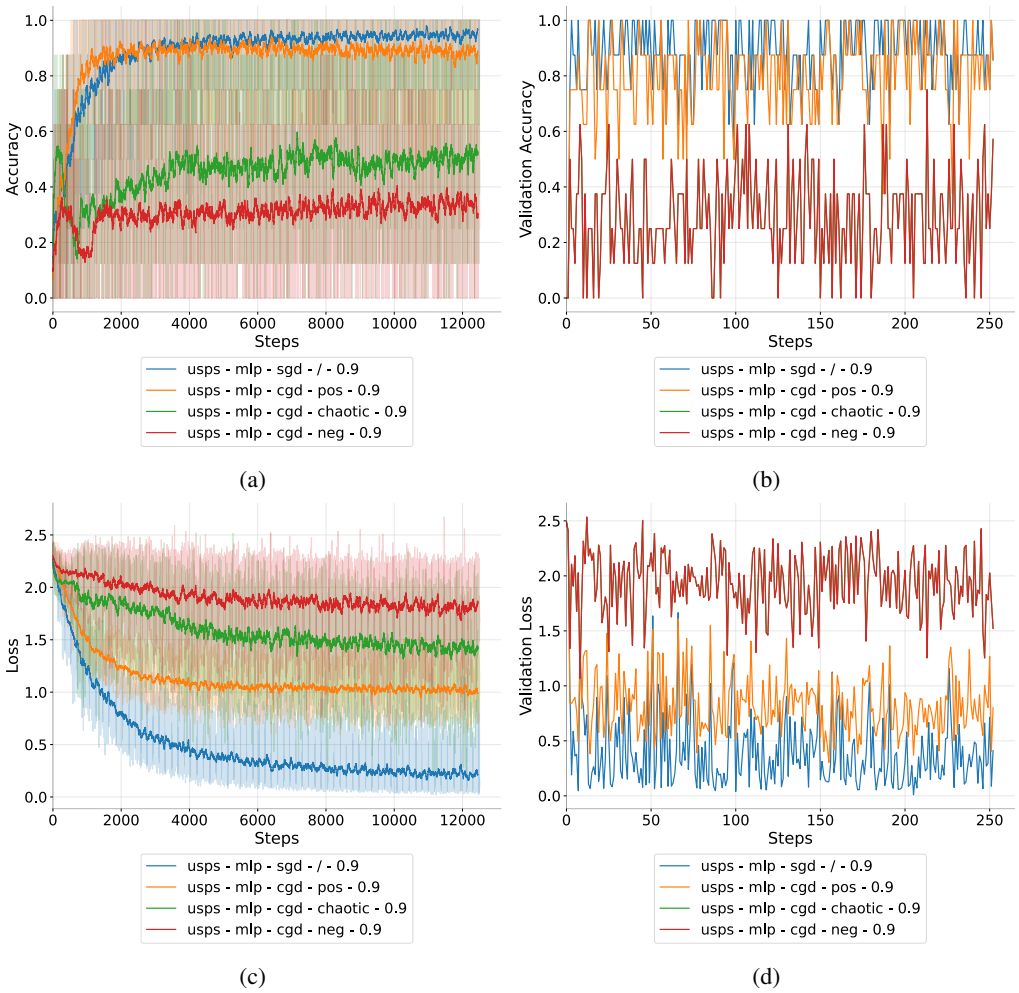


Figure B.1.1: Line plots of the (a) training accuracy and (b) validation accuracy, as well as (c) training loss and (d) validation loss for an MLP (relu activation) trained on the USPS dataset with momentum (0.9). The models trained with CGD use pruning of the full chaotic Hessian spectrum at every time 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.

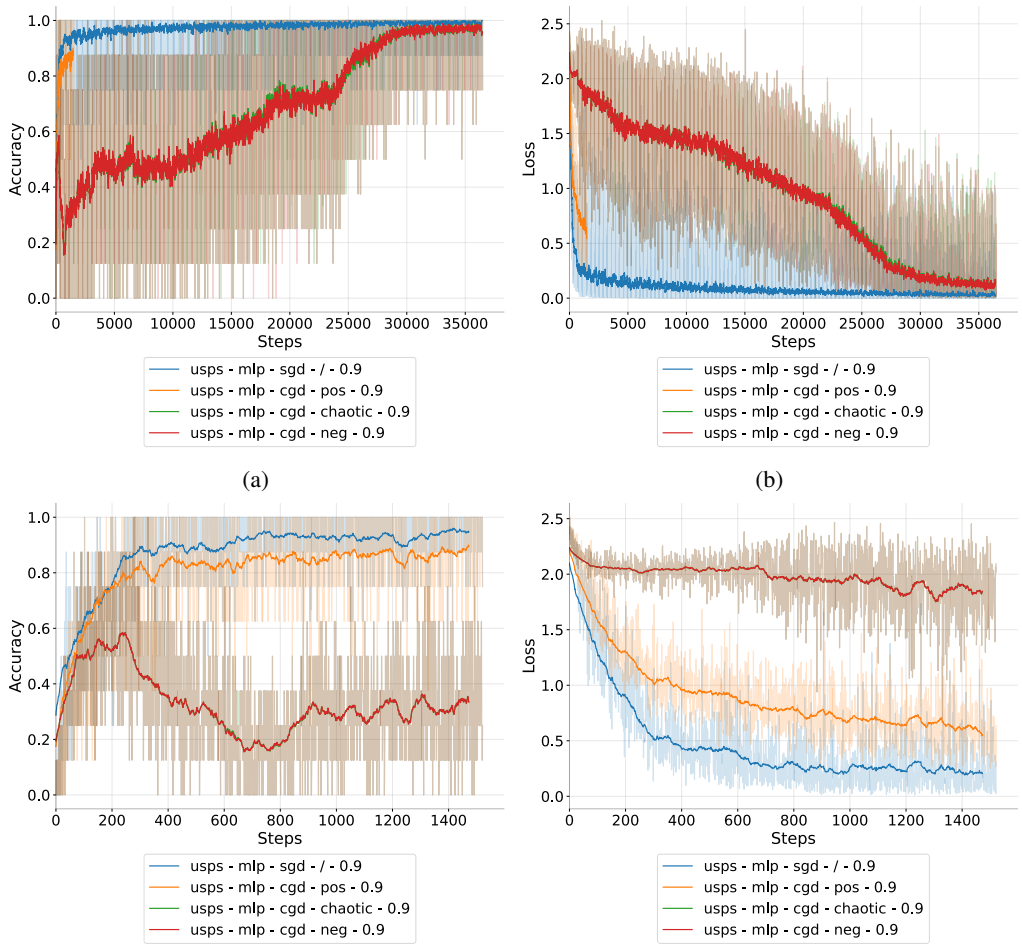


Figure B.1.2: Line plots of the (a)/(c) accuracy and (b)/(d) loss curves for an MLP (relu activation) trained on the USPS dataset with momentum (0.9). The models trained with CGD use pruning of the full chaotic, negative or positive Hessian spectrum at every time step. For greater clarity, the plots 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.

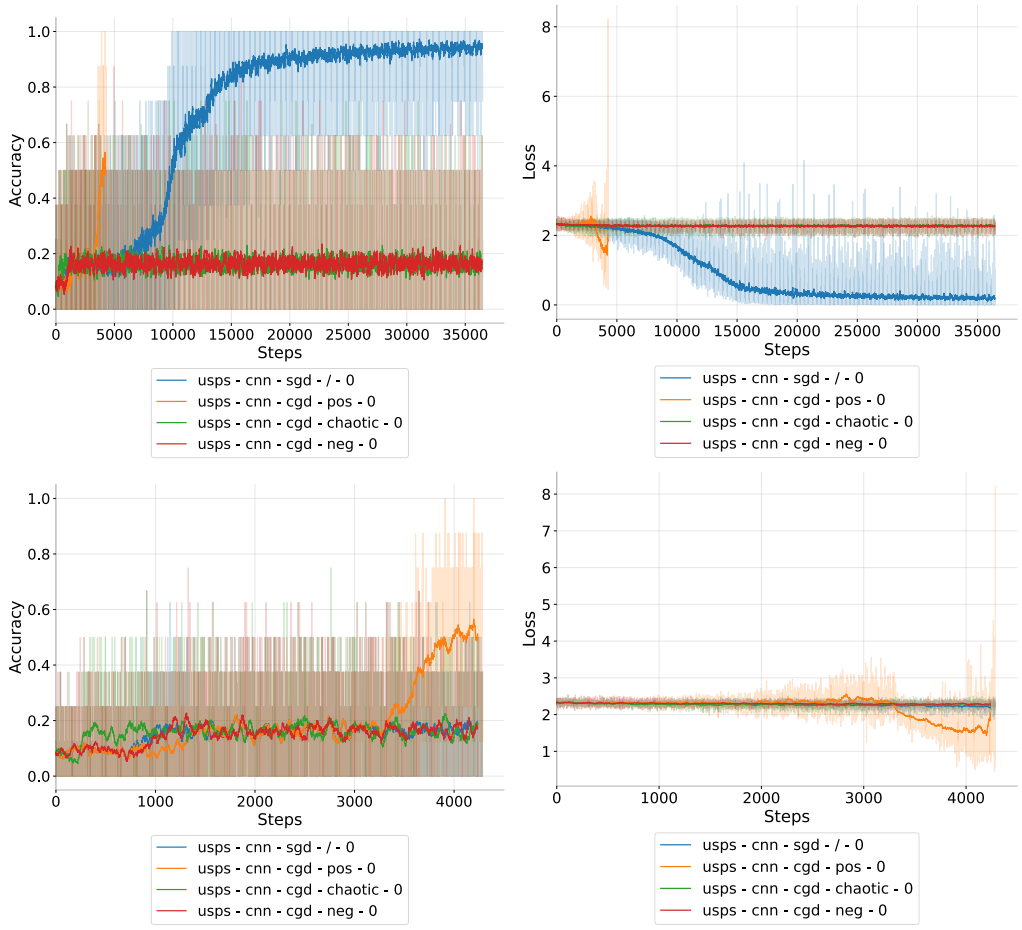


Figure B.1.3: Line plots of the (a)/(c) accuracy and (b)/(d) loss curves for a CNN trained on the USPS dataset without momentum. The models trained with CGD use pruning of the full chaotic, negative or positive Hessian spectrum at every time step. For greater clarity, the plots in the bottom row show 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.

FashionMNIST

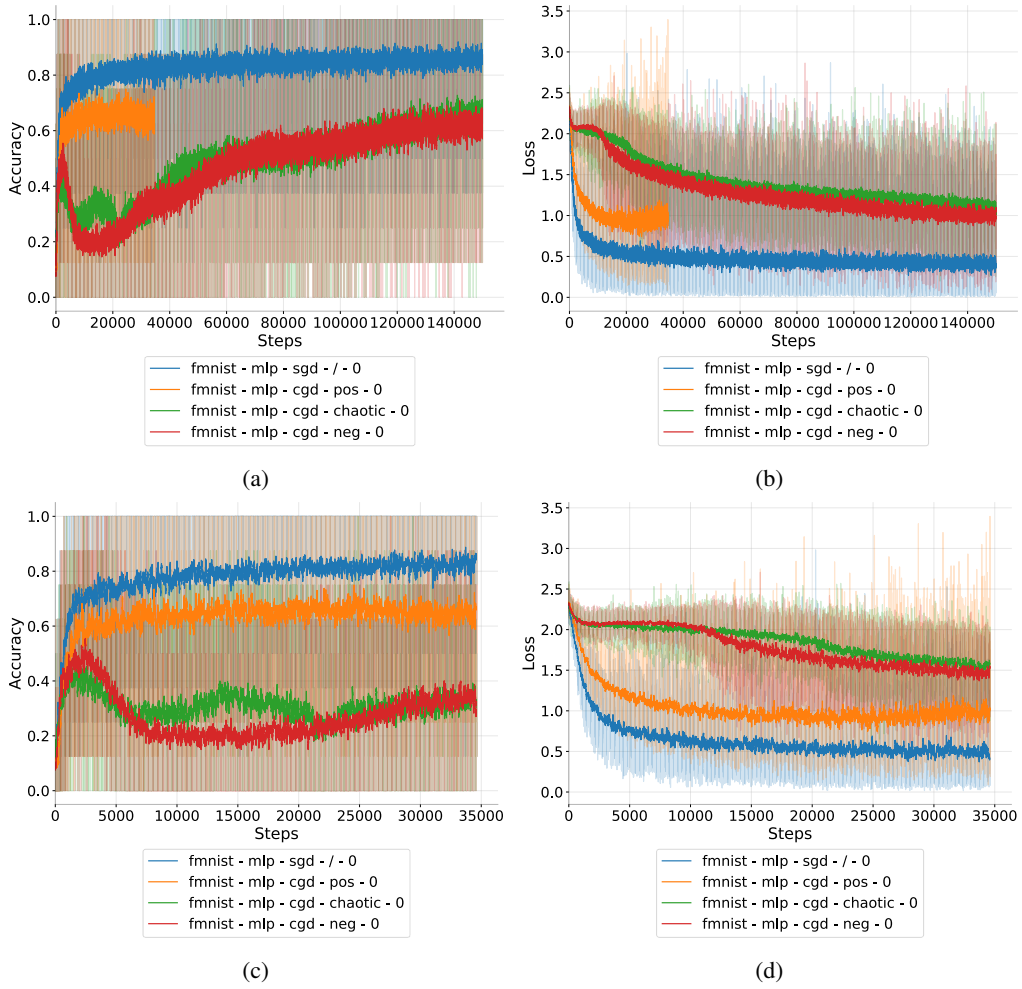


Figure B.1.4: Line plots of the (a)/(c) accuracy and (b)/(d) loss curves for an MLP trained on the FashionMNIST dataset without momentum. The models trained with CGD use pruning of the full chaotic, negative or positive Hessian spectrum at every time step. For greater clarity, the plots in the bottom 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.

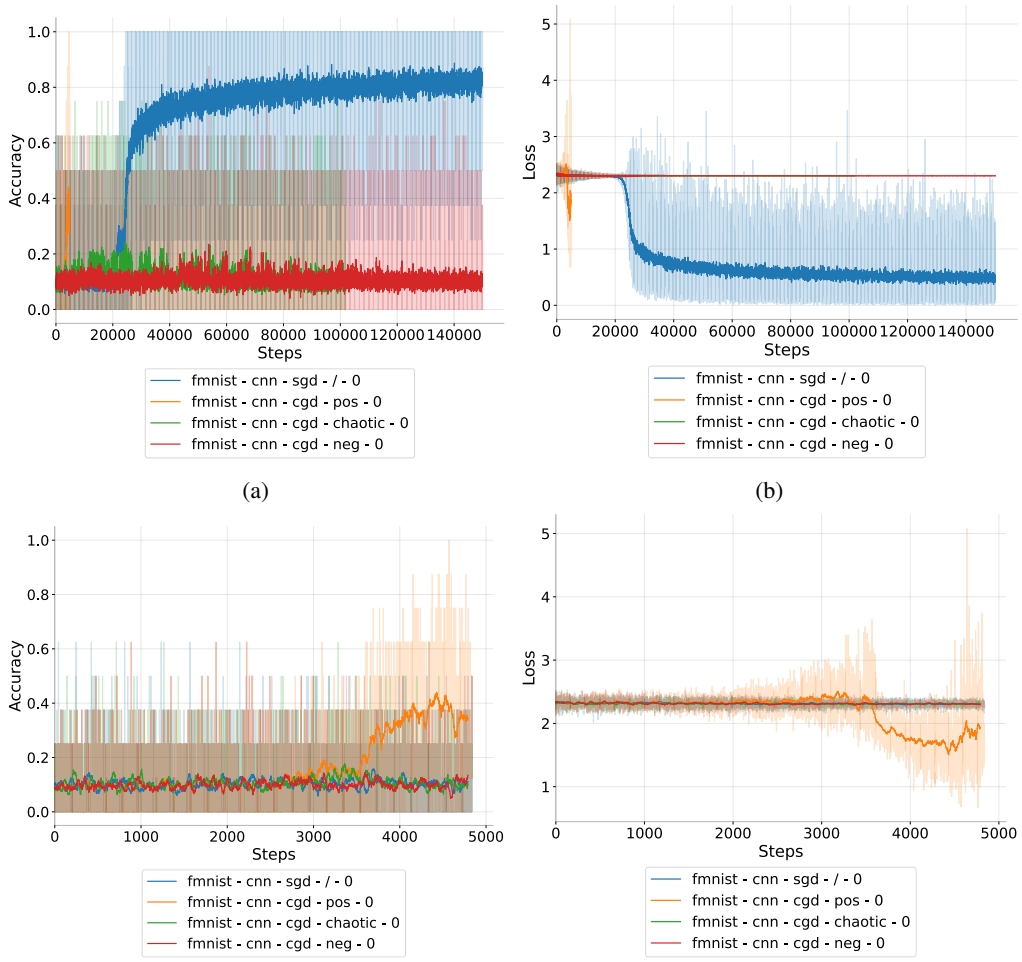


Figure B.1.5: Line plots of the (a)/(c) accuracy and (b)/(d) loss curves for a 2D CNN trained on the FashionMNIST dataset without momentum. The models trained with CGD use pruning of the full chaotic, negative or positive Hessian spectrum at every time step. For greater clarity, the plots in the bottom 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.

B.2 Pruning Experiments: Eigenspaces

USPS

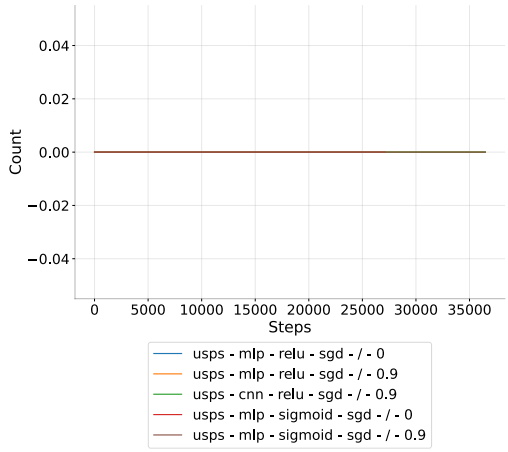


Figure B.2.1: Count of positive eigenvalues of the Hessian that satisfy the chaos criterion $\lambda_i > \frac{2}{\gamma}$ throughout the training, for several runs. The count is effectively zero.

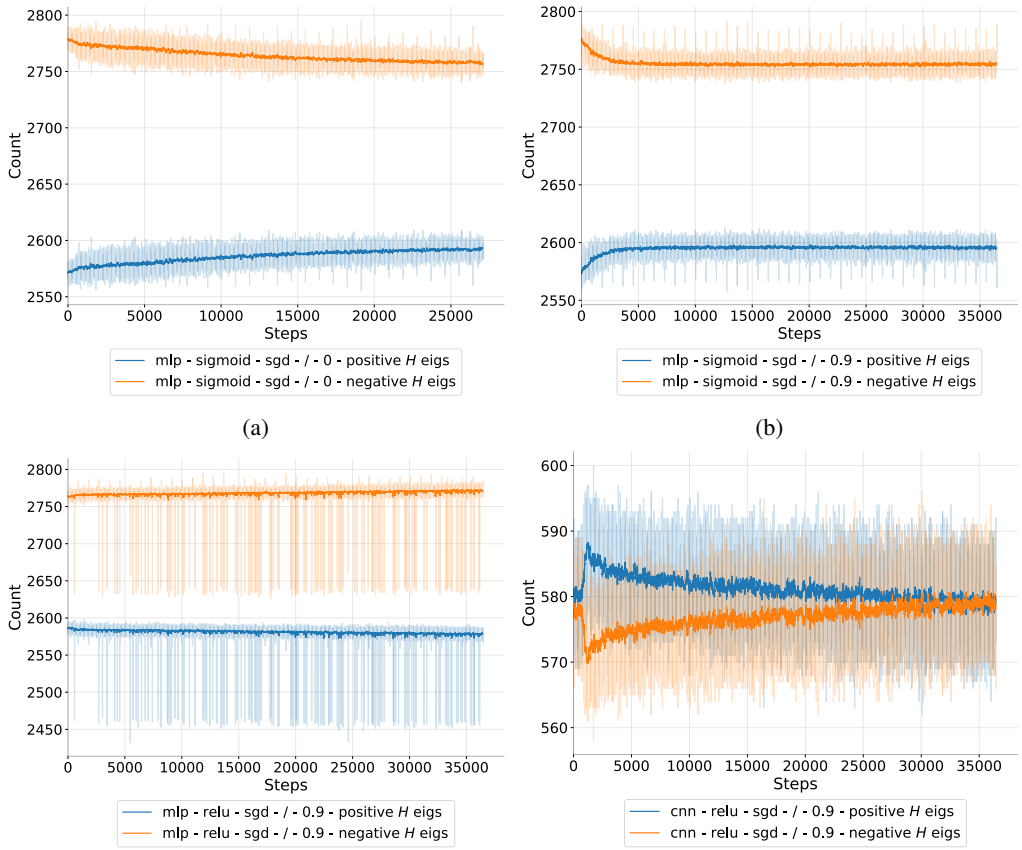
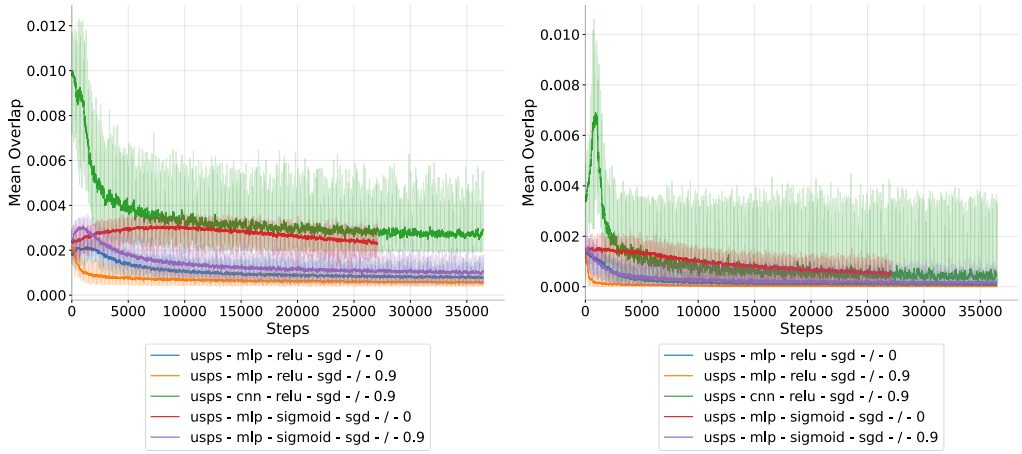


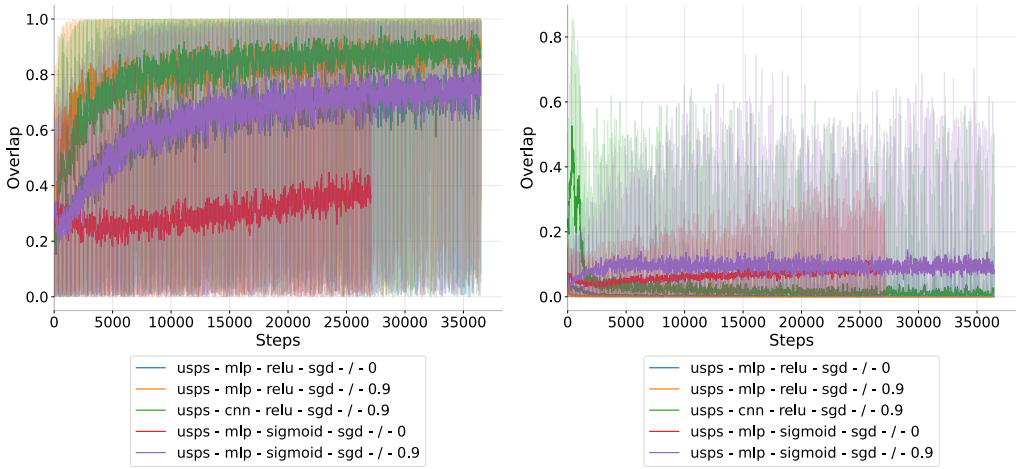
Figure B.2.2: **Number of positive** ($\lambda_i > 0$) and **negative** ($\lambda_i < 0$) eigenvalues of the Hessian for different runs on the USPS dataset. For the MLP models with sigmoid activation (a) and (b), the number of positive eigenvalues increases as training proceeds, while the number of negative 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.



(a) Positive-gradient overlap

(b) Negative-gradient overlap

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.

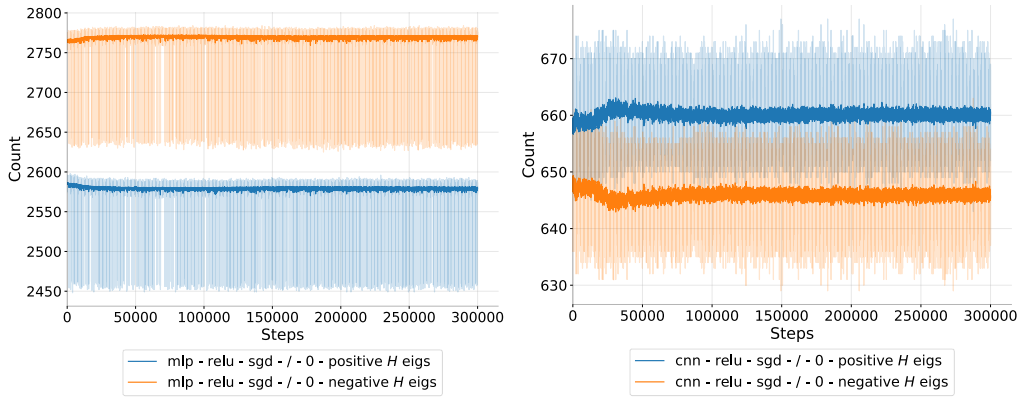


(a) Top-gradient overlap

(b) Bottom-gradient overlap

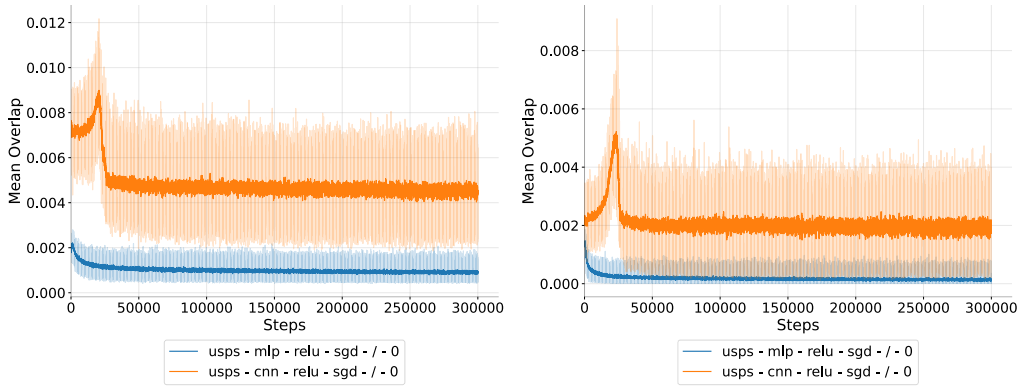
Figure B.2.4: Overlaps between the (normalized) (a) top Hessian eigenvector and the gradient, and (b) the minimum (i.e. most negative) Hessian eigenvector and the gradient, where each curve corresponds to a different run configuration. The overlap between two vectors is measured as absolute cosine similarity. 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 fluctations, 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**.

FashionMNIST



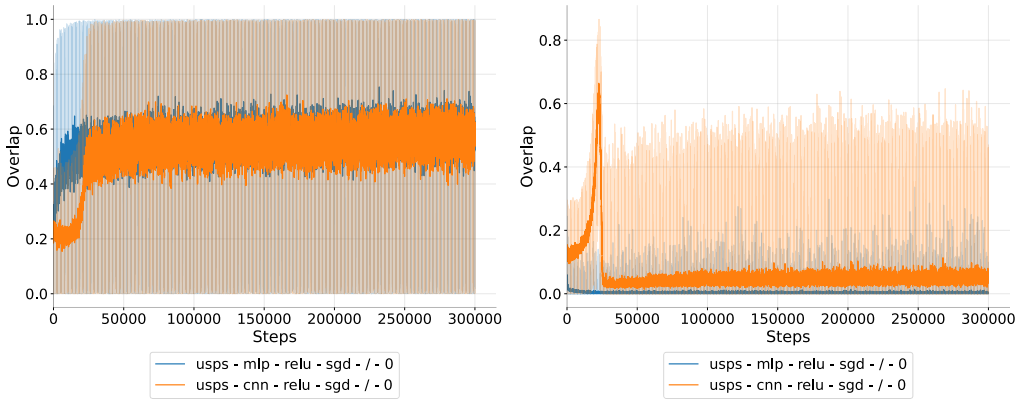
(a) (b)

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 an then relaxes to lower levels. The evolution of the negative eigenvalue count is complementary.



(a) Positive-gradient overlap (b) Negative-gradient overlap

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.



(a) Top-gradient overlap

(b) Bottom-gradient overlap

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 fluctations, 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.

B.3 Additional

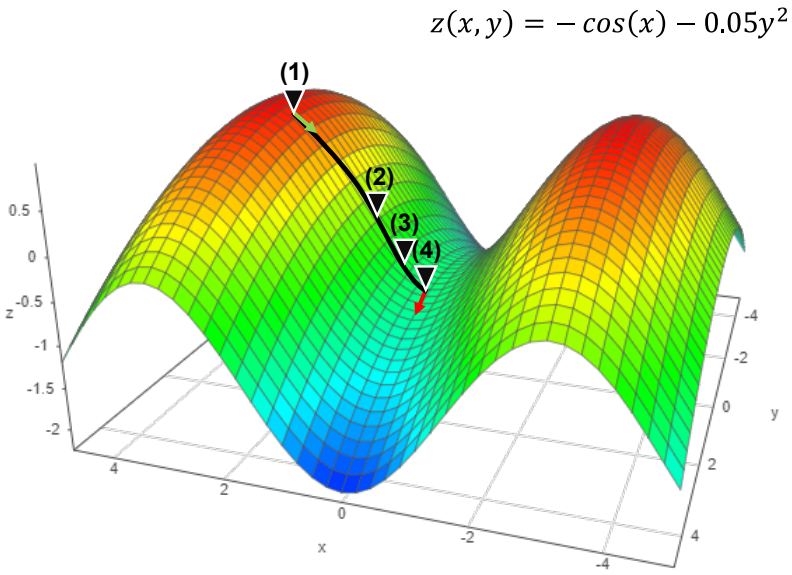
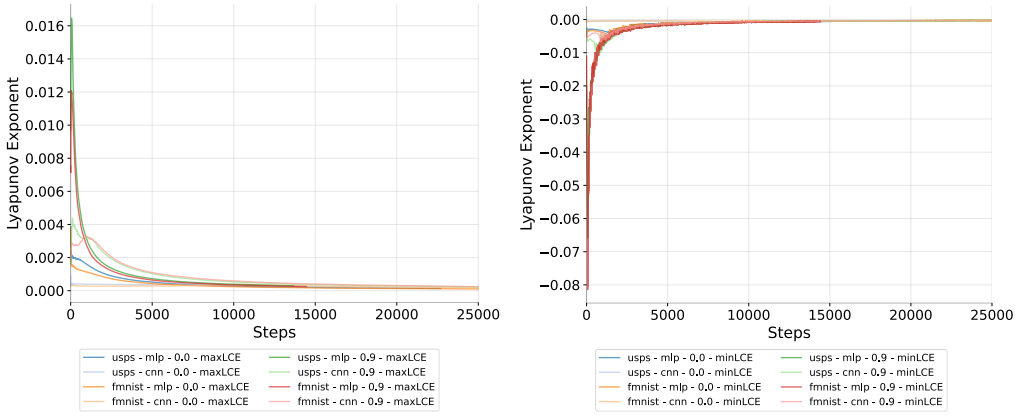


Figure B.3.1: A simplified model of training by SGD illustrating how negative curvature precedes positive curvature: At the beginning of the training, the optimizer is near a high-dimensional local maximum or saddle-point (1), from which the loss can be reduced by updating the parameters in directions 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`.

C Further Analysis on Globally Chaotic Dynamics

C.1 Lyapunov Exponents



(a) Maximum LCE (mLCE)

(b) Minimum LCE

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.

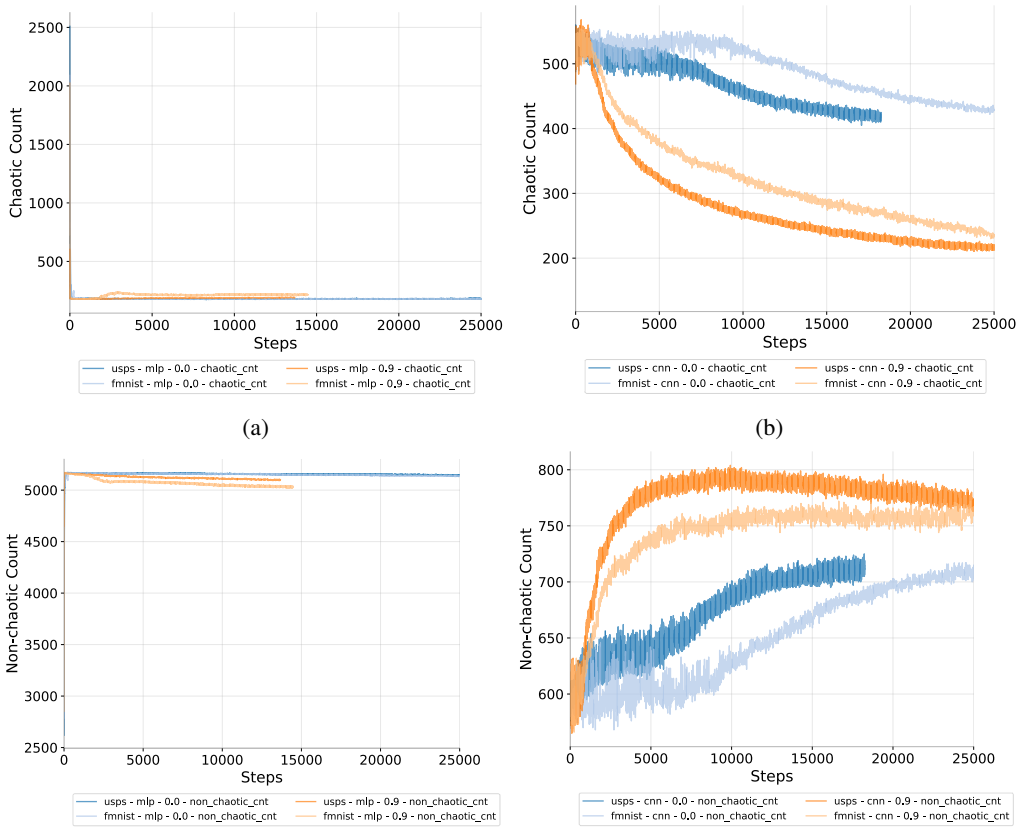


Figure C.1.2: Chaotic (a)/(b) and non-chaotic count (c)/(d) for MLP (a)/(c) and CNN (b)/(d). The counts are defined as $\dim(V_{\text{chaotic}}(t))$ and $\dim(V_{\text{non-chaotic}}(t))$, respectively. The chaotic count for the MLP drops quickly from a high value to a low value and stays almost constant 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 $D -$ 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.

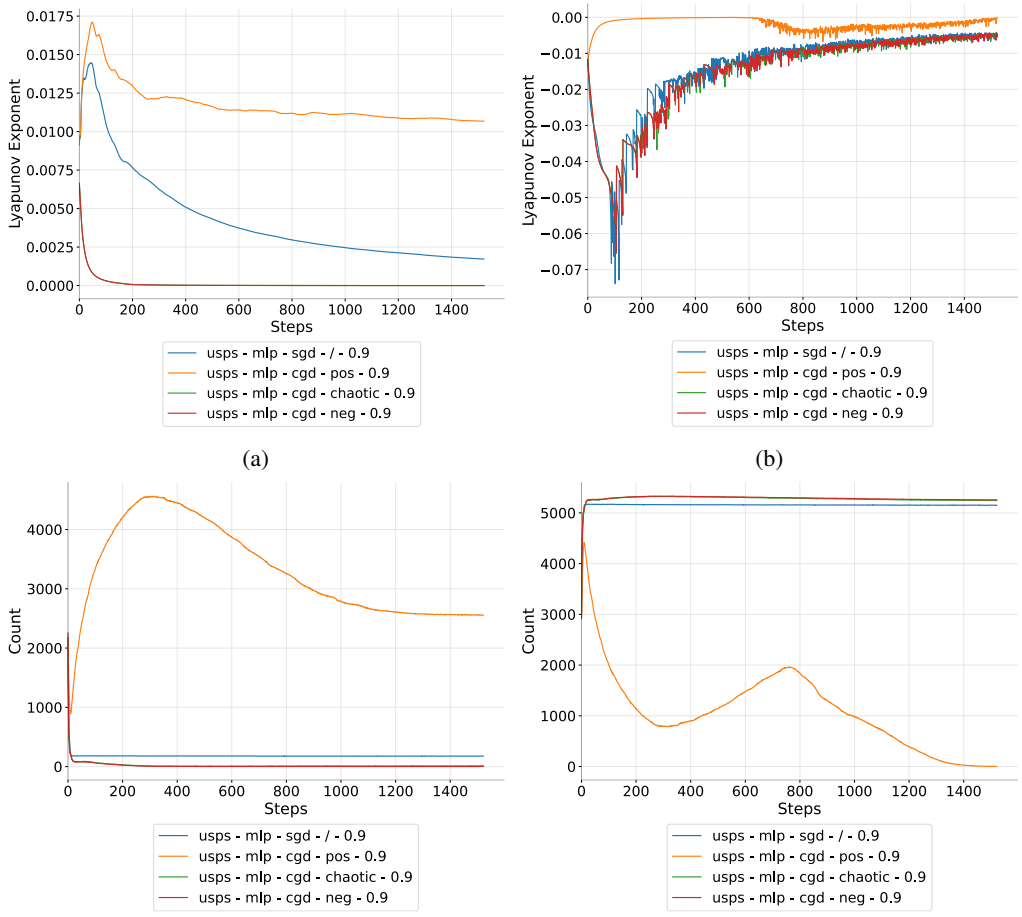


Figure C.1.3: Maximum (a) and minimum (b) Lyapunov exponent for an MLP trained on the USPS dataset with momentum (0.9). Even with momentum, CGD is effective at quickly dampening the maximum Lyapunov exponent to values close to 0 for chaotic and negative pruning. Without pruning, the Lyapunov exponent 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.

C.2 Subspace Similarity of Chaotic spaces

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

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

For comparison, we also estimated the overlap of two random subspaces of the same size as $V_{\text{chaotic}}(t_0)$ 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 $d_{(t,i)} = \dim(V_{\text{chaotic}}(t, \theta_i))$ for different experiments $i = 1, ..., 8 =: N$. The baseline is defined as

$$\frac{1}{N} \sum_{i=1}^{N} \mathop{\mathbb{E}}_{V_{(0,i)}, V_{(t,i))}} [\text{overlap}(V_{(0,i)}, V_{(t,i)})] \tag{65}$$

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

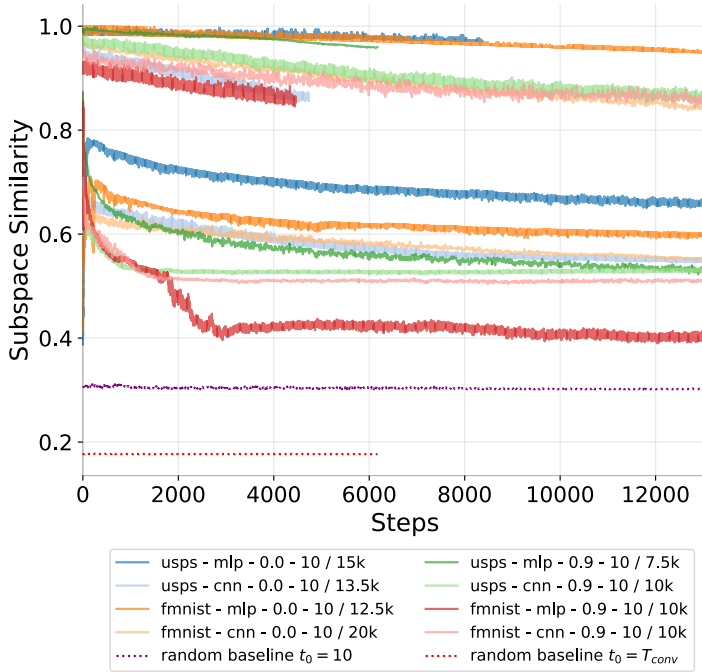
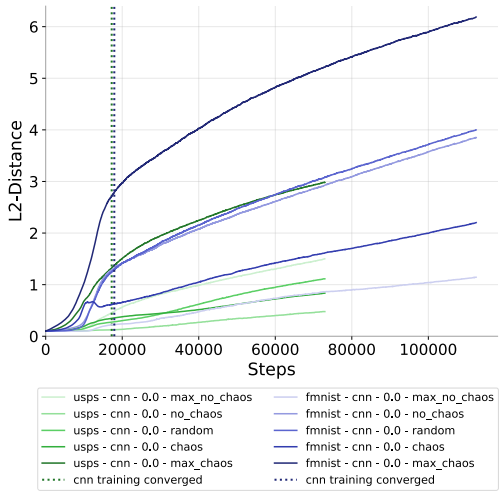
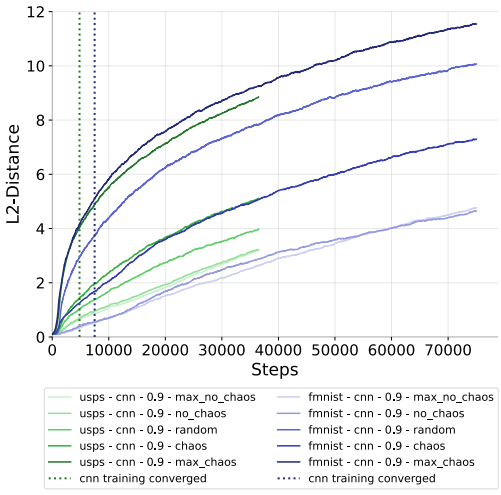


Figure C.2.1: Subspace similarities measured for the same configuration of models (CNN, MLP) and datasets (USPS, FashionMNIST) with and without momentum as we used in our other experiments. We fix a chaotic subspace $V_{\text{chaotic}}(t_0)$ (see section 4.2) at the start $t_0 = 10$ and the end of trainng $t_0 = T_{\text{conv}}$, 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.11). 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 (65)).

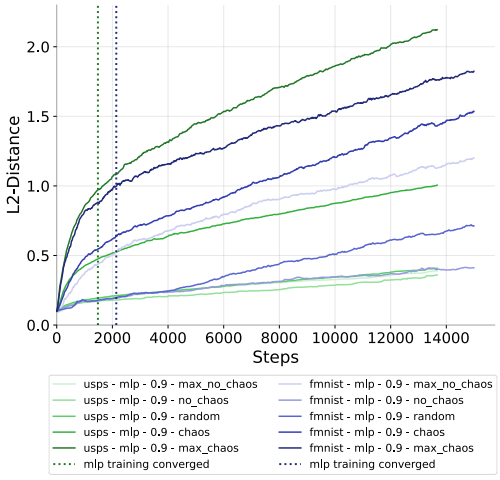
C.3 Distance Experiments



(a) CNN without momentum

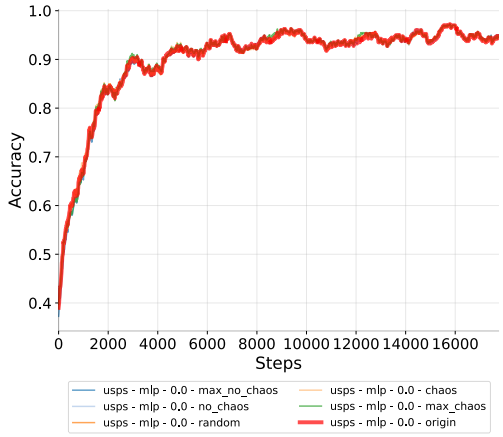


(b) CNN with momentum

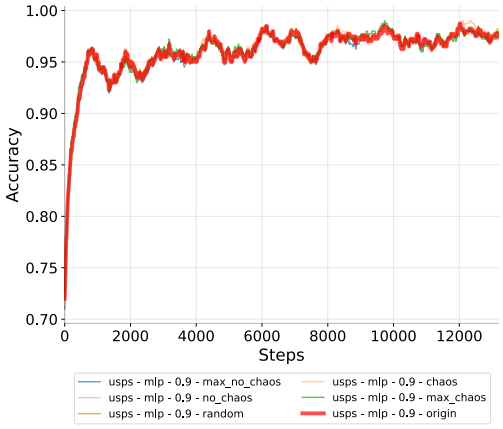


(c) MLP with momentum

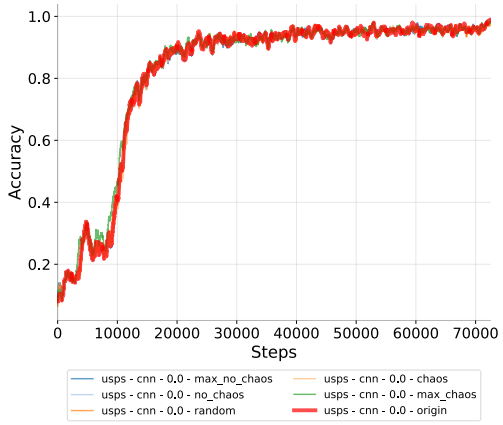
Figure C.3.1: Further distance experiments of Figure 3 (a) with a perturbation magnitude of 0.1. We only see minor differences between datasets so it seems that chaotic behavior is more affected by the model choice and general training parameters. In fact, the CNN model exhibits more global chaotic behavior 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.



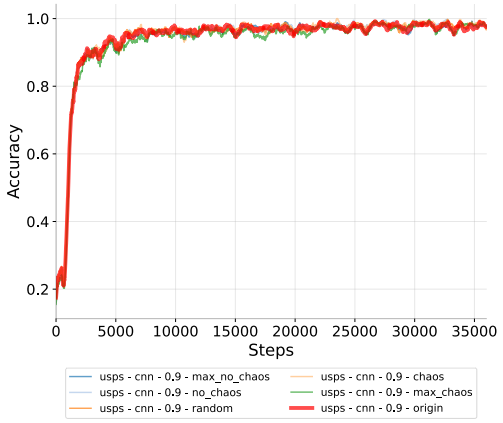
(a) MLP without momentum



(b) MLP with momentum

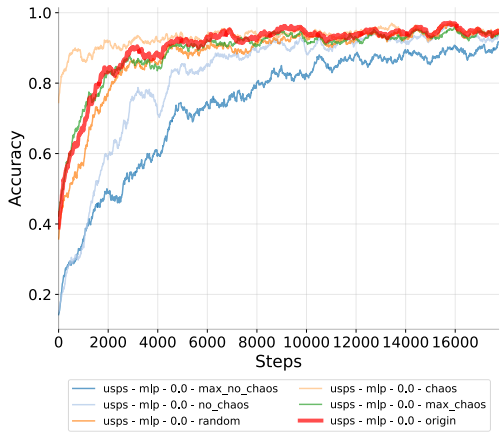


(c) CNN without momentum

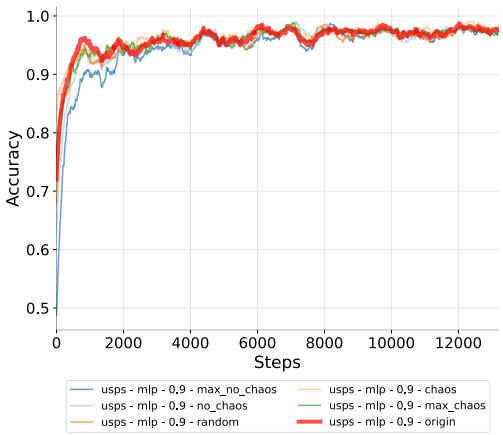


(d) CNN with momentum

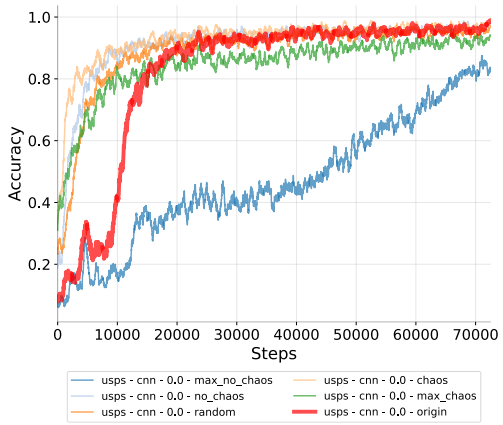
Figure C.3.2: Averaged accuracy (moving average over the last 500 steps) of runs with a perturbation magnitude of $\varepsilon = 0.1$ over the different strategies. We see no significant difference between the models with different perturbation strategies compared to the original model.



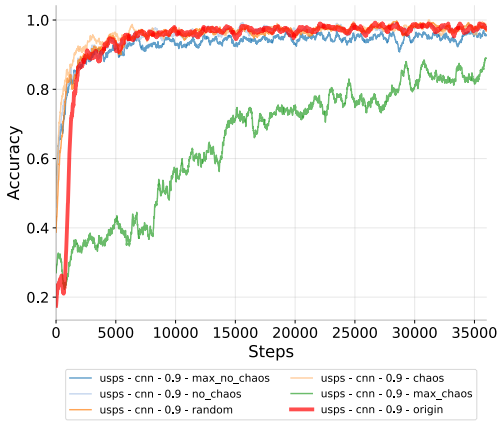
(a) MLP without momentum



(b) MLP with momentum



(c) CNN without momentum



(d) CNN with momentum

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 phenomenons.

C.4 Choice of Perturbation Magnitude

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

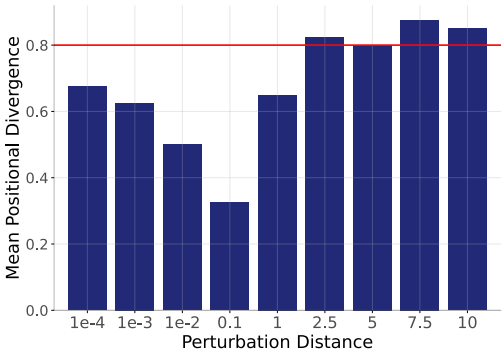
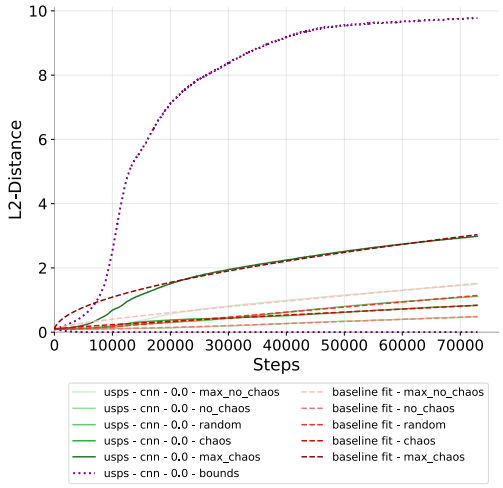


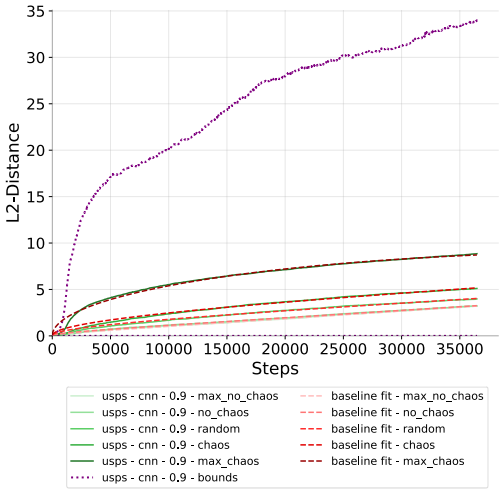
Figure C.4.1: The mean positional divergence: $1/(2N)\sum_{i=1}^{N}|\text{pos}_{\text{strat}(i)}-i|$ over all $N=5$ different strategies. We expect the strategies to be ordered in the following way: maximal chaotic, chaotic, random, non-chaotic, maximal non-chaotic (from highest divergence to lowest).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

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

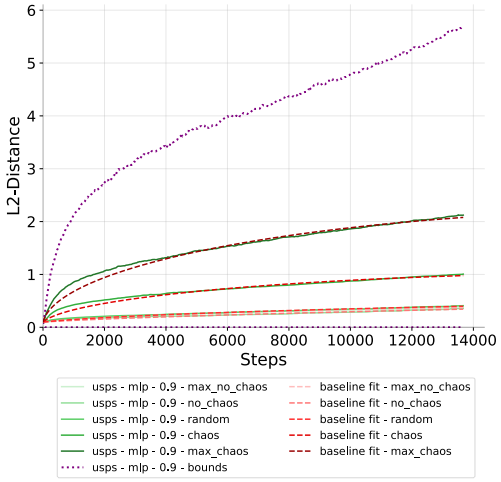
C.5 Theoretical Distance Analysis



(a) CNN without momentum



(b) CNN with momentum



(c) MLP with momentum

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 do determine the exact direction of maximal/minimal global chaos due to a numerical error (see section C.4)