A. Proofs

Proposition 1. *The Levi-Civita connection of a neural network model manifold is given by*

$$\Gamma^{\mu}_{\alpha\beta} = g^{\mu\nu}\mathbb{E}_{q(\mathbf{x})}\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}\left\{\partial_\nu \log p_\theta(\mathbf{t} \mid \mathbf{x})\left[\partial_\alpha\partial_\beta \log p_\theta(\mathbf{t} \mid \mathbf{x}) + \frac{1}{2}\partial_\alpha \log p_\theta(\mathbf{t} \mid \mathbf{x})\partial_\beta \log p_\theta(\mathbf{t} \mid \mathbf{x})\right]\right\}$$

Proof of Proposition 1. Let $r_\theta(\mathbf{z})$ be the joint distribution defined by $q(\mathbf{x})p_\theta(\mathbf{t} \mid \mathbf{x})$. The Fisher information metric is $g_{\mu\nu} = \mathbb{E}_{\mathbf{z}}[\partial_\mu \log r_\theta(\mathbf{z})\partial_\nu \log r_\theta(\mathbf{z})]$. The partial derivative $\partial_\mu g_{\alpha\beta}$ can be computed via scoring function trick, *i.e.*, $\partial_\mu g_{\alpha\beta} = \partial_\mu \mathbb{E}_{\mathbf{z}}[\partial_\alpha \log r_\theta(\mathbf{z})\partial_\beta \log r_\theta(\mathbf{z})] = \mathbb{E}_{\mathbf{z}}[\partial_\mu \log r_\theta(\mathbf{z})\partial_\alpha \log r_\theta(\mathbf{z})\partial_\beta \log r_\theta(\mathbf{z}) + \partial_\mu\partial_\alpha \log r_\theta(\mathbf{z})\partial_\beta \log r_\theta(\mathbf{z}) + \partial_\alpha \log r_\theta(\mathbf{z})\partial_\mu\partial_\beta \log r_\theta(\mathbf{z})]$. All other partial derivatives can be obtained via symmetry.

According to the definition,

$$\begin{aligned}\Gamma^{\mu}_{\alpha\beta} &= \frac{1}{2}g^{\mu\nu}(\partial_\alpha g_{\nu\beta} + \partial_\beta g_{\nu\alpha} - \partial_\nu g_{\alpha\beta}) \\ &= g^{\mu\nu}\mathbb{E}_{\mathbf{z}}\big\{\partial_\nu \log r_\theta(\mathbf{z})\big[\partial_\alpha\partial_\beta \log r_\theta(\mathbf{z}) + \frac{1}{2}\partial_\alpha \log r_\theta(\mathbf{z})\partial_\beta \log r_\theta(\mathbf{z})\big]\big\}.\end{aligned}$$

The proposition is proved after replacing $r_\theta(\mathbf{z})$ with $q(\mathbf{x})p_\theta(\mathbf{t} \mid \mathbf{x})$. □

Theorem 1. *Consider the initial value problem $\dot{x} = f(t, x(t)), x(0) = a, 0 \le t \le T$, where $f(t, x(t))$ is a continuous function in a region U containing*

$$\mathcal{D} := \{(t, x) \mid 0 \le t \le T, \|x - a\| \le X\}.$$

Suppose $f(t, x)$ also satisfies Lipschitz condition, so that

$$\|f(t, x) - f(t, y)\| \le L_f \|x - y\|, \quad \forall (t, x) \in \mathcal{D} \wedge (t, y) \in \mathcal{D}.$$

In addition, let $M = \sup_{(t,x)\in\mathcal{D}} \|f(t, x)\|$ and assume that $TM \le X$. Next, suppose x to be the global coordinate of a Riemannian C^∞-manifold $\mathcal{M}$ with Levi-Civita connection $\Gamma^{\mu}_{\alpha\beta}(x)$. Then, let the interval $[0, T]$ be subdivided into n equal parts by the grid points $0 = t_0 < t_1 < \cdots < t_n = T$, with the grid size $h = T/n$. Denote x_k as the numerical solution given by Euler's update with geodesic correction (7), so that

$$x^{\mu}_{k+1} = x^{\mu}_k + hf^{\mu}(t_k, x_k) - \frac{1}{2}h^2\Gamma^{\mu}_{\alpha\beta}(x_k)f^{\alpha}(t_k, x_k)f^{\beta}(t_k, x_k), \qquad x_0 = a. \tag{11}$$

Let $\hat{x}_k$ be the numerical solution obtained by faster geodesic correction (9)

$$\hat{x}^{\mu}_{k+1} = \hat{x}^{\mu}_k + hf^{\mu}(t_k, \hat{x}_k) - \frac{1}{2}h^2\Gamma^{\mu}_{\alpha\beta}(\hat{x}_k)\delta\hat{x}^{\alpha}_k\delta\hat{x}^{\beta}_k, \tag{12}$$

where $\delta\hat{x}_k = (\hat{x}_k - \hat{x}_{k-1})/h$. Finally, define the error e_k at each grid point x_k by $e_k = x'_k - x_k$, and $\hat{e}_k = x'_k - \hat{x}_k$, where x'_k is the numerical solution given by Riemannian Euler method, i.e.*,*

$$x'_{k+1} = \mathrm{Exp}(x'_k, hf(t_k, x'_k)), \qquad x'_0 = a. \tag{13}$$

Then, assuming $\|x_k - a\| \le X$, $\|\hat{x}_k - a\| \le X$, and $\|x'_k - a\| \le X$, it follows that

$$\|e_k\| \le \mathcal{O}(h^2) \quad \textit{and} \quad \|\hat{e}_k\| \le \mathcal{O}(h^2), \quad h \to 0, \quad \forall k \in [n].$$

As a corollary, both Euler's update with geodesic correction and its faster variant converge to the solution of ODE in 1st order.

Proof of Theorem 1. Since $f(t, x)$ is continuous in the compact region $\mathcal{D}$ satisfying Lipschitz condition, and $TM \le X$, Picard-Lindelf theorem ensures that there exists a unique continuously differentiable solution in $\mathcal{D}$.

On the smooth Riemannian manifold $\mathcal{M}$, the corresponding region $\mathcal{X} \subset \mathcal{M}$ for the set of coordinates $\{x \mid \|x - a\| \le X\}$ is compact. Following Lemma 5.7, Lemma 5.8 in (Petersen, 2006) and a standard covering-of-compact-set argument,

there exists a constant $\epsilon > 0$ such that geodesics are defined everywhere within the region $\mathcal{G} := \{(p, v, s) \mid p \in \mathcal{X}, v \in \mathcal{T}_p\mathcal{M} \wedge \|v\| \leq M, 0 \leq s \leq \epsilon\}$. Tychonoff's theorem affirms that $\mathcal{G}$ is compact in terms of product topology and we henceforth assume $h < \epsilon$. Next, let $\gamma(p, v, s) := \mathrm{Exp}(p, sv)$, where $s \in (-\epsilon, \epsilon)$. From continuous dependence on initial conditions of ODE, we have $\gamma(p, v, s) \in C^\infty$ (see, *e.g.*, Theorem 5.11 in (Petersen, 2006)).

Using Taylor expansion with Lagrange remainder, we can rewrite update rule (13) as

$$x'_{k+1} = x'_k + \frac{\partial\gamma(x'_k, v'_k, 0)}{\partial s}h + \frac{1}{2}\frac{\partial^2\gamma(x'_k, v'_k, 0)}{\partial s^2}h^2 + \frac{1}{6}\frac{\partial^3\gamma(x'_k, v'_k, \xi h)}{\partial s^3}h^3, \tag{14}$$

where $v'_k = f(t_k, x'_k)$ and $\xi \in [0, 1]$. In the meanwhile, update rule (11) can be equivalently written as

$$x_{k+1} = x_k + \frac{\partial\gamma(x_k, v_k, 0)}{\partial s}h + \frac{1}{2}\frac{\partial^2\gamma(x_k, v_k, 0)}{\partial s^2}h^2, \tag{15}$$

where $v_k = f(t_k, x_k)$. Subtracting (15) from (14), we obtain

$$\begin{aligned} e_{k+1} = e_k &+ \left[\frac{\partial\gamma(x'_k, v'_k, 0)}{\partial s} - \frac{\partial\gamma(x_k, v_k, 0)}{\partial s}\right]h \\ &+ \frac{1}{2}\left[\frac{\partial^2\gamma(x'_k, v'_k, 0)}{\partial s^2} - \frac{\partial^2\gamma(x_k, v_k, 0)}{\partial s^2}\right]h^2 \\ &+ \frac{1}{6}\frac{\partial^3\gamma(x'_k, v'_k, \xi h)}{\partial s^3}h^3. \end{aligned} \tag{16}$$

Recall that $\gamma(p, v, s)$ is C^∞ on $\mathcal{G}$, by extreme value theorem we can denote

$$\begin{gathered} \Gamma_1 = \sup_{(p,v,s)\in\mathcal{G}} \left\|\frac{\partial^2\gamma(p, v, s)}{\partial p\partial s}\right\|, \quad \Gamma_2 = \sup_{(p,v,s)\in\mathcal{G}} \left\|\frac{\partial^2\gamma(p, v, s)}{\partial v\partial s}\right\| \\ \Gamma_3 = \sup_{(p,v,s)\in\mathcal{G}} \left\|\frac{\partial^3\gamma(p, v, s)}{\partial p\partial s^2}\right\|, \quad \Gamma_4 = \sup_{(p,v,s)\in\mathcal{G}} \left\|\frac{\partial^3\gamma(p, v, s)}{\partial v\partial s^2}\right\| \\ \Gamma_5 = \sup_{(p,v,s)\in\mathcal{G}} \left\|\frac{\partial^3\gamma(p, v, s)}{\partial s^3}\right\|, \end{gathered}$$

where $\|\cdot\|$ can be chosen arbitrarily as long as they are compatible (*e.g.*, use operator norms for tensors), since norms in a finite dimensional space are equivalent to each other.

With upper bounds $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4$ and Γ_5, Eq. (16) has the estimation via Lagrange mean value theorem,

$$\begin{aligned} \|e_{k+1}\| &\leq \|e_k\| + h(\Gamma_1\|e_k\| + \Gamma_2\|v'_k - v_k\|) + \frac{1}{2}h^2(\Gamma_3\|e_k\| + \Gamma_4\|v'_k - v_k\|) + \frac{1}{6}\Gamma_5 h^3 \\ &\leq \|e_k\| + h(\Gamma_1\|e_k\| + L_f\Gamma_2\|e_k\|) + \frac{1}{2}h^2(\Gamma_3\|e_k\| + L_f\Gamma_4\|e_k\|) + \frac{1}{6}\Gamma_5 h^3 \\ &\leq \left(1 + \Gamma_1 h + \Gamma_2 h L_f + \frac{1}{2}\Gamma_3 h^2 + \frac{1}{2}\Gamma_4 L_f h^2\right)\|e_k\| + \frac{1}{6}\Gamma_5 h^3 \\ &\leq \cdots \\ &\leq \left(1 + C_1 h + \frac{1}{2}C_2 h^2\right)^k \left(\|e_0\| + \frac{\Gamma_5 h^2}{6C_1 + 3C_2 h}\right) - \frac{\Gamma_5 h^2}{6C_1 + 3C_2 h} \\ &\leq \left(e^{C_1 kh + \frac{1}{2}C_2 kh^2} - 1\right)\left(\frac{\Gamma_5 h^2}{6C_1 + 3C_2 h}\right) \\ &\leq \left(e^{C_1 T + \frac{1}{2}C_2 T^2} - 1\right)\left(\frac{\Gamma_5 h^2}{6C_1 + 3C_2 h}\right) \\ &\leq \frac{\Gamma_5}{6C_1}\left(e^{C_1 T + \frac{1}{2}C_2 T^2} - 1\right)h^2 = \mathcal{O}(h^2), \end{aligned}$$

where from the fifth line we substitute C_1 for $\Gamma_1 + \Gamma_2 L_f$ and C_2 for $\Gamma_3 + \Gamma_4 L_f$, and we used the fact that $\|e_0\| = 0$ and the inequality $(1+w)^k \leq \exp(kw)$. This means geodesic correction converges to the invariant solution obtained using a Riemannian Euler method with a 2nd-order rate. Since Riemannian Euler method is itself a first-order algorithm (Bielecki, 2002), geodesic correction also converges to the exact solution in 1st order.

Then, we consider the faster geodesic update rule (12)

$$\begin{aligned}
&\hat{x}^\mu_{k+1} = \hat{x}^\mu_k + \frac{\partial \gamma^\mu(\hat{x}_k, \hat{v}_k, 0)}{\partial s} h - \frac{1}{2}\Gamma^\mu_{\alpha\beta}(\hat{x}_k)(\hat{x}^\alpha_k - \hat{x}^\alpha_{k-1})(\hat{x}^\beta_k - \hat{x}^\beta_{k-1}) \\
=&\hat{x}^\mu_k + \frac{\partial \gamma^\mu(\hat{x}_k, \hat{v}_k, 0)}{\partial s} h - \frac{1}{2}\Gamma^\mu_{\alpha\beta}(\hat{x}_k)\left[h\frac{\partial \gamma^\alpha(\hat{x}_{k-1}, \hat{v}_{k-1}, 0)}{\partial s} + \frac{1}{2}h^2\Gamma^\alpha_{ab}(\hat{x}_{k-1}) f^a(t_{k-1}, \hat{x}_{k-1})\right. \\
&\left. f^b(t_{k-1}, \hat{x}_{k-1})\right] \cdot \left[h\frac{\partial \gamma^\beta(\hat{x}_{k-1}, \hat{v}_{k-1}, 0)}{\partial s} + \frac{1}{2}h^2\Gamma^\beta_{cd}(\hat{x}_{k-1}) f^c(t_{k-1}, \hat{x}_{k-1}) f^d(t_{k-1}, \hat{x}_{k-1})\right] \\
=&\hat{x}^\mu_k + \frac{\partial \gamma^\mu(\hat{x}_k, \hat{v}_k, 0)}{\partial s} h - \frac{1}{2}\Gamma^\mu_{\alpha\beta}\frac{\partial \gamma^\alpha(\hat{x}_{k-1}, \hat{v}_{k-1}, 0)}{\partial s}\frac{\partial \gamma^\beta(\hat{x}_{k-1}, \hat{v}_{k-1}, 0)}{\partial s} h^2 \\
&+ \Phi^\mu(h, \hat{x}_k, \hat{x}_{k-1}, t_{k-1}, \hat{v}_{k-1}) \\
=&\hat{x}^\mu_k + \frac{\partial \gamma^\mu(\hat{x}_k, \hat{v}_k, 0)}{\partial s} h + \frac{1}{2}\frac{\partial^2 \gamma^\mu(\hat{x}_{k-1}, \hat{v}_{k-1}, 0)}{\partial s^2} h^2 + \Phi^\mu(h, \hat{x}_k, \hat{x}_{k-1}, t_{k-1}, \hat{v}_{k-1}),
\end{aligned} \tag{17}$$

where the last line utilizes geodesic equation (1) and $\Phi^\mu(h, \hat{x}_k, \hat{x}_{k-1}, t_{k-1}, \hat{v}_{k-1})$ is

$$\begin{aligned}
&h^3 \underbrace{\frac{1}{2}\Gamma^\mu_{\alpha\beta}(\hat{x}_k)\Gamma^\alpha_{ab}(\hat{x}_{k-1}) f^a f^b \frac{\partial \gamma^\beta(\hat{x}_{k-1}, \hat{v}_{k-1}, 0)}{\partial s}}_{=:\Phi_1(\hat{x}_k, \hat{x}_{k-1}, \hat{v}_{k-1}, t_{k-1})} \\
&+h^4 \underbrace{\frac{1}{8}\Gamma^\mu_{\alpha\beta}(\hat{x}_k)\Gamma^\alpha_{ab}(\hat{x}_{k-1})\Gamma^\beta_{cd}(\hat{x}_{k-1}) f^a f^b f^c f^d}_{=:\Phi_2(\hat{x}_k, \hat{x}_{k-1}, t_{k-1})},
\end{aligned}$$

where $f^i = f^i(t_{k-1}, \hat{x}_{k-1}), i \in \{a, b, c, d\}$. Both Φ_1 and Φ_2 are C^∞ functions on compact sets. As a result, extreme value theorem states that there exist constants A and B so that $\sup \|\Phi_1\| = A$ and $\sup \|\Phi_2\| = B$.

Subtracting (17) from (15) and letting $\theta_k = x_k - \hat{x}_k$ we obtain

$$\begin{aligned}
\theta_{k+1} = \theta_k &+ \left[\frac{\partial \gamma(x_k, v_k, 0)}{\partial s} - \frac{\partial \gamma(\hat{x}_k, \hat{v}_k, 0)}{\partial s}\right] h \\
&+ \frac{1}{2}\left[\frac{\partial^2 \gamma(x_k, v_k, 0)}{\partial s^2} - \frac{\partial^2 \gamma(\hat{x}_k, \hat{v}_k, 0)}{\partial s^2}\right] h^2 \\
&+ h^3 \Phi_1(\hat{x}_k, \hat{x}_{k-1}, \hat{v}_{k-1}, t_{k-1}) + h^4 \Phi_2(\hat{x}_k, \hat{x}_{k-1}, t_{k-1}),
\end{aligned}$$

and the error can be bounded by

$$\begin{aligned}
\|\theta_{k+1}\| &\leq \|\theta_k\| + h(\Gamma_1 \|\theta_k\| + \Gamma_2 \|v_k - \hat{v}_k\|) + \frac{1}{2}h^2(\Gamma_3 \|\theta_k\| + \Gamma_4 \|v_k - \hat{v}_k\|) + Ah^3 + Bh^4 \\
&\leq \|\theta_k\| + h(\Gamma_1 \|\theta_k\| + L_f\Gamma_2 \|\theta_k\|) + \frac{1}{2}h^2(\Gamma_3 \|\theta_k\| + L_f\Gamma_4 \|\theta_k\|) + Ah^3 + Bh^4 \\
&\leq (1 + C_1 h + \frac{1}{2}C_2 h^2) \|\theta_k\| + Ah^3 + Bh^4 \\
&\leq \cdots \\
&\leq \left(1 + C_1 h + \frac{1}{2}C_2 h^2\right)^k \left(\|\theta_0\| + \frac{Ah^2 + Bh^3}{C_1 + \frac{1}{2}C_2 h}\right) - \frac{Ah^2 + Bh^3}{C_1 + \frac{1}{2}C_2 h} \\
&\leq \left(e^{C_1 kh + \frac{1}{2}C_2 kh^2} - 1\right)\frac{Ah^2 + Bh^3}{C_1 + \frac{1}{2}C_2 h} \\
&\leq \left(e^{C_1 T + \frac{1}{2}C_2 T^2} - 1\right)\frac{Ah^2 + Bh^3}{C_1} = \mathcal{O}(h^2).
\end{aligned}$$

Finally, $\|\hat{e}_k\| \leq \|e_k\| + \|\theta_k\| = \mathcal{O}(h^2)$ as $h \to 0$. This shows that faster geodesic correction converges to the invariant solution of Riemannian Euler method with a 2nd-order rate and the exact solution of ODE in 1st order. □

B. Derivations of Connections for Different Losses

In this section we show how to derive the formulas of Fisher information matrices and Levi-Civita connections for three common losses used in our experiments.

B.1. Squared Loss

The squared loss is induced from negative log-likelihood of the probabilistic model

$$p_\theta(\mathbf{t} \mid \mathbf{x}) = \prod_{i=1}^{o} \mathcal{N}(t_i \mid y_i, \sigma^2),$$

and the log-likelihood is

$$\ln p_\theta(\mathbf{t} \mid \mathbf{x}) = -\frac{1}{2\sigma^2} \sum_{i=1}^{o} (t_i - y_i)^2 + \text{const.}$$

According to the definition of Fisher information matrix,

$$\begin{aligned}
g_{\mu\nu} &= \mathbb{E}_{q(\mathbf{x})}[\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}[\partial_\mu \ln p_\theta(\mathbf{t} \mid \mathbf{x}) \partial_\nu \ln p_\theta(\mathbf{t} \mid \mathbf{x})]] \\
&= \mathbb{E}_{q(\mathbf{x})} \left[\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})} \left[\frac{1}{\sigma^4} \sum_{ij} (t_i - y_i)(t_j - y_j) \partial_\mu y_i \partial_\nu y_j \right] \right] \\
&= \mathbb{E}_{q(\mathbf{x})} \left[\frac{1}{\sigma^4} \sum_{ij} \delta_{ij} \sigma^2 \partial_\mu y_i \partial_\nu y_j \right] \\
&= \frac{1}{\sigma^2} \sum_{i=1}^{o} \mathbb{E}_{q(\mathbf{x})} \left[\partial_\mu y_i \partial_\nu y_j \right].
\end{aligned}$$

To compute the Levi-Civita connection $\Gamma^\mu_{\alpha\beta}$, we first calculate the following component

$$\begin{aligned}
&\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}[\partial_\nu \ln p_\theta(\mathbf{t} \mid \mathbf{x}) \partial_\alpha \partial_\beta \ln p_\theta(\mathbf{t} \mid \mathbf{x})] \\
=&\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})} \left[\left(\frac{1}{\sigma^2} \sum_{i=1}^{o} (t_i - y_i) \partial_\nu y_i \right) \left(\frac{1}{\sigma^2} \sum_{j=1}^{o} -\partial_\alpha y_j \partial_\beta y_j + (t_j - y_j) \partial_\alpha \partial_\beta y_j \right) \right] \\
=&\frac{1}{\sigma^4} \mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})} \left[\sum_{ij} (t_i - y_i)(t_j - y_j) \partial_\nu y_i \partial_\alpha \partial_\beta y_j \right] \\
=&\frac{1}{\sigma^4} \mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})} \left[\sum_{ij} \sigma^2 \delta_{ij} \partial_\nu y_i \partial_\alpha \partial_\beta y_j \right] \\
=&\frac{1}{\sigma^2} \sum_{i=1}^{o} \mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})} \left[\partial_\nu y_i \partial_\alpha \partial_\beta y_i \right].
\end{aligned}$$

The second component of $\Gamma^\mu_{\alpha\beta}$ we have to consider is

$$\begin{aligned}
&\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})} \left[\frac{1}{2} \partial_\nu \ln p_\theta(\mathbf{t} \mid \mathbf{x}) \partial_\alpha \ln p_\theta(\mathbf{t} \mid \mathbf{x}) \partial_\beta \ln p_\theta(\mathbf{t} \mid \mathbf{x}) \right] \\
=&\frac{1}{2\sigma^6} \sum_{ijk} \mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}[(t_i - y_i)(t_j - y_j)(t_k - y_k) \partial_\nu y_i \partial_\alpha y_i \partial_\beta y_i] \\
=&0,
\end{aligned}$$

where the last equality uses the property that third moment of a Gaussian distribution is 0.

Combining the above two components, we obtain the Levi-Civita connection

$$\Gamma^{\mu}_{\alpha\beta} = \frac{1}{\sigma^2}\sum_{i=1}^{o} g^{\mu\nu}\mathbb{E}_{q(\mathbf{x})}[\partial_\nu y_i \partial_\alpha \partial_\beta y_i].$$

B.2. Binary Cross-Entropy

The binary cross-entropy loss is induced from negative log-likelihood of the probabilistic model

$$p_\theta(\mathbf{t} \mid \mathbf{x}) = \prod_{i=1}^{o} y_i^{t_i}(1-y_i)^{1-t_i},$$

and the log-likelihood is

$$\ln p_\theta(\mathbf{t} \mid \mathbf{x}) = \sum_{i=1}^{o} t_i \ln y_i + (1-t_i)\ln(1-y_i).$$

According to the definition of Fisher information matrix,

$$\begin{aligned}
g_{\mu\nu} &= \mathbb{E}_{q(\mathbf{x})}[\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}[\partial_\mu \ln p_\theta(\mathbf{t} \mid \mathbf{x})\partial_\nu \ln p_\theta(\mathbf{t} \mid \mathbf{x})]] \\
&= \mathbb{E}_{q(\mathbf{x})}\left[\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}\left[\sum_{ij}\frac{(t_i-y_i)(t_j-y_j)}{y_i y_j (1-y_i)(1-y_j)}\partial_\mu y_i \partial_\nu y_j\right]\right] \\
&= \mathbb{E}_{q(\mathbf{x})}\left[\sum_{ij}\delta_{ij}\frac{y_i(1-y_i)}{y_i y_j (1-y_i)(1-y_j)}\partial_\mu y_i \partial_\nu y_j\right] \\
&= \sum_{i=1}^{o}\mathbb{E}_{q(\mathbf{x})}\left[\frac{1}{y_i(1-y_i)}\partial_\mu y_i \partial_\nu y_j\right].
\end{aligned}$$

To compute the Levi-Civita connection $\Gamma^{\mu}_{\alpha\beta}$, we first calculate the following component

$$\begin{aligned}
&\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}[\partial_\nu \ln p_\theta(\mathbf{t} \mid \mathbf{x})\partial_\alpha\partial_\beta \ln p_\theta(\mathbf{t} \mid \mathbf{x})] \\
=&\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}\left[\left(\sum_{i=1}^{o}\frac{t_i-y_i}{y_i(1-y_i)}\partial_\mu y_i\right)\left(\sum_{j=1}^{o}-\frac{(t_j-y_j)^2}{y_j^2(1-y_j)^2}\partial_\alpha y_j\partial_\beta y_j + \frac{t_j-y_j}{y_j(1-y_j)}\partial_\alpha\partial_\beta y_j\right)\right] \\
=&\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}\left[-\sum_{ij}\frac{(t_i-y_i)(t_j-y_j)^2}{y_i(1-y_i)y_j^2(1-y_j)^2}\partial_\mu y_i\partial_\alpha y_j\partial_\beta y_j + \sum_{ij}\frac{(t_i-y_i)(t_j-y_j)}{y_i(1-y_i)y_j(1-y_j)}\partial_\mu y_i\partial_\alpha\partial_\beta y_j\right] \\
=&\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}\left[-\sum_{i=1}^{o}\frac{(t_i-y_i)^3}{y_i^3(1-y_i)^3}\partial_\mu y_i\partial_\alpha y_i\partial_\beta y_i + \sum_{i=1}^{o}\frac{(t_i-y_i)^2}{y_i^2(1-y_i)^2}\partial_\mu y_i\partial_\alpha\partial_\beta y_i\right] \\
=&-\sum_{i=1}^{o}\frac{(1-y_i)^3 y_i + (-y_i)^3(1-y_i)}{y_i^3(1-y_i)^3}\partial_\mu y_i\partial_\alpha y_i\partial_\beta y_i + \sum_{i=1}^{o}\frac{1}{y_i(1-y_i)}\partial_\mu y_i\partial_\alpha\partial_\beta y_i \\
=&\sum_{i=1}^{o}\frac{2y_i-1}{y_i^2(1-y_i)^2}\partial_\mu y_i\partial_\alpha y_i\partial_\beta y_i + \sum_{i=1}^{o}\frac{1}{y_i(1-y_i)}\partial_\mu y_i\partial_\alpha\partial_\beta y_i.
\end{aligned}$$

where in the first line we use the equality $(t_j-y_j)^2 = (t_j - 2t_j y_j + y_j^2)$ which holds given that $t_j \in \{0,1\}$.

The second component of $\Gamma^{\mu}_{\alpha\beta}$ is

$$\begin{aligned}
&\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}\left[\frac{1}{2}\partial_\nu \ln p_\theta(\mathbf{t}\mid\mathbf{x})\partial_\alpha \ln p_\theta(\mathbf{t}\mid\mathbf{x})\partial_\beta \ln p_\theta(\mathbf{t}\mid\mathbf{x})\right]\\
=&\frac{1}{2}\sum_{ijk}\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}\left[\frac{(t_i-y_i)(t_j-y_j)(t_k-y_k)}{y_iy_jy_k(1-y_i)(1-y_j)(1-y_k)}\partial_\nu y_i\partial_\alpha y_j\partial_\beta y_k\right]\\
=&\frac{1}{2}\sum_{i=1}^{o}\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}\left[\frac{(t_i-y_i)^3}{y_i^3(1-y_i)^3}\partial_\nu y_i\partial_\alpha y_i\partial_\beta y_i\right]\\
=&\sum_{i=1}^{o}\frac{1-2y_i}{2y_i^2(1-y_i)^2}\partial_\nu y_i\partial_\alpha y_i\partial_\beta y_i,
\end{aligned}$$

Combining the above two components, we obtain the Levi-Civita connection

$$\Gamma^{\mu}_{\alpha\beta}=g^{\mu\nu}\sum_{i=1}^{o}\mathbb{E}_{q(\mathbf{x})}\left[\frac{2y_i-1}{2y_i^2(1-y_i)^2}\partial_\nu y_i\partial_\alpha y_i\partial_\beta y_i+\frac{1}{y_i(1-y_i)}\partial_\nu y_i\partial_\alpha\partial_\beta y_i\right].$$

B.3. Multi-Class Cross-Entropy

The multi-class cross-entropy loss is induced from negative log-likelihood of the probabilistic model

$$p_\theta(\mathbf{t}\mid\mathbf{x})=\prod_{i=1}^{o}y_i^{t_i},$$

and the log-likelihood is

$$\ln p_\theta(\mathbf{t}\mid\mathbf{x})=\sum_{i=1}^{o}t_i\ln y_i.$$

According to the definition of Fisher information matrix,

$$\begin{aligned}
g_{\mu\nu}&=\mathbb{E}_{q(\mathbf{x})}[\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}[\partial_\mu \ln p_\theta(\mathbf{t}\mid\mathbf{x})\partial_\nu \ln p_\theta(\mathbf{t}\mid\mathbf{x})]]\\
&=\mathbb{E}_{q(\mathbf{x})}\left[\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}\left[\sum_{ij}\frac{t_it_j}{y_iy_j}\partial_\mu y_i\partial_\nu y_j\right]\right]\\
&=\mathbb{E}_{q(\mathbf{x})}\left[\sum_{i=1}^{o}\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}\left[\frac{t_i^2}{y_i^2}\right]\partial_\mu y_i\partial_\nu y_i\right]\\
&=\sum_{i=1}^{o}\mathbb{E}_{q(\mathbf{x})}\left[\frac{1}{y_i}\partial_\mu y_i\partial_\nu y_i\right].
\end{aligned}$$

To compute the Levi-Civita connection $\Gamma^{\mu}_{\alpha\beta}$, we first calculate the following component

$$\begin{aligned}
&\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}[\partial_\nu \ln p_\theta(\mathbf{t}\mid\mathbf{x})\partial_\alpha\partial_\beta \ln p_\theta(\mathbf{t}\mid\mathbf{x})]\\
=&\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}\left[\left(\sum_{i=1}^{o}\frac{t_i}{y_i}\partial_\nu y_i\right)\left(\sum_{j=1}^{o}-\frac{t_j}{y_j^2}\partial_\alpha y_j\partial_\beta y_j+\frac{t_j}{y_j}\partial_\alpha\partial_\beta y_j\right)\right]\\
=&\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}\left[\sum_{i=1}^{o}-\frac{t_i^2}{y_i^3}\partial_\nu y_i\partial_\alpha y_i\partial_\beta y_i+\frac{t_i^2}{y_i^2}\partial_\nu y_i\partial_\alpha\partial_\beta y_i\right]\\
=&\sum_{i=1}^{o}-\frac{1}{y_i^2}\partial_\nu y_i\partial_\alpha y_i\partial_\beta y_i+\frac{1}{y_i}\partial_\nu y_i\partial_\alpha\partial_\beta y_i.
\end{aligned}$$

The second component of $\Gamma^{\mu}_{\alpha\beta}$ we have to consider is

$$\begin{aligned}
&\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}\left[\frac{1}{2}\partial_\nu \ln p_\theta(\mathbf{t}\mid\mathbf{x})\partial_\alpha \ln p_\theta(\mathbf{t}\mid\mathbf{x})\partial_\beta \ln p_\theta(\mathbf{t}\mid\mathbf{x})\right]\\
=&\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}\left[\sum_{ijk}\frac{t_i t_j t_k}{2y_i y_j y_k}\partial_\nu y_i \partial_\alpha y_j \partial_\beta y_k\right]\\
=&\mathbb{E}_{p_\theta(\mathbf{t}|\mathbf{x})}\left[\sum_{i=1}^{o}\frac{t_i^3}{2y_i^3}\partial_\nu y_i \partial_\alpha y_i \partial_\beta y_i\right]\\
=&\sum_{i=1}^{o}\frac{1}{2y_i^2}\partial_\nu y_i \partial_\alpha y_i \partial_\beta y_i.
\end{aligned}$$

Combining the above two components, we obtain the Levi-Civita connection

$$\Gamma^{\mu}_{\alpha\beta} = g^{\mu\nu}\sum_{i=1}^{o}\mathbb{E}_{q(\mathbf{x})}\left[\frac{1}{y_i}\partial_\nu y_i \partial_\alpha\partial_\beta y_i - \frac{1}{2y_i^2}\partial_\nu y_i \partial_\alpha y_i \partial_\beta y_i\right].$$

C. Computing Connection Products via Backpropagation

It is not tractable to compute $\Gamma^{\mu}_{\alpha\beta}$ for large neural networks. Fortunately, to evaluate (7) we only need to know $\Gamma^{\mu}_{\alpha\beta}\dot\gamma^\alpha\dot\gamma^\beta$. For the typical losses in Proposition 2, this expression contains two main terms:

1. $\sum_{i=1}^{o}\lambda_i\partial_\nu y_i\partial_\alpha\partial_\beta y_i\dot\gamma^\alpha\dot\gamma^\beta$. Note that $\partial_\alpha\partial_\beta y_i\dot\gamma^\alpha\dot\gamma^\beta$ is the directional second derivative of y_i along the direction of $\dot\gamma$ (it's a scalar).
2. $\sum_{i=1}^{o}\lambda_i\partial_\nu y_i\partial_\alpha y_i\partial_\beta y_i\dot\gamma^\alpha\dot\gamma^\beta$. Note that $\partial_\alpha y_i\partial_\beta y_i\dot\gamma^\alpha\dot\gamma^\beta = (\partial_\alpha y_i\dot\gamma^\alpha)^2$ (recall Einstein's notation), where $\partial_\alpha y_i\dot\gamma^\alpha$ is the directional derivative of y_i along the direction of $\dot\gamma$.

After obtaining the directional derivatives of y_i (scalars μ_i), both terms have the form of $\sum_{i=1}^{o}\lambda_i\mu_i\partial_\nu y_i$. It can be computed via backpropagation with loss function $L=\sum_{i=1}^{o}\lambda_i\mu_i y_i$ while treating $\lambda_i\mu_i$ as constants.

Inspired by the "Pearlmutter trick" for computing Hessian-vector and curvature matrix-vector products (Pearlmutter, 1994; Schraudolph, 2002), we propose a similar method to compute directional derivatives and connections.

As a first step, we use the following notations. Given an input $\mathbf{x}$ and parameters $\theta=(W_1,\cdots,W_l,b_1,\cdots,b_l)$, a feed-forward neural network computes its output $\mathbf{y}(\mathbf{x},\theta)=a_l$ by the recurrence

$$s_i = W_i a_{i-1} + b_i \tag{18}$$
$$a_i = \phi_i(s_i), \tag{19}$$

where W_i is the weight matrix, b_i is the bias, and $\phi_i(\cdot)$ is the activation function. Here a_i, b_i and s_i are all vectors of appropriate dimensions. The loss function $L(\mathbf{t},\mathbf{y})$ measures the distance between the ground-truth label $\mathbf{t}$ of $\mathbf{x}$ and the network output $\mathbf{y}$. For convenience, we also define

$$\begin{aligned}
\mathcal{D}(v) &= \frac{\mathrm{d}L(\mathbf{t},\mathbf{y})}{\mathrm{d}v}\\
\mathcal{R}_v(g(\theta)) &= \lim_{\epsilon\to 0}\frac{1}{\epsilon}[g(\theta+\epsilon v)-g(\theta)]\\
\mathcal{S}_v(g(\theta)) &= \lim_{\epsilon\to 0}\frac{1}{\epsilon^2}[g(\theta+2\epsilon v)-2g(\theta+\epsilon v)+g(\theta)]\\
&= \mathcal{R}_v(\mathcal{R}_v(g(\theta))),
\end{aligned}$$

which represents the gradient of $L(\mathbf{t},\mathbf{y})$, directional derivative of $g(\theta)$ and directional second derivative of $g(\theta)$ along the direction of v respectively.

The following observation is crucial for our calculation:

Proposition 3. *For any differentiable scalar function $g(\theta), g_1(\theta), g_2(\theta), f(x)$ and vector v, we have*

$$\begin{aligned}
\mathcal{R}_v(g_1+g_2) &= \mathcal{R}_v(g_1)+\mathcal{R}_v(g_2)\\
\mathcal{R}_v(g_1 g_2) &= \mathcal{R}_v(g_1)g_2 + g_1\mathcal{R}_v(g_2)\\
\mathcal{R}_v(f(g)) &= f'\mathcal{R}_v(g)\\
\mathcal{S}_v(g_1+g_2) &= \mathcal{S}_v(g_1)+\mathcal{S}_v(g_2)\\
\mathcal{S}_v(g_1 g_2) &= \mathcal{S}_v(g_1)g_2 + 2\mathcal{R}_v(g_1)\mathcal{R}_v(g_2) + g_1\mathcal{S}_v(g_2)\\
\mathcal{S}_v(f(g)) &= f''\mathcal{R}_v(g)^2 + f'\mathcal{S}_v(g)
\end{aligned}$$

Using those new notations, the directional derivatives $\partial_\alpha\partial_\beta y_i\dot{\gamma}^\alpha\dot{\gamma}^\beta$, $\partial_\alpha y_i\dot{\gamma}^\alpha$ can be written as $\mathcal{S}_{\dot{\gamma}}(y_i)$ and $\mathcal{R}_{\dot{\gamma}}(y_i)$. We can obtain recurrent equations for them by applying Proposition 3 to (18) and (19). The results are

$$\begin{aligned}
\mathcal{R}_{\dot{\gamma}}(s_i) &= \mathcal{R}_{\dot{\gamma}}(W_i)a_{i-1} + W_i\mathcal{R}_{\dot{\gamma}}(a_{i-1}) + \mathcal{R}_{\dot{\gamma}}(b_i)\\
\mathcal{R}_{\dot{\gamma}}(a_i) &= \phi'(s_i)\odot\mathcal{R}_{\dot{\gamma}}(s_i)\\
\mathcal{S}_{\dot{\gamma}}(s_i) &= \mathcal{S}_{\dot{\gamma}}(W_i)a_{i-1} + 2\mathcal{R}_{\dot{\gamma}}(W_i)\mathcal{R}_{\dot{\gamma}}(a_{i-1})\\
&\quad + W_i\mathcal{S}_{\dot{\gamma}}(a_{i-1}) + \mathcal{S}_{\dot{\gamma}}(b_i)\\
\mathcal{S}_{\dot{\gamma}}(a_i) &= \phi''(s_i)\odot\mathcal{R}_{\dot{\gamma}}(s_i)^2 + \phi'(s_i)\odot\mathcal{S}_{\dot{\gamma}}(s_i),
\end{aligned}$$

which can all be computed during the forward pass, given that $\mathcal{R}_{\dot{\gamma}}(W_i)=\dot{\gamma}$ and $\mathcal{S}_{\dot{\gamma}}(W_i)=0$. Based on the above recurrent rules, we summarize our algorithms for those two terms of connections in Alg. 1 and Alg. 2.

Algorithm 1 Calculating $\sum_{i=1}^{o}\lambda_i\partial_\nu y_i\partial_\alpha\partial_\beta y_i\dot{\gamma}^\alpha\dot{\gamma}^\beta$ (term 1)

Require: $\dot{\gamma}$. Here we abbreviate $\mathcal{R}_{\dot{\gamma}}$ to $\mathcal{R}$ and $\mathcal{S}_{\dot{\gamma}}$ to $\mathcal{S}$.

$a_0 \leftarrow x$
$\mathcal{R}(a_0) \leftarrow 0$
$\mathcal{S}(a_0) \leftarrow 0$

for $i \leftarrow 1$ to l **do** ▷ forward pass
  $s_i \leftarrow W_i a_{i-1} + b_i$
  $a_i \leftarrow \phi_i(s_i)$
  $\mathcal{R}(s_i) \leftarrow \mathcal{R}(W_i)a_{i-1} + W_i\mathcal{R}(a_{i-1}) + \mathcal{R}(b_i)$
  $\mathcal{R}(a_i) \leftarrow \phi'(s_i)\odot\mathcal{R}(s_i)$
  $\mathcal{S}(s_i) \leftarrow \mathcal{S}(W_i)a_{i-1} + 2\mathcal{R}(W_i)\mathcal{R}(a_{i-1}) + W_i\mathcal{S}(a_{i-1}) + \mathcal{S}(b_i)$
  $\mathcal{S}(a_i) \leftarrow \phi''(s_i)\odot\mathcal{R}(s_i)^2 + \phi'(s_i)\odot\mathcal{S}(s_i)$
end for

Compute $\vec{\lambda}$ from a_l
$\mathcal{D}(a_l) = \vec{\lambda}\mathcal{S}(a_l)$

for $i = l$ to 1 **do** ▷ backward pass
  $\mathcal{D}(s_i) \leftarrow \mathcal{D}(a_i)\odot\phi'(s_i)$
  $\mathcal{D}(W_i) \leftarrow \mathcal{D}(s_i)a_{i-1}^\intercal$
  $\mathcal{D}(b_i) \leftarrow \mathcal{D}(s_i)$
  $\mathcal{D}(a_{i-1}) \leftarrow W_i^\intercal\mathcal{D}(s_i)$
end for
return $(\mathcal{D}(W_1),\cdots,\mathcal{D}(W_l),\mathcal{D}(b_1),\cdots,\mathcal{D}(b_l))$.

D. Practical Considerations

Practically, the Fisher information matrix could be ill-conditioned for inversion. In experiments, we compute $[g_{\mu\nu} + \epsilon\,\mathrm{diag}(g_{\mu\nu})]^{-1}\partial_\nu L$ instead of $(g_{\mu\nu})^{-1}\partial_\nu L$, where ϵ is the damping coefficient and $\mathrm{diag}(g_{\mu\nu})$ is the diagonal part of $g_{\mu\nu}$.

Algorithm 2 Calculating $\sum_{i=1}^{o} \lambda_i \partial_\nu y_i \partial_\alpha y_i \partial_\beta y_i \dot{\gamma}^\alpha \dot{\gamma}^\beta$ (term 2)

Require: $\dot{\gamma}$. Here we abbreviate $\mathcal{R}_{\dot{\gamma}}$ to $\mathcal{R}$.

$a_0 = x$
$\mathcal{R}(a_0) = 0$

for $i \leftarrow 1$ to l **do** ▷ forward pass
  $s_i \leftarrow W_i a_{i-1} + b_i$
  $a_i \leftarrow \phi(s_i)$
  $\mathcal{R}(s_i) \leftarrow \mathcal{R}(W_i) a_{i-1} + W_i \mathcal{R}(a_{i-1}) + \mathcal{R}(b_i)$
  $\mathcal{R}(a_i) \leftarrow \phi'(s_i) \odot \mathcal{R}(s_i)$
end for

Compute $\vec{\lambda}$ from a_l
$\mathcal{D}(a_l) = \vec{\lambda} \mathcal{R}(a_l)^2$

for $i = l$ to 1 **do** ▷ backward pass
  $\mathcal{D}(s_i) \leftarrow \mathcal{D}(a_i) \odot \phi'(s_i)$
  $\mathcal{D}(W_i) \leftarrow \mathcal{D}(s_i) a_{i-1}^\intercal$
  $\mathcal{D}(b_i) \leftarrow \mathcal{D}(s_i)$
  $\mathcal{D}(a_{i-1}) \leftarrow W_i^\intercal \mathcal{D}(s_i)$
end for
return $(\mathcal{D}(W_1), \cdots, \mathcal{D}(W_l), \mathcal{D}(b_1), \cdots, \mathcal{D}(b_l))$.

When $g_{\mu\nu}$ is too large to be inverted accurately, we use truncated conjugate gradient for solving the corresponding linear system.

Moreover, in line with the pioneering work of Martens (2010), we use backtracking search to adaptively shrink the step size and adopt a Levenberg-Marquardt style heuristic for adaptively choosing the damping coefficient.

As pointed out in Ollivier (2013), there are also two other sources of invariance loss, initialization and damping. Simple random initialization obviously depends on the network architecture. Unfortunately, there is no clear way to make it independent of parameterization. Large damping wipes out small eigenvalue directions and swerves optimization towards naïve gradient descent, which is not invariant. When the damping coefficient selected according to the Marquardt heuristic (Marquardt, 1963) is very large, it becomes meaningless to use either midpoint integrator or geodesic correction. In the experiments of training deep neural networks, we found it beneficial to set a threshold for the damping coefficient and switch off midpoint integrator or geodesic correction at the early stage of optimization when damping is very large.

E. Additional Details on Experimental Evaluations

E.1. Settings for Deep Neural Network Training

For the deep network experiments, we use the hyper-parameters in Martens (2010) as a reference, the modifications are that we fix the maximum number of CG iterations to 50 and maximum number of epochs to 140 for all algorithms and datasets. The initial damping coefficient is 45 across all tasks, and the damping thresholds for CURVES, MNIST and FACES are set to 5, 10 and 0.1 respectively. As mentioned in Appendix D, we use a threshold on the damping to switch on / off our corrections. However, in reality our methods will only be switched off for a small number of iterations in the early stage of training. Note that more careful tuning of these thresholds, *e.g.*, using different thresholds for different acceleration methods, may lead to better results.

Since both midpoint integrator and geodesic correction are direct modifications of natural gradient method, we incorporate all the improvements in Martens (2010), including sparse initialization, CG iteration backtracking, *etc*. For determining the learning rate $h\lambda$, we use the default value $h\lambda = 1$ with standard backtracking search. To highlight the effectiveness of the algorithmic improvements we introduced, the same set of hyper-parameters and random seed is used across all algorithms

on all datasets.

For deep autoencoders, network structures are the same as in Hinton & Salakhutdinov (2006) and Martens (2010) and we adopt their training / test partitions and choice of loss functions. For deep classifiers on MNIST, the network structure is 784-1000-500-250-30-10, all with fully connected layers, and as preprocessing, we center and normalize all training and test data.

Deep autoencoders for CURVES and MNIST datasets are trained with binary cross-entropy losses while FACES is trained with squared loss. All results are reported in squared losses. Although there is discrepancy between training and test losses, they align with each other pretty well and thus we followed the setting in (Martens, 2010). According to our observation, performance is robust to different random seeds and the learning curves measured by errors on training and test datasets are similar, except that we slightly overfitted FACES dataset.

Our implementation is based on the MATLAB code provided by (Martens, 2010). However, we used MATLAB Parallel Computing Toolbox for GPU, instead of the Jacket package used in (Martens, 2010), because Jacket is not available anymore. Computation times are not directly comparable as the Parallel Computing Toolbox is considerably slower than Jacket. The programs were run on Titan Xp GPUs.

E.2. Settings for Model-Free Reinforcement Learning

We consider common hyperparameter choices for ACKTR as well as our midpoint integrator and geodesic correction methods, where both the policy network and the value network is represented as a two layer fully-connected neural network with 64 neurons in each layer. Specifically, we consider our methods (and subsequent changes to the hyperparameters) only on the policy networks. We select constant learning rates for each environment since it eliminates the effect of the learning rate schedule in (Wu et al., 2017) over sample efficiency. The learning rates are set so that ACKTR achieves the highest episodic reward at 1 million timesteps. We select learning rates of 1.0, 0.03, 0.03, 0.03, 0.3, 0.01 for `HalfCheetah`, `Hopper`, `Reacher`, `Walker2d`, `InvertedPendulum`, `InvertedDoublePendulum` respectively. We set momentum to be zero for all methods, since we empirically find that this improves sample efficiency for ACKTR with the fixed learning rate schedule. For example, our ACKTR results for the `Walker2d` environment is over 1500 for 1 million timesteps, whereas (Wu et al., 2017) reports no more than 800 for the same number of timesteps (even with the learning rate schedule).

The code is based on OpenAI baselines (Dhariwal et al., 2017) and connection-vector products are computed with TensorFlow (Abadi et al., 2016) automatic differentiation.

F. Experiments on the Small-Curvature Approximation

Our geodesic correction is inspired by geodesic acceleration (Transtrum et al., 2011), a method to accelerate the Gauss-Newton algorithm for nonlinear least squares problems. In (Transtrum & Sethna, 2012), geodesic acceleration is derived from a high-order approximation to Hessian under the so-called *small-curvature assumption*. In this section, we demonstrate empirically that the small-curvature approximation generally does not hold for deep neural networks. To this end, we need to generalize the method in (Transtrum & Sethna, 2012) (which is only applicable to square loss) to general losses.

F.1. Derivation Based on Perturbation

It can be shown that Fisher information matrix is equivalent to the Gauss-Newton matrix when the loss function is appropriately chosen. Let's analyze the acceleration terms from this perspective.

Let the loss function be $\mathcal{L}(y, f)$ and $z^i(x;\theta)$, $i = 1, \cdots, o$ be the top layer values of the neural network. To show the equivalence of Gauss-Newton matrix and Fisher information matrix, we usually require $\mathcal{L}$ to also include the final layer activation (non-linearity) applied on z (Pascanu & Bengio, 2013; Martens, 2014). Hence different from y, z is usually the value *before* final layer activation function. To obtain the conventional Gauss-Newton update, we analyze the following problem:

$$\min_{\delta\theta} \sum_{(x,y)\in S} \mathcal{L}(y, z + \partial_j z \delta\theta^j) + \kappa F_{ij}\delta\theta^i\delta\theta^j,$$

where S is the training dataset and F is a metric measuring the distance between two models with parameter difference $\delta\theta$.

Note that without loss of generality, we omit $\sum_{(x,y)\in S}$ in the sequel.

By approximating $\mathcal{L}(y,\cdot)$ with a second-order Taylor expansion, we obtain

$$\mathcal{L}(y,z)+\partial_k\mathcal{L}(y,z)\partial_j z^k\delta\theta^j+\frac{1}{2}\partial_m\partial_n\mathcal{L}(y,z)\partial_i z^m\partial_j z^n\delta\theta^i\delta\theta^j+\kappa F_{ij}\delta\theta^i\delta\theta^j.$$

The normal equations obtained by setting derivatives to 0 are

$$\left(\partial_m\partial_n\mathcal{L}\partial_i z^m\partial_j z^n+2\kappa F_{ij}\right)\delta\theta^j=-\partial_k\mathcal{L}(y,z)\partial_i z^k,$$

which exactly gives the natural gradient update

$$\delta\theta_1^j=-\left(\partial_m\partial_n\mathcal{L}\partial_i z^m\partial_j z^n+\lambda F_{ij}\right)^{-1}\partial_k\mathcal{L}(y,z)\partial_i z^k.$$

where we fold 2κ to λ. Hence natural gradient is an approximation to the Hessian with linearized model output $z+\partial_j z\delta\theta^j$.

Now let us correct the error of linearizing z using higher order terms, *i.e.*,

$$\mathcal{L}\left(y,z+\partial_j z\delta\theta^j+\frac{1}{2}\partial_j\partial_l z\delta\theta^j\delta\theta^l\right)+\lambda F_{ij}\delta\theta^i\delta\theta^j.$$

Expanding $\mathcal{L}(y,\cdot)$ to second-order gives us

$$\begin{aligned}\mathcal{L}(y,z)+\partial_k\mathcal{L}(y,z)\left(\partial_j z^k\delta\theta^j+\frac{1}{2}\partial_j\partial_l z^k\delta\theta^j\delta\theta^l\right)+\\ \frac{1}{2}\partial_m\partial_n\mathcal{L}(y,z)\left(\partial_j z^m\delta\theta^j+\frac{1}{2}\partial_j\partial_l z^m\delta\theta^j\delta\theta^l\right)\left(\partial_k z^n\delta\theta^k+\frac{1}{2}\partial_k\partial_p z^n\delta\theta^k\delta\theta^p\right)+\lambda F_{ij}\delta\theta^i\delta\theta^j.\end{aligned}$$

The normal equations are

$$\begin{aligned}\partial_k\mathcal{L}(y,z)\partial_\mu z^k+(\partial_k\mathcal{L}(y,z)\partial_\mu\partial_j z^k+\partial_m\partial_n\mathcal{L}(y,z)\partial_\mu z^m\partial_j z^n+\lambda F_{\mu j})\delta\theta^j+\\ \left(\partial_m\partial_n\mathcal{L}(y,z)\partial_\mu\partial_j z^m\partial_k z^n+\frac{1}{2}\partial_m\partial_n\mathcal{L}(y,z)\partial_j\partial_k z^m\partial_\mu z^n\right)\delta\theta^j\delta\theta^k=0.\end{aligned}$$

Let $\delta\theta=\delta\theta_1+\delta\theta_2$ and assume $\delta\theta_2$ to be small. Dropping in $\delta\theta_1$ will turn the normal equation to

$$\begin{aligned}(\partial_k\mathcal{L}(y,z)\partial_i\partial_j z^k+\partial_m\partial_n\mathcal{L}(y,z)\partial_i z^m\partial_j z^n+\lambda F_{ij})\delta\theta_2^j+\partial_k\mathcal{L}(y,z)\partial_i\partial_j z^k\delta\theta_1^j+\\ \left(\partial_m\partial_n\mathcal{L}(y,z)\partial_\mu\partial_j z^m\partial_k z^n+\frac{1}{2}\partial_m\partial_n\mathcal{L}(y,z)\partial_j\partial_k z^m\partial_\mu z^n\right)\delta\theta_1^j\delta\theta_1^k=0.\end{aligned}$$

The approximation for generalized Gauss-Newton matrix is $\partial_k\mathcal{L}(y,z)\partial_i\partial_j z^k=0$. After applying it to $\delta\theta_2$, we have

$$\begin{aligned}\delta\theta_2^\mu=-(\partial_m\partial_n\mathcal{L}(y,z)\partial_i z^m\partial_j z^n+\lambda F_{ij})^{-1}\\ \left[\left(\partial_m\partial_n\mathcal{L}(y,z)\partial_\mu\partial_j z^m\partial_k z^n+\frac{1}{2}\partial_m\partial_n\mathcal{L}(y,z)\partial_j\partial_k z^m\partial_\mu z^n\right)\delta\theta_1^j\delta\theta_1^k+\partial_k\mathcal{L}(y,z)\partial_\mu\partial_j z^k\delta\theta_1^j\right].\end{aligned}\tag{20}$$

If we combine $\partial_k\mathcal{L}(y,z)\partial_\mu\partial_j z^k\delta\theta_1^j$ and $[\partial_m\partial_n\mathcal{L}(y,z)\partial_\mu\partial_j z^m\partial_k z^n]\delta\theta_1^j\delta\theta_1^k$ and use the following *small-curvature* approximation (Transtrum & Sethna, 2012)

$$\partial_l\mathcal{L}[\delta_{ml}-(\partial_\alpha\partial_\beta\mathcal{L}\partial_i f^\alpha\partial_k f^\beta)^{-1}\partial_i f^l\partial_m\partial_n\mathcal{L}\partial_k z^n]\partial_\mu\partial_j z^m=0,$$

we will obtain an expression of $\delta\theta_2$ which is closely related to the geodesic correction term.

$$\delta\theta_2^\mu=-\frac{1}{2}(\partial_m\partial_n\mathcal{L}(y,z)\partial_i z^m\partial_j z^n+\lambda F_{ij})^{-1}\left(\partial_m\partial_n\mathcal{L}(y,z)\partial_j\partial_k z^m\partial_\mu z^n\right)\delta\theta_1^j\delta\theta_1^k.\tag{21}$$

The final update rule is

$$\delta\theta = \delta\theta_1 + \delta\theta_2.$$

Whenever the loss function satisfies $\mathcal{L}(y,z) = -\log r(y \mid z) = -z^\intercal T(y) + \log Z(z)$ we have $\mathbf{F}_\theta = \sum_{(x,y)\in S} \partial_m\partial_n\mathcal{L}(y,z)\partial_i\partial_j z^m z^n$, *i.e.*, Gauss-Newton method coincides with natural gradient and the Fisher information matrix is the Gauss-Newton matrix.

Here are the formulas for squared loss and binary cross-entropy loss, where we follow the notation in the main text and denote y as the final network output *after* activation.

Proposition 4. *For linear activation function and squared loss, we have the following formulas*

$$\begin{aligned}
\partial_m\partial_n\mathcal{L}(t,f)\partial_\mu\partial_j z^m\partial_k z^n\delta\theta_1^j\delta\theta_1^k &= \frac{1}{\sigma^2}\partial_\mu\partial_j y^m\partial_k y^m\delta\theta_1^j\delta\theta_1^k \\
\frac{1}{2}\partial_m\partial_n\mathcal{L}(t,f)\partial_j\partial_k z^m\partial_\mu z^n\delta\theta_1^j\delta\theta_1^k &= \frac{1}{2\sigma^2}\partial_j\partial_k y^m\partial_\mu y^m\delta\theta_1^j\delta\theta_1^k \\
\partial_k\mathcal{L}(t,f)\partial_\mu\partial_j z^k\delta\theta_1^j &= \frac{1}{\sigma^2}(y_k - t_k)\partial_\mu\partial_j y^k\delta\theta_1^j.
\end{aligned}$$

In this case, (21) is equivalent to $-\frac{1}{2}\Gamma^\mu_{jk}\delta\theta_1^j\delta\theta_1^k$, which is our geodesic correction term for squared loss.

Proposition 5. *For sigmoid activation function and binary cross-entropy loss, we have the following terms*

$$\begin{aligned}
y_i := \mathrm{sigmoid}(z) &:= \frac{1}{1+e^{-z^i}} \\
\partial_m\partial_n\mathcal{L}(t,f) &= \delta_{mn}y_m(1-y_m) \\
\partial_m\partial_n\mathcal{L}(t,f)\partial_\mu\partial_j z^m\partial_k z^n\delta\theta_1^j\delta\theta_1^k &= \left[\frac{1}{y_m(1-y_m)}\partial_\mu\partial_j y_m\partial_k y_m + \right. \\
&\quad \left.\frac{2y_m-1}{y_m^2(1-y_m)^2}\partial_\mu y_m\partial_j y_m\partial_k y_m\right]\delta\theta_1^j\delta\theta_1^k \\
\frac{1}{2}\partial_m\partial_n\mathcal{L}(t,f)\partial_j\partial_k z^m\partial_\mu z^n\delta\theta_1^j\delta\theta_1^k &= \frac{1}{2}\left[\frac{1}{y_m(1-y_m)}\partial_j\partial_k y_m\partial_\mu y_m\right. \\
&\quad \left.+\frac{2y_m-1}{y_m^2(1-y_m)^2}\partial_j y_m\partial_k y_m\partial_\mu y_m\right]\delta\theta_1^j\delta\theta_1^k \\
\partial_k\mathcal{L}(t,f)\partial_\mu\partial_j z^k\delta\theta_1^j &= \left[\frac{1}{y_k(1-y_k)}\partial_\mu\partial_j y_k + \frac{2y_k-1}{y_k^2(1-y_k)^2}\partial_\mu y_k\partial_j y_k\right](y_k-t_k)\delta\theta_1^j
\end{aligned}$$

In this case, (21) will give a similar result as geodesic correction, which is $-\frac{1}{2}\Gamma^{(1)\mu}_{jk}\delta\theta_1^j\delta\theta_1^k$. The only difference is using 1-connection $\Gamma^{(1)\mu}_{jk}$ (Amari et al., 1987) instead of Levi-Civita connection Γ^μ_{jk}.

We also need an additional algorithm to calculate $\lambda_i\partial_\nu\partial_\alpha y_i\partial_\beta y_i\dot\theta^\alpha\dot\theta^\beta$, as provided by Alg. 3.

F.2. Empirical Results

If the small curvature approximation holds, (20) should perform similarly as (21). The power of geodesic correction can thus be viewed as a higher-order approximation to the Hessian than natural gradient / Gauss-Newton and the interpretation of preserving higher-order invariance would be doubtful.

In order to verify the small curvature approximation, we use both (20) (named "**perturb**") and (21) (named "**geodesic**") for correcting the natural gradient update. The difference between their performance shows how well small curvature approximation holds. Using the same settings in main text, we obtain the results on CURVES, MNIST and FACES.

As shown in Figure 5, (20) never works well except for the beginning. This is anticipated since Newton's method is susceptible to negative curvature and will blow up at saddle points (Dauphin et al., 2014). Therefore the direction of approximating Newton's method more accurately is not reasonable. The close match of (20) and (21) indicates that the

Algorithm 3 An algorithm for computing $\lambda_i \partial_\nu \partial_\alpha y_i \partial_\beta y_i \dot{\theta}^\alpha \dot{\theta}^\beta$ (term 3)

Require: $\dot{\theta}$. We abbreviate $\mathcal{R}_{\dot{\theta}}$ to $\mathcal{R}$.

$\mathcal{R}(a_0) \leftarrow 0$ ▷ Since a_0 is not a function of the parameters

for $i \leftarrow 1$ to l **do** ▷ forward pass
  $\mathcal{R}(s_i) \leftarrow \mathcal{R}(W_i) a_{i-1} + W_i \mathcal{R}(a_{i-1}) + \mathcal{R}(b_i)$ ▷ product rule
  $\mathcal{R}(a_i) \leftarrow \mathcal{R}(s_i) \phi_i'(s_i)$ ▷ chain rule
end for

▷ By here we have computed all $\partial_\beta y_i \dot{\theta}^\beta$

Compute $\vec{\lambda}$ from a_l

Let $L = \sum_{i=1}^{o} \lambda_i \mathcal{R}(a_l^i) y_i$

$\mathcal{R}(\mathcal{D}(a_l)) \leftarrow \mathcal{R}\left(\frac{\partial L}{\partial \mathbf{y}}\Big|_{\mathbf{y}=a_l} \right) = \frac{\partial^2 L}{\partial \mathbf{y}^2}\Big|_{\mathbf{y}=a_l} \mathcal{R}(a_l) = 0$

$\mathcal{D}a_l \leftarrow \frac{\partial L}{\partial \mathbf{y}}\Big|_{\mathbf{y}=a_l} = (\lambda_1(\mathcal{R}a_l^1), \cdots, \lambda_o(\mathcal{R}a_l^o))$

for $i \leftarrow l$ to 1 **do**
  $\mathcal{D}(s_i) \leftarrow \mathcal{D}(a_i) \odot \phi_i'(s_i)$
  $\mathcal{D}(W_i) \leftarrow \mathcal{D}(s_i) a_{i-1}^\intercal$
  $\mathcal{D}(b_i) \leftarrow \mathcal{D}(s_i)$
  $\mathcal{D}(a_{i-1}) \leftarrow W_i^\intercal \mathcal{D}(s_i)$
  $\mathcal{R}(\mathcal{D}(s_i)) \leftarrow \mathcal{R}(\mathcal{D}(a_i)) \odot \phi_i'(s_i) + \mathcal{D}(a_i) \odot \mathcal{R}(\phi_i'(s_i)) = \mathcal{R}(\mathcal{D}(a_i)) \odot \phi_i'(s_i) + \mathcal{D}(a_i) \odot \phi_i''(s_i) \odot \mathcal{R}(s_i)$
  $\mathcal{R}(\mathcal{D}(W_i)) \leftarrow \mathcal{R}(\mathcal{D}(s_i)) a_{i-1}^\intercal + \mathcal{D}(s_i) \mathcal{R}(a_{i-1}^\intercal)$
  $\mathcal{R}(\mathcal{D}(b_i)) \leftarrow \mathcal{R}(\mathcal{D}(s_i))$
  $\mathcal{R}(\mathcal{D}(a_{i-1})) \leftarrow \mathcal{R}(W_i^\intercal) \mathcal{D}(s_i) + W_i^\intercal \mathcal{R}(\mathcal{D}(s_i))$
end for
return $\lambda_i \partial_\nu \partial_\alpha y_i \partial_\beta y_i \dot{\theta}^\alpha \dot{\theta}^\beta = (\mathcal{R}(\mathcal{D}(W_i)), \cdots, \mathcal{R}(\mathcal{D}(W_l)), \mathcal{R}(\mathcal{D}(b_1)), \cdots, \mathcal{R}(\mathcal{D}(b_l)))$

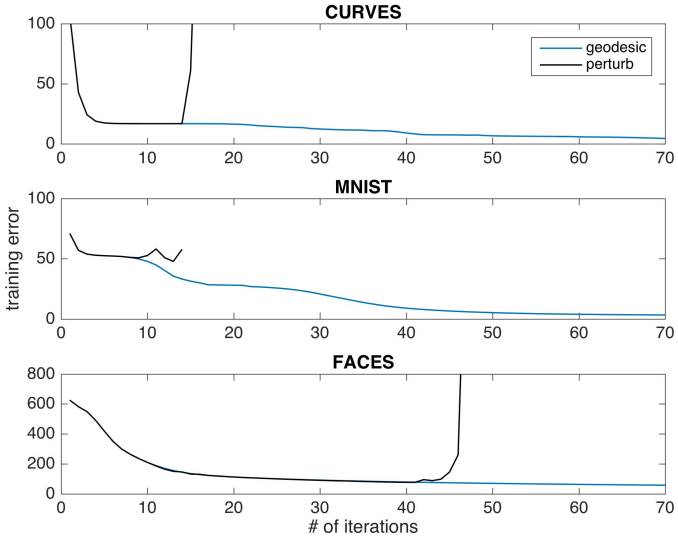

Figure 5. Study of small curvature approximation on different datasets.

small curvature assumption indeed holds temporarily, and Newton's method is very close to natural gradient at the beginning. However, the latter divergence of (20) demonstrates that the effectiveness of geodesic correction does not come from approximating Newton's method to a higher order.