Supplementary Materials for Understanding Synthetic Gradients and Decoupled Neural Interfaces

A. Relation to critic methods

Instead of estimating the gradient directly, one could estimate loss instead (thus use some trainable $\phi(h|\theta) \approx \mathbb{E}[L|h]$) and then use its gradient wrt. to its inputs ($\partial\phi/\partial h$) as a surrogate for the synthetic gradient. These kind of approaches are known in Reinforcement Learning as critic methods (Fairbank, 2014; Heess et al., 2015), but in terms of gradient approximation they do not guarantee any alignment between these signals if only critic is a non-linear function. As an example lets consider a function $\phi(h(x_i)|\theta) = L(h(x_i))$ for every x_i, such that it is constant (in terms of its output) in some ϵ balls around each x_i. As a consequence gradients of ϕ are 0 everywhere, yet as a critic it receives no learning signal (since loss is approximated perfectly). This example shows that in general alignment between critic gradient and true gradient can be arbitrary, and completely independent from the loss error itself.

B. Additional examples

Critical points

We can show an example of SG introducing new critical points. Consider a small one-dimensional training dataset $\{-2, -1, 1, 2\} \subset \mathbb{R}$, and let us consider a simple system where the model $f : \mathbb{R} \to \mathbb{R}$ is parametrised with two scalars, a and b and produces $ax + b$. We train it to minimise $L(a, b) = \sum_{i=1}^{4} |ax_i + b|$. This has a unique minimum which is obtained for $a = b = 0$, and standard gradient based methods will converge to this solution. Let us now attach a SG module between f and L. This module produces a (trainable) scalar value $c \in \mathbb{R}$ (thus it produces a single number, independent from the input). Regardless of the value of a, we have a critical point of the SG module when $b = 0$ and $c = 0$. However, solutions with $a = 1$ and $c = 0$ are clearly not critical points of the original system. Figure 6 shows the loss surface and the fitting of SG module when it introduces new critical point.

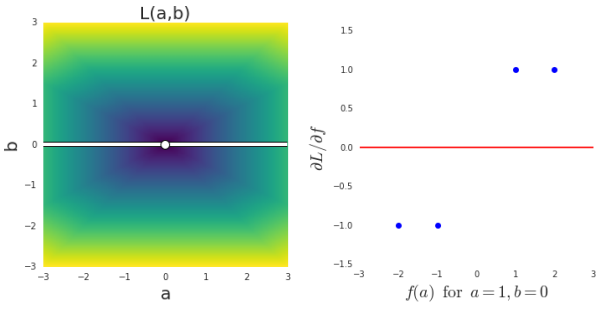


Figure 6. Left: The loss surface with a white marker represents critical point of the original optimisation and white line a set of critical points of SG based one. *Right:* A situation when SG finds a solution $d = 0$ which introduces new critical point, which is not a critical point of the original problem.

C. Proofs

Theorem 1 *Let us consider linear regression trained with a linear SG module attached between its output and the loss. If one chooses the learning rate of the SG module using line search, then in every iteration there exists small enough, positive learning rate of the main network such that it converges to the global solution.*

Proof. Let $\mathbf{X} = \{\mathbf{x}^s\}_{s=1}^S \in \mathbb{R}^{d\times S}$ be the data, let $\{y_s\}_{s=1}^S \in \mathbb{R}^{1\times S}$ be the labels. Throughout the proof k will be the iteration of training.
We denote by $\mathbf{1} \in \mathbb{R}^{1\times S}$ a row vector in which every element is 1. We also follow the standard convention of including the bias in the weight matrix by augmenting the data $\mathbf{X}$ with one extra coordinate always equal to 1. Thus, we denote $\bar{\mathbf{X}} = (\mathbf{X}^T|\mathbf{1}^T)^T$, $\bar{\mathbf{X}} \in \mathbb{R}^{(d+1)\times S}$ and $\bar{\mathbf{x}}^s$-the columns of $\bar{\mathbf{X}}$. Using that convention, the weight matrix is $\mathbf{W}_k \in \mathbb{R}^{1\times(d+1)}$. We have

$$p_k^s := \mathbf{W}_k \bar{\mathbf{x}}^s,$$

$$L = \frac{1}{2}\sum_{s=1}^{S}(y^s - p_k^s)^2 = \frac{1}{2}\sum_{i=1}^{n}(y^s - \mathbf{W}_k\bar{\mathbf{x}}^s)^2.$$

Our aim is to find

$$\arg\min_{\mathbf{W},b} L.$$

We use

$$\frac{\partial L}{\partial \mathbf{W}} = \frac{\partial L}{\partial \mathbf{p}}\frac{\partial \mathbf{p}}{\partial \mathbf{W}} = \sum_{s=1}^{S}\frac{\partial L}{\partial p^s}\frac{\partial p^s}{\partial \mathbf{W}} =$$

$$\sum_{s=1}^{S} \frac{\partial L}{\partial p^s} \bar{\mathbf{x}}^s = \sum_{s=1}^{S} \left(y^s - \mathbf{W}_k \bar{\mathbf{x}}^s\right) \left(\bar{\mathbf{x}}^s\right)^T$$

$$\frac{\partial L}{\partial \mathbf{p}} = \left(p^1 - y^1, \ldots, p^S - y^S\right)$$

We will use the following parametrization of the synthetic gradient $\widetilde{\nabla \mathcal{L}}_k = (\alpha_k+1)\mathbf{p}_k - (\beta_k+1)\mathbf{y} + \gamma_k \mathbf{1}$. The reason for using this form instead of simply $a_k \mathbf{p}_k + b_k \mathbf{y} + c_k \mathbf{1}$ is that we are going to show that under DNI this synthetic gradient will converge to the "real gradient" $\frac{\partial L}{\partial \mathbf{p}}$, which means showing that $\lim_{k\to\infty} (\alpha_k, \beta_k, \gamma_k) = (0, 0, 0)$. Thanks to this choice of parameters α_k, β_k, γ_k we have the simple expression for the error

$$E_k = \left\| \widetilde{\nabla \mathcal{L}}_k - \frac{\partial L}{\partial \mathbf{p}} \right\|_2^2 =$$

$$\|(\alpha_k + 1)\mathbf{p}_k - (\beta_k + 1)\mathbf{y} + \gamma_k \mathbf{1} -$$

$$\left(p_k^1 - y^1, \ldots, p_k^S - y^S\right)\|_2^2 =$$

$$\left\|\left(\alpha_k p_k^1 - \beta_k y^1 + \gamma_k, \ldots, \alpha_k p_k^S - \beta_k y^S + \gamma_k\right)\right\|_2^2$$

Parameters α_k, β_k, γ_k will be updated using the gradient descent minimizing the error E. We have

$$\frac{\partial E}{\partial \alpha} = \sum_{s=1}^{S} (\alpha_k p_k^s - \beta_k y^s + \gamma_k) p_k^s$$

$$\frac{\partial E}{\partial \beta} = -\sum_{s=1}^{S} (\alpha_k p_k^s - \beta_k y^s + \gamma_k) y^s$$

$$\frac{\partial E}{\partial \gamma} = \sum_{s=1}^{S} (\alpha_k p_k^s - \beta_k y^s + \gamma_k).$$

As prescribed in Jaderberg et al. (2016), we start our iterative procedure from the synthetic gradient being equal to zero and we update the parameters by adding the (negative) gradient multiplied by a learning rate ν. This means that we apply the iterative procedure:

$$\alpha_0 = -1, \;\; \beta_0 = -1, \;\; \gamma_0 = 0$$

$$\begin{aligned}
\mathbf{W}_{k+1} =& \mathbf{W}_k - \mu \sum_{s=1}^{S} \left((\alpha_k + 1)\mathbf{p}_k^s - \right. \\
& \left. (\beta_k + 1)\mathbf{y}^s + \gamma_k\right) \left(\bar{\mathbf{x}}^s\right)^T \\
\alpha_{k+1} =& \alpha_k - \nu \sum_{s=1}^{S} (\alpha_k p_k^s - \beta_k y^s + \gamma_k) p_k^s \\
\beta_{k+1} =& \beta_k + \nu \sum_{s=1}^{S} (\alpha_k p_k^s - \beta_k y^s + \gamma_k) y^s \\
\gamma_{k+1} =& \gamma_k - \nu \sum_{s=1}^{S} (\alpha_k p_k^s - \beta_k y^s + \gamma_k).
\end{aligned}$$

Using matrix notation

$$\begin{aligned}
\mathbf{W}_{k+1} &= \mathbf{W}_k - \mu((\alpha_k + 1)\mathbf{p}_k - (\beta_k + 1)\mathbf{y} + \gamma_k \mathbf{1})\bar{\mathbf{X}}^T \\
\alpha_{k+1} &= \alpha_k - \nu \left(\alpha_k \|\mathbf{p}_k\|_2^2 - \beta_k \langle \mathbf{y}, \mathbf{p}_k \rangle + \gamma_k \langle \mathbf{1}, \mathbf{p}_k \rangle\right) \\
\beta_{k+1} &= \beta_k + \nu \left(\alpha_k \langle \mathbf{p}_k, \mathbf{y} \rangle - \beta_k \|\mathbf{y}\|_2^2 + \gamma_k \langle \mathbf{1}, \mathbf{y} \rangle\right) \\
\gamma_{k+1} &= \gamma_k - \nu \left(\alpha_k \langle \mathbf{1}, \mathbf{p}_k \rangle - \beta_k \langle \mathbf{1}, \mathbf{y} \rangle + S\gamma_k\right)
\end{aligned}$$

Note, that the subspace given by $\alpha = \beta = \gamma = 0$ is invariant under this mapping. As noted before, this corresponds to the synthetic gradient being equal to the real gradient. Proving the convergence of SG means showing, that a trajectory starting from $\alpha_0 = -1$, $\beta_0 = -1$, $\gamma_0 = 0$ converges to $\mathbf{W} = \mathbf{W}_0$, $\alpha = \beta = \gamma = 0$, where $\mathbf{W}_0$ are the "true" weigts of the linear regression. We are actually going to prove more, we will show that $\mathbf{W} = \mathbf{W}_0$, $\alpha = \beta = \gamma = 0$ is in fact a global attractor, i.e. that any trajectory converges to that point. Denoting $\omega = (\alpha, \beta, \gamma)^t$ we get

$$\begin{aligned}
\mathbf{W}_{k+1} &= \mathbf{W}_k - \mu((\alpha_k + 1)\mathbf{p}_k - (\beta_k + 1)\mathbf{y} + \gamma_k \mathbf{1})\bar{\mathbf{X}}^T \\
\omega_{k+1} &= \omega_k - \nu \left[\mathbf{p}_k^T| - \mathbf{y}^T|\mathbf{1}^T\right]^T \left[\mathbf{p}_k^T| - \mathbf{y}^T|\mathbf{1}^T\right] \omega_k
\end{aligned}$$

$$\begin{aligned}
\mathbf{W}_{k+1} &= \mathbf{W}_k - \mu(\mathbf{p}_k - \mathbf{y})\bar{\mathbf{X}}^T - \mu\omega_k^T \left[\mathbf{p}_k^T| - \mathbf{y}^T|\mathbf{1}^T\right]^T \bar{\mathbf{X}}^T \\
\omega_{k+1} &= \omega_k - \nu \left[\mathbf{p}_k^T| - \mathbf{y}^T|\mathbf{1}^T\right]^T \left[\mathbf{p}_k^T| - \mathbf{y}^T|\mathbf{1}^T\right] \omega_k.
\end{aligned}$$

Denoting by $\mathbf{A}_k = \left[\mathbf{p}_k^T| - \mathbf{y}^T|\mathbf{1}^T\right]$ we get

$$\begin{aligned}
\mathbf{W}_{k+1} &= \mathbf{W}_k - \mu(\mathbf{p}_k - \mathbf{y})\bar{\mathbf{X}}^T - \mu\omega^T \mathbf{A}_k^T \bar{\mathbf{X}}^T \\
\omega_{k+1} &= \omega_k - \nu \mathbf{A}_k^T \mathbf{A}_k \omega_k.
\end{aligned}$$

Multiplying both sides of the first equation by $\bar{\mathbf{X}}$ we obtain

$$\begin{aligned}
\mathbf{W}_{k+1}\bar{\mathbf{X}} &= \mathbf{W}_k\bar{\mathbf{X}} - \mu(\mathbf{p}_k - \mathbf{y})\bar{\mathbf{X}}^T\bar{\mathbf{X}} - \mu\omega^T \mathbf{A}_k^T \bar{\mathbf{X}}^T\bar{\mathbf{X}} \\
\omega_{k+1} &= \omega_k - \nu \mathbf{A}_k^T \mathbf{A}_k \omega_k.
\end{aligned}$$

Denote $\boldsymbol{B} = \bar{\mathbf{X}}^T\bar{\mathbf{X}}$. We get

$$\begin{aligned}
\mathbf{p}_{k+1} &= \mathbf{p}_k - \mu\mathbf{p}_k\mathbf{B} + \mu\mathbf{y}\mathbf{B} - \mu\omega_k^T \mathbf{A}_k^T \mathbf{B} \\
\omega_{k+1} &= \omega_k - \nu \mathbf{A}_k^T \mathbf{A}_k \omega_k.
\end{aligned}$$

Denoting $\mathbf{e}_k = (\mathbf{y} - \mathbf{p}_k)^T$ we get

$$\begin{aligned}
\mathbf{e}_{k+1} &= \mathbf{e}_k - \mu\mathbf{B}\mathbf{e}_k + \mu\mathbf{B}\mathbf{A}_k\omega_k \\
\omega_{k+1} &= \omega_k - \nu \mathbf{A}_k^T \mathbf{A}_k \omega_k.
\end{aligned}$$

We will use the symbol $\xi = \mathbf{A}_k\omega_k$. Then

$$\begin{aligned}
\mathbf{e}_{k+1} &= \mathbf{e}_k - \mu\mathbf{B}\mathbf{e}_k + \mu\mathbf{B}\xi_k \\
\xi_{k+1} &= \xi_k - \nu \mathbf{A}_k \mathbf{A}_k^T \xi_k.
\end{aligned} \tag{1}$$

Every vector v can be uniquely expressed as a sum $v = v^\perp + v^\parallel$ with $\bar{\mathbf{X}}v^\perp = \mathbf{0}$ and $v^\parallel = \bar{\mathbf{X}}^T\theta$ for some θ ($v^\parallel$ is a projection of v onto the linear subspace spanned by

the columns of $\bar{\mathbf{X}}$). Applying this decomposition to $\mathbf{e}_k = \mathbf{e}_k^\perp + \mathbf{e}_k^\parallel$ we get

$$\begin{aligned}\mathbf{e}_{k+1}^\perp &= \mathbf{e}_k^\perp - \mu(\boldsymbol{B}\mathbf{e}_k)^\perp + \mu(\boldsymbol{B}\xi_k)^\perp\\ \mathbf{e}_{k+1}^\parallel &= \mathbf{e}_k^\parallel - \mu(\boldsymbol{B}\mathbf{e}_k)^\parallel + \mu(\boldsymbol{B}\xi_k)^\parallel\\ \xi_{k+1} &= \xi_k - \nu\mathbf{A}_k\mathbf{A}_k^T\xi_k.\end{aligned}$$

Note now, that as $\boldsymbol{B} = \bar{\mathbf{X}}^T\bar{\mathbf{X}}$, for any vector v there is $(\boldsymbol{B}v)^\perp = \mathbf{0}$, and $(\boldsymbol{B}v)^\parallel = \boldsymbol{B}v$ (because the operator $v \mapsto v^\parallel$ is a projection). Moreover, $\boldsymbol{B}v = \boldsymbol{B}v^\parallel$. Therefore

$$\begin{aligned}\mathbf{e}_{k+1}^\perp &= \mathbf{e}_k^\perp\\ \mathbf{e}_{k+1}^\parallel &= \mathbf{e}_k^\parallel - \mu(\boldsymbol{B}\mathbf{e}_k^\parallel) + \mu(\boldsymbol{B}\xi_k)^\parallel\\ \xi_{k+1} &= \xi_k - \nu\mathbf{A}_k\mathbf{A}_k^T\xi_k.\end{aligned}$$

The value $\mathbf{e}_k^\perp$ does not change. Thus, we will be omitting the first equation. Note, that $\mathbf{e}_k^\perp$ is "the residue", the smallest error that can be obtained by a linear regression.
For the sake of visual appeal we will denote $\mathbf{f} = \mathbf{e}_k^\parallel$

$$\begin{aligned}\mathbf{f}_{k+1} &= \mathbf{f}_k - \mu\boldsymbol{B}\mathbf{f}_k + \mu\boldsymbol{B}\xi_k\\ \xi_{k+1} &= \xi_k - \nu\mathbf{A}_k\mathbf{A}_k^T\xi_k.\end{aligned}$$

Taking norms and using $\|u+v\| \le \|u\| + \|v\|$ we obtain

$$\begin{aligned}\|\mathbf{f}_{k+1}\|_2 &\le \|\mathbf{f}_k - \mu\boldsymbol{B}\mathbf{f}_k\|_2 + \mu\|\boldsymbol{B}\xi_k\|_2\\ \|\xi_{k+1}\|_2^2 &= \|\xi_k\|_2^2 - 2\nu\|\mathbf{A}_k^T\xi_k\|_2^2 + \nu^2\|\mathbf{A}_k\mathbf{A}_k^T\xi_k\|_2^2.\end{aligned}$$

Observe that $\|\mathbf{f}_k - \mu\boldsymbol{B}\mathbf{f}_k\|_2^2 = \|\mathbf{f}_k\|_2^2 - 2\mu\mathbf{f}_k\boldsymbol{B}\mathbf{f}_k + \mu^2\|\boldsymbol{B}\mathbf{f}_k\|_2^2$. As $\boldsymbol{B}$ is a constant matrix, there exists a constant $b > 0$ such that $v^T\boldsymbol{B}v \ge b\|v\|_2^2$ for any v satisfying $v^\parallel = v$. Therefore $\|\mathbf{f}_k - \mu\boldsymbol{B}\mathbf{f}_k\|_2^2 \le \|\mathbf{f}_k\|_2^2 - 2\mu b\|\mathbf{f}_k\|_2^2 + \mu^2\|\boldsymbol{B}\|^2\|\mathbf{f}_k\|_2^2$. Using that and $\|\boldsymbol{B}\xi_k\|_2 \le \|\boldsymbol{B}\|\|\xi_k\|_2$ we get

$$\begin{aligned}\|\mathbf{f}_{k+1}\|_2 &\le \sqrt{1-2\mu b+\mu^2\|\boldsymbol{B}\|^2}\|\mathbf{f}_k\|_2 + \mu\|\boldsymbol{B}\|\|\xi_k\|_2\\ \|\xi_{k+1}\|_2^2 &= \|\xi_k\|_2^2 - 2\nu\|\mathbf{A}_k^T\xi_k\|_2^2 + \nu^2\|\mathbf{A}_k\mathbf{A}_k^T\xi_k\|_2^2.\end{aligned}$$

Let us assume that $\mathbf{A}_k\mathbf{A}_k^T\xi_k \neq 0$. In that case the right-hand side of the second equation is a quadratic function is ν, whose minimum value is attained for $\nu = \frac{\|\mathbf{A}_k^T\xi_k\|_2^2}{\|\mathbf{A}_k\mathbf{A}_k^T\xi_k\|_2^2}$. For so-chosen ν we have

$$\begin{aligned}\|\mathbf{f}_{k+1}\|_2 &\le \sqrt{1-2\mu b+\mu^2\|\boldsymbol{B}\|^2}\|f_k\|_2 + \mu\|\boldsymbol{B}\|\|\xi_k\|_2\\ \|\xi_{k+1}\|_2^2 &= \left(1 - \frac{\|\mathbf{A}_k^T\xi_k\|_2^2}{\|\mathbf{A}_k\mathbf{A}_k^T\xi_k\|_2^2}\frac{\|\mathbf{A}_k^T\xi_k\|_2^2}{\|\xi_k\|_2^2}\right)\|\xi_k\|_2^2.\end{aligned}$$

Consider a space $\{\mathbf{f}\}\oplus\{\xi\}$ (concatenation of vectors) with a norm $\|\{\mathbf{f}\}\oplus\{\xi\}\|_\oplus = \|\mathbf{f}\|_2 + \|\xi\|_2$.

$$\|\{\mathbf{f}_{k+1}\}\oplus\{\xi_{k+1}\}\|_\oplus \le$$

$$\sqrt{1-2\mu b+\mu^2\|\boldsymbol{B}\|^2}\|f_k\|_2 + \mu\|\boldsymbol{B}\|\|\xi_k\|_2 +$$

$$\sqrt{1 - \frac{\|\mathbf{A}_k^T\xi_k\|_2^2}{\|\mathbf{A}_k\mathbf{A}_k^T\xi_k\|_2^2}\frac{\|\mathbf{A}_k^T\xi_k\|_2^2}{\|\xi_k\|_2^2}}\|\xi_k\|_2 \le$$

Using $\sqrt{1-h} \le 1 - \frac{1}{2}h$ we get

$$\|\{\mathbf{f}_{k+1}\}\oplus\{\xi_{k+1}\}\|_\oplus \le \sqrt{1-2\mu b+\mu^2\|\boldsymbol{B}\|^2}\|f_k\|_2 +$$

$$\left(1 - \frac{\|\mathbf{A}_k^T\xi_k\|_2^2}{2\|\mathbf{A}_k\mathbf{A}_k^T\xi_k\|_2^2}\frac{\|\mathbf{A}_k^T\xi_k\|_2^2}{\|\xi_k\|_2^2} + \mu\right)\|\xi_k\|_2$$

Note, that $\sqrt{1-2\mu b+\mu^2\|\boldsymbol{B}\|^2} < 1$ for $0 < \mu \le \frac{b}{\|\boldsymbol{B}\|^2}$. Thus, for

$$\mu < \min\left\{\frac{b}{\|\boldsymbol{B}\|^2}, 1 - \frac{\|\mathbf{A}_k^T\xi_k\|_2^2}{2\|\mathbf{A}_k\mathbf{A}_k^T\xi_k\|_2^2}\frac{\|\mathbf{A}_k^T\xi_k\|_2^2}{\|\xi_k\|_2^2}\right\},$$

for every pair $\{\mathbf{f}_{k+1}\}\oplus\{\xi_{k+1}\} \neq \{0\}\oplus\{0\}$ (and if they are zeros then we already converged) there is

$$\|\{\mathbf{f}_{k+1}\}\oplus\{\xi_{k+1}\}\|_\oplus < \|\{\mathbf{f}_k\}\oplus\{\xi_k\}\|_\oplus.$$

Therefore, by Theorem 2, the error pair $\{\mathbf{f}_{k+1}\}\oplus\{\xi_{k+1}\}$ has to converge to $\mathbf{0}$, which ends the proof in the case $\mathbf{A}_k\mathbf{A}_k^T\xi_k \neq 0$. It remains to investigate what happens if $\mathbf{A}_k\mathbf{A}_k^T\xi_k = 0$.

We start by observing that either $\xi_k = 0$ or $\mathbf{A}_k^T\xi_k \neq 0$ and $\mathbf{A}_k\mathbf{A}_k^T\xi_k \neq 0$. This follows directly from the definition $\xi_k = \mathbf{A}_k\omega_k$. Indeed, if $\xi_k \neq 0$ there is $0 < \|\mathbf{A}_k\omega_k\|_2^2 = \omega_k^T\mathbf{A}_k^T\xi_k$ and analogously $0 < \|\mathbf{A}_k^T\xi_k\| = \xi_k^T\mathbf{A}_k\mathbf{A}_k^T\xi_k$.

In case $\xi_k = 0$ there is $\|\{\mathbf{f}_{k+1}\}\oplus\{\xi_{k+1}\}\|_\oplus = \|f_{k+1}\|_2 < \sqrt{1-2\mu b+\mu^2\|\boldsymbol{B}\|^2}\|f_k\|_2 = \sqrt{1-2\mu b+\mu^2\|\boldsymbol{B}\|^2}\|\{\mathbf{f}_k\}\oplus\{\xi_k\}\|_\oplus$ and the theorem follows. □

Theorem 2. *Let B be a finite-dimensional Banach space. Let $f : B \to B$ be a continuous map such that for every $\mathbf{x} \in B$ there is $\|f(x)\| < \|x\|$. Then for every x there is $\lim_{n\to\infty} f^n(x) = 0$.*

Proof. Let $\omega(x) = \{y : \exists_{i_1<i_2<\ldots} \lim_{n\to\infty} f^{i_n}(x) = y\}$. Because $\|f(x)\| < \|x\|$, the sequence $x, f(x), f^2(x), \ldots$ is contained in a ball of a radius $\|x\|$, which due to a finite dimensionality of B is a compact set. Thus, $\omega(x)$ is nonempty. Moreover, from the definition, $\omega(x)$ is a closed set, and therefore it is a compact set. Let $y_0 = \inf_{y\in\omega(x)}\|y\|$ – which we know exists, due to the compactness of $\omega(x)$ and the continuity of $\|\cdot\|$ (Weierstraß theorem). But for every $y \in \omega(x)$ there is $f(y) \in \omega(x)$, thus there must be $y_0 = 0$. By definition, for every ε, there exists n_0 such that $\|f^{n_0}(x)\| < \varepsilon$. Therefore, for $n > n_0$ $\|f^n(x)\| < \varepsilon$. Therefore, $f^n(x)$ must converge to 0. □

Proposition 2. *Let us assume that a SG module is trained in each iteration in such a way that it ϵ-tracks true gradient,*

i.e. that $\|\mathrm{SG}(h,y) - \partial L/\partial h\| \leq \epsilon$. *If* $\|\partial h/\partial \theta_{<h}\|$ *is upper bounded by some* K *and there exists a constant* $\delta \in (0,1)$ *such that in every iteration* $\epsilon K \leq \|\partial L/\partial \theta_{<h}\| \frac{1-\delta}{1+\delta}$, *then the whole training process converges to the solution of the original problem.*

Proof. Directly from construction we get that $\|\partial L/\partial \theta_{<h} - \hat{\partial} L/\hat{\partial}\theta_{<h}\| = \|(\partial L/\partial h - \mathrm{SG}(h,y))\partial h/\partial \theta_{<h}\| \leq \epsilon K$ thus in each iteration there exists such a vector e, that $\|e\| \leq \epsilon K$ and $\hat{\partial} L/\hat{\partial}\theta_{<h} = \partial L/\partial \theta_{<h} + e$. Consequently, we get a model trained with noisy gradients, where the noise of the gradient is bounded in norm by ϵK so, directly from assumptions, it is also upper bounded by $\|\partial L/\partial \theta_{<h}\| \frac{1-\delta}{1+\delta}$ and we we get that the direction followed is sufficient for convergence as this means that cosine between true gradient and synthetic gradient is uniformly bounded away (by δ) from zero (Zoutendijk, 1970; Gratton et al., 2011). At the same time, due to Proposition 1, we know that the assumptions do not form an empty set as the SG module can stay in an ϵ neighborhood of the gradient, and both norm of the synthetic gradient and $\|\partial h/\partial \theta_{<h}\|$ can go to zero around the true critical point. □

Corollary 1. *For a deep linear model and an MSE objective, trained with a linear SG module attached between two of its hidden layers, there exist learning rates in each iteration such that it converges to the critical point of the original problem.*

Proof. Denote the learning rate of the main model by μ and learning rate of the SG module by $\nu > 0$ and put $\mu = \epsilon \max(0, \|e\| - 1/(3\|\partial h/\partial \theta_{<h}\|)\|\partial L/\partial \theta_{<h}\|)$, where ϵ is a small learning rate (for example found using line search) and e is the error SG will make in the next iteration. The constant $1/3$ appears here as it is equal to $(1-\delta)/(1+\delta)$ for $\delta = 0.5$ which is a constant from Proposition 2, which we will need later on. Norm of e consists of the error fitting term L_{SG} which we know, and the term depending on the previous μ value, since this is how much the solution for the SG problem evolved over last iteration. In such a setting, the main model changes iff

$$\|e\|\|\partial h/\partial \theta_{<h}\| < 1/3\|\partial L/\partial \theta_{<h}\|. \tag{2}$$

First of all, this takes place as long as ν is small enough since the linear SG is enough to represent $\partial L/\partial h$ with arbitrary precision (Proposition 1) and it is trained to do so in a way that always converges (as it is a linear regression fitted to a linear function). So in the worst case scenario for a few first iterations we choose very small μ (it always exists since in the worst case scenario $\mu = 0$ agrees with the inequality). Furthermore, once this happens we follow true gradient on $\theta_{>h}$ and a noisy gradient on $\theta_{<h}$. Since the noise is equal to $e\partial h/\partial \theta_{<h}$ we get that

$$\|e\partial h/\partial \theta_{<h}\| \leq \|e\|\|\partial h/\partial \theta_{<h}\| < 1/3\|\partial L/\partial \theta_{<h}\|,$$

which is equivalent to error for $\theta_{<h}$ being upper bounded by $(1-\delta)/(1+\delta)\|\partial L/\partial h\|$ for $\delta = 0.5$ which matches assumptions of Proposition 2, thus leading to the convergence of the model considered. If at any moment we lose track of the gradient again – the same mechanism kicks in - μ goes down for as long as the inequality (2) does not hold again (and it has to at some point, given ν is positive and small enough). □

D. Technical details

All experiments were performed using TensorFlow (Abadi et al., 2016). In all the experiments SG loss is the MSE between synthetic and true gradients. Since all SGs considered were linear, weights were initialized to zeros so initially SG produces zero gradients, and it does not affect convergence (since linear regression is convex).

Datasets

Each of the artificial datasets is a classification problem, consisting of $\mathbf{X}$ sampled from k-dimensional Gaussian distribution with zero mean and unit standard deviation. For $k = 2$ we sample 100 points and for $k = 100$ we sample 1000. Labels $\mathbf{y}$ are generated in a way depending on the dataset name:

- linear$k$ - we randomly sample an origin-crossing hyperplane (by sampling its parameters from standard Gaussians) and label points accordingly,
- noisy$k$ - we label points according to linear$k$ and then randomly swap labels of 10% of samples,
- random$k$ - points are labeled completely randomly.

We used one-hot encoding of binary labels to retain compatibility with softmax-based models, which is consistent with the rest of experiments. However we also tested the same things with a single output neuron and regular sigmoid-based network and obtained analogous results.

Optimisation

Optimisation is performed using the Adam optimiser (Kingma & Ba, 2014) with a learning rate of $3e-5$. This applies to both main model and to SG module.

Artificial datasets

Table 2 shows results for training linear regression (shallow MSE), 10 hidden layer deep linear regression (deep MSE),

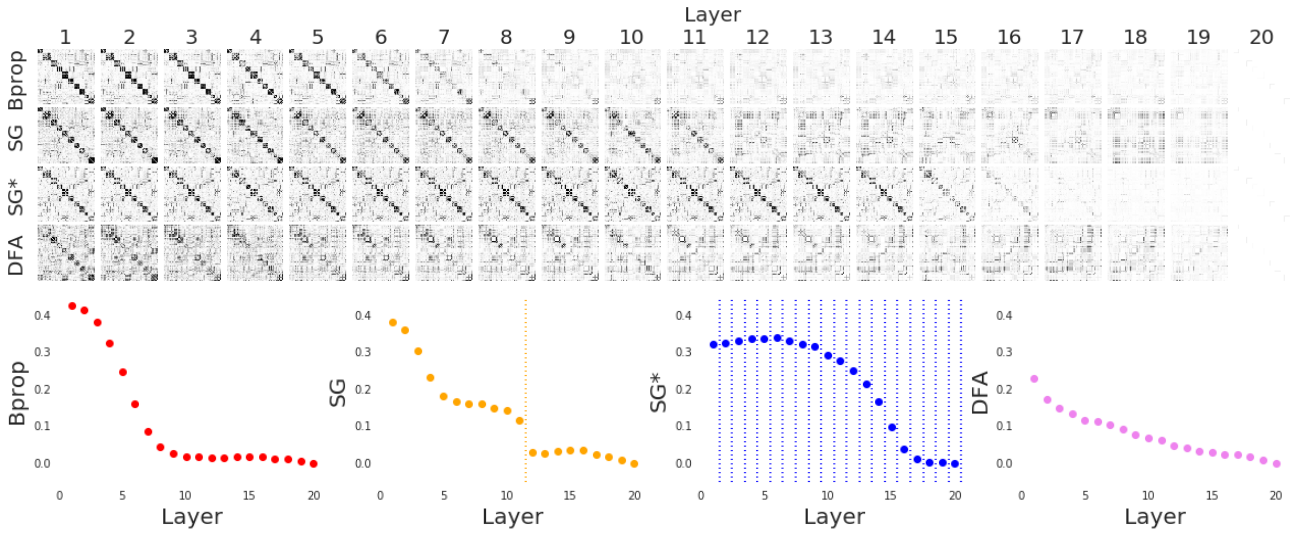


Figure 7. (top) Representation Dissimilarity Matrices for a label ordered sample from MNIST dataset pushed through 20-hidden layer deep relu networks trained with backpropagation (top row), a single SG attached between layers 11 and 12 (2nd row), SG between every pair of layers (3rd row), and the DFA model (4th row). Notice the moment of appearance of dark blue squares on a diagonal in each learning method, which shows when a clear inner-class representation has been learned. For visual confidence off block diagonal elements are semi transparent. (bottom) L2 distance between diagonal elements at a given layer and the same elements at layer 20. Dotted lines show where SGs are inserted. With a single SG module we can see that there is the representation is qualitatively different for the first part of the network (up to layer 11) and the rest. For fully unlocked model the representation constantly evolves through all the layers, as opposed to backprop which has a nearly constant representation correlation from layer 9 forward. Also due to DFA mathematical formulation it tries to solve the task as early as possible thus leading to nearly non-evolving representation correlation after the very first layer.

logistic regression (shallow log loss) and 10 hidden layer deep linear classifier (deep log loss). Since all these problems (after proper initialisation) converge to the global optima, we report the difference between final loss obtained for SG enriched models and the true global optimum.

MNIST experiments

Networks used are simple feed forward networks with h layers of 512 hidden relu units followed by batch normalisation layers. The final layer is a regular 10-class softmax layer. Inputs were scaled to $[0, 1]$ interval, besides that there was no preprocessing applied.

Representational Dissimilarity Matrices

In order to build RDMs for a layer h we sample 400 points (sorted according to their label) from the MNIST dataset, $\{x_i\}_{i=1}^{400}$ and record activations on each of these points, $h_i = h(x_i)$. Then we compute a matrix RDM such that $\mathrm{RDM}_{ij} = 1 - \mathrm{corr}(h_i, h_j)$. Consequently a perfect RDM is a block diagonal matrix, thus elements of the same class have a representation with high correlation and the representations of points from two distinct classes are not correlated. Figure 7 is the extended version of the analogous Figure 3 from the main paper where we show RDMs for backpropagation, a single SG, SG in-between every two layers, and also the DFA model, when training 20 hidden layer deep relu network.

Linear classifier/regression probes

One way of checking the degree to which the actual classification problem is solved at every layer of a feedforward network is to attach linear classifiers to every hidden layer and train them on the main task without backpropagating through the rest of the network. This way we can make a plot of train accuracy obtained from the representation at each layer. As seen in Figure 8 (left) there is not much of the difference between such analysis for backpropagation and a single SG module, confirming our claim in the paper that despite different representations in both sections of SG based module - they are both good enough to solve the main problem. We can also that DFA tries to solve the classification problem bottom-up as opposed to up-bottom – notice that for DFA we can have 100% accuracy after the very first hidden layer, which is not true even for backpropagation.

We also introduced a new kind of linear probe, which tries to capture how much computation (non-linear transformations) are being used in each layer. To achieve this, we attach a linear regressor module after each hidden layer and

dataset	model	MSE	log loss
linear2	shallow	0.00000	**0.03842**
linear100	shallow	0.00002	**0.08554**
noisy2	shallow	0.00000	**0.00036**
noisy100	shallow	0.00002	**0.00442**
random2	shallow	0.00000	0.00000
random100	shallow	0.00004	0.00003
noisy2	deep	0.00000	0.00000
noisy100	deep	0.00001	**0.00293**
random2	deep	0.00000	0.00000
random100	deep	0.00001	0.00004

Table 2. Differences in final losses obtained for various models/datasets when trained with SG as compared to model trained with backpropagation. Bolded entries denote experiments which converged to a different solution. *lineark* is k dimensional, linearly separable dataset, *noisy* is linearly separable up to 10% label noise, and *random* has completely random labeling. Shallow models means linear ones, while deep means 10 hidden layer deep linear models. Reported differences are averaged across 10 different datasets from the same distributions.

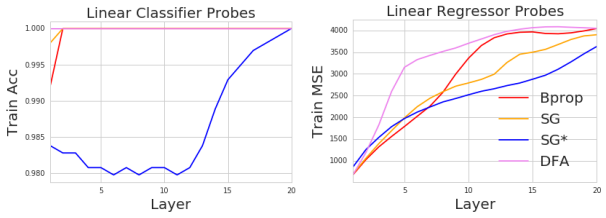


Figure 8. Left: Training accuracy at each linear classifier probe. *Right:* MSE for each linear regressor probe.

regress it (with MSE) to the input of the network. This is obviously label agnostic approach, but measures how non-linear the transformations are up to the given hidden layer. Figure 8 (right) again confirms that with a single SG we have two parts of the network (thus results are similar to RDM experiments) which do have slightly different behaviour, and again show clearly that DFA performs lots of non-linear transformations very early on compared to all other methods.

Loss estimation

In the main paper we show how SG modules using both activations and labels are able to implicitly describe the loss surface reasonably well for most of the training, with different datasets and losses. For completeness, we also include the same experiment for SG modules which do not use label information (Figure 9 (a) - (d)) as well as a module which does not use activations at all[6] (Figure 9 (e) - (h))). There are two important observations here: Firstly, none of these two approaches provide a loss estimation fidelity comparable with the full SG (conditioned on both activations and labels). This gives another empirical confirmation for correct conditioning of the module. Secondly, models which used only labels did not converge to a good solutions after 100k iterations, while without the label SG was able to do so (however it took much longer and was far noisier).

[6]This is more similar to a per-label stale gradient model.

References

Abadi, Martín, Agarwal, Ashish, Barham, Paul, Brevdo, Eugene, Chen, Zhifeng, Citro, Craig, Corrado, Greg S, Davis, Andy, Dean, Jeffrey, Devin, Matthieu, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Fairbank, M. *Value-gradient learning*. PhD thesis, City University London, UK, 2014.

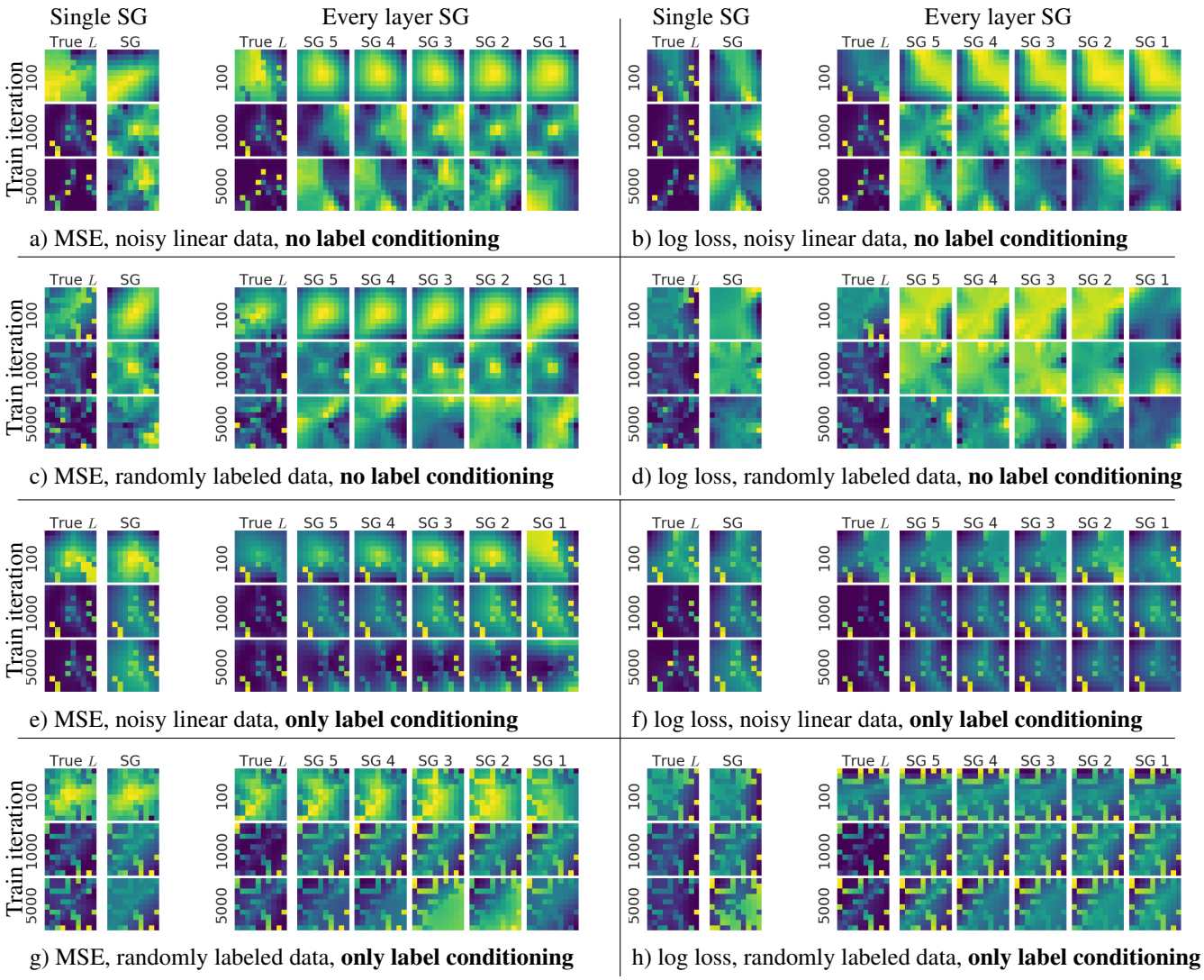
Gratton, Serge, Toint, Philippe L, and Tröltzsch, Anke. How much gradient noise does a gradient-based linesearch method tolerate. Technical report, Citeseer, 2011.

Heess, N, Wayne, G, Silver, D, Lillicrap, T P, Erez, T, and Tassa, Y. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, pp. 2944–2952, 2015.

Jaderberg, Max, Czarnecki, Wojciech Marian, Osindero, Simon, Vinyals, Oriol, Graves, Alex, and Kavukcuoglu, Koray. Decoupled neural interfaces using synthetic gradients. *arXiv preprint arXiv:1608.05343*, 2016.

Kingma, Diederik and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Zoutendijk, G. Nonlinear programming, computational methods. *Integer and nonlinear programming*, 143(1): 37–86, 1970.



a) MSE, noisy linear data, **no label conditioning**

b) log loss, noisy linear data, **no label conditioning**

c) MSE, randomly labeled data, **no label conditioning**

d) log loss, randomly labeled data, **no label conditioning**

e) MSE, noisy linear data, **only label conditioning**

f) log loss, noisy linear data, **only label conditioning**

g) MSE, randomly labeled data, **only label conditioning**

h) log loss, randomly labeled data, **only label conditioning**

Figure 9. Visualisation of the true loss and the loss extracted from the SG module. In each block left plot shows an experiment with a single SG attached and the right one with a SG after each hidden layer. Note, that in this experiment the final loss is actually big, thus even though the loss reassembles some part of the noise surface, the bright artifact lines are actually keeping it away from the true solution.