8 Appendix

8.1 Proof of Lemma 1

For this proof, we will use the result of Thm. 1 from [Bennani et al., 2020] (particularly Remark 1) and notice that the expression of $f^\star_{\tau_T}$ can be espressed recursively with respect to $f^\star_{\tau_S}$:

Proof.

$$\begin{aligned}
f^\star_{\tau_T}(x) &= f^\star_{\tau_T-1}(x) + \langle \nabla_\omega f^\star_{\tau_T-1}(x), \omega^\star_{\tau_T} - \omega^\star_{\tau_T-1}\rangle \\
&= f^\star_{\tau_T-k}(x) + ... + \langle \nabla_\omega f^\star_{\tau_T-2}(x), \omega^\star_{\tau_T-1} - \omega^\star_{\tau_T-2}\rangle + \langle \nabla_\omega f^\star_{\tau_T-1}(x), \omega^\star_{\tau_T} - \omega^\star_{\tau_T-1}\rangle \\
&= f^\star_{\tau_S}(x) + \sum_{k=\tau_S+1}^{\tau_T} \langle \nabla_\omega f^\star_k(x), \omega^\star_k - \omega^\star_{k-1}\rangle \\
&= f^\star_{\tau_S}(x) + \sum_{k=\tau_S+1}^{\tau_T} \langle \nabla_\omega f_0(x), \omega^\star_k - \omega^\star_{k-1}\rangle \quad \text{(NTK constant)} \\
&= f^\star_{\tau_S}(x) + \langle \nabla_\omega f_0(x), \omega^\star_{\tau_T} - \omega^\star_{\tau_S}\rangle
\end{aligned}$$

where we used constant NTK assumption, i.e $\nabla_\omega f^\star_\tau(x) = \nabla_{\omega_0} f(x)$, $\forall \tau \in [T]$.

Using the fact that the kernel is given by $\phi(x) = \nabla_{w_0} f(x)$, we have that:

$$\begin{aligned}
\delta^{\tau_S \to \tau_T}(X^{\tau_S}) &= f^\star_{\tau_T}(X^{\tau_S}) - f^\star_{\tau_S}(X^{\tau_S}) \\
&= \langle \phi(X^{\tau_S}), \omega^\star_{\tau_T} - \omega^\star_{\tau_S}\rangle \\
\Delta^{\tau_S \to \tau_T}(X^{\tau_S}) &= \left\| \phi(X^{\tau_S})(\omega^\star_{\tau_T} - \omega^\star_{\tau_S}) \right\|_2^2
\end{aligned}$$

This concludes the proof. □

8.2 Proof of Theorem 1

For this proof, we will decompose the drift from task τ_S to τ_T into a telescopic sum. We will then use SVD to factorize the expression of $(\omega^\star_\tau - \omega^\star_{\tau-1})$ and get the upper bound showed.

Proof.

$$\Delta^{\tau_S \to \tau_T}(X^{\tau_S}) = \left\| \phi(X^{\tau_S})(\omega^\star_{\tau_T} - \omega^\star_{\tau_S}) \right\|_2^2 \quad \text{(Lemma 1)} \tag{13}$$

$$= \left\| \sum_{k=\tau_S+1}^{\tau_T} \phi(X^{\tau_S})(\omega^\star_k - \omega^\star_{k-1}) \right\|_2^2 \tag{14}$$

$$= \left\| \sum_{k=\tau_S+1}^{\tau_T} \phi(X^{\tau_S}) \underbrace{\phi(X^k)^\top [\phi(X^k)\phi(X^k)^\top + \lambda I_{n_k}]^{-1} \tilde{y}_k}_{\text{from Thm. 1 of [Bennani et al., 2020]}} \right\|_2^2 \tag{15}$$

$$= \left\| \sum_{k=\tau_S+1}^{\tau_T} U_{\tau_S} \Sigma_{\tau_S} V_{\tau_S}^\top V_k \Sigma_k U_k^\top [U_k \Sigma_k^2 U_k^\top + \lambda I_{n_k}]^{-1} \tilde{y}_k \right\|_2^2 \quad \text{(SVD)} \tag{16}$$

$$= \left\| \sum_{k=\tau_S+1}^{\tau_T} U_{\tau_S} \Sigma_{\tau_S} \underbrace{V_{\tau_S}^\top V_k}_{O_{SGD}^{\tau_S \to k}} \underbrace{\Sigma_k [\Sigma_k^2 + \lambda I_{n_k}]^{-1} U_k^\top}_{M_k} \tilde{y}_k \right\|_2^2 \tag{17}$$

$$\tag{18}$$

Where we used the SVD $\phi(X^\tau) = U_\tau \Sigma_\tau V_\tau^T, \forall \tau \in [T]$. This concludes the proof. □

8.3 Proof of Corolary 1

For this proof, we will bound the Catastrophic Forgetting as a function of the principal angles between the source and target subspaces. Indeed, given two subspace τ_S and τ_T represented by their orthonormal basis concatenated respectively in V_{τ_S} and V_{τ_T}, the elements of the diagonal matrix $\Theta^{\tau_S \to \tau_T}$ resulting from the SVD of $V_{\tau_S}^\top V_{\tau_T}$ are the cosines of the principal angles between these two subspace [Wedin, 1983, Zhu and Knyazev, 2013].

Proof.

$$\Delta^{\tau_S \to \tau_T}(X^{\tau_S}) \leq \sum_{k=\tau_S+1}^{\tau_T} \left\| U_{\tau_S} \Sigma_{\tau_S} V_{\tau_S}^\top V_k \Sigma_k [\Sigma_k^2 + \lambda I_{n_k}]^{-1} U_k^\top \tilde{y}_k \right\|_2^2 \tag{19}$$

$$\leq \sum_{k=\tau_S+1}^{\tau_T} \|U_{\tau_S} \Sigma_{\tau_S}\|_2^2 \left\| V_{\tau_S}^\top V_k \right\|_2^2 \left\| \Sigma_k [\Sigma_k^2 + \lambda I_{n_k}]^{-1} U_k^\top \tilde{y}_k \right\|_2^2 \quad \text{(sub-multiplicativity of norm 2)} \tag{20}$$

$$\leq \sum_{k=\tau_S+1}^{\tau_T} \|\Sigma_{\tau_S}\|_2^2 \left\| V_{\tau_S}^\top V_k \right\|_2^2 \|M_k \tilde{y}_k\|_2^2 \qquad (U_{\tau_S} \text{ is an orthonormal matrix}) \tag{21}$$

$$\leq \sigma_{\tau_S,1}^2 \sum_{k=\tau_S+1}^{\tau_T} \left\| Y \Theta^{\tau_S \to k} Z^\top \right\|_2^2 \|M_k \tilde{y}_k\|_2^2 \qquad \text{(SVD)} \tag{22}$$

$$\leq \sigma_{\tau_S,1}^2 \sum_{k=\tau_S+1}^{\tau_T} \left\| \Theta^{\tau_S \to k} \right\|_2^2 \|M_k \tilde{y}_k\|_2^2 \qquad (Y, Z \text{ are orthonormal matrices}) \tag{23}$$

$$\tag{24}$$

where $Y \Theta^{\tau_S \to k} Z^\top$ is the SVD of $V_{\tau_S}^\top V_k$. This concludes the proof. □

8.4 Proof of Corollary 2

We first need to prove a corollary that is exactly the same as Corollary 4 (shown below), the difference lies in the kernel definition. Under the same notation as in Corollary 4, the solution after training on task τ for GEM-NT is such that:

$$\omega_\tau^\star - \omega_{\tau-1}^\star = \overline{\phi}_\tau(X^\tau)^\top (\kappa_\tau(X^\tau, X^\tau) + \lambda I_{n_\tau})^{-1} \tilde{y}_\tau \tag{25}$$

where:

$$\begin{aligned}
\kappa_\tau(x, x') &= \overline{\phi}_\tau(x) \overline{\phi}_\tau(x')^\top, \\
\overline{\phi}_\tau(x) &= \phi(x) T_{\tau-1}, \\
T_\tau &= I_p - G_\tau (G_\tau)^\top, \\
\tilde{y}_\tau &= y_\tau - y_{\tau-1 \to \tau}, \\
y_{\tau-1 \to \tau} &= f_{\tau-1}^\star(X^\tau),
\end{aligned}$$

Proof. of Corollary 2 Similarly to Proof of Theorem 1:

$$\Delta^{\tau_S \to \tau_T}(X^{\tau_S}) = \left\|\phi(X^{\tau_S})(\omega^\star_{\tau_T} - \omega^\star_{\tau_S})\right\|_2^2 \quad \text{(Lemma 1)} \tag{26}$$

$$= \left\| \sum_{k=\tau_S+1}^{\tau_T} \phi(X^{\tau_S})(\omega^\star_k - \omega^\star_{k-1}) \right\|_2^2 \tag{27}$$

$$= \left\| \sum_{k=\tau_S+1}^{\tau_T} \phi(X^{\tau_S}) \underbrace{\overline{\phi}(X^k)^\top [\overline{\phi}(X^k)\overline{\phi}(X^k)^\top + \lambda I_{n_k}]^{-1} \overline{y}_k}_{\text{as shown above}} \right\|_2^2 \tag{28}$$

$$= \left\| \sum_{k=\tau_S+1}^{\tau_T} U_{\tau_S}\Sigma_{\tau_S} V_{\tau_S}^\top T_{k-1}^\top V_k \Sigma_k U_k^\top [\overline{\phi}(X^k)\overline{\phi}(X^k)^\top + \lambda I_{n_k}]^{-1} \overline{y}_k \right\|_2^2 \quad \text{(SVD)} \tag{29}$$

$$= \left\| \sum_{k=\tau_S+1}^{\tau_T} U_{\tau_S}\Sigma_{\tau_S} \underbrace{V_{\tau_S}^\top T_{k-1} T_{k-1}^\top V_k}_{O^{\tau_S \to k}_{\text{GEM-NT}}} \underbrace{\Sigma_k U_k^\top [\overline{\phi}(X^k)\overline{\phi}(X^k)^\top + \lambda I_{n_k}]^{-1}}_{M_k} \overline{y}_k \right\|_2^2 \quad (T_{k-1})^n = T_{k-1}, \forall n \geq 1 \tag{30}$$

$$\tag{31}$$

This concludes the proof. □

8.5 Forgetting for PCA-OGD

To prove the forgetting expression for PCA-OGD, we will use a corollary arising naturally from Theorem 1 of [Bennani et al., 2020] which extends the expression of the learned weights $(\omega^\star_{\tau+1} - \omega^\star_\tau)$ from the **infinite** to the **finite** memory case. The proof will be shown after the proof of Corollary 3 for the flow of the understanding.

Corollary 4 (Convergence of PCA-OGD under finite memory)**.**

Given $\mathcal{T}_1, ..., \mathcal{T}_T$ a sequence of tasks. If the learning rate satisfies: $\eta_\tau < \frac{1}{\|\kappa_\tau(X^\tau, X^\tau) + \lambda I_{n_\tau}\|}$, $\kappa_\tau, \forall \tau \in [T]$ is invertible with a weight decay regularizer $\lambda > 0$, the solution after training on task τ is such that:

$$\omega^\star_\tau - \omega^\star_{\tau-1} = \tilde{\phi}_\tau(X^\tau)^\top (\kappa_\tau(X^\tau, X^\tau) + \lambda I_{n_\tau})^{-1} \tilde{y}_\tau \tag{32}$$

where:

$$\begin{aligned}
\kappa_\tau(x, x') &= \tilde{\phi}_\tau(x)\tilde{\phi}_\tau(x')^\top, \\
\tilde{\phi}_\tau(x) &= \phi(x) T_{\tau-1,:d}, \\
T_{\tau,:d} &= I_p - P_{\tau,:d} P_{\tau,:d}^\top, \\
\phi(x) &= \nabla_{\omega_0} f_0^\star(x), \\
\tilde{y}_\tau &= y_\tau - y_{\tau-1\to\tau}, \\
y_{\tau-1\to\tau} &= f^\star_{\tau-1}(X^\tau),
\end{aligned}$$

where $T_{0,:d} = I_p$ since there are no previous task when training on task 1.

Proof. of Corollary 3

Similarly to Proof of Theorem 1:

$$\Delta^{\tau_S \to \tau_T}(X^{\tau_S}) = \left\|\phi(X^{\tau_S})(\omega^\star_{\tau_T} - \omega^\star_{\tau_S})\right\|_2^2 \quad \text{(Lemma 1)} \tag{33}$$

$$= \left\| \sum_{k=\tau_S+1}^{\tau_T} \phi(X^{\tau_S})(\omega^\star_k - \omega^\star_{k-1}) \right\|_2^2 \tag{34}$$

$$= \left\| \sum_{k=\tau_S+1}^{\tau_T} \phi(X^{\tau_S}) \underbrace{\tilde{\phi}(X^k)^\top [\tilde{\phi}(X^k)\tilde{\phi}(X^k)^\top + \lambda I_{n_k}]^{-1} \tilde{y}_k}_{\text{from Corollary 4}} \right\|_2^2 \tag{35}$$

$$= \left\| \sum_{k=\tau_S+1}^{\tau_T} U_{\tau_S}\Sigma_{\tau_S} V_{\tau_S}^\top T_{k-1,:d}^\top V_k \Sigma_k U_k^\top [\tilde{\phi}(X^k)\tilde{\phi}(X^k)^\top + \lambda I_{n_k}]^{-1} \tilde{y}_k \right\|_2^2 \quad \text{(SVD)} \tag{36}$$

$$= \left\| \sum_{k=\tau_S+1}^{\tau_T} U_{\tau_S}\Sigma_{\tau_S} \underbrace{V_{\tau_S}^\top R_{k-1,d:} R_{k-1,d:}^\top V_k}_{O_{PCA}^{\tau_S \to k}} \Sigma_k \underbrace{U_k^\top [\tilde{\phi}(X^k)\tilde{\phi}(X^k)^\top + \lambda I_{n_k}]^{-1}}_{M_k} \tilde{y}_k \right\|_2^2 \tag{37}$$

□

Proof of Corollary 4.

In the same fashion as [Bennani et al., 2020], we prove Corollary 4 by induction. Our induction hypothesis H_τ is the following : $\mathcal{H}_\tau$: For all $k \leq \tau$, Corollary 4 holds.

First, we prove that $\mathcal{H}_1$ holds.

The proof is straightforward. For the first task, since there were no previous tasks, PCA-OGD on this task is the same as SGD.

Therefore, it is equivalent to minimising the following objective :

$$\arg\min_{\omega \in \mathbb{R}^p} \left\|f_0(X^1) + \phi(X^1)(\omega - \omega_0^\star) - y_1\right\|_2^2 + \frac{1}{2}\lambda \left\|\omega - \omega_0\right\|_2^2$$

where $\phi(x) = \nabla_{\omega_0^\star} f_0^\star(x)$.

Substituing the residual term $\tilde{y}_1 = y_1 - f_0(X^1)$, we get:

$$\arg\min_{\omega \in \mathbb{R}^p} \left\|\phi(X^1)(\omega - \omega_0^\star) - \tilde{y}_1\right\|_2^2 + \frac{1}{2}\lambda \left\|\omega - \omega_0\right\|_2^2$$

The objective is quadratic and the Hessian is positive definite, therefore the minimum exists and is unique

$$\omega_1^\star - \omega_0^\star = \phi(X^1)^\top(\phi(X^1)\phi(X^1)^\top + \lambda I_{n_1})^{-1}\tilde{y}_1$$

Under the NTK regime assumption :

$$f_1^\star(x) = f_0^\star(x) + \nabla_{\omega_0} f_0^\star(x)^\top(\omega_1^\star - \omega_0^\star)$$

Then, by replacing into $\omega_1^\star - \omega_0^\star$:

$$f_1^\star(x) = f_0^\star(x) + \nabla_{\omega_0} f_0^\star(x)\phi(X^1)^\top(\phi(X^1)\phi(X^1)^\top + \lambda I_{n_1})^{-1}\tilde{y}_1$$
$$f_1^\star(x) = f_0^\star(x) + \kappa_1(x, X^1)(\kappa_1(X^1, X^1) + \lambda I_{n_1})^{-1}\tilde{y}_1$$

Finally :

$$f_1^\star(x) - f_0^\star(x) = \kappa_1(x, X^1)(\kappa_1(X^1, X^1) + \lambda I_{n_1})^{-1}\tilde{y}_1$$

Where :

$$\begin{aligned}\kappa_1(X^1, X^1) &= \tilde{\phi_1}(X^1)\tilde{\phi_1}(X^1)^\top \\ &= \phi(X^1)T_{0,:d}T_{0,:d}^\top\phi(X^1)^\top \\ &= \phi(X^1)\phi(X^1)^\top\end{aligned}$$

Since there is no previous task and $T_{0,:d}$ contains no eigenvectors yet, we have $T_{0,:d} = I_p$ and $\tilde{y}_1 = y_1$.

This completes the proof of $\mathcal{H}_1$.

Let $\tau \in \mathcal{N}^\star$, we assume that $\mathcal{H}_\tau$ is true, then we show that $\mathcal{H}_{\tau+1}$ is true.

At the end of training of task τ, we add the first d eigenvectors of $\phi(X^\tau)\phi(X^\tau)^\top$ to $P_{\tau-1,:d} \in \mathbb{R}^{p\times(\tau-1)\cdot d}$ to form the matrix $P_{\tau,:d} \in \mathbb{R}^{p\times\tau\cdot d}$ through PCA decomposition

.

The update during the training of task $\tau+1$ is projected orthogonally to the first d components of task 1 until τ via the matrix $T_{\tau,:d}$:

$$\begin{aligned}\omega_{\tau+1}(t+1) &= \omega_\tau^\star - \eta T_{\tau,:d}\nabla_\omega\mathcal{L}_\lambda^\tau(\omega_{\tau+1}(t)) \\ \omega_{\tau+1}(t+1) - \omega_\tau^\star &= -\eta T_{\tau,:d}\nabla_\omega\mathcal{L}_\lambda^\tau(\omega_{\tau+1}(t)) \\ \omega_{\tau+1}(t+1) - \omega_\tau^\star &= T_{\tau,:d}\tilde{\omega}_{\tau+1}\end{aligned}$$

Where η is the learning rate and $T_{\tau,:d} = I_p - P_{\tau,:d}P_{\tau,:d}^\top$.

We rewrite the objective by plugging in the variables we just defined. The two objectives are equivalent :

$$\underset{\tilde{\omega}_{\tau+1}\in\mathbb{R}^p}{\arg\min}\left\|\underbrace{\phi(X^{\tau+1})T_{\tau,:d}}_{\phi_{\tau+1}(X^{\tau+1})}\tilde{w}_{\tau+1} - \tilde{y}_{\tau+1}\right\|_2^2$$

The optimisation objective is quadratic, unconstrainted, with a positive definite hessian. Therefore, an optimum exists and is unique :

$$\begin{aligned}\tilde{\omega}_{\tau+1}^\star &= \phi_{\tau+1}(X^{\tau+1})^\top(\phi_{\tau+1}(X^{\tau+1})\phi_{\tau+1}(X^{\tau+1})^\top)^{-1}\tilde{y}_{\tau+1} \\ \omega_{\tau+1}^\star - \omega_\tau^\star &= \phi_{\tau+1}(X^{\tau+1})^\top(\phi_{\tau+1}(X^{\tau+1})\phi_{\tau+1}(X^{\tau+1})^\top)^{-1}\tilde{y}_{\tau+1} \\ \omega_{\tau+1}^\star - \omega_\tau^\star &= \phi_{\tau+1}(X^{\tau+1})^\top(\kappa_{\tau+1}(X^{\tau+1}, X^{\tau+1}))^{-1}\tilde{y}_{\tau+1}\end{aligned}$$

Recall from the induction hypothesis of $\mathcal{H}_\tau$ the general form of $f_\tau^\star(x)$:

$$f_\tau^\star(x) = f_{\tau-1}^\star(x) + \langle\nabla_{\omega_0}f_0^\star(x), \omega_{\tau+1}^\star - \omega_\tau^\star\rangle$$

After training on task $\tau+1$:

$$\begin{aligned}f_{\tau+1}^\star(x) &= f_{\tau-1}^\star(x) + \langle\nabla_{\omega_0}f_0^\star(x), \omega_{\tau+1}^\star - \omega_{\tau-1}^\star\rangle \\ f_{\tau+1}^\star(x) &= f_{\tau-1}^\star(x) + \langle\nabla_{\omega_0}f_0^\star(x), \omega_{\tau+1}^\star - \omega_{\tau-1}^\star + \underbrace{\omega_\tau^\star - \omega_\tau^\star}_{=0}\rangle \\ f_{\tau+1}^\star(x) &= \underbrace{f_{\tau-1}^\star(x) + \langle\nabla_{\omega_0}f_0^\star(x), \omega_\tau^\star - \omega_{\tau-1}^\star\rangle}_{f_\tau^\star(x)} + \langle\nabla_{\omega_0}f_0^\star(x), \omega_{\tau+1}^\star - \omega_\tau^\star\rangle \\ f_{\tau+1}^\star(x) &= f_\tau^\star(x) + \langle\nabla_{\omega_0}f_0^\star(x), \omega_{\tau+1}^\star - \omega_\tau^\star\rangle \\ f_{\tau+1}^\star(x) &= f_\tau^\star(x) + \phi(x)\phi_{\tau+1}(X^{\tau+1})^\top(\kappa_{\tau+1}(X^{\tau+1}, X^{\tau+1}))^{-1}\tilde{y}_{\tau+1} \\ f_{\tau+1}^\star(x) &= f_\tau^\star(x) + \phi(x)T_{\tau,:d}^\top\phi_{\tau+1}(X^{\tau+1})^\top(\kappa_{\tau+1}(X^{\tau+1}, X^{\tau+1}))^{-1}\tilde{y}_{\tau+1} \\ f_{\tau+1}^\star(x) &= f_\tau^\star(x) + \underbrace{\phi(x)T_{\tau,:d}T_{\tau,:d}^\top\phi_{\tau+1}(X^{\tau+1})^\top}_{\kappa_{\tau+1}(x,X^{\tau+1})}(\kappa_{\tau+1}(X^{\tau+1}, X^{\tau+1}))^{-1}\tilde{y}_{\tau+1} \quad (\text{ since } (T_{\tau,:d})^\top = T_{\tau,:d}) \\ f_{\tau+1}^\star(x) &= f_\tau^\star(x) + \kappa_{\tau+1}(x, X^{\tau+1})(\kappa_{\tau+1}(X^{\tau+1}, X^{\tau+1}))^{-1}\tilde{y}_{\tau+1}\end{aligned}$$

We have proven $\mathcal{H}_{t+1}$ and conclude the proof of Corollary 4.

□

8.6 Algorithms summary

Properties	$O_X^{\tau_S \to \tau_T}$ $= V_{\tau_S}^\top \mathbf{X}\mathbf{X}^\top V_{\tau_T}$	X contains	Elements stored in the memory	Recompute NTK?
SGD	X=I_{τ_S}	NA	NA	NA
GEM-NT	X=$\overline{\mathbf{G}}_{\tau_\mathbf{T}-\mathbf{1}}$	samples of $\nabla\mathcal{L}(X^\tau)$	samples of X^τ	Yes
OGD	X=$\mathbf{R}_{\tau_\mathbf{T}-\mathbf{1}}$	samples of $\nabla f(X^\tau)$	samples of $\nabla f(X^\tau)$	No
PCA-OGD	X=$\mathbf{R}_{\tau_\mathbf{T}-\mathbf{1},:\mathbf{d}}$	top eigenvectors of $\nabla f(X^\tau)$	top eigenvectors of $\nabla f(X^\tau)$	No

Table 2: Property of the Overlap matrix for each method which is responsible for mitigating Catastrophic Forgetting. NA: Not applicable

NTK overlap matrix First of all, the three methods GEM-NT, OGD, PCA-OGD differs from SGD by the the matrix X (1st column) that contains either the features map $\nabla_\omega f(x)$ or the gradient loss function.

Elements stored GEM-NT and OGD both samples random elements at the end of each task τ to store in the memory. For the sake of understanding, if we assume a mean square loss function, with a batch size equal to one, the gradient loss function becomes: $g_\tau^{(\text{GEM-NT})} = \underbrace{\nabla_\omega f_\tau(x)}_{g_\tau^{(\text{OGD})}}(f_\tau(x) - y_\tau)$. From here, we see that GEM-NT weights the features maps by the residual of a given task $k < \tau$ when training on task τ.

Information compression PCA-OGD compresses the information contained in the data by storing the principal components of $\nabla_\omega f(X^\tau)$ through PCA. If the data has structure such as Rotated MNIST or Split CIFAR (See Section 9), storing few components will explain a high percentage of variance of the data of component in order to explain the dataset variance.

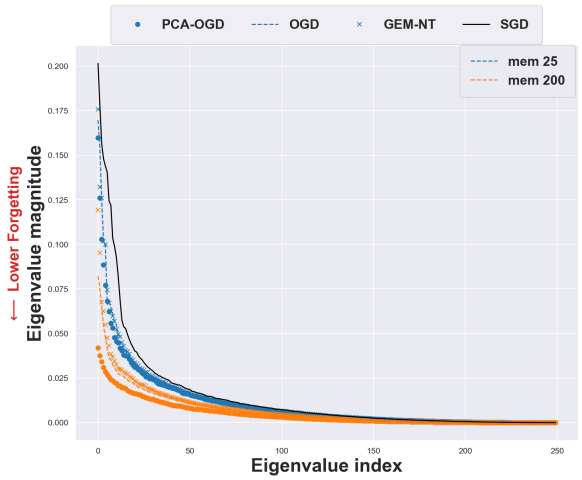
Accounting for the NTK variation The drawback OGD and PCA-OGD compared to GEM-NT is that the NTK is assumed to be constant which is not always the case in practice (See Section 8.10). PCA-OGD and OGD will then project orthogonally to a vector that is outdated.

8.7 The counter-example of Permuted MNIST: no structure

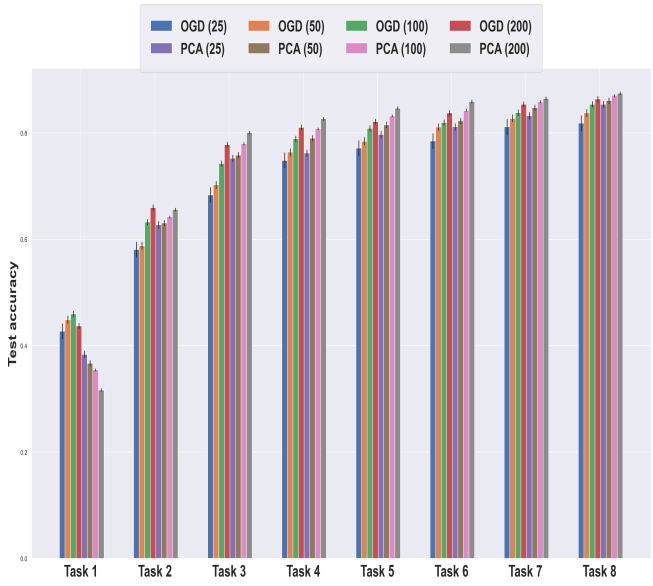
We now examine the dataset Permuted MNIST and try to understand why PCA-OGD is not efficient in such case. Each task is an MNIST dataset where a different and uniform permutation of pixels is applied. This has the particularity of removing any extra-task correlations and patterns.

Eigenvalues of the NTK overlap matrix : Figure 5a shows the eigenvalues of the NTK overlap matrix when increasing buffer size. First of all, we notice that the magnitude of the eigenvalues is very small compared to Figure 3. This is explained by the fact that each task shares almost no correlations, meaning that the cosine of angle of the two subspaces might be close to 0 (small eigenvalues). Additionally, we see that increasing the memory size does not reduce much the eigenvalue magnitude. This is due to the distribution of eigenvalues (See Figure 9) which are spread more uniformly than Permuted MNIST and Split CIFAR, meaning that more components need to be kept in order to explain a high % of the variance. In this situation, PCA-OGD does not have much advantage compared to OGD (See also toy example, Supplementary Material Section 8.12).

Final accuracy with OGD : We now compare final performance against OGD (See Figure 5b). PCA-OGD does sensitively well compared to OGD (except for the first task where performance are much worse). This can be explained by the fact that PCA-OGD needs to keep a lot of components to explain a high percentage of variance such that selecting random element like OGD will results in comparable results. This is all the more confirmed by Table 3, keeping 50 components only explains 50% of the variance while it respectively explains 81% for Rotated MNIST and 72% for Split CIFAR. As mentionned by [Farquhar and Gal, 2018], even though such datasets meets the definition of CL, it is an irrealistic setting since "new situations look confusingly similar to old ones'". Hence methods that leverage structure like PCA-OGD can be useful.



(a) Comparison of the eigenvalues of $O^{1\to 2}$ on **Permuted MNIST** with increasing memory size. Lower values imply less forgetting.



(b) Final accuracy on **Permuted MNIST** for different memory size (averaged over 5 seeds ±1 std). OGD and PCA-OGD have comparable performance (except for the first task).

8.8 Comparison PCA-OGD versus OGD

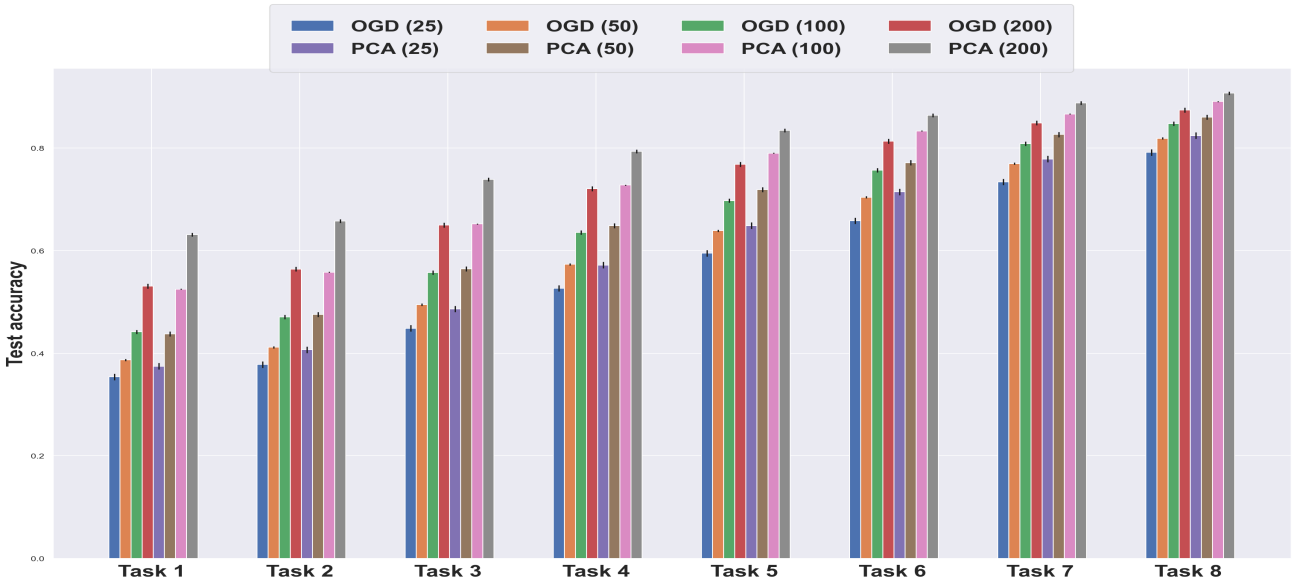


Figure 6: Final accuracy on **Rotated MNIST** for different memory size (averaged over 5 seeds ±1 std). OGD needs twice as much memory as PCA-OGD in order to achieve the same performance (i.e compare OGD (200) and PCA (100).

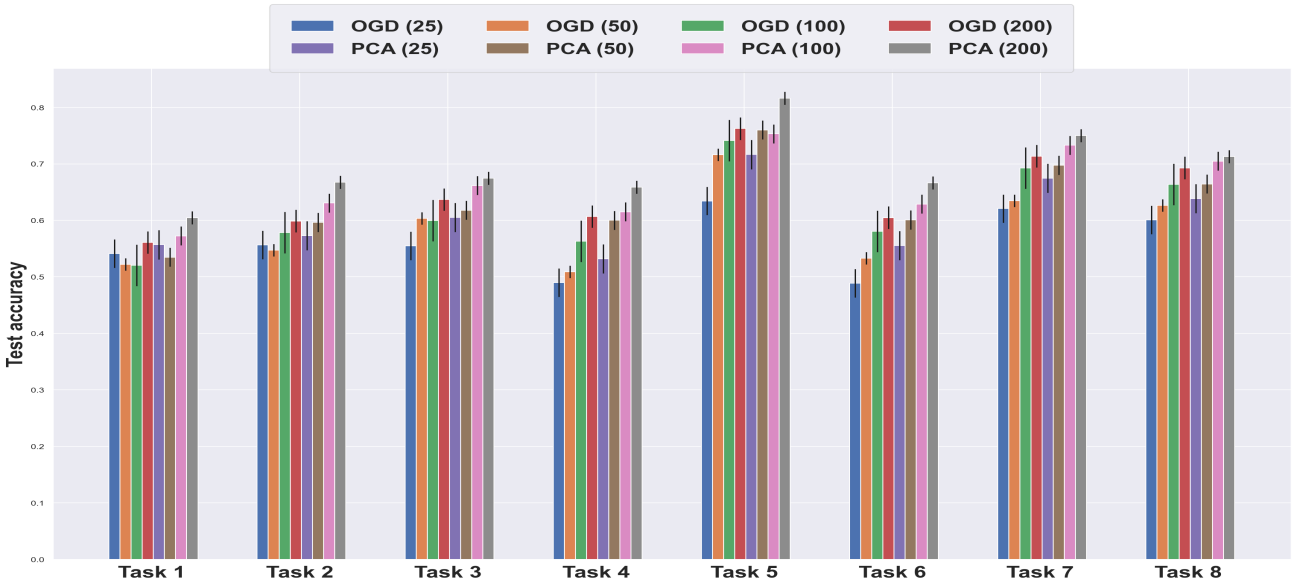


Figure 7: Final accuracy on **Split CIFAR** for different memory size (averaged over 5 seeds ±1 std). When dataset is well structured PCA-OGD efficiently leverages the pattern (i.e compare OGD (200) and PCA (100).

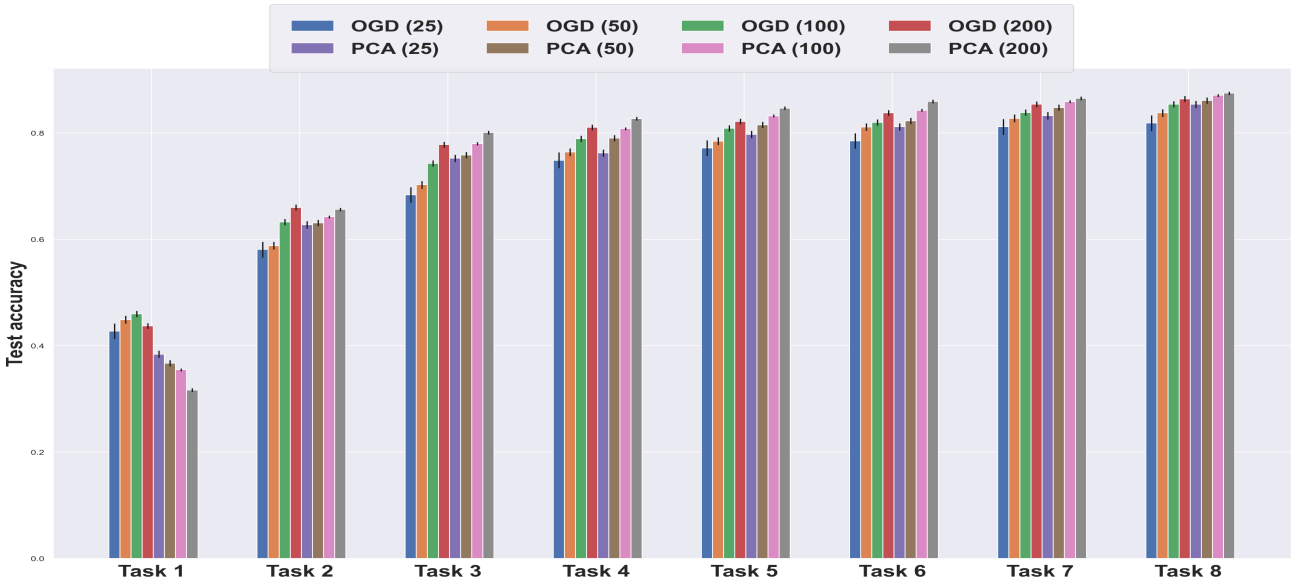


Figure 8: Final accuracy on **Permuted MNIST** for different memory size. When there is no structure, information captured by PCA-OGD from previous tasks cannot be leveraged for future tasks. OGD and PCA-OGD have comparable performance (except for the first task).

8.9 Structure in the data

We sample a subset of $s = 3,000$ samples from different datasets $x_j^\tau, j = 1, ..., 3000$ (Permuted and Rotated MNIST), then we perform PCA on $\phi(x_j^\tau)\phi(x_j^\tau)^\top$ and keep the d top components. Having a total memory size of $M = 200, 500, 1000, 2000, 3000$ and training on 15 tasks means that each task will be allocated $M/14$ since we omit the last task. As seen in Figure 9 for a total memory size of 200, we only keep 14 components which corresponds to 38.84% of the variance explained in Permuted MNIST while it represents 71.06% for Rotated MNIST. This is naturally explained by the fact that having random permutations breaks the structure of the data and in order to keep the most information would we need to allocate a large amount of memory.

components kept	Permuted MNIST	Rotated MNIST	Split CIFAR
10	35.13	68.52	58.87
25	45.14	75.54	65.99
50	53.33	81.09	72.27
100	62.37	86.23	79.19
200	71.65	90.71	85.99
500	83.20	94.42	93.24

Table 3: Percentage of variance explained with different memory size when performing PCA on $s = 3,000$ samples (except for Split CIFAR where $s = 1,500$).

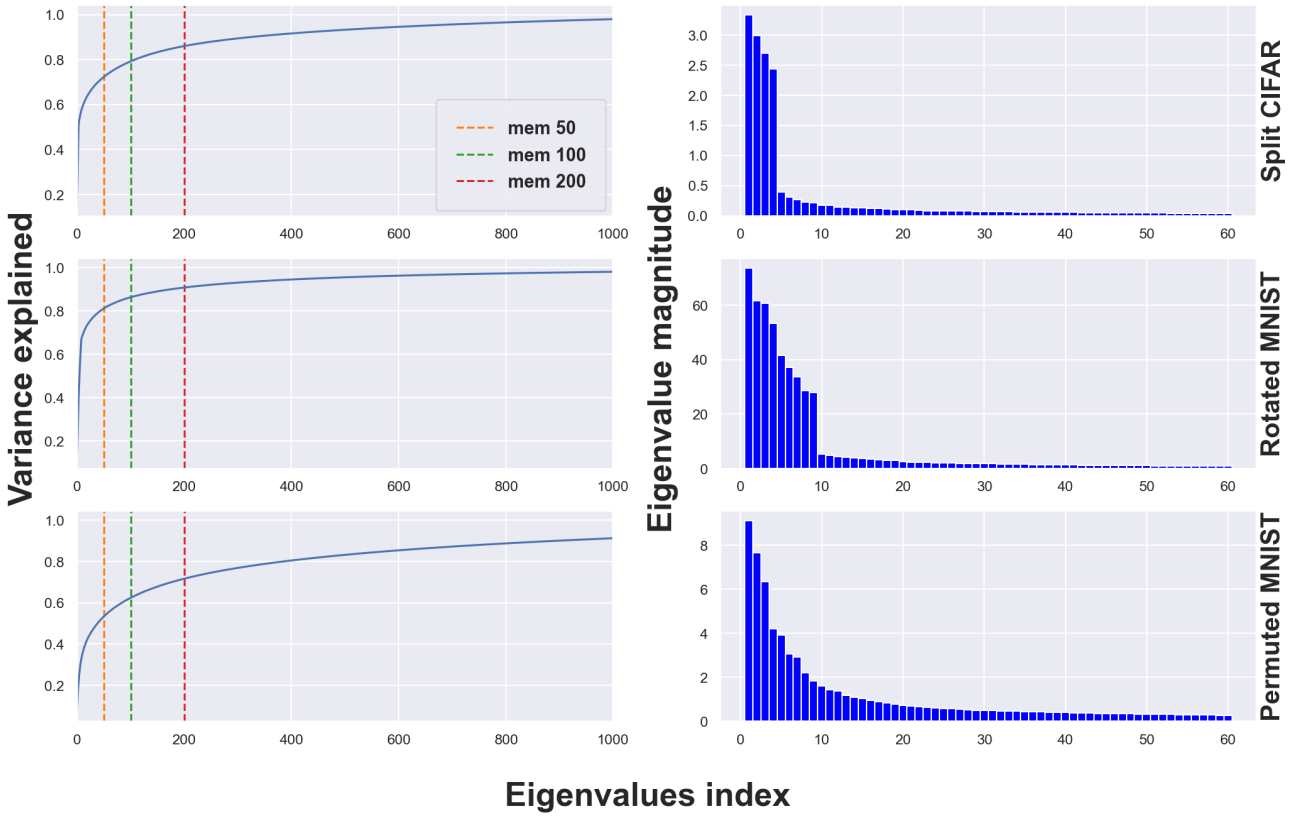


Figure 9: Percentage of variance explained for different datasets. Verticale lines on the left represents the number of components kept or the memory allocated per task. We have truncated the x-axis to focus on the interesting part.

8.10 NTK changes

We measure the change in NTK of PCA-OGD from its initialization value for different dataset size for a fixed architecture after each task (See Figure 10). The green curve shows the actual parameters used for the experiments. Although, there is linear increase of the NTK for MNIST datasets, it is approximately constant (after the first task) for Split CIFAR which validates the constant NTK assumption and explains the good result of PCA-OGD for this dataset.

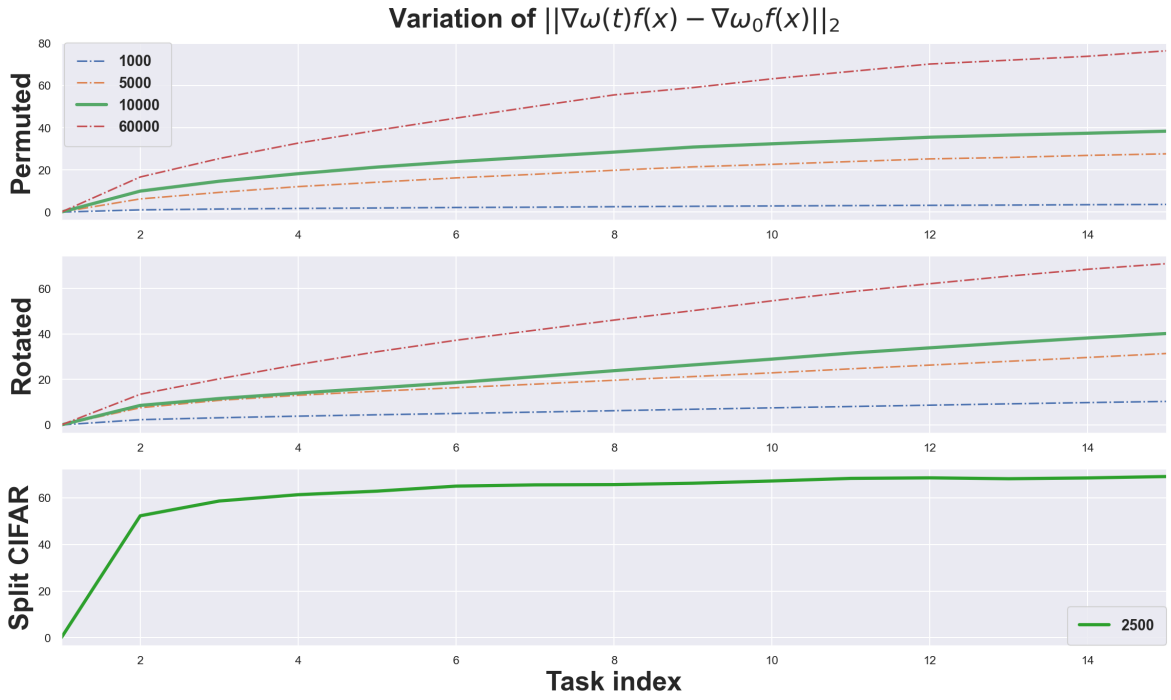


Figure 10: Variation of the NTK for different datasets size (legend).

8.11 Experimental setup and general performance

Datasets We are considering four datasets **Permuted MNIST** [Kirkpatrick et al., 2017], **Rotated MNIST** [Lopez-Paz and Ranzato, 2017], **Split MNIST** and **Split CIFAR** [Zenke et al., 2017]. For MNIST dataset, we sampled $1,000$ examples from each task leading to a total training set size of $10,000$ as in [Lopez-Paz and Ranzato, 2017, Aljundi et al., 2019a].

- Permuted MNIST is coming from the 0-9 digit dataset MNIST [LeCun et al., 1998] where each pixels have been permuted randomly. Each task corresponds to a new permutation randomly generated (but fixed along all the dataset samples).
- Rotated MNIST is the same MNIST dataset where each new task corresponds to a fixed rotation of each digit by a fixed angle. Our 15 tasks correspond to a fixed rotation of 5 degres with respect to the previous task.
- Split MNIST consists of 5 binary classification tasks where we split the digit such as: 0/1 , 2/3 , 4/5 , 6/7 , 8/9.
- Split CIFAR comes from CIFAR-100 dataset [Krizhevsky et al., 2009] which contains 100 classes that can be grouped again into 20 superclasses. Split CIFAR-100 [Lopez-Paz and Ranzato, 2017] is constructed by splitting the dataset into 20 disjoincts classes sampled without replacement. The 20 tasks are then composed of 5 classes.

Baselines We are comparing our method PCA-OGD along with SGD, A-GEM [Chaudhry et al., 2018] and OGD [Farajtabar et al., 2020].

Optimizer We use Stochastic Gradient for each method and grid search to find hyperparameters that gave best results: learning rate of $1e-3$, batch size of 32 and 10 epochs for each tasks.

Performance Metrics Following [Chaudhry et al., 2018] we report the Average Accuracy A_T and the Forgetting Measure F_T:

$$A_T = \frac{1}{T}\sum_{\tau=1}^{T} a_{T,\tau}$$

Where $a_{\tau,T}$ represents the accuracy of task τ at the end of the training of task $T \geq \tau$.

Forgetting Measure [Lopez-Paz and Ranzato, 2017] used the average forgetting as the performance drop of task τ over the training of later tasks:

$$F_T = \frac{1}{T-1}\sum_{\tau=1}^{T-1} f_\tau^T$$

where f_τ^T is defined as the highest forgetting from task τ until T:

$$f_\tau^T = \max_{l=\tau,\ldots,T} a_{l,\tau} - a_{T,\tau}$$

Hyperparameters	Split MNIST	Rotated/Permuted MNIST	CIFAR-100
Dataset size (per task)	2,000	10,000	2,500
Epochs	5	10	50
Architecture	MLP	MLP	LeNet[1]
Hidden dimension	100	100	200
EWC regularizer	10	10	25
# tasks	5	15	20
Optimizer	SGD		
Learning rate	1e-03		
Batch size	32		
Torch seeds	0 to 4		
Memory size	100		
PCA sample size s	3,000		
samples used to compute $O^{\tau_S \to \tau_T}$ (per task)	250		

Table 4: Hyperparameters used across experiments.

[1]For this architecture, we observed better results when only projecting orthogonally to the fully connected layers.

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
SGD	25.55 ± 0.99	27.79 ± 0.85	33.39 ± 1.08	40.97 ± 0.92	49.05 ± 1.07	56.05 ± 0.95	64.48 ± 0.83
EWC	49.16 ± 1.54	52.58 ± 1.85	61.0 ± 1.81	67.77 ± 1.55	72.85 ± 1.18	76.8 ± 1.1	80.05 ± 0.76
AGEM	**65.48 ± 1.38**	**65.93 ± 1.15**	**72.95 ± 0.65**	**77.23 ± 0.55**	**79.38 ± 0.32**	81.9 ± 0.29	84.09 ± 0.25
OGD	44.16 ± 1.52	47.06 ± 1.26	55.75 ± 0.96	63.53 ± 1.37	69.75 ± 0.36	75.67 ± 0.44	80.86 ± 0.19
PCA-OGD	52.51 ± 2.3	55.81 ± 1.79	65.21 ± 1.76	72.79 ± 1.34	79.0 ± 0.8	**83.34 ± 0.57**	**86.65 ± 0.31**

	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Task 14	Task 15
SGD	72.75 ± 0.6	78.55 ± 0.43	84.4 ± 0.26	88.45 ± 0.27	91.08 ± 0.13	92.48 ± 0.09	93.28 ± 0.19	**92.74 ± 0.15**
EWC	82.71 ± 0.65	84.45 ± 0.37	86.32 ± 0.24	87.34 ± 0.31	87.47 ± 0.37	86.9 ± 0.38	85.23 ± 0.58	82.42 ± 0.65
AGEM	85.84 ± 0.12	87.47 ± 0.16	89.6 ± 0.18	91.28 ± 0.09	92.54 ± 0.18	93.13 ± 0.09	**93.33 ± 0.11**	92.71 ± 0.12
OGD	84.75 ± 0.41	87.39 ± 0.38	90.09 ± 0.5	91.7 ± 0.3	92.84 ± 0.28	93.03 ± 0.23	92.9 ± 0.22	91.86 ± 0.19
PCA-OGD	**89.08 ± 0.1**	**90.62 ± 0.2**	**92.02 ± 0.3**	**92.87 ± 0.09**	**93.52 ± 0.17**	**93.18 ± 0.12**	92.85 ± 0.14	91.3 ± 0.15

Table 5: Final Accuracy for Rotated MNIST (averaged over 5 seeds ±1 std).

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7
SGD	56.88 ± 2.85	62.18 ± 6.06	65.96 ± 2.87	66.62 ± 10.57	71.74 ± 3.0	70.86 ± 5.42	77.27 ± 1.69
EWC	**77.5 ± 2.13**	**79.78 ± 1.63**	**81.91 ± 0.94**	**81.92 ± 0.63**	81.26 ± 1.02	80.88 ± 0.89	80.91 ± 1.1
AGEM	75.34 ± 1.92	75.16 ± 1.37	79.41 ± 0.9	79.78 ± 3.22	79.87 ± 1.65	81.16 ± 1.88	82.48 ± 1.28
OGD	45.97 ± 3.6	63.25 ± 3.43	74.25 ± 2.8	78.9 ± 3.15	80.9 ± 1.42	81.99 ± 1.27	83.86 ± 0.61
PCA-OGD	35.47 ± 3.34	64.23 ± 2.05	77.98 ± 1.59	80.82 ± 1.98	**83.21 ± 1.09**	**84.25 ± 1.39**	**85.89 ± 0.45**

	Task 8	Task 9	Task 10	Task 11	Task 12	Task 13	Task 14	Task 15
SGD	75.14 ± 2.34	81.54 ± 2.61	83.31 ± 1.17	85.27 ± 1.71	86.48 ± 0.49	88.34 ± 0.29	89.73 ± 0.48	**90.85 ± 0.16**
EWC	80.98 ± 0.94	80.4 ± 1.33	80.2 ± 1.18	79.77 ± 1.27	78.6 ± 0.91	77.92 ± 0.8	77.3 ± 0.58	76.28 ± 0.67
AGEM	82.81 ± 1.09	85.86 ± 0.67	85.56 ± 0.85	86.44 ± 1.22	87.65 ± 0.81	88.81 ± 0.34	89.91 ± 0.25	90.79 ± 0.21
OGD	85.42 ± 0.59	86.69 ± 0.5	86.84 ± 0.62	87.86 ± 0.65	88.68 ± 0.36	89.28 ± 0.31	89.88 ± 0.23	90.45 ± 0.16
PCA-OGD	**87.08 ± 0.26**	**87.46 ± 0.77**	**87.9 ± 0.5**	**88.34 ± 0.44**	**89.16 ± 0.39**	**89.65 ± 0.14**	**89.93 ± 0.17**	90.29 ± 0.2

Table 6: Final Accuracy for Permuted MNIST (averaged over 5 seeds ±1 std).

	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	Task 7	Task 8	Task 9	Task 10
SGD	48.92 ± 5.11	40.96 ± 2.7	46.52 ± 6.45	40.2 ± 7.27	56.36 ± 8.12	44.76 ± 6.08	54.32 ± 4.9	44.52 ± 5.41	46.24 ± 6.29	59.88 ± 7.56
EWC	**63.28 ± 2.44**	62.32 ± 7.5	57.36 ± 5.88	56.0 ± 6.39	73.56 ± 6.42	52.32 ± 5.96	64.92 ± 3.03	62.44 ± 2.89	53.04 ± 6.24	70.48 ± 5.09
AGEM	38.88 ± 2.81	37.0 ± 5.34	37.28 ± 5.62	32.44 ± 8.53	42.4 ± 12.14	36.72 ± 4.66	41.12 ± 5.16	39.36 ± 4.22	41.92 ± 6.3	47.08 ± 8.52
OGD	52.04 ± 2.92	57.84 ± 5.83	59.96 ± 3.77	56.32 ± 1.47	74.12 ± 2.25	58.04 ± 2.75	69.24 ± 1.18	66.36 ± 3.66	60.84 ± 2.17	77.84 ± 2.2
PCA-OGD	57.24 ± 2.55	**63.08 ± 4.96**	**66.16 ± 0.83**	**61.52 ± 1.09**	**75.32 ± 6.88**	**62.88 ± 1.45**	**73.28 ± 1.06**	**70.48 ± 1.67**	**66.32 ± 1.14**	**80.28 ± 1.22**

	Task 11	Task 12	Task 13	Task 14	Task 15	Task 16	Task 17	Task 18	Task 19	Task 20
SGD	67.56 ± 4.21	49.64 ± 8.46	66.4 ± 4.6	58.68 ± 4.78	65.68 ± 5.11	54.8 ± 13.41	63.6 ± 3.84	57.84 ± 4.5	75.16 ± 2.54	80.12 ± 2.41
EWC	77.92 ± 2.36	67.4 ± 1.51	74.44 ± 1.83	68.28 ± 3.08	72.64 ± 1.87	67.88 ± 3.73	67.08 ± 3.67	53.08 ± 5.02	73.32 ± 1.18	71.32 ± 6.05
AGEM	54.68 ± 7.95	39.48 ± 11.16	52.52 ± 12.46	50.36 ± 7.41	55.4 ± 12.14	55.16 ± 1.8	54.92 ± 9.64	50.28 ± 10.96	65.56 ± 5.36	78.52 ± 1.01
OGD	80.44 ± 1.93	67.8 ± 2.99	80.4 ± 1.04	74.56 ± 0.8	77.92 ± 1.93	72.36 ± 0.82	75.0 ± 0.83	**73.0 ± 1.48**	79.52 ± 1.39	**81.88 ± 1.73**
PCA-OGD	**82.56 ± 0.74**	**71.84 ± 1.81**	**81.88 ± 1.52**	**76.08 ± 1.51**	**80.4 ± 0.95**	**73.52 ± 1.39**	**76.52 ± 0.53**	**73.0 ± 1.63**	**80.36 ± 1.82**	81.28 ± 0.72

Table 7: Final Accuracy for Split CIFAR (averaged over 5 seeds ±1 std).

	Task 1	Task 2	Task 3	Task 4	Task 5
SGD	99.35 ± 0.2	88.62 ± 5.21	94.85 ± 1.69	98.1 ± 0.38	**94.6 ± 0.4**
EWC	99.34 ± 0.19	88.36 ± 5.38	94.96 ± 1.37	98.09 ± 0.39	94.56 ± 0.49
AGEM	99.5 ± 0.22	85.92 ± 7.36	93.31 ± 2.13	98.07 ± 0.42	94.44 ± 0.82
OGD	99.64 ± 0.09	**92.24 ± 1.49**	**95.75 ± 0.4**	98.22 ± 0.39	94.41 ± 0.4
PCA-OGD	**99.67 ± 0.08**	92.22 ± 1.67	95.37 ± 0.72	**98.39 ± 0.28**	94.14 ± 0.42

Table 8: Final Accuracy for Split MNIST (averaged over 5 seeds ±1 std).

8.12 Worst-case scenario for PCA-OGD: data spread uniformly along all directions

In this section, we present a toy example which highlights the drawbacks of PCA-OGD against Catastrophic Forgetting in comparison with OGD, in the special case where magnitude of eigenvalues are spread out.

Experiments In this section, we build a worst case scneario where datapoints $\{X^\tau\}_{\tau=1}^T$ are spread uniformly across all directions. We consider a regression task with a linear model $f_\tau(X^\tau) = (X^\tau)^\top(\omega_\tau(t) - \omega^\star_{\tau-1})$ where $X^\tau \in \mathbb{R}^{n_\tau \times p}$, $\omega \in \mathbb{R}^p$, $\tau \in [T]$. We generate the data as follows for all $\tau \in [T]$:

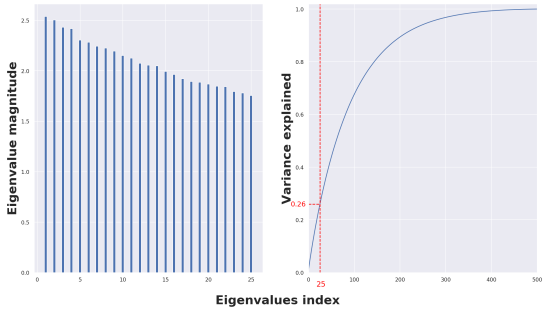
$$\begin{aligned} X^\tau &\sim \mathcal{N}(\mu_{x_\tau}, \sigma_{x_\tau}) \\ \omega^\star_\tau &\sim \mathcal{N}(\mu_{\omega_\tau}, \sigma_{\omega_\tau}) \\ y_\tau &= (X^\tau)^\top \omega^\star_\tau + \epsilon_\tau \\ \epsilon_\tau &\sim \mathcal{N}(0, \sigma_{\epsilon_\tau}) \end{aligned}$$

We are considering Mean Square Error (MSE) for the loss function: $\mathcal{L}_\tau = \frac{1}{n_\tau}\sum_{i=1}^{n_\tau}(y_i^\tau - f_\tau(x_i^\tau))^2, \quad \forall \tau \in [T]$.

Note in this setting, the kernel is simply the which is simply the gradient kernel matrix of the dataset :

$$\phi(X^\tau)\phi(X^\tau)^\top = \nabla_\omega f_\tau(X^\tau)\nabla_\omega f_\tau(X^\tau)^\top = X^\tau(X^\tau)^\top \in \mathbb{R}^{n_\tau \times n_\tau}$$

As shown below in Figure 11a the eigenvalues of the PCA decomposition of $X^\tau(X^\tau)^\top$ are of the same magnitude order and taking the first 25 components only represents 26% of the explained variance. We trained the model on 15 tasks with a total memory of 25 per tasks. We only show below the testing error and forgetting error of the first 9 tasks. As expected, PCA-OGD incurs drastic variation of its loss function while OGD shows practically no forgetting.



(a) Eigenvalues structure of the dataset. The first eigenvalues are sensitively of the same magnitude order (left) such that taking the first 25(5%) only explains 26% of the data variance (right).

(b) Testing loss of OGD (left) versus PCA-OGD (right). OGD incurs almost no forgetting while PCA-OGD has drastic variation in the testing loss over the time.

8.13 Pseudo-code for GEM-NT

Algorithm 2: GEM-NT for Continual Learning

Input : A task sequence $\mathcal{T}_1, \mathcal{T}_2, \ldots$, learning rate η, components to keep d

1. Initialize $S_1 \leftarrow \{\}$; $\omega \leftarrow \omega_0$
2. **for** *Task ID* $\tau = 1, 2, 3, \ldots$ **do**
   repeat
   $\mathbf{g} \leftarrow$ Stochastic Batch Gradient for $\mathcal{T}_\tau$ at ω;
   `// Orthogonal updates`
   $\tilde{\mathbf{g}} = \mathbf{g} - \sum_{(x_k, y_k) \in \mathcal{S}_k, k=1,..,\tau-1} \nabla \mathcal{L}^\tau_\lambda(x_k; y_k))$;
   $\omega \leftarrow \omega - \eta \tilde{\mathbf{g}}$
   until *convergence*;
   `// Compute loss gradient`
   Sample d elements (x_τ, y_τ) from $\mathcal{T}_\tau$
   $S_\tau \leftarrow \{(x_\tau, y_\tau)\}$
   end for

8.14 Computational overhead of PCA-OGD

The only additional step of PCA-OGD over OGD is the PCA operation. The complexity of the Graham Schmidt (GS) step is $O(M^2 p)$ where M is the memory size and p the number of parameters. The PCA operation has complexity $O(n^2 p)$ where $n \ll p$ are the first n components to keep. However, $n = M/T$ with T being the number of tasks hence there is no computational overhead for the PCA step: PCA-OGD and OGD have the same asymptotic complexity.

8.15 Eigenvalues evolution of the NTK overlap matrix between the source and target task

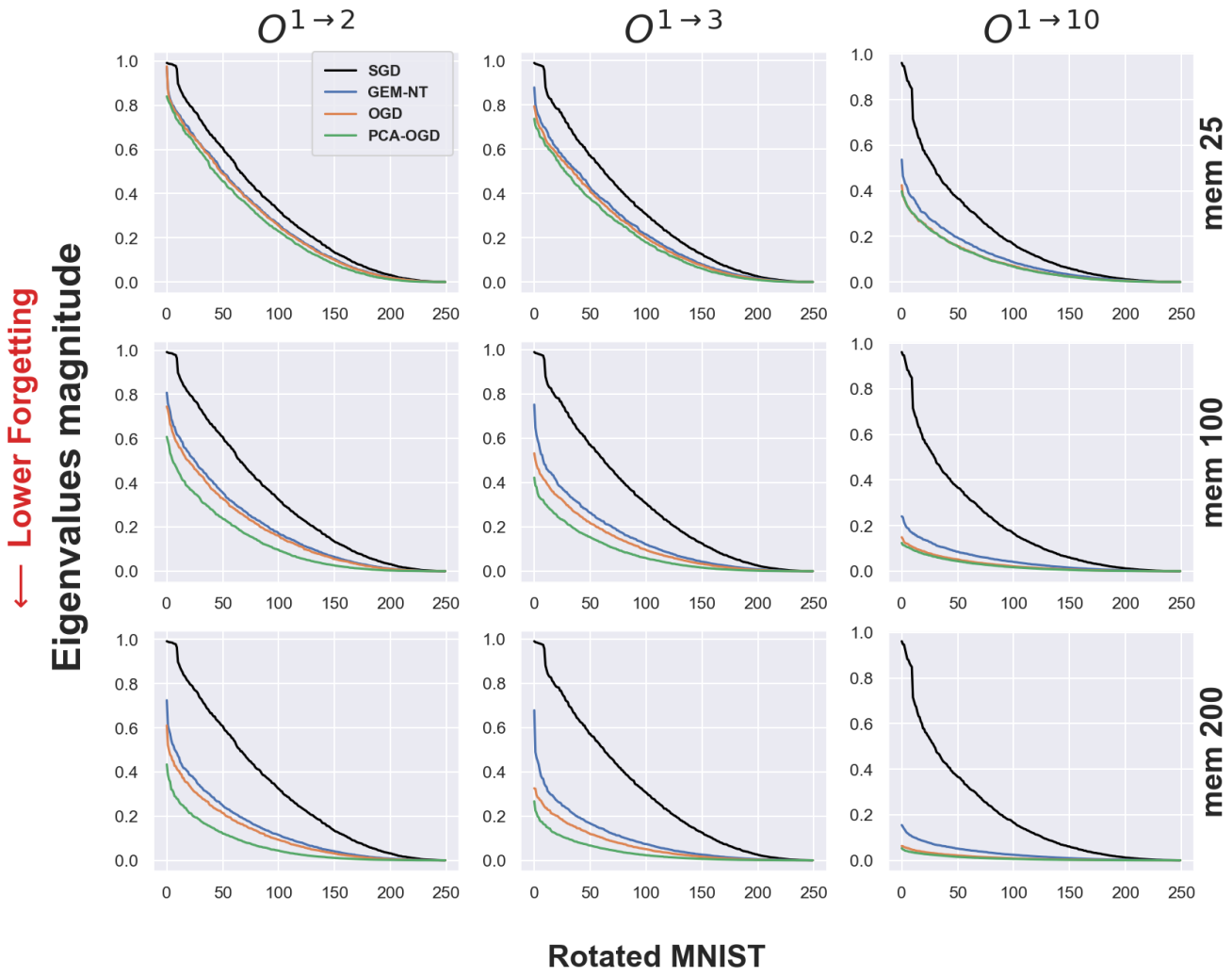


Figure 12: Eigenvalues of the overlap matrix $O^{\tau_S \to \tau_T}$ for different memozy size and methods. Increasing memory gives better advantage to PCA-OGD.

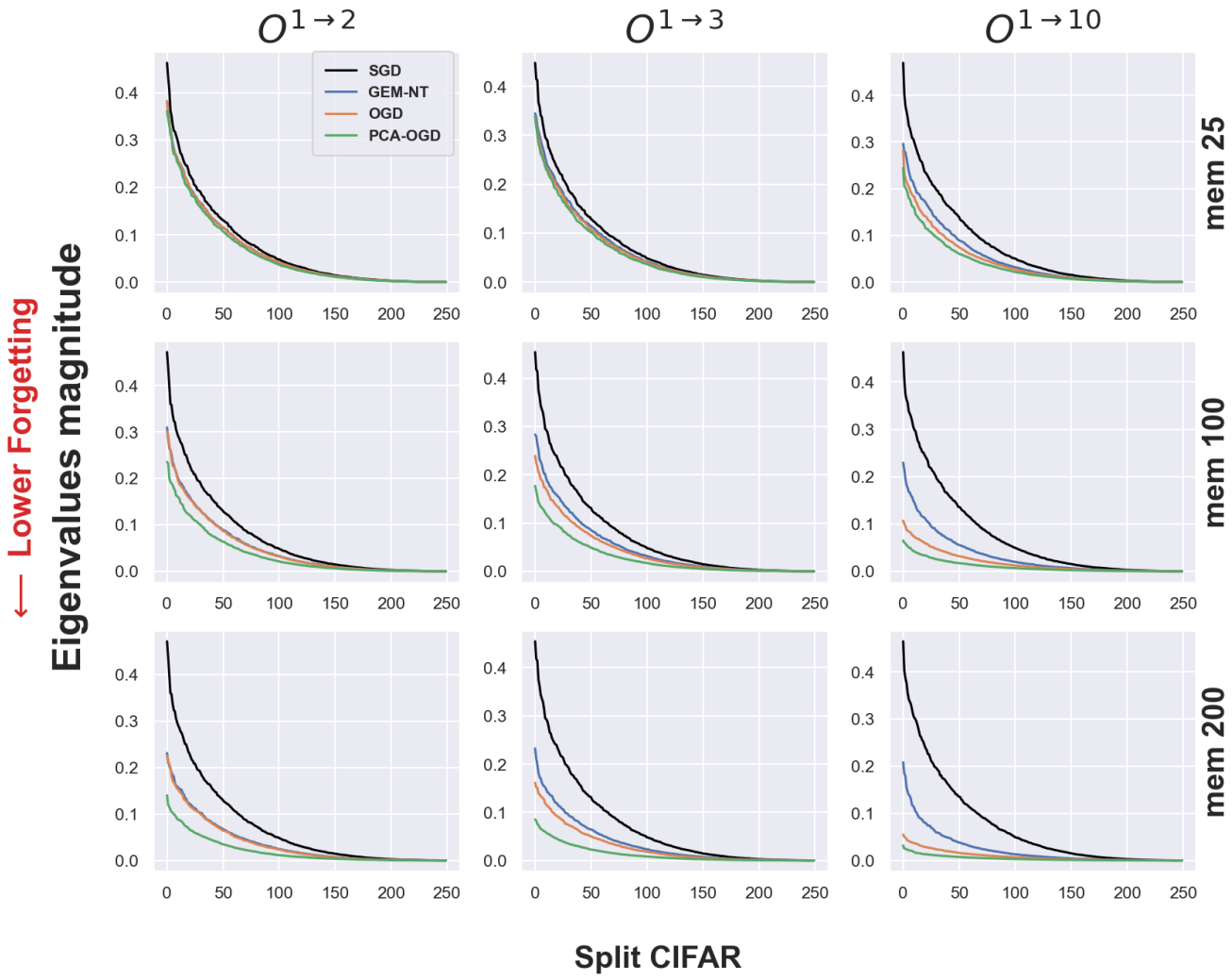


Figure 13: Eigenvalues of the overlap matrix $O^{\tau_S \to \tau_T}$ for different memozy size and methods. Increasing memory gives better advantage to PCA-OGD.

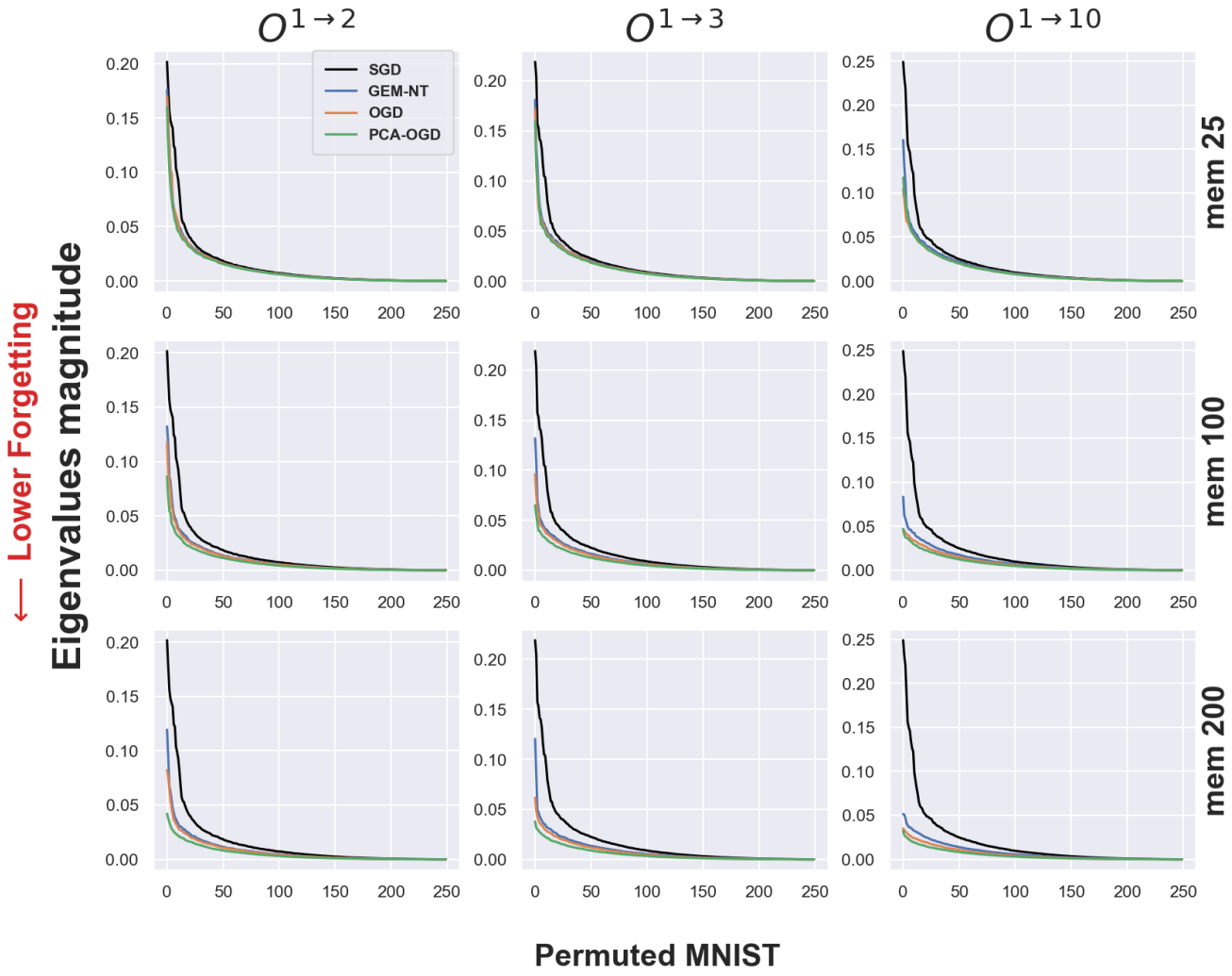


Figure 14: Eigenvalues of the overlap matrix $O^{\tau_S \to \tau_T}$ for different memozy size and methods. Since there is no Pattern accross task of Permuted MNIST, PCA-OGD does not take advantage of keeping principal eigenvalues directions.