A Technical Proofs

Before talking about the main results, the following lemma is used in our analysis.

Lemma 2. *(Proposition 5 in [Mokhtari et al., 2016]) Consider the sample sets $\mathcal{S}_m$ with size m and $\mathcal{S}_n$ with size n such that $\mathcal{S}_m \subset \mathcal{S}_n$. Let w_m is V_m-suboptimal solution of the risk R_m. If assumptions 1 and 2 hold, then the following is true:*

$$R_n(w_m) - R_n(w_n^*) \leq V_m + \tfrac{2(n-m)}{n}(V_{n-m} + V_m) + \\ 2(V_m - V_n) + \tfrac{c(V_m - V_n)}{2}\|w^*\|^2, \;\; w.h.p. \tag{17}$$

If we consider $V_n = \mathcal{O}(\frac{1}{n^\gamma})$ where $\gamma \in [0.5, 1]$, and assume that $n = 2m$ (or $\alpha = 2$), then (17) can be written as (w.h.p):

$$R_n(w_m) - R_n(w_n^*) \leq \Big[3 + \Big(1 - \tfrac{1}{2^\gamma}\Big)\Big(2 + \tfrac{c}{2}\|w^*\|^2\Big)\Big]V_m. \tag{18}$$

A.1 Practical stopping criterion

Here we discuss two stopping criteria to fulfill the 10^{th} line of Algorithm 1. At first, considering w_n^* is unknown in practice, we can use strong convexity inequality as $R_n(\tilde{w}_k) - R_n(w_n^*) \leq \frac{1}{2cV_n}\|\nabla R_n(\tilde{w}_k)\|^2$ to find a stopping criterion for the inner loop, which satisfies $\|\nabla R_n(\tilde{w}_k)\| < (\sqrt{2c})V_n$. Another stopping criterion is discussed by [Zhang and Lin, 2015], using the fact that the risk R_n is self-concordant. This criterion can be written as $\delta_n(\tilde{w}_k) \leq (1-\beta)\sqrt{V_n}$, where $\beta \leq \frac{1}{20}$. The later stopping criterion implies that $R_n(\tilde{w}_k) - R_n(w_n^*) \leq V_n$ whenever $V_n \leq 0.68^2$. For the risk R_n, the same as [Zhang and Lin, 2015] we can define the following auxiliary function and vectors:

$$\omega_*(t) = -t - \log(1-t), \qquad 0 \leq t < 1. \tag{19}$$

$$\tilde{u}_n(\tilde{w}_k) = [\nabla^2 R_n(\tilde{w}_k)]^{-1/2}\nabla R_n(\tilde{w}_k), \tag{20}$$

$$\tilde{v}_n(\tilde{w}_k) = [\nabla^2 R_n(\tilde{w}_k)]^{1/2} v_n. \tag{21}$$

We can note that $\|\tilde{u}_n(\tilde{w}_k)\| = \sqrt{\nabla R_n(\tilde{w}_k)[\nabla^2 R_n(\tilde{w}_k)]^{-1}\nabla R_n(\tilde{w}_k)}$, which is the exact Newton decrement, and, the norm $\|\tilde{v}_n(\tilde{w}_k)\| = \delta_n(\tilde{w}_k)$ which is the approximation of Newton decrement (and $\tilde{u}_n(\tilde{w}_k) = \tilde{v}_n(\tilde{w}_k)$ in the case when $\epsilon_k = 0$). As a result of Theorem 1 in the study of [Zhang and Lin, 2015], we have:

$$(1-\beta)\|\tilde{u}_n(\tilde{w}_k)\| \leq \|\tilde{v}_n(\tilde{w}_k)\| \leq (1+\beta)\|\tilde{u}_n(\tilde{w}_k)\|, \tag{22}$$

where $\beta \leq \frac{1}{20}$. Also, by the equation in (21), we know that $\|\tilde{v}_n(\tilde{w}_k)\| = \delta_n(\tilde{w}_k)$.

As it is discussed in the section 9.6.3. of the study of [Boyd and Vandenberghe, 2004], we have $\omega_*(t) \leq t^2$ for $0 \leq t \leq 0.68$.

According to Theorem 4.1.13 in the study of [Nesterov, 2013], if $\|\tilde{u}_n(\tilde{w}_k)\| < 1$ we have:

$$\omega(\|\tilde{u}_n(\tilde{w}_k)\|) \leq R_n(\tilde{w}_k) - R_n(w_n^*) \leq \omega_*(\|\tilde{u}_n(\tilde{w}_k)\|). \tag{23}$$

Therefore, if $\|\tilde{u}_n(\tilde{w}_k)\| \leq 0.68$, we have:

$$R_n(\tilde{w}_k) - R_n(w_n^*) \leq \omega_*(\|\tilde{u}_n(\tilde{w}_k)\|) \leq \|\tilde{u}_n(\tilde{w}_k)\|^2 \\ \overset{(22)}{\leq} \tfrac{1}{(1-\beta)^2}\|\tilde{v}_n(\tilde{w}_k)\|^2 = \tfrac{1}{(1-\beta)^2}\delta_n^2(\tilde{w}_k) \tag{24}$$

Thus, we can note that $\delta_n(\tilde{w}_k) \leq (1-\beta)\sqrt{V_n}$ concludes that $R_n(\tilde{w}_k) - R_n(w_n^*) \leq V_n$ when $V_n \leq 0.68^2$.

A.2 Proof of Theorem 1

According to the Theorem 1 in [Zhang and Lin, 2015]1, we can derive the iteration complexity by starting from w_m as a good warm start, to reach w_n which is V_n-suboptimal solution for the risk R_n. By Corollary 1 in

[Zhang and Lin, 2015], we can note that if we set ϵ_k the same as (12), after K_n iterations we reach the solution w_n such that $R_n(w_n) - R_n(w_n^*) \le V_n$ where

$$K_n = \left\lceil \frac{R_n(w_m) - R_n(w_n^*)}{\frac{1}{2}\omega(1/6)} \right\rceil + \left\lceil \log_2\left(\frac{2\omega(1/6)}{V_n}\right) \right\rceil. \tag{25}$$

Also, in Algorithm 2, before the main loop, 1 communication round is needed, and in every iteration of the main loop in this algorithm, 1 round of communication happens. According to Lemma 1, we can note that the number of PCG steps needed to reach the approximation of Newton direction with precision ϵ_k is as following:

$$\begin{aligned} C_n(\epsilon_k) &= \left\lceil \sqrt{1 + \frac{2\mu_n}{cV_n}} \log_2\left(\frac{2\sqrt{\frac{cV_n+M}{cV_n}} \|\nabla R_n(\tilde{w}_k)\|}{\epsilon_k} \right) \right\rceil \\ &\overset{(12)}{=} \left\lceil \sqrt{1 + \frac{2\mu_n}{cV_n}} \log_2\left(\frac{2(cV_n+M)}{\beta c V_n} \right) \right\rceil. \end{aligned} \tag{26}$$

Therefore, in every call of Algorithm 2, the number of communication rounds is not larger than $1 + C_n(\epsilon_k)$. Thus, we can note that when we start from w_m, which is V_m-suboptimal solution for the risk R_m, T_n communication rounds are needed, where $T_n \le K_n(1 + C_n(\epsilon_k))$, to reach the point w_n which is V_n-suboptimal solution of the risk R_n, which follows (13).
Suppose the initial sample set contains m_0 samples, and consider the set $\mathcal{P} = \{m_0, \alpha m_0, \alpha^2 m_0, \ldots, N\}$, then with high probability with $\mathcal{T}$ rounds of communication, we reach V_N-suboptimal solution for the whole data set:

$$\mathcal{T} \le \sum_{i=2}^{|\mathcal{P}|} \left(\left\lceil \frac{R_{\mathcal{P}[i]}(w_{\mathcal{P}[i-1]}) - R_{\mathcal{P}[i]}(w^*_{\mathcal{P}[i]})}{\frac{1}{2}\omega(1/6)} \right\rceil + \left\lceil \log_2\left(\frac{2\omega(1/6)}{V_{\mathcal{P}[i]}}\right) \right\rceil \right) \left(1 + \left\lceil \sqrt{1 + \frac{2\mu_{\mathcal{P}[i]}}{cV_{\mathcal{P}[i]}}} \log_2\left(\frac{2(cV_{\mathcal{P}[i]}+M)}{\beta c V_{\mathcal{P}[i]}} \right) \right\rceil \right). \tag{27}$$

A.3 Proof of Corollary 1

The proof of the first part is trivial. According to Lemma 2, we can find the upper bound for $R_n(w_m) - R_n(w_n^*)$, and when $\alpha = 2$, by utilizing the bound (18) we have:

$$\begin{aligned} K_n &= \left\lceil \frac{R_n(w_m) - R_n(w_n^*)}{\frac{1}{2}\omega(1/6)} \right\rceil + \left\lceil \log_2\left(\frac{2\omega(1/6)}{V_n}\right) \right\rceil \\ &\overset{(18)}{\le} \underbrace{\left\lceil \frac{\left(3 + \left(1 - \frac{1}{2^\gamma}\right)\left(2 + \frac{c}{2}\|w^*\|^2\right)\right) V_m}{\frac{1}{2}\omega(1/6)} \right\rceil + \left\lceil \log_2\left(\frac{2\omega(1/6)}{V_n}\right) \right\rceil}_{:=\tilde{K}_n}. \end{aligned} \tag{28}$$

Therefore, we can notice that when we start from w_m, which is V_m-suboptimal solution for the risk R_m, with high probability with $\tilde{T}_n$ communication rounds, where $\tilde{T}_n \le \tilde{K}(1 + C_n(\epsilon_k))$, and $C_n(\epsilon_k)$ is defined in (26), we reach the point w_n which is V_n-suboptimal solution of the risk R_n, which follows (14).
Suppose the initial sample set contains m_0 samples, and consider the set $\mathcal{P} = \{m_0, 2m_0, 4m_0, \ldots, N\}$, then the total rounds of communication, $\tilde{\mathcal{T}}$, to reach V_N-suboptimal solution for the whole data set is bounded as following:

$$\begin{aligned} \tilde{\mathcal{T}} \le& \sum_{i=2}^{|\mathcal{P}|} \left(\left\lceil \frac{\left(3 + \left(1 - \frac{1}{2^\gamma}\right)\left(2 + \frac{c}{2}\|w^*\|^2\right)\right) V_{\mathcal{P}[i-1]}}{\frac{1}{2}\omega(1/6)} \right\rceil + \left\lceil \log_2\left(\frac{2\omega(1/6)}{V_{\mathcal{P}[i]}}\right) \right\rceil \right) \Bigg) \\ & \left(\left\lceil \sqrt{1 + \frac{2\mu}{cV_{\mathcal{P}[i]}}} \log_2\left(\frac{2(cV_{\mathcal{P}[i]}+M)}{\beta c V_{\mathcal{P}[i]}} \right) \right\rceil \right) \\ \le& \Bigg(\log_2 \frac{N}{m_0} + \left(\frac{\left(3 + \left(1 - \frac{1}{2^\gamma}\right)\left(2 + \frac{c}{2}\|w^*\|^2\right)\right)}{\frac{1}{2}\omega(1/6)} \frac{1 - \left(\frac{1}{2^\gamma}\right)^{\log_2 \frac{N}{m_0}}}{1 - \frac{1}{2^\gamma}} V_{m_0} \right) \\ & + \sum_{i=2}^{|\mathcal{P}|} \left\lceil \log_2\left(\frac{2\omega(1/6)}{V_{\mathcal{P}[i]}}\right) \right\rceil \Bigg) \left(\left\lceil \sqrt{1 + \frac{2\mu}{cV_N}} \log_2\left(\frac{2}{\beta} + \frac{2M}{\beta c} \cdot \frac{1}{V_N} \right) \right\rceil \right) \\ \le& \Bigg(2\log_2 \frac{N}{m_0} + \left(\frac{\left(3 + \left(1 - \frac{1}{2^\gamma}\right)\left(2 + \frac{c}{2}\|w^*\|^2\right)\right)}{\frac{1}{2}\omega(1/6)} \frac{1 - \left(\frac{1}{2^\gamma}\right)^{\log_2 \frac{N}{m_0}}}{1 - \frac{1}{2^\gamma}} V_{m_0} \right) \\ & + \log_2 \frac{N}{m_0} \log_2\left(\frac{2\omega(1/6)}{V_N}\right) \Bigg) \left(\left\lceil \sqrt{1 + \frac{2\mu}{cV_N}} \log_2\left(\frac{2}{\beta} + \frac{2M}{\beta c} \cdot \frac{1}{V_N} \right) \right\rceil \right), \quad \text{w.h.p.} \end{aligned}$$

where $\mu = \max\{\mu_{m_0}, \mu_{\alpha m_0}, \ldots, \mu_N\}$.

A.4 Proof of Corollary 2

By Corollary 1, it is shown that. after $\tilde{\mathcal{T}}$ rounds of communication we reach a point with the statistical accuracy of V_N of the full training set, where with high probability $\tilde{\mathcal{T}}$ is bounded above by

$$\begin{aligned}\tilde{\mathcal{T}} \leq \Bigg(& 2\log_2 \tfrac{N}{m_0} + \log_2 \tfrac{N}{m_0} \log_2(\tfrac{2\omega(1/6)}{V_N}) \\ & + \Big(\frac{\left(3+\left(1-\frac{1}{2^\gamma}\right)\left(2+\frac{c}{2}\|w^*\|^2\right)\right)}{\frac{1}{2}\omega(1/6)} \frac{1-(\frac{1}{2^\gamma})^{\log_2 \frac{N}{m_0}}}{1-\frac{1}{2^\gamma}} V_{m_0}\Big)\Bigg) \\ & \left(1 + \left\lceil \sqrt{(1+\tfrac{2\mu}{cV_N})} \log_2\left(\tfrac{2}{\beta} + \tfrac{2M}{\beta c}\cdot\tfrac{1}{V_N}\right)\right\rceil\right), \end{aligned} \tag{29}$$

where m_0 is the size of the initial training set. Note that the result in (29) implies that the overall rounds of communication to obtain the statistical accuracy of the full training set is $\tilde{\mathcal{T}} = \tilde{\mathcal{O}}(\gamma(\log_2 N)^2\sqrt{N^\gamma}\log_2 N^\gamma)$. Hence, when $\gamma = 1$, we have $\tilde{\mathcal{T}} = \tilde{\mathcal{O}}((\log_2 N)^3\sqrt{N})$, and for $\gamma = 0.5$, the result is $\tilde{\mathcal{O}} = \mathcal{O}((\log_2 N)^3 N^{\frac{1}{4}})$.

A.5 Proof of Theorem 2

By using Woodbury Formula [Ma and Takáč, 2016, Press et al., 2007], every PCG iteration has the cost of $\mathcal{O}(d^2)$. The reason comes from the fact that the total computations needed for applying Woodbury Formula are $\mathcal{O}(\Lambda^3)$, where $\Lambda = \max\{|\mathcal{A}_{m_0}|, |\mathcal{A}_{\alpha m_0}|, \ldots, |\mathcal{A}_N|\}$, and $\Lambda \leq 100$. The complexity of every iteration of PCG is $\mathcal{O}(d^2 + \Lambda^3)$ or equivalently $\mathcal{O}(d^2)$, and by using the results in the proof of Corrolary 2, the total complexity of DANCE is $\tilde{\mathcal{O}}((\log_2(N))^3 N^{1/4} d^2)$.

B Details Concerning Experimental Section

In this section, we describe our datasets and implementation details. Along this work, we select four datasets to demonstrate the efficiency of our Algorithm 1. Two of them are for convex loss case for a binary classification task using logistic model and the other two are non-convex loss for a multi-labels classification task using convolutional neural networks. The details of the dataset are summarized in Table 2.

Table 2: Summary of two binary classification datasets and two multi-labels classification datasets

Dataset	**# of samples**	**# of features**	**# of categories**
rcv1	20,242	47,326	2
gisette	7,242	5,000	2
Mnist	60,000	28*28	10
Cifar10	60,000	28*28*3	10

In terms of non-convex cases, we select two convolutional structure for the demonstration. NaiveCNet is a simple two convolutional layer network for Mnist dataset, and Vgg11 is a relative larger model with 8 convolutional layers. The details of the network architecture is summarized in Table 3. Note that for vgg11, a batch normalization layer is applied right after each convolutional layer.

C Additional Plots

Besides the plots in Section 5, we also experimented different data sets, and the other corresponding settings are described in the main body.

Table 3: Summary of two convolutional neural network architecture

Architecture	**NaiveCNet**	**Vgg11**
conv-1	$(5 \times 5 \times 16)$, stride=1	$(3 \times 3 \times 64)$, stride=1
max-pool-1	(2×2), stride=2	(2×2),stride=2
conv- 2	$(5 \times 5 \times 32)$, stride=1	$(3 \times 3 \times 128)$, stride=1
max-pool-2	(2×2), stride=2	(2×2), stride=2
conv- 3		$(3 \times 3 \times 256)$, stride=1
max-pool-3		(2×2), stride=2
conv- 4		$(3 \times 3 \times 256)$
max-pool-4		(2×2), stride=2
conv- 5		$(3 \times 3 \times 512)$, stride $= 1$
max-pool-5		(2×2), stride=2
conv- 6		$(3 \times 3 \times 512)$, stride $= 1$
max-pool-6		(2×2), stride=2
conv- 7		$(3 \times 3 \times 512)$, stride $= 1$
max-pool-7		(2×2), stride=2
conv- 8		$(3 \times 3 \times 512)$, stride $= 1$
max-pool-8		(2×2), stride=2
fc		512
output	10	10

C.1 Sensitivity Analysis of DANCE's Parameters

In this section, we consider different possible values of hyper-parameters for DANCE, and as it is clear from Figures 6, 7, 8 and 9, DANCE behaves in a robust way when changes in the hyper-parameters happen. In other words, DANCE is not that much sensitive to the choice of hyper-parameters.

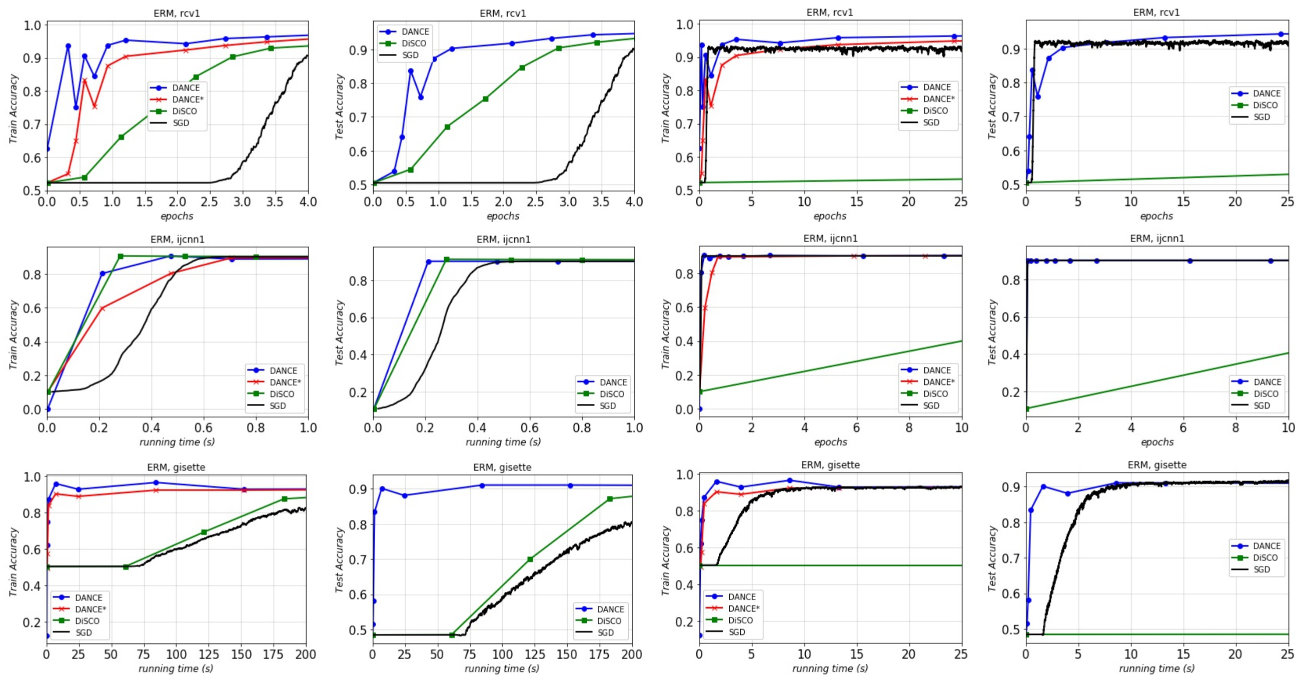

Figure 5: Performance of different algorithms on a logistic regression problem with rcv1, ijcnn1 and gissete datasets. In the left two figures, the plot *DANCE** is the training accuracy based on the entire training set, while the plot *DANCE* represents the training accuracy based on the current sample size.

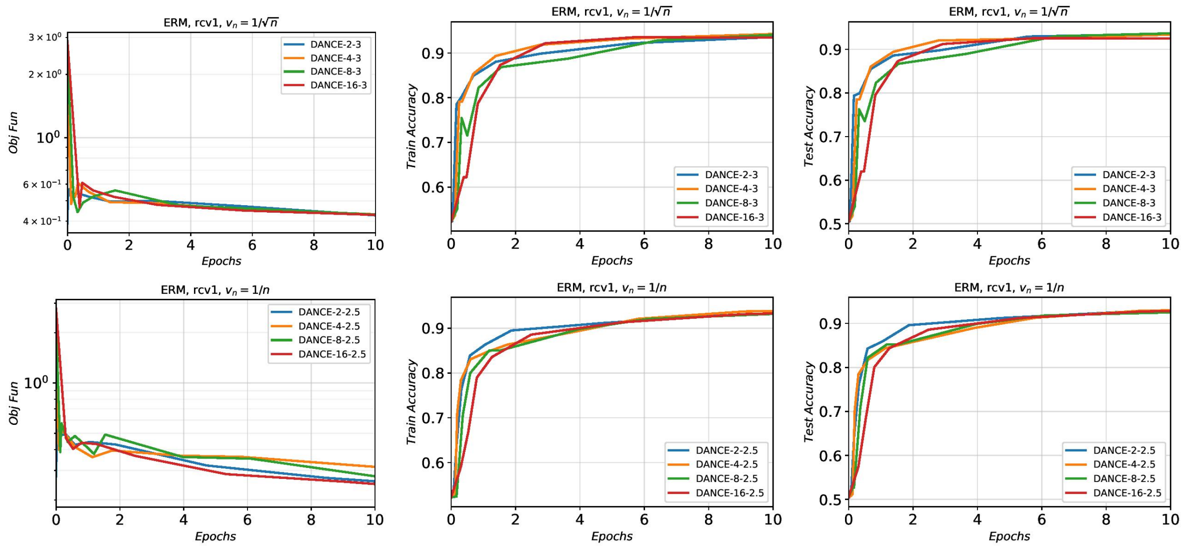

Figure 6: DANCE performance with respect to fixed α and different values of initial samples. The first row shows the results of DANCE for “rcv1” dataset when $V_n = \dfrac{1}{\sqrt{n}}$. The second row shows the results of DANCE for “rcv1” dataset when when $V_n = \dfrac{1}{n}$.

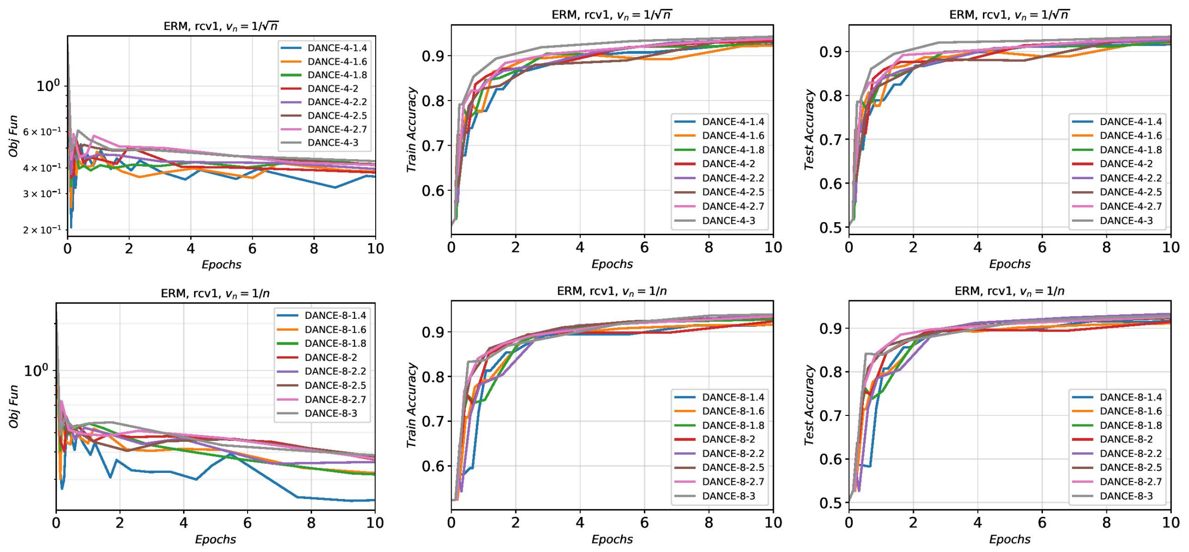

Figure 7: DANCE performance with respect to different values of α and fixed number of initial sample. The first row shows the results of DANCE for "rcv1" dataset when $V_n = \frac{1}{\sqrt{n}}$. The second row shows the results of DANCE for "rcv1" dataset when when $V_n = \frac{1}{n}$.

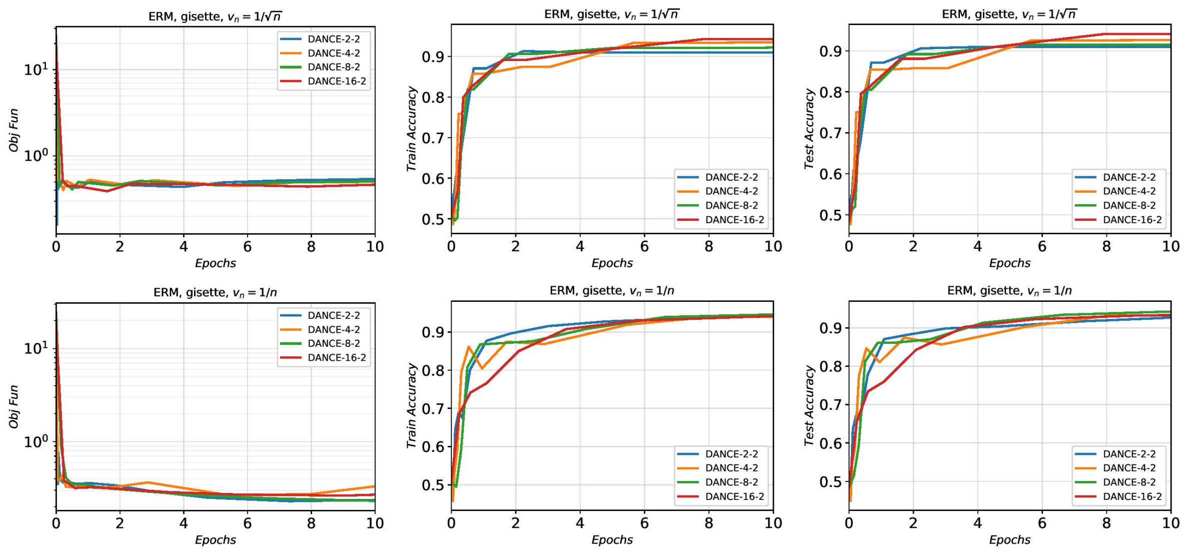

Figure 8: DANCE performance with respect to fixed α and different values of initial samples. The first row shows the results of DANCE for "gisette" dataset when $V_n = \frac{1}{\sqrt{n}}$. The second row shows the results of DANCE for "gisette" dataset when when $V_n = \frac{1}{n}$.

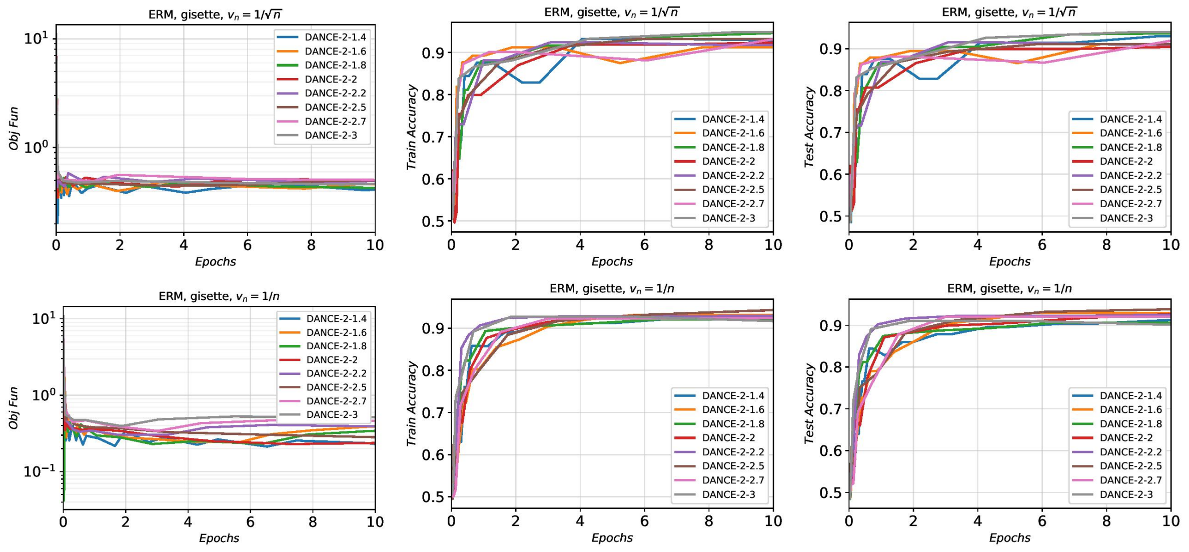

Figure 9: DANCE performance with respect to different values of α and fixed number of initial sample. The first row shows the results of DANCE for "gisette" dataset when $V_n = \dfrac{1}{\sqrt{n}}$. The second row shows the results of DANCE for "gisette" dataset when when $V_n = \dfrac{1}{n}$.

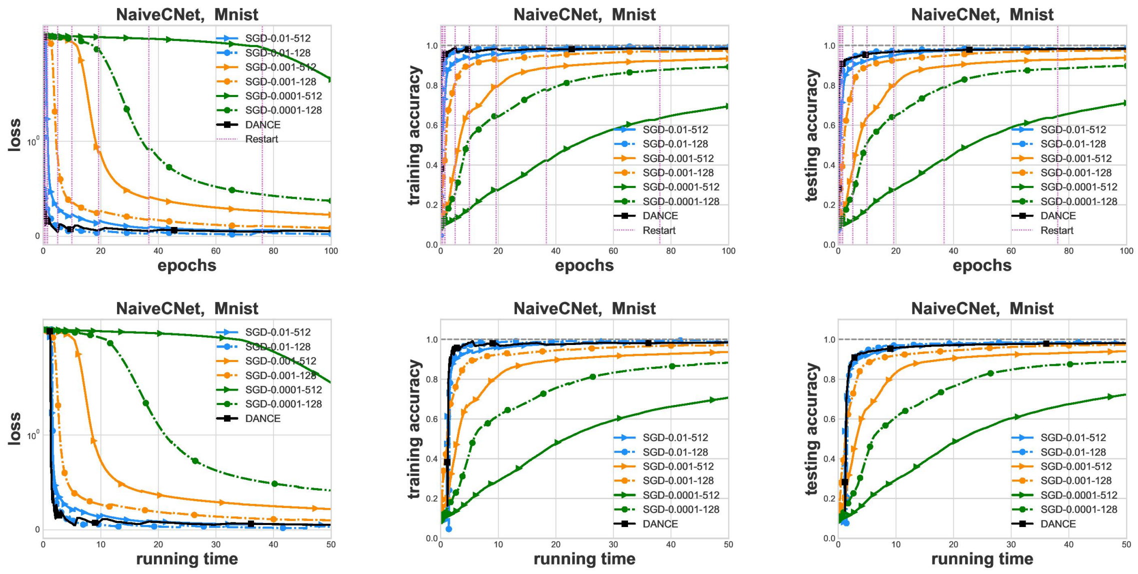

Figure 10: Comparison between DANCE and SGD with various hyper-parameters on Mnist dataset and NaiveCNet. NaiveCNet is a basic CNN with 2 convolution layers and 2 max-pool layers (see details at Appendix B). Figures on the top and bottom show how loss values, training accuracy and test accuracy are changing with respect to epochs and running time. We force two algorithms to restart (double training sample size) after achieving the following number of epochs: $0.075, 0.2, 0.6.1.6, 4.8, 9.6, 18, 36, 72$. For SGD, we varies learning rate from $0.01, 0.001, 0.0001$ and batchsize from $128, 512$. One can observe that SGD is sensitive to hyper-parameter settings, while DANCE has few parameters to tune but still shows competitive performance.

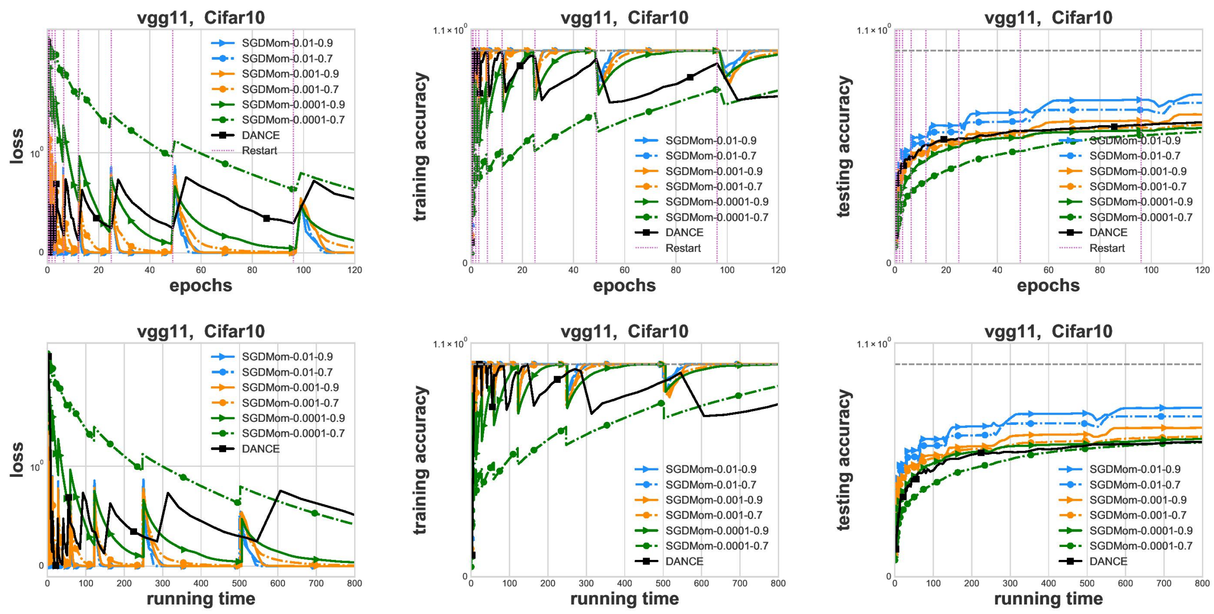

Figure 11: Comparison between DANCE and with momentum for various hyper-parameters on Cifar10 dataset and vgg11 network. Figures on the top and bottom show how loss values, training accuracy and test accuracy are changing regarding epochs and running time, respectively. We force two algorithms to restart (double training sample size) after running the following number of epochs: $0.2, 0.8, 1.6.3.2, 6.4, 12, 24, 48, 96$. For SGD with momentum, we fix the batchsize to be 256 and varies learning rate from $0.01, 0.001, 0.0001$ and momentum parameter from $0.7, 0.9$. One can observe that SGD with momentum is sensitive to hyper-parameter settings, while DANCE has few hyper-parameters to tune but still shows competitive performance.

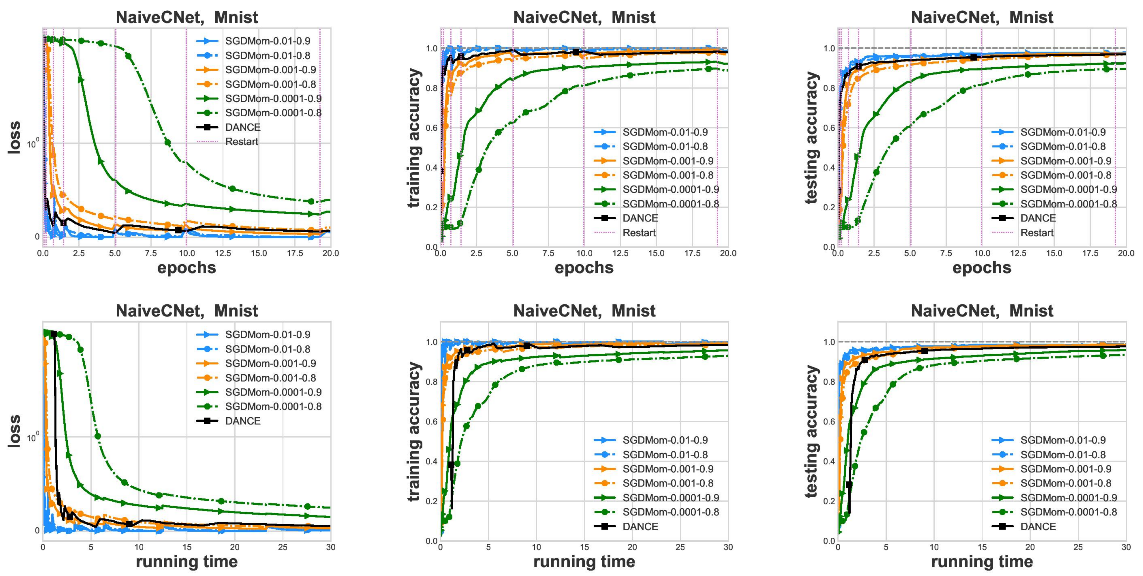

Figure 12: Comparison between DANCE and SGD with momentum for various hyper-parameters on Mnist dataset and NaiveCNet. Figures on the top and bottom show how loss values, training accuracy and test accuracy are changing regarding epochs and running time, respectively. We force two algorithms to restart (double training sample size) after running the following number of epochs: $0.075, 0.2, 0.6.1.6, 4.8, 9.6, 18, 36, 72$. For SGD with momentum, we fix the batchsize to be 128 and set learning rate to be $0.01, 0.001, 0.0001$ and momentum parameter to be $0.8, 0.9$. One could observe that SGD with momentum is sensitive to hyper-parameter settings, while DANCE has few hyper-parameters to tune but still shows competitive performance.