A Detailed Proofs for Binary Classification

A.1 Proof of Theorem 1

Theorem 1. *The dual problem of Eq.* (5) *is given by:*

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & -c\sum_{i=1}^{n}\ell^*\left(-\frac{\alpha_i}{c}\right) \\ \text{subject to} \quad & \lambda_{\max}\Big(\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j)\Big)\leq 1\,, \\ & \alpha_i\geq 0\,, \quad \forall i\in[n]\,, \end{aligned} \tag{6}$$

in which α_i's are the dual variables, $\ell^*(\cdot)$ is the Fenchel conjugate of the loss function $\ell(\cdot)$, $K\left(\mathbf{x}_i,\mathbf{x}_j\right)=\Phi\left(\mathbf{x}_i\right)^\top\Phi\left(\mathbf{x}_j\right)$ is a kernel generating matrix, such that its entries are composed of the kernel function $\mathcal{K}\left(\cdot,\cdot\right)=\phi\left(\cdot\right)^\top\phi\left(\cdot\right)$ taking one patch from each of the two samples as input:

$$K\left(\mathbf{x}_i,\mathbf{x}_j\right)=\begin{pmatrix}\mathcal{K}\big(z_1(\mathbf{x}_i),z_1(\mathbf{x}_j)\big) & \mathcal{K}\big(z_1(\mathbf{x}_i),z_2(\mathbf{x}_j)\big) & \cdots & \mathcal{K}\big(z_1(\mathbf{x}_i),z_p(\mathbf{x}_j)\big)\\ \mathcal{K}\big(z_2(\mathbf{x}_i),z_1(\mathbf{x}_j)\big) & \mathcal{K}\big(z_2(\mathbf{x}_i),z_2(\mathbf{x}_j)\big) & \cdots & \mathcal{K}\big(z_2(\mathbf{x}_i),z_p(\mathbf{x}_j)\big)\\ \vdots & \vdots & \ddots & \vdots\\ \mathcal{K}\big(z_p(\mathbf{x}_i),z_1(\mathbf{x}_j)\big) & \mathcal{K}\big(z_p(\mathbf{x}_i),z_2(\mathbf{x}_j)\big) & \cdots & \mathcal{K}\big(z_p(\mathbf{x}_i),z_p(\mathbf{x}_j)\big)\end{pmatrix}. \tag{7}$$

Proof. For Lagrangian variables $\alpha_i\geq 0$, we can write the Lagrangian function of the optimization problem in Eq. (5) as

$$\begin{aligned}\mathcal{L}\left(A,\xi,\alpha\right)&=\|A\|_*+c\sum_{i=1}^{n}\ell(\xi_i)+\sum_{i=1}^{n}\alpha_i\left(\xi_i-y_i\operatorname{Tr}\left(\Phi\left(\mathbf{x}_i\right)^\top A\right)\right)\\ &=\|A\|_*-\sum_{i=1}^{n}\alpha_i y_i\operatorname{Tr}\left(\Phi\left(\mathbf{x}_i\right)^\top A\right)+c\sum_{i=1}^{n}\ell(\xi_i)+\sum_{i=1}^{n}\alpha_i\xi_i\\ &=\|A\|_*-\operatorname{Tr}\left(\sum_{i=1}^{n}\alpha_i y_i\Phi\left(\mathbf{x}_i\right)^\top A\right)+c\sum_{i=1}^{n}\ell(\xi_i)+\sum_{i=1}^{n}\alpha_i\xi_i\\ &=\|A\|_*-\left\langle\sum_{i=1}^{n}\alpha_i y_i\Phi\left(\mathbf{x}_i\right),A\right\rangle+c\sum_{i=1}^{n}\ell(\xi_i)+\sum_{i=1}^{n}\alpha_i\xi_i,\end{aligned} \tag{8}$$

in which Eq. (8) utilizes the trace definition of the matrix inner product.

By the definition of the dual function,

$$\begin{aligned}g\left(\alpha\right)&=\min_{A,\xi}\mathcal{L}\left(A,\xi,\alpha\right)\\ &=\min_{A}\|A\|_*-\left\langle\sum_{i=1}^{n}\alpha_i y_i\Phi\left(\mathbf{x}_i\right),A\right\rangle+\min_{\xi}c\sum_{i=1}^{n}\ell(\xi_i)+\sum_{i=1}^{n}\alpha_i\xi_i\\ &=-\underbrace{\max_{A}\left\{\left\langle\sum_{i=1}^{n}\alpha_i y_i\Phi\left(\mathbf{x}_i\right),A\right\rangle-\|A\|_*\right\}}_{g_1}\underbrace{-\max_{\xi}\left\{-c\sum_{i=1}^{n}\ell(\xi_i)-\sum_{i=1}^{n}\alpha_i\xi_i\right\}}_{g_2}\end{aligned} \tag{18}$$

We notice that by the definition of conjugate function (Boyd & Vandenberghe, 2004), g_1 takes the form of the conjugate function of the nuclear norm, i.e. for $f_0 = \|A\|_*$

$$g_1 = f_0^* \left(\sum_{i=1}^{n} \alpha_i y_i \Phi\left(\mathbf{x}_i\right) \right), \tag{10}$$

in which

$$f_0^*\left(\cdot\right) = \begin{cases} 0\,, & \|\cdot\|_2 \leq 1 \\ \infty, & otherwise \end{cases}$$

Also,

$$\begin{aligned} g_2 &= \max_{\xi} \left\{ -c \sum_{i=1}^{n} \ell(\xi_i) - \sum_{i=1}^{n} \alpha_i \xi_i \right\} \\ &= c \sum_{i=1}^{n} \max_{\xi_i} \left(-\frac{\alpha_i}{c} \cdot \xi_i - \ell(\xi_i) \right) \\ &= c \sum_{i=1}^{n} \ell^* \left(-\frac{\alpha_i}{c} \right), \end{aligned}$$

in which $\ell^*(\cdot) = \sup_{\xi_i} \left\{ \langle \cdot, \xi_i \rangle - \ell(\xi_i) \right\}$ is the conjugate function of the loss function $\ell(\xi_i)$. Therefore, the dual problem of Eq. (5) is as follows:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & -c \sum_{i=1}^{n} \ell^* \left(-\frac{\alpha_i}{c} \right) \\ \text{subject to} \quad & \left\| \sum_{i=1}^{n} \alpha_i y_i \Phi\left(\mathbf{x}_i\right) \right\|_2 \leq 1\,, \\ & \alpha_i \geq 0\,, \quad \forall i \in [n]\,. \end{aligned} \tag{11}$$

The spectral norm constraint in Eq. (11) is equivalent to

$$\left\| \sum_{i=1}^{n} \alpha_i y_i \Phi\left(\mathbf{x}_i\right) \right\|_2^2 \leq 1.$$

Furthermore,

$$\begin{aligned} \left\| \sum_{i=1}^{n} \alpha_i y_i \Phi\left(\mathbf{x}_i\right) \right\|_2^2 &= \sigma_{\max} \left(\sum_{i=1}^{n} \alpha_i y_i \Phi\left(\mathbf{x}_i\right) \right)^2 \\ &= \lambda_{\max} \left(\left(\sum_{i=1}^{n} \alpha_i y_i \Phi\left(\mathbf{x}_i\right) \right)^\top \left(\sum_{j=1}^{n} \alpha_j y_j \Phi\left(\mathbf{x}_j\right) \right) \right) \\ &= \lambda_{\max} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \Phi\left(\mathbf{x}_i\right)^\top \Phi\left(\mathbf{x}_j\right) \right). \end{aligned} \tag{19}$$

Therefore we have:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & -c\sum_{i=1}^{n}\ell^*\left(-\frac{\alpha_i}{c}\right) \\ \text{subject to} \quad & \lambda_{\max}\Big(\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j)\Big)\leq 1\,, \\ & \alpha_i\geq 0\,, \quad \forall i\in[n]\,, \end{aligned} \tag{6}$$

in which $K\left(\mathbf{x}_i,\mathbf{x}_j\right)=\Phi\left(\mathbf{x}_i\right)^\top\Phi\left(\mathbf{x}_j\right)$ is the kernel generating matrix, and

$$K\left(\mathbf{x}_i,\mathbf{x}_j\right)=\begin{pmatrix} \mathcal{K}\left(z_1\left(\mathbf{x}_i\right),z_1\left(\mathbf{x}_j\right)\right) & \mathcal{K}\left(z_1\left(\mathbf{x}_i\right),z_2\left(\mathbf{x}_j\right)\right) & \cdots & \mathcal{K}\left(z_1\left(\mathbf{x}_i\right),z_p\left(\mathbf{x}_j\right)\right) \\ \mathcal{K}\left(z_2\left(\mathbf{x}_i\right),z_1\left(\mathbf{x}_j\right)\right) & \mathcal{K}\left(z_2\left(\mathbf{x}_i\right),z_2\left(\mathbf{x}_j\right)\right) & \cdots & \mathcal{K}\left(z_2\left(\mathbf{x}_i\right),z_p\left(\mathbf{x}_j\right)\right) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{K}\left(z_p\left(\mathbf{x}_i\right),z_1\left(\mathbf{x}_j\right)\right) & \mathcal{K}\left(z_p\left(\mathbf{x}_i\right),z_2\left(\mathbf{x}_j\right)\right) & \cdots & \mathcal{K}\left(z_p\left(\mathbf{x}_i\right),z_p\left(\mathbf{x}_j\right)\right) \end{pmatrix}, \tag{7}$$

in which $\mathcal{K}\left(\cdot,\cdot\right)=\phi\left(\cdot\right)^\top\phi\left(\cdot\right)$ is the kernel function. □

A.2 Proof of Theorem 2

Theorem 2. *Given the optimal dual solution $\{\hat{\alpha}_i\}_{i=1}^n$ to the problem in Eq. (6), let $\tilde{V}\in\mathbb{R}^{p\times p}$ be the matrix of eigenvectors from the eigendecomposition*

$$\tilde{V}\Lambda\tilde{V}^{-1}=\sum_{i=1}^{n}\sum_{j=1}^{n}\hat{\alpha}_i\hat{\alpha}_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j), \tag{20}$$

where $\Lambda=\mathrm{diag}(\lambda_1,...,\lambda_p)\in\mathbb{R}^{p\times p}$. The linear weight $\hat{V}_1$ can be recovered by the eigenvectors in $\tilde{V}$ corresponding to the eigenvalue of 1, that is,

$$\hat{V}_1=\left[\tilde{V}(i)\right]\in\mathbb{R}^{p\times r},\ \textit{for all } i \textit{ such that } \lambda_i=1, \tag{14}$$

in which $\tilde{V}(i)$ denotes the i^{th} column of $\tilde{V}$ and $[\cdot]$ denotes the concatenation of these columns.

Proof. By the stationary condition in Lemma 1, we know that the there exists one element in the subdifferential of $\|A\|$ at value $\hat{A}$ that satisfies

$$\sum_{i=1}^{n}\hat{\alpha}_i y_i\Phi\left(\mathbf{x}_i\right)\in\partial\|\hat{A}\|_*.$$

Then for $\hat{A}$, consider its compact SVD $\hat{A}=\hat{U}_1\hat{D}_1\hat{V}_1^\top$ and the full SVD $\hat{A}=\hat{U}\hat{D}\hat{V}^\top$ in which $\hat{U}=[\hat{U}_1\mid\hat{U}_2]$, $\hat{V}=[\hat{V}_1\mid\hat{V}_2]$. By Lemma 2, we know that $\exists\ \hat{E}$ such that

$$\hat{U}_1\hat{V}_1^\top+\hat{U}_2\hat{E}\hat{V}_2^\top=\sum_{i=1}^{n}\hat{\alpha}_i y_i\Phi\left(\mathbf{x}_i\right) \tag{21}$$

Now we construct the kernel generating matrix with Eq. (21) under the intuition to avoid the basis function matrix $\Phi(\mathbf{x}_i)$. Let $T=\hat{U}_1\hat{V}_1^\top+\hat{U}_2\hat{E}\hat{V}_2^\top=\sum_{i=1}^{n}\hat{\alpha}_i y_i\Phi\left(\mathbf{x}_i\right)$ and compute $S=T^\top T$. For the right side of Eq. (21), we get the expression containing the kernel generating matrix $K(\mathbf{x}_i,\mathbf{x}_j)$:

$$\begin{aligned} S&=\sum_{i=1}^{n}\sum_{j=1}^{n}\hat{\alpha}_i\hat{\alpha}_j y_i y_j\Phi\left(\mathbf{x}_i\right)^\top\Phi\left(\mathbf{x}_j\right) \\ &=\sum_{i=1}^{n}\sum_{j=1}^{n}\hat{\alpha}_i\hat{\alpha}_j y_i y_j K(\mathbf{x}_i,\mathbf{x}_j). \end{aligned} \tag{22}$$

For the left side of Eq. (21), we know by the properties of SVD that $\hat{U}_1$ and $\hat{U}_2$ are semi-orthogonal matrices, i.e., $\hat{U}_1^\top \hat{U}_1 = \mathbf{I}_r$, $\hat{U}_2^\top \hat{U}_2 = \mathbf{I}_{(d_2-r)}$, and also $\hat{U}_1 \perp \hat{U}_2$, i.e., $\hat{U}_1^\top \hat{U}_2 = \hat{U}_2^\top \hat{U}_1 = \mathbf{0}$. Therefore, by plugging in $T = \hat{U}_1 \hat{V}_1^\top + \hat{U}_2 \hat{E} \hat{V}_2^\top$, we get

$$\begin{aligned} S &= \left(\hat{U}_1 \hat{V}_1^\top + \hat{U}_2 \hat{E} \hat{V}_2^\top\right)^\top \left(\hat{U}_1 \hat{V}_1^\top + \hat{U}_2 \hat{E} \hat{V}_2^\top\right) \\ &= \hat{V}_1 \hat{U}_1^\top \hat{U}_1 \hat{V}_1^\top + \hat{V}_1 \hat{U}_1^\top \hat{U}_2 \hat{E} \hat{V}_2^\top + \hat{V}_2 \hat{E}^\top \hat{U}_2^\top \hat{U}_1 \hat{V}_1^\top + \hat{V}_2 \hat{E}^\top \hat{U}_2^\top \hat{U}_2 \hat{E} \hat{V}_2^\top \\ &= \hat{V}_1 \hat{V}_1^\top + \hat{V}_2 \hat{E}^\top \hat{E} \hat{V}_2^\top. \end{aligned} \tag{23}$$

Now we show that the eigenvectors of $\hat{V}_1 \hat{V}_1^\top$ can be found in the eigenvectors of S with corresponding eigenvalues being 1. One way to see this is that, since $\hat{V}_1 \perp \hat{V}_2$, we have,

$$\left(\hat{V}_1 \hat{V}_1^\top\right)\left(\hat{V}_2 \hat{E}^\top \hat{E} \hat{V}_2^\top\right) = \left(\hat{V}_2 \hat{E}^\top \hat{E} \hat{V}_2^\top\right)\left(\hat{V}_1 \hat{V}_1^\top\right) = \mathbf{0},$$

meaning that $\hat{V}_1 \hat{V}_1^\top$ and $\hat{V}_2 \hat{E}^\top \hat{E} \hat{V}_2^\top$ commute. Since $\hat{V}_1 \hat{V}_1^\top$ and $\hat{V}_2 \hat{E}^\top \hat{E} \hat{V}_2^\top$ are both real symmetric matrices that are diagonalizable, they are simultaneously diagonalizable. That is to say, $\exists\ \tilde{V} \in \mathbb{R}^{p \times p}$ from the eigendecomposition of S, such that

$$S = \tilde{V} \Lambda \tilde{V}^{-1},\ \hat{V}_1 \hat{V}_1^\top = \tilde{V} \Lambda_v \tilde{V}^{-1},\ \textit{and}\ \hat{V}_2 \hat{E}^\top \hat{E} \hat{V}_2^\top = \tilde{V} \Lambda_e \tilde{V}^{-1}. \tag{24}$$

On the diagonal of $\Lambda \in \mathbb{R}^{p \times p}$ are the eigenvalues of S, i.e. $\Lambda = \text{diag}(\lambda_1, ..., \lambda_p)$, Λ_v the eigenvalues of $\hat{V}_1 \hat{V}_1^\top$, and Λ_e the eigenvalues of $\hat{T}^\top \hat{T}$. We also have

$$\Lambda = \Lambda_v + \Lambda_e. \tag{25}$$

Since $\hat{V}_1$ is semi-orthogonal, there are only 0's and 1's on the diagonal of Λ_v. Meanwhile, by the Lemma 2, we know that $\sigma_{\max}(E) \leq 1$. For the moment, assume that $\sigma_{\max}(E) < 1$. That is to say, the elements on the diagonal of Λ_e lie in the range $[0, 1)$. Moreover, given that $\hat{V}_1 \perp \hat{V}_2$, we know that

$$\Lambda_{v,ii} = \begin{cases} 0\ , & \forall i\ \textit{such that}\ \Lambda_{e,ii} > 0 \\ 1 & \forall i\ \textit{such that}\ \Lambda_{e,ii} = 0 \end{cases} \tag{26}$$

Therefore, the eigenvectors $\left[\tilde{V}(i)\right]$ with $\Lambda_{ii} = 1$ are the eigenvectors of $\hat{V}_1 \hat{V}_1^\top$. Since $\hat{V}_1$ is from the compact SVD with $\text{rank}(\hat{V}_1) = r$, $\hat{V}_1 = \left[\tilde{V}(i)\right]$ is the linear weight we recover. We can compute $\tilde{V}$ and Λ by

$$\tilde{V} \Lambda \tilde{V}^{-1} = \sum_{i=1}^{n} \sum_{j=1}^{n} \hat{\alpha}_i \hat{\alpha}_j y_i y_j K\left(\mathbf{x}_i, \mathbf{x}_j\right).$$

Finally, recall that in general $\sigma_{\max}(E) \leq 1$. Therefore, the eigenvalues of $\hat{V}_2 \hat{E}^\top \hat{E} \hat{V}_2^\top$ could also be 1, and thus, we are recovering a solution in a superset of the set of vectors in the linear weight.[5] □

A.3 Proof of Theorem 3

Theorem 3. *Given the optimal dual solution $\hat{\alpha}_i$, $i \in [n]$ to the problem in Eq. (11), the convolutional weight $\hat{U}_1$ can be implicitly recovered by the convolution output, that is, $\forall i \in [n]$,*

$$\Phi\left(\mathbf{x}_i\right)^\top \hat{U}_1 = \sum_{j=1}^{n} \hat{\alpha}_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \hat{V}_1. \tag{15}$$

Proof. From Eq. (21) we know that

$$\hat{V}_1 \hat{U}_1^\top = \sum_{j=1}^{n} \hat{\alpha}_j y_j \Phi\left(\mathbf{x}_j\right)^\top - \hat{V}_2 \hat{E}^\top \hat{U}_2^\top. \tag{27}$$

[5]Experimentally, the total number of vectors associated to eigenvalues close to 1 was very small. Thus, most likely we are recovering only $\hat{V}_1$ and not $\hat{V}_2$, and so we do not believe this is an issue in practical terms.

Then to construct the convolution output, we multiply $\Phi(\mathbf{x}_i)$ on the right for both sides of the equation:

$$\begin{aligned}\hat{V}_1\hat{U}_1^\top\Phi(\mathbf{x}_i) &= \sum_{j=1}^{n}\hat{\alpha}_j y_j \Phi\left(\mathbf{x}_j\right)^\top \Phi(\mathbf{x}_i) - \hat{V}_2\hat{E}^\top\hat{U}_2^\top\Phi(\mathbf{x}_i)\\ &= \sum_{j=1}^{n}\hat{\alpha}_j y_j K\left(\mathbf{x}_j,\mathbf{x}_i\right) - \hat{V}_2\hat{E}^\top\hat{U}_2^\top\Phi(\mathbf{x}_i)\end{aligned}$$

For the first term, we have generated the kernel generating matrix. Now we need to eliminate the $\hat{V}_1$ on the left side of the equation to get the exact form of the convolution output, and also avoid computing $\Phi(\mathbf{x}_i)$ on the right side of the equation. Therefore, for both sides of the equation, we multiply $\hat{V}_1^\top$ on the left, then we get

$$\hat{V}_1^\top\hat{V}_1\hat{U}_1^\top\Phi(\mathbf{x}_i) = \sum_{j=1}^{n}\hat{\alpha}_j y_j \hat{V}_1^\top K\left(\mathbf{x}_j,\mathbf{x}_i\right) - \hat{V}_1^\top\hat{V}_2\hat{E}^\top\hat{U}_2^\top\Phi(\mathbf{x}_i).$$

Knowing that $\hat{V}_1^\top\hat{V}_1 = \mathbf{I}$ and $\hat{V}_1^\top\hat{V}_2 = \mathbf{0}$, we have

$$\hat{U}_1^\top\Phi(\mathbf{x}_i) = \sum_{j=1}^{n}\hat{\alpha}_j y_j \hat{V}_1^\top K\left(\mathbf{x}_j,\mathbf{x}_i\right). \tag{28}$$

Taking transpose on both sides of the equation completes the proof. □

B Detailed Proofs for Multi-class Classification

B.1 Proof of Theorem 4

Theorem 4. *For all dual variables $\alpha_{k,i} \geq 0$ with $i \in [n]$ and $k \in [m]$, the dual problem of Eq. (16) is given by:*

$$\begin{aligned}\underset{\alpha}{\text{maximize}} \quad & -c\sum_{i=1}^{n}\sum_{k=1}^{m}\ell^*\left(-\frac{\alpha_{k,i}}{c}\right)\\ \text{subject to} \quad & \lambda_{\max}\Big(\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m}\alpha'_{k,i}\alpha'_{k,j}K'_k(\mathbf{x}_i,\mathbf{x}_j)\Big) \leq 1,\\ & \alpha_{k,i}\geq 0,\ \alpha_{y_i,i}=0, \quad \forall i\in[n],\ \forall k\in[m],\end{aligned} \tag{17}$$

in which $\alpha'_{k,i} = \sum_{s=1}^{m}\alpha_{s,i}\mathbb{1}[k=y_i] - \alpha_{k,i}$, $\ell^*(\cdot)$ is the Fenchel conjugate of the loss function $\ell(\cdot)$, and the kernel generating matrix is constructed in $K'_k(\mathbf{x}_i,\mathbf{x}_j) = \operatorname{diag}\big(\mathbf{0}_{p\times p},...,\underbrace{K(\mathbf{x}_i,\mathbf{x}_j)}_{k^{th}\ block\ diagonal},...,\mathbf{0}_{p\times p}\big)$, with $K(\mathbf{x}_i,\mathbf{x}_j) = \Phi(\mathbf{x}_i)^\top\Phi(\mathbf{x}_j)$.

Proof. For Lagrangian variables $\alpha_{k,i}\geq 0$, $i\in[n]$, $k\in[m]\setminus\{y_i\}$, we can write the Lagrangian function of Eq. (16) as

$$\mathcal{L}(A,\xi,\alpha) = \|A\|_* + c\sum_{i=1}^{n}\sum_{k\neq y_i}\ell(\xi_{k,i}) + \sum_{i=1}^{n}\sum_{k\neq y_i}\alpha_{k,i}\left(\xi_{k,i} - \operatorname{Tr}\left(\Phi'_{y_i}(\mathbf{x}_i)^\top A\right) + \operatorname{Tr}\left(\Phi'_k(\mathbf{x}_i)^\top A\right)\right).$$

For the simplicity of notation, we define $\alpha_{y_i,i}=0$, $\xi_{y_i,i}=0$, and denote $\mathcal{L} = \mathcal{L}(A,\xi,\alpha)$. Then the Lagrangian function becomes

$$\mathcal{L} = \|A\|_* + \sum_{i=1}^{n}\sum_{k=1}^{m}\alpha_{k,i}\left(\operatorname{Tr}\left(\Phi'_k(\mathbf{x}_i)^\top A\right) - \operatorname{Tr}\left(\Phi'_{y_i}(\mathbf{x}_i)^\top A\right)\right) + c\sum_{i=1}^{n}\sum_{k=1}^{m}\ell(\xi_{k,i}) + \sum_{i=1}^{n}\sum_{k=1}^{m}\alpha_{k,i}\xi_{k,i}. \tag{29}$$

By the definition of the dual function,

$$
\begin{aligned}
g(\alpha) &= \min_{A,\xi} \mathcal{L}(A, \xi, \alpha) \\
&= \underbrace{\min_{A} \left\{ \|A\|_* - \sum_{i=1}^{n}\sum_{k=1}^{m} \alpha_{k,i} \left(\operatorname{Tr}\left(\Phi'_{y_i}(\mathbf{x}_i)^\top A \right) - \operatorname{Tr}\left(\Phi'_k(\mathbf{x}_i)^\top A \right) \right) \right\}}_{g_1} \\
&\quad + \underbrace{\min_{\xi} \left\{ c \sum_{i=1}^{n}\sum_{k=1}^{m} \ell(\xi_{k,i}) + \sum_{i=1}^{n}\sum_{k=1}^{m} \alpha_{k,i}\xi_{k,i} \right\}}_{g_2}
\end{aligned} \tag{30}
$$

For g_1, we have

$$
\begin{aligned}
g_1 &= \min_A \|A\|_* - \sum_{k=1}^{m} \left(\sum_{i=1}^{n} \alpha_{k,i} \operatorname{Tr}\left(\Phi'_{y_i}(\mathbf{x}_i)^\top A \right) - \sum_{i=1}^{n} \alpha_{k,i} \operatorname{Tr}\left(\Phi'_k(\mathbf{x}_i)^\top A \right) \right) \\
&= \min_A \|A\|_* - \sum_{k=1}^{m} \left(\sum_{i=1}^{n}\sum_{s=1}^{m} \alpha_{s,i} \mathbb{1}[k = y_i] \operatorname{Tr}\left(\Phi'_k(\mathbf{x}_i)^\top A \right) - \sum_{i=1}^{n} \alpha_{k,i} \operatorname{Tr}\left(\Phi'_k(\mathbf{x}_i)^\top A \right) \right) \\
&= \min_A \|A\|_* - \sum_{k=1}^{m}\sum_{i=1}^{n} \left(\sum_{s=1}^{m} \alpha_{s,i} \mathbb{1}[k = y_i] - \alpha_{k,i} \right) \operatorname{Tr}\left(\Phi'_k(\mathbf{x}_i)^\top A \right) \\
&= -\max_A \sum_{k=1}^{m}\sum_{i=1}^{n} \left(\sum_{s=1}^{m} \alpha_{s,i} \mathbb{1}[k = y_i] - \alpha_{k,i} \right) \operatorname{Tr}\left(\Phi'_k(\mathbf{x}_i)^\top A \right) - \|A\|_* \\
&= -\max_A \left\langle \sum_{k=1}^{m}\sum_{i=1}^{n} \left(\sum\nolimits_{s=1}^{m} \alpha_{s,i} \mathbb{1}[k = y_i] - \alpha_{k,i} \right) \Phi'_k(\mathbf{x}_i), A \right\rangle - \|A\|_* \\
&= -\max_A \left\langle \sum_{k=1}^{m}\sum_{i=1}^{n} \alpha'_{k,i} \Phi'_k(\mathbf{x}_i), A \right\rangle - \|A\|_*
\end{aligned} \tag{31}
$$

in which Eq. (31) utilizes the linearity of trace and the definition of matrix inner product, and for a shorthand notation, $\alpha'_{k,i} = \sum_{s=1}^{m} \alpha_{s,i} \mathbb{1}[k = y_i] - \alpha_{k,i}$.

We can see that g_1 can be re-written as the conjugate function of the nuclear norm, i.e. for f_0^* in Eq. (10),

$$
g_1 = -f_0^* \left(\sum_{i=1}^{n}\sum_{k=1}^{m} \alpha'_{k,i} \Phi'_k(\mathbf{x}_i) \right). \tag{32}
$$

Furthermore, we must have $\left\| \sum_{i=1}^{n}\sum_{k=1}^{m} \alpha'_{k,i} \Phi'_k(\mathbf{x}_i) \right\|_2 \leq 1$ for g_1 to be bounded. Also,

$$
\begin{aligned}
g_2 &= \min_\xi c \sum_{i=1}^{n}\sum_{k=1}^{m} \ell(\xi_{k,i}) + \sum_{i=1}^{n}\sum_{k=1}^{m} \alpha_{k,i}\xi_{k,i} \\
&= c \sum_{i=1}^{n}\sum_{k=1}^{m} \left(\min_{\xi_{k,i}} \ell(\xi_{k,i}) + \frac{\alpha_{k,i}}{c} \xi_{k,i} \right) \\
&= -c \sum_{i=1}^{n}\sum_{k=1}^{m} \left(\max_{\xi_{k,i}} \langle -\frac{\alpha_{k,i}}{c}, \xi_{k,i} \rangle - \ell(\xi_{k,i}) \right) \\
&= -c \sum_{i=1}^{n}\sum_{k=1}^{m} \ell^* \left(-\frac{\alpha_{k,i}}{c} \right),
\end{aligned}
$$

in which $\ell^*(\cdot) = \sup_{\xi_{k,i}} \left\{ \langle \cdot, \xi_{k,i} \rangle - \ell(\xi_{k,i}) \right\}$ is the conjugate function of the loss function $\ell(\xi_{k,i})$. Therefore, the dual problem of Eq. 16 is as follows:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & -c \sum_{i=1}^{n} \sum_{k=1}^{m} \ell^* \left(-\frac{\alpha_{k,i}}{c} \right) \\ \text{subject to} \quad & \left\| \sum_{i=1}^{n} \sum_{k=1}^{m} \alpha'_{k,i} \Phi'_k \left(\mathbf{x}_i \right) \right\|_2 \leq 1 \,, \\ & \alpha_{k,i} \geq 0, \ \alpha_{y_i,i} = 0, \quad \forall i \in [n], \ \forall k \in [m] \,, \end{aligned} \tag{33}$$

The spectral norm constraint in Eq. (33) is equivalent to

$$\left\| \sum_{i=1}^{n} \sum_{k=1}^{m} \alpha'_{k,i} \Phi'_k \left(\mathbf{x}_i \right) \right\|_2^2 \leq 1, \tag{34}$$

which can be further transformed by

$$\begin{aligned} \left\| \sum_{i=1}^{n} \sum_{k=1}^{m} \alpha'_{k,i} \Phi'_k \left(\mathbf{x}_i \right) \right\|_2^2 &= \sigma_{\max} \left(\sum_{i=1}^{n} \sum_{k=1}^{m} \alpha'_{k,i} \Phi'_k \left(\mathbf{x}_i \right) \right)^2 \\ &= \lambda_{\max} \left(\left(\sum_{i=1}^{n} \sum_{k=1}^{m} \alpha'_{k,i} \Phi'_k \left(\mathbf{x}_i \right) \right)^\top \left(\sum_{j=1}^{n} \sum_{l=1}^{m} \alpha'_{l,j} \Phi'_l \left(\mathbf{x}_j \right) \right) \right) \\ &= \lambda_{\max} \left(\sum_{i,j} \sum_{k,l} \alpha'_{k,i} \alpha'_{l,j} \Phi'_k \left(\mathbf{x}_i \right)^\top \Phi'_l \left(\mathbf{x}_j \right) \right). \end{aligned} \tag{35}$$

Notice that $\forall k, l \in [m]$,

$$\Phi'_k \left(\mathbf{x}_i \right)^\top \Phi'_l \left(\mathbf{x}_j \right) = \begin{cases} \mathbf{0}_{mp \times mp} \,, & k \neq l \\ K'_k \left(\mathbf{x}_i, \mathbf{x}_j \right) & k = l \end{cases}, \tag{36}$$

$$K'_k \left(\mathbf{x}_i, \mathbf{x}_j \right) = \operatorname{diag} \left(\mathbf{0}_{p \times p}, ..., \underbrace{K \left(\mathbf{x}_i, \mathbf{x}_j \right)}_{k^{th} \ block \ diagonal}, ..., \mathbf{0}_{p \times p} \right),$$

in which $K \left(\mathbf{x}_i, \mathbf{x}_j \right) = \Phi \left(\mathbf{x}_i \right)^\top \Phi \left(\mathbf{x}_j \right)$. Therefore we have the following dual optimization problem for multiclass classification:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & -c \sum_{i=1}^{n} \sum_{k=1}^{m} \ell^* \left(-\frac{\alpha_{k,i}}{c} \right) \\ \text{subject to} \quad & \lambda_{\max} \Big(\sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{m} \alpha'_{k,i} \alpha'_{k,j} K'_k (\mathbf{x}_i, \mathbf{x}_j) \Big) \leq 1 \,, \\ & \alpha_{k,i} \geq 0, \ \alpha_{y_i,i} = 0, \quad \forall i \in [n], \ \forall k \in [m] \,. \end{aligned} \tag{17}$$

□

B.2 Proof of Theorem 5

Theorem 5. *We can recover the linear weight $\hat{V}_1 \in \mathbb{R}^{mp \times r}$ by*

$$\hat{V}_1 = \left[\tilde{V}(i) \right], \forall \ i \ \textit{such that} \ \lambda_i = 1, \tag{37}$$

in which $\tilde{V} \in \mathbb{R}^{mp \times mp}$ *comes from*

$$\tilde{V}\Lambda\tilde{V}^{-1} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} \alpha'_{k,i}\alpha'_{k,j}K'_k\left(\mathbf{x}_i, \mathbf{x}_j\right), \tag{38}$$

and $\Lambda = \operatorname{diag}(\lambda_1, ..., \lambda_{mp}) \in \mathbb{R}^{mp \times mp}$, $\hat{\alpha}'_{k,i} = \sum_{s=1}^{m} \hat{\alpha}_{s,i}\mathbb{1}\left[k = y_i\right] - \hat{\alpha}_{k,i}$.

Proof. The proof of this theorem also leverages the stationary condition and the subdifferential of nuclear norm as in Theorem 2. For the stationary condition of problem (16) with respect to parameter A,

$$0 \in \partial\left\|\hat{A}\right\|_* - \sum_{i=1}^{n}\sum_{k=1}^{m} \hat{\alpha}'_{k,i}\Phi'_k\left(\mathbf{x}_i\right), \tag{39}$$

Then for $\hat{A} \in \mathbb{R}^{d_2 \times mp}$, consider its compact SVD $\hat{A} = \hat{U}_1\hat{D}_1\hat{V}_1^\top$, in which $\hat{U}_1 \in \mathbb{R}^{d_2 \times r}$, $\hat{D}_1 \in \mathbb{R}^{r \times r}$, and $\hat{V}_1 \in \mathbb{R}^{mp \times r}$, and the full SVD $\hat{A} = \hat{U}\hat{D}\hat{V}^\top$, in which $\hat{U} = [\hat{U}_1 \mid \hat{U}_2] \in \mathbb{R}^{d_2 \times d_2}$, $\hat{V} = [\hat{V}_1 \mid \hat{V}_2] \in \mathbb{R}^{mp \times mp}$. By Lemma 2, we know that $\exists\ \hat{E} \in \mathbb{R}^{(d_2 - r) \times (mp - r)}$ such that

$$\hat{U}_1\hat{V}_1^\top + \hat{U}_2\hat{E}\hat{V}_2^\top = \sum_{i=1}^{n}\sum_{k=1}^{m} \hat{\alpha}'_{k,i}\Phi'_k\left(\mathbf{x}_i\right). \tag{40}$$

By computing the Gram matrix of both sides of Eq. (40),

$$\begin{aligned}\left(\hat{U}_1\hat{V}_1^\top + \hat{U}_2\hat{E}\hat{V}_2^\top\right)^\top\left(\hat{U}_1\hat{V}_1^\top + \hat{U}_2\hat{E}\hat{V}_2^\top\right) &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m}\sum_{l=1}^{m} \hat{\alpha}'_{k,i}\hat{\alpha}'_{l,j}\Phi'_k\left(\mathbf{x}_i\right)^\top\Phi'_l\left(\mathbf{x}_j\right) \\ &= \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} \hat{\alpha}'_{k,i}\hat{\alpha}'_{k,j}K'_k\left(\mathbf{x}_i, \mathbf{x}_j\right)\end{aligned}$$

where we generate the kernel generating matrix in the form of Eq. (36) on the right side. For the left side, by a similar simplification procedure as in the proof of Theorem 2, we have

$$\hat{V}_1\hat{V}_1^\top + \hat{V}_2\hat{E}^\top\hat{E}\hat{V}_2^\top = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} \hat{\alpha}'_{k,i}\hat{\alpha}'_{k,j}K'_k\left(\mathbf{x}_i, \mathbf{x}_j\right).$$

Consequently, following the proof steps of Theorem 2, we can say that $\hat{V}_1 = \left[\tilde{V}(i)\right]$ for all i such that $\Lambda_{ii} = 1$ is the linear weight we recover, where we compute $\tilde{V}$ and Λ by

$$\tilde{V}\Lambda\tilde{V}^{-1} = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{m} \hat{\alpha}'_{k,i}\hat{\alpha}'_{k,j}K'_k\left(\mathbf{x}_i, \mathbf{x}_j\right).$$

□

B.3 Proof of Theorem 6

Theorem 6. *We can implicitly recover the convolutional weight in the convolution output for multi-class classification. That is, for all* $i \in [n]$*, we have*

$$\Phi\left(\mathbf{x}_i\right)^\top\hat{U}_1 = \sum_{j=1}^{n}\sum_{k=1}^{m} \alpha'_{k,j}\left[\mathbf{0}_{p \times (k-1)p}, K\left(\mathbf{x}_i, \mathbf{x}_j\right), \mathbf{0}_{p \times (m-k)p}\right]\hat{V}_1, \tag{41}$$

where $\hat{\alpha}'_{k,i} = \sum_{s=1}^{m} \hat{\alpha}_{s,i}\mathbb{1}\left[k = y_i\right] - \hat{\alpha}_{k,i}$.

Proof. From Eq. (40) we know that

$$\hat{V}_1\hat{U}_1^\top = \sum_{j=1}^{n}\sum_{k=1}^{m} \hat{\alpha}'_{k,j}\Phi'_k\left(\mathbf{x}_j\right)^\top - \hat{V}_2\hat{E}^\top\hat{U}_2^\top. \tag{42}$$

To form the convolution output on the left and the kernel generating matrix on the right, we multiply $\Phi(\mathbf{x}_i)$ on the right for both sides of the equation:

$$\begin{aligned}
\hat{V}_1\hat{U}_1^\top\Phi(\mathbf{x}_i) &= \sum_{j=1}^{n}\sum_{k=1}^{m}\hat{\alpha}'_{k,j}\Phi'_k\left(\mathbf{x}_j\right)^\top\Phi(\mathbf{x}_i) - \hat{V}_2\hat{E}^\top\hat{U}_2^\top\Phi(\mathbf{x}_i) \\
&= \sum_{j=1}^{n}\sum_{k=1}^{m}\hat{\alpha}'_{k,j}\left[\mathbf{0}_{d_2\times(k-1)p}, \Phi\left(\mathbf{x}_j\right), \mathbf{0}_{d_2\times(m-k)p}\right]^\top\Phi(\mathbf{x}_i) - \hat{V}_2\hat{E}^\top\hat{U}_2^\top\Phi(\mathbf{x}_i) \\
&= \sum_{j=1}^{n}\sum_{k=1}^{m}\hat{\alpha}'_{k,j}\left[\mathbf{0}_{p\times(k-1)p}, \left(\Phi\left(\mathbf{x}_j\right)^\top\Phi(\mathbf{x}_i)\right), \mathbf{0}_{p\times(m-k)p}\right]^\top - \hat{V}_2\hat{E}^\top\hat{U}_2^\top\Phi(\mathbf{x}_i) \\
&= \sum_{j=1}^{n}\sum_{k=1}^{m}\hat{\alpha}'_{k,j}\left[\mathbf{0}_{p\times(k-1)p}, K\left(\mathbf{x}_j,\mathbf{x}_i\right)^\top, \mathbf{0}_{p\times(m-k)p}\right]^\top - \hat{V}_2\hat{E}^\top\hat{U}_2^\top\Phi(\mathbf{x}_i)\,. \qquad (43)
\end{aligned}$$

For both sides of the equation, multiply with $\hat{V}_1^\top$ on the left, we have

$$\begin{aligned}
\hat{V}_1^\top\hat{V}_1\hat{U}_1^\top\Phi(\mathbf{x}_i) = \sum_{j=1}^{n}\sum_{k=1}^{m}\hat{\alpha}'_{k,j}\hat{V}_1^\top\left[\mathbf{0}_{p\times(k-1)p}, K\left(\mathbf{x}_j,\mathbf{x}_i\right)^\top, \mathbf{0}_{p\times(m-k)p}\right]^\top \\
+ \hat{V}_1^\top\hat{V}_2\hat{E}^\top\hat{U}_2^\top\Phi(\mathbf{x}_i)\,.
\end{aligned}$$

Knowing that $\hat{V}_1^\top\hat{V}_1 = \mathbf{I}$ and $\hat{V}_1^\top\hat{V}_2 = \mathbf{0}$, we have

$$\hat{U}_1^\top\Phi(\mathbf{x}_i) = \sum_{j=1}^{n}\sum_{k=1}^{m}\hat{\alpha}'_{k,j}\hat{V}_1^\top\left[\mathbf{0}_{p\times(k-1)p}, K\left(\mathbf{x}_j,\mathbf{x}_i\right)^\top, \mathbf{0}_{p\times(m-k)p}\right]^\top. \qquad (44)$$

Taking transpose on both sides gives the form of the convolution output and completes the proof. □

C Fenchel conjugate of Common Losses

C.1 Hinge Loss

The hinge loss function is given by $\ell_H(x) = \max\{0, 1-x\}$, $\forall x \in \mathbb{R}$. Its Fenchel conjugate is

$$\ell_H^*(x^*) = \begin{cases} x^*\,, & x^* \in [-1, 0] \\ \infty, & otherwise \end{cases}. \qquad (45)$$

Derivation can be found in (Heinrich, 2013).

C.2 Squared Hinge Loss

The squared hinge loss function is given by $\ell_{SH}(x) = (\max\{0, 1-x\})^2$, $\forall x \in \mathbb{R}$. Its Fenchel conjugate is

$$\ell_{SH}^*(x^*) = \begin{cases} x^* + \frac{x^{*2}}{4}\,, & x^* \le 0 \\ \infty, & otherwise \end{cases}. \qquad (46)$$

Derivation can be found in (Heinrich, 2013).

C.3 Logistic Loss

The logistic loss function is given by $\ell_L(x) = \log(1+e^{-x})$, $\forall x \in \mathbb{R}$. Its Fenchel conjugate is

$$\ell_L^*(x^*) = \begin{cases} -x^*\log(-x^*) + (1+x^*)\log(1+x^*)\,, & x^* \in [-1, 0] \\ \infty, & otherwise \end{cases}. \qquad (47)$$

Derivation can be found in (Borwein & Lewis, 2005).

C.4 Exponential Loss

The exponential loss function is given by $\ell_E(x) = e^{-x}$, $\forall x \in \mathbb{R}$. Its Fenchel conjugate is

$$\ell_E^*(x^*) = \begin{cases} x^* - x^* \log(-x^*) \,, & x^* \leq 0 \\ \infty, & otherwise \end{cases} . \tag{48}$$

Derivation can be found in (Borwein & Lewis, 2005).

D Dual Optimization with Hinge Loss

We apply hinge loss for experimental evaluation. Given the dual optimization problem derived in Theorem 1:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & -c \sum_{i=1}^{n} \ell^* \left(-\frac{\alpha_i}{c} \right) \\ \text{subject to} \quad & \lambda_{\max}\Big(\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \Big) \leq 1 \,, \\ & \alpha_i \geq 0 \,, \quad \forall i \in [n] \,, \end{aligned} \tag{6}$$

and the Fenchel conjugate of the hinge loss function given in Appendix C.1, we know the objective function becomes

$$\begin{aligned} -c \sum_{i=1}^{n} \ell^* \left(-\frac{\alpha_i}{c} \right) &= -c \sum_{i=1}^{n} \left(-\frac{\alpha_i}{c} \right) \\ &= \sum_{i=1}^{n} \alpha_i, \end{aligned}$$

for $-\frac{\alpha_i}{c} \in [-1, 0]$, i.e. $\forall i \in [n]$, $\alpha_i \in [0, c]$. Therefore the dual optimization problem with hinge loss is:

$$\begin{aligned} \underset{\alpha}{\text{maximize}} \quad & \sum_{i=1}^{n} \alpha_i \\ \text{subject to} \quad & \lambda_{\max}\Big(\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \Big) \leq 1 \,, \\ & 0 \leq \alpha_i \leq c \,, \quad \forall i \in [n] \,. \end{aligned}$$

E Algorithms

E.1 Coordinate Descent Optimization

Algorithm 2 Coordinate Descent Optimization of the Dual Problem (6)

Input: Data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$; Kernel function $\mathcal{K}$; Hyperparameter c.

1: Compute $\lambda_{\max}(K(\mathbf{x}_i, \mathbf{x}_i))$ for $i = 1$ to n.
2: Let $Indices$ be the sorted coordinates according to the ascending order of $\lambda_{\max}(K(\mathbf{x}_i, \mathbf{x}_i))$
3: **for** $iteration = 1, 2, \dots$ **do**
4: Let $S = \varnothing$ be the set of the optimized coordinates.
5: Let $R = \mathbf{0}_{p\times p}$.
6: **for** i in $Indices$ **do**
7: Compute $T = \sum_{j\in S} \hat{\alpha}_j y_i y_j \big(K(\mathbf{x}_i, \mathbf{x}_j) + K(\mathbf{x}_j, \mathbf{x}_i)\big)$
8: **if** $\lambda_{\max}\big(c^2 K(\mathbf{x}_i, \mathbf{x}_i) + cT + R\big) \leq 1$ **then**
9: Let $\hat{\alpha}_i = c$.
10: **else**
11: Find the largest $\hat{\alpha}_i \in [0, c]$ such that $\lambda_{\max}\big(\hat{\alpha}_i^2 K(\mathbf{x}_i, \mathbf{x}_i) + \hat{\alpha}_i T + R\big) \leq 1$ using binary search.
12: **end**
13: Add i to S.
14: $R = R + \hat{\alpha}_i^2 K(\mathbf{x}_i, \mathbf{x}_i) + \hat{\alpha}_i T$.
15: **end for**
16: **end for**

Output: Dual solution $\{\hat{\alpha}_i\}_{i=1}^n$.

Experimentally, we run Algorithm 2 above for only one iteration. To formally argue for the convergence of the above algorithm, we have added the several iterations in Line 3. By changing the maximization problem to the minimization of the negative of the objective function, and by using the extended-value extension (Boyd & Vandenberghe, 2004), the dual problem in Eq. (6) becomes $\text{minimize}_{\alpha\in\mathbb{R}^n} f(\alpha)$ where:

$$f(\alpha) = \begin{cases} c\sum_{i=1}^n \ell^*(-\alpha_i/c), & \text{if } \lambda_{\max}\left(\sum_{i=1}^n \sum_{j=1}^n \alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)\right) \leq 1 \text{ and } \alpha_i \geq 0, \forall i \in [n] \\ \infty, \text{otherwise} \end{cases}$$

We can now invoke Theorem 4.1 in (Tseng, 2001) to show the convergence of our algorithm. (See also Example 6.4 therein.)

E.2 Training $\mathcal{D}$-layer DCCNNs

Algorithm 3 Training $\mathcal{D}$-layer DCCNNs

Input: Number of convolutional layers $\mathcal{D}$; Data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$; Kernel function $\mathcal{K}$.

1: **for** $\ell = 1$ to $\mathcal{D}$ **do**
2: Construct dual problem in Eq. (6) with $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and $\mathcal{K}$
3: Solve for the dual solution in the ℓ^{th} layer $\{\hat{\alpha}_i^\ell\}_{i=1}^n$.
4: Recover the convolution output $\{\mathbf{x}_i^\ell\}_{i=1}^n$ and the linear weight $\hat{L}_\ell$ with Algorithm 1.
5: Let $\mathbf{x}_i = \mathbf{x}_i^\ell$.
6: **end for**

Output: Dual solution of every layer $\{\{\hat{\alpha}_i^\ell\}_{i=1}^n\}_{\ell=1}^{\mathcal{D}}$; Convolution output of every layer $\{\{\mathbf{x}_i^\ell\}_{i=1}^n\}_{\ell=1}^{\mathcal{D}}$; Linear weight of the every layer $\{\hat{L}_\ell\}_{\ell=1}^{\mathcal{D}}$.

E.3 Making Predictions with $\mathcal{D}$-layer DCCNNs

Algorithm 4 Making Predictions with $\mathcal{D}$-layer DCCNNs

Input: Test data $\mathbf{x}_{new}$; Kernel function $\mathcal{K}$; Dual solution of every layer $\{\{\hat{\alpha}_i^\ell\}_{i=1}^n\}_{\ell=1}^{\mathcal{D}}$; Convolution output of every layer $\{\{\mathbf{x}_i^\ell\}_{i=1}^n\}_{\ell=1}^{\mathcal{D}}$; Linear weight of the every layer $\{\hat{L}_\ell\}_{\ell=1}^{\mathcal{D}}$.

1: **for** $\ell = 1$ to $\mathcal{D}-1$ **do**

2: Compute the convolution output for $\mathbf{x}_{new}$ by

$$\mathbf{x}'_{new} = \mathrm{vec}\big(\sum\nolimits_{j=1}^n \hat{\alpha}_j^\ell y_j K(\mathbf{x}_{new}, \mathbf{x}_j)\hat{L}_\ell\big).$$

3: Let $\mathbf{x}_{new} = \mathbf{x}'_{new}$.

4: **end for**

5: In layer $\mathcal{D}$, compute the prediction by {Apply the final linear weight for classification}

$$f(\mathbf{x}_{new}) = \mathrm{sign}\big(\mathrm{Tr}(\sum\nolimits_{j=1}^n \hat{\alpha}_j^{\mathcal{D}} y_j K(\mathbf{x}_{new}, \mathbf{x}_j)\hat{L}_{\mathcal{D}}\hat{L}_{\mathcal{D}}^\top)\big).$$

Output: Prediction $f(\mathbf{x}_{new})$.

F Average Pooling Matrix

Convolutional output $\Phi(\mathbf{x}_i)^\top \hat{U}_1 \in \mathbb{R}^{p\times r}$ is formed by the output of r filters, i.e. the convolution output has r channels. Pooling is done for each channel, also known as a *feature map*, of the convolutional output. Denote one column of $\Phi(\mathbf{x}_i)^\top \hat{U}_1$, i.e. one feature map as $\mathbf{x}'_{i,k} \in \mathbb{R}^p$ for $k \in [r]$. In an average pooling operation, a subset of entries, or a patch of the feature map is multiplied element-wise with an average pooling filter. The total number of pooling operations depends on the pooling filter width and stride. We use q to represent the total number of pooling operation on one feature map, then we can generate an average pooling matrix $G \in \mathbb{R}^{q\times p}$. Assume the pooling filter has b entries, then

$$G_{s,t} = \begin{cases} \frac{1}{b} & \text{pooling filter multiplied with the } t^{th} \text{ feature map entry in the } s^{th} \text{ pooling operation} \\ 0 & \text{otherwise} \end{cases}.$$

To perform average pooling on the convolution output, we can directly multiply the average pooling matrix on the left of the convolution output before vectorizing it and feeding it to the next layer, i.e.

$$G\Phi(\mathbf{x}_i)^\top \hat{U}_1 = \sum_{j=1}^n \hat{\alpha}_j y_j GK(\mathbf{x}_i, \mathbf{x}_j)\hat{V}_1.$$

Then for the next layerwise training, $\{\mathrm{vec}\big(\sum_{j=1}^n \hat{\alpha}_j y_j GK(\mathbf{x}_i, \mathbf{x}_j)\hat{V}_1\big)\}_{i=1}^n$ is regarded as the input.

G Experiment

G.1 Experiment Settings

Data. We use the MNIST (Lecun et al., 1998) and the ImageNet datasets (Deng et al., 2009) under the terms of the Creative Commons Attribution-Share Alike 3.0 license. In binary classification experiments, for both the MNIST and ImageNet datasets, we randomly pick two classes, images of digit 2 versus 3 from MNIST, images containing tench versus bathroom tissue from ImageNet, and use 2000 images for training, 100 images for validation, and 600 images for testing. We center crop the ImageNet images to 224×224, and transfer the pixel values to $[0, 1]$ independently for each of the three RGB channels. No other preprocessing is applied.

Architecture. (1) MNIST experiments: we implement 1-layer and 2-layer DCCNN for binary classification and 1-layer DCCNN for multiclass classification, and compare with its counterparts under the CCNN framework and the CNN trained with back-propagation SGD. We test all methods with filter width 5, stride

1, and padding size 2, and set the number of filters for CCNN and SGD-trained CNN to 16 for each layer. (2) ImageNet experiments: we follow the architectures similar to AlexNet (Krizhevsky et al., 2012) and VGG11 (Simonyan & Zisserman, 2015) with some few modifications. For DCCNN and CCNN, we only count the number of convolutional layers, that is, 5 layers in AlexNet and 8 layers in VGG11, and concatenate one final linear layer for the purpose of classification. In order to reduce the number of patches for computational efficiency, we change the stride of the first layer from 4 to 8 for both AlexNet and VGG11. To ensure VGG11 keeps all information of the input image, we correspondingly change the filter width in the first layer from 3 to 9, and remove the pooling after layer 4 and 6. For the simplicity of matrix multiplication, we choose average pooling as the pooling operation. Other settings of filter width, stride and padding size are kept the same. All methods are trained with the same architecture.

Kernel function and activation function. For both DCCNN and CCNN, we choose the Gaussian RBF kernel as in (Zhang et al., 2017) that has the form $\mathcal{K}(\mathbf{z}_i, \mathbf{z}_j) = \exp\{-\gamma\|\mathbf{z}_i - \mathbf{z}_j\|_2^2\}$, $\|\mathbf{z}_i\|_2 = \|\mathbf{z}_j\|_2 = 1$. For a fair comparison, we choose the sinusoid function $\rho(\cdot) = \sin(\cdot)$ (Isa et al., 2010; Sopena et al., 1999) as the activation function, as it is proved in (Zhang et al., 2017) to be contained in the RKHS of Gaussian RBF kernel.

CCNN kernel matrix factorization. For the approximate kernel factorization $K \approx QQ^\top$ in the algorithm of (Zhang et al., 2017), we use a 25-column matrix Q from either: (1) $UD^{\frac{1}{2}}$ for $K = UDU^\top$; (2) $UD^{\frac{1}{2}}V$ for $K = UDU^\top$, in which V is a random orthonormal matrix; (3) $K^{\frac{1}{2}}$; (4) the Cholesky decomposition of K. For the consideration of computational complexity, we only construct the block diagonal of the kernel matrix K, and approximate the sample feature by the factorization of its corresponding block diagonal.

Loss function. For CCNN and DCCNN, we use the hinge loss for optimization, and for the nuclear norm constraint in CCNN, we implement it as regularization in the optimization step. For back-propagation SGD, we use the binary cross-entropy loss for optimization.

Hyperparameters. For CNNs trained with SGD, we use 50 epochs with batch size 50 and learning rate 0.1. For a fair comparison, we do not employ dropout, weight decay, or other tricks for training. We run 10 trials for each experiment and report the average and standard deviation of accuracy. For CCNN, since the block diagonal kernel matrix is much smaller than the full kernel matrix, we set the CCNN hyperparameters $m = 25$ and $r = 16$ for each layer. For DCCNN, even though we theoretically take the eigenvectors with eigenvalue 1, there could be numerical issues during the optimization process, making the eigenvalues not strictly 1. Therefore we set a threshold for the eigenvalues we take. For 1-layer DCCNN, 2-layer DCCNN, AlexNet DCCNN, and VGG11 DCCNN, the threshold is set to 0.8, 0.9, 0.975, 0.85, respectively.

Platform and implementation. We implement DCCNN with Matlab 2018a. The code is tested on a server with 4 CPU cores and 16GB memory size.

G.2 Results

We demonstrate the results of prediction accuracy in Table 2. The rows are organized by the datasets and architectures while each column shows the performance of one method across different tasks.

For SGD experiments, in particular, we run 10 trials for each setting with different random seeds and report the average and standard deviation of accuracy. The average accuracy is shown in Table 2. For binary classification, we observe a 1.8% standard deviation for one-layer CNN trained with SGD on the MNIST dataset, and 0.6% for the two-layer CNN trained end-to-end with SGD. For the network structure similar to AlexNet on the ImageNet dataset, we observe a standard deviation of 2.5%, and for the architecture similar to VGG11, the standard deviation is 1.7%. For multiclass classification, the standard deviation of SGD is 0.5%.

G.3 Discussion on Performance Level

In the binary classification task of the MNIST data, we can see that DCCNN outperforms CNN optimized by SGD and all different kernel matrix factorizations for CCNN on one-layer and two-layer networks with only one exception for the two-layer CCNN with Cholesky decomposition. This verifies the effectiveness of

Dataset	Architecture	CCNN				SGD (end-to-end / layerwise)	DCCNN
		$UD^{\frac{1}{2}}$	$UD^{\frac{1}{2}}V$	$K^{\frac{1}{2}}$	Cholesky		
MNIST	1-Conv-Layer	90.3%	90.3%	88.8%	93.7%	90.2% / ——	94.8%
	2-Conv-Layer	93.7%	93.5%	93.5%	96.7%	95.2% / 92.4%	96.0%
ImageNet	AlexNet	62.3%	62.3%	53.2%	62.0%	87.1% / 86.3%	83.3%
	VGG11	55.3%	52.5%	59.8%	57.0%	89.2% / 86.7%	85.0%

(a) Test Accuracies on MNIST and ImageNet Binary Classification.

Dataset	Architecture	CCNN				SGD	DCCNN
		$UD^{\frac{1}{2}}$	$UD^{\frac{1}{2}}V$	$K^{\frac{1}{2}}$	Cholesky		
MNIST	1-Conv-Layer	75.4%	75.4%	82.3%	85.9%	87.0%	85.3%

(b) Test Accuracies on MNIST Multiclass Classification.

Table 2: Experiment results for binary and multiclass classification. Methods compared include CCNN, CNN trained with SGD, and our proposed DCCNN. For CCNN we use a 25-column matrix Q from 4 different ways of kernel matrix factorization: (1) $UD^{\frac{1}{2}}$ for $K = UDU^\top$; (2) $UD^{\frac{1}{2}}V$ for $K = UDU^\top$, and random orthonormal matrix V; (3) $K^{\frac{1}{2}}$; (4) the Cholesky decomposition of K. The X-Conv-Layer in the architecture column refers to the number of convolutional layers, as only one linear layer is concatenated at the end for classification. For architectures with more than one convolutional layer, we list the result of SGD trained both end-to-end and layerwise. The result of SGD only reflects its performance in our specific experiment setting, e.g. few convolutional filters, and is listed for DCCNN sanity check purposes.

DCCNN. Furthermore, we observe that different factorization approaches introduce turbulence to the CCNN method, while our algorithm does not suffer from such ambiguity.

On the more complicated ImageNet dataset, DCCNN also performs comparably well with the end-to-end SGD optimized CNNs under both AlexNet and VGG11 architectures, and significantly outperforms the CCNN method. With the heuristic of kernel matrix factorization and cutting the weight matrix, CCNN does not necessarily generalize to complex datasets like ImageNet. As the task gets difficult, the performance level of CCNN with different factorization methods gets more arbitrary. This further highlights the merits of our proposed DCCNN.

Moreover, we observe that there is a decrease of accuracy for CCNN with the increase on the number of layers. That may be caused by the accumulated impact of the weight-cutting heuristic, i.e., the number of filters heuristically enforced by a hyperparameter in each layer may not accurately reflect the information learned. Setting it too large would include unnecessary noise while setting it too small may discard relevant information. This further highlights the effectiveness of DCCNN on encouraging a small number of filters without introducing heuristics or ambiguity.

G.4 Discussion on Computational Complexity

We first analyze theoretically that

- spatially, CCNN has to construct the whole kernel matrix of size $np \times np$ with all of the n samples and p patches, leading to spatial complexity of $\mathcal{O}(n^2p^2)$, while DCCNN only need the kernel generating matrix of one sample at a time, which is of size $p \times p$, leading to the cost of $\mathcal{O}(p^2)$ space;
- temporally, the factorization of the kernel matrix in CCNN takes $\mathcal{O}(n^3p^3)$, while the runtime of DCCNN is dominated by the eigendecomposition in Algorithm 2, leading to time complexity of $\mathcal{O}(n^2p^3)$.

Experimentally, we now demonstrate how the running time of each method scales with the number of samples n and the number of patches p for the following reasons: (1) It takes an extremely long time to run the full kernel matrix (size $np \times np$) factorization for CCNN without approximation tricks, thus making such comparisons on actual running time trivial. So for CCNN baseline we only construct the kernel matrix of a batch of samples as applied in the CCNN paper (Zhang et al., 2017) to accelerate CCNN. (2) Methods are implemented differently, e.g. MATLAB versus PyTorch, hand-crafted matrix multiplications versus built-in conv & pooling operations, etc.

For 2000 samples 784 patches and 200 samples and 100 patches on 1-Layer MNIST experiment, the running time ratio is shown in Table 3:

1-Layer MNIST running time	DCCNN	CCNN	SGD
$(n, p) = (2000, 784)$ / $(n, p) = (200, 100)$	580x	1300x	55x

Table 3: Running time ratio between data and models of different scales.

As we see from the result, DCCNN is more computationally efficient than CCNN. Though SGD runs even faster, we emphasize that SGD is a stochastic approximation algorithm in nature.

Furthermore, the improvement on space complexity originates naturally from the dual formulation of the problem without the kernel matrix construction, which once again shows the significance of DCCNN. On the other hand, time complexity relies more on the solving algorithm, which is only an initial design in DCCNN as we focus more on the formulation of the dual problem and weight recovery from the optimized dual solution, and may be improved with better engineering.

H Further Discussion on Related Works

On learning convolutional neural networks with kernel methods, following the work of Neural Tangent Kernel (NTK) (Jacot et al., 2018), Arora et al. (2019) proposes the Convolutional Neural Tangent Kernel (CNTK) that studies the exact computation of CNNs with infinitely many convolutional filters, i.e. infinitely wide CNNs. As our approach uses kernel information and implies an infinite size of the convolutional weight matrix, it is fundamentally different from CNTK in both formulation and derivation: (1) In CCNN formulation described in Eq. (1), the convolutional weight is $W \in \mathbb{R}^{d_2 \times r}$. r is the number of filters, which is not only finite but encouraged to be low by the nuclear norm constraint, while CNTK studies the case of infinite many filters. In CCNN formulation, d_2, the size of each filter, is the part that may go to infinity, while CNTK assumes each filter has a fixed size. Therefore, the CCNN and CNTK frameworks are fundamentally different. (2) CNTK is to take the infinite limit of width so that the inputs of activation function tend to i.i.d. centered Gaussian processes with fixed covariance (Jacot et al., 2018; Arora et al., 2019), under which condition the output of the neural network would converge to the output of CNTK, asymptotically (Theorem 1 in Jacot et al. (2018)) or non-asymptotically (Theorem 3.2 in Arora et al. (2019)). On the other hand, the infinity size of the weight matrix in CCNN or DCCNN comes from the infinite dimension of the kernel basis function.

On deriving the convex equivalence of CNNs, Ergen & Pilanci (2020) derives a convex analytic framework utilizing semi-infinite duality, and regards the CNN architecture as an implicit convex regularizer following previous work (Pilanci & Ergen, 2020). As our approach utilizes the power of duality, it is not for the purpose of convexifying CNNs, but to eliminate the need and the ambiguity of factorizing the very large kernel matrix in CCNN (Zhang et al., 2017). As a result, the optimization problem is completely different. The most significant difference, the nuclear norm regularization in DCCNN poses challenges as it is non-differentiable, leading to no closed-form solution for recovering the primal solution from the dual. For this particular challenge, we proposed a highly-novel weight recovery algorithm to recover the linear weight and the output of the convolutional layer directly, instead of the convolutional weight.

On optimization with low-rank constraint, (Shalev-Shwartz et al., 2011; Li & Fu, 2015; Cabral et al., 2013; He et al., 2015) and many others have built up the connection between low-rankness and minimizing the nuclear

norm, in the context of large-scale optimization, learning discriminative subspace, matrix factorization, image restoration, etc. This further validates the soundness and significance of the DCCNN problem formulation.