A. Appendix

A.1. Proof for Lemma 1

At the optimal decision boundary the probabilities of any point $\boldsymbol{x} \in \mathbb{R}^d$ belonging to class $y = -1$ and $y = 1$ modeled by $\mathcal{D}$ are the same. Here, $\boldsymbol{\mu} = \boldsymbol{\mu_1} = -\boldsymbol{\mu_{-1}}$ and $I = \Sigma_{-1} = \Sigma_1$.

$$
\begin{aligned}
&\implies \frac{\exp[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_{-1}})^\top \Sigma_{-1}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_{-1}})]}{\sqrt{(2\pi)^d|\Sigma_{-1}|}} = \\
&\frac{\exp[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu_1})^\top \Sigma_{1}^{-1}(\boldsymbol{x}-\boldsymbol{\mu_1})]}{\sqrt{(2\pi)^d|\Sigma_{1}|}} \\
&\implies -\frac{1}{2}\log|I| - \frac{1}{2}(\boldsymbol{x}^\top\boldsymbol{x} - 2\boldsymbol{x}^\top\boldsymbol{\mu_{-1}} + \boldsymbol{\mu_{-1}}^\top\boldsymbol{\mu_{-1}}) = \\
&-\frac{1}{2}\log|I| - \frac{1}{2}(\boldsymbol{x}^\top\boldsymbol{x} - 2\boldsymbol{x}^\top\boldsymbol{\mu_1} + \boldsymbol{\mu_1}^\top\boldsymbol{\mu_1}) \\
&\implies \boldsymbol{x}^\top(\boldsymbol{\mu_{-1}} - \boldsymbol{\mu_1}) - \frac{1}{2}(\boldsymbol{\mu_{-1}}^\top\boldsymbol{\mu_{-1}} - \boldsymbol{\mu_1}^\top\boldsymbol{\mu_1}) = 0 \\
&\implies -\boldsymbol{x}^\top\boldsymbol{\mu} = 0 \\
&\implies P(\boldsymbol{x}) \equiv \boldsymbol{x}^\top\boldsymbol{\mu} = 0.
\end{aligned}
$$

Now, the accuracy of the clean model P is to be computed. Note that if $P(\boldsymbol{x}) < 0$ the Bayes optimal classification is class -1, else the classification is class 1. Let $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, I)$, and $Z \sim \mathcal{N}(0,1)$, and $sgn(.)$ be the signum function.

$$
\begin{aligned}
\tau_{\mathcal{D}}(P) &= \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[\mathbb{1}(y = sgn(P(\boldsymbol{x})))\right] = \mathbb{P}[y\boldsymbol{x}^\top\boldsymbol{\mu} > 0] \\
&= \mathbb{P}[y(y\boldsymbol{\mu} + \boldsymbol{z})^\top\boldsymbol{\mu} > 0] \\
&= \mathbb{P}[(\boldsymbol{\mu} + \boldsymbol{z})^\top\boldsymbol{\mu} > 0] \\
&= \mathbb{P}[\|\boldsymbol{\mu}\|_2^2 + \|\boldsymbol{\mu}\|_2 Z > 0] = \phi(\|\boldsymbol{\mu}\|_2).
\end{aligned}
$$

□

A.2. Proof for Lemma 2

Let $\mathcal{D}_1 = \mathcal{N}(\boldsymbol{\mu}, I)$. For every data point $(\boldsymbol{x}, y) \sim \mathcal{D}_1$, let the perturbed data $(A_1\boldsymbol{x}, y)$ be modelled by a distribution $\tilde{D}_1$. We prove that $\tilde{D}_1 = \mathcal{N}(A_1\boldsymbol{\mu}, A_1^\top A_1)$.

$$
\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_1}\ A_1\boldsymbol{x} = A_1\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_1}\ \boldsymbol{x} = A_1\boldsymbol{\mu}.
$$

$$
\begin{aligned}
&\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_1}\ (A_1\boldsymbol{x} - A_1\boldsymbol{\mu})(A_1\boldsymbol{x} - A_1\boldsymbol{\mu})^\top \\
&= \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_1}\ A_1(\boldsymbol{x}-\boldsymbol{\mu})[A_1(\boldsymbol{x}-\boldsymbol{\mu})]^\top \\
&= \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_1}\ A_1(\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^\top A_1^\top \\
&= A_1\mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}_1}\ (\boldsymbol{x}-\boldsymbol{\mu})(\boldsymbol{x}-\boldsymbol{\mu})^\top A_1^\top \\
&= A_1 I A_1^\top = A_1 A_1^\top.
\end{aligned}
$$

Tri-diagonal Toeplitz matrices $A_y = T(d; a_y, 1, a_y)$ are symmetric. Hence, $\tilde{\mathcal{D}} = \mathcal{N}(yA_y\boldsymbol{\mu}, A_y^2)$. □

A.3. Remarks on Lemma 3

A tri-diagonal Toeplitz matrix $T(d; a_1, a_2, a_3)$ is represented as

$$
\begin{bmatrix}
a_2 & a_3 & 0 & 0 & \dots & 0 \\
a_1 & a_2 & a_3 & 0 & \dots & 0 \\
0 & a_1 & a_2 & a_3 & \dots & 0 \\
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
0 & \dots & \dots & 0 & a_1 & a_2
\end{bmatrix} \in \mathbb{R}^{d\times d}
$$

The class of matrices $A_y = T(d; a_y, 1, a_y)$ are symmetric and can be diagonalized as $QDQ^\top$. $Q = \left(\left(\frac{2}{d+1}\right)^{1/2}\sin\left(\frac{ij\pi}{d+1}\right)\right)_{i,j}$ is symmetric and it is the common eigenvector matrix to all A_y matrices. As shown in Lemma 3, Q and D can be represented using trigonometric functions. Also, we have $A(n) := A_1^n \pm A_{-1}^n = Q(D_1^n \pm D_{-1}^n)Q$ where $A_1 = QD_1Q$ and $A_{-1} = QD_{-1}Q$. Further, $\mathrm{Tr}(A(n)) = \mathrm{Tr}(Q(D_1^n \pm D_{-1}^n)Q) = \mathrm{Tr}((D_1^n \pm D_{-1}^n)Q^2) = \mathrm{Tr}(D_1^n \pm D_{-1}^n)$.

A.4. Proof for Lemma 4

At the optimal decision boundary the probabilities of any point $\boldsymbol{x} \in \mathbb{R}^d$ belonging to class $y = -1$ and $y = 1$ modeled by $\tilde{\mathcal{D}}$ are the same. Here, $\boldsymbol{\mu} = \boldsymbol{\mu_1} = -\boldsymbol{\mu_{-1}}$ and A_y's are symmetric.

$$
\begin{aligned}
&\frac{\exp[-\frac{1}{2}(\boldsymbol{x}-A_{-1}\boldsymbol{\mu_{-1}})^\top(A_{-1}IA_{-1}^\top)^{-1}(\boldsymbol{x}-A_{-1}\boldsymbol{\mu_{-1}})]}{\sqrt{(2\pi)^d|A_{-1}IA_{-1}^\top|}} \\
&= \frac{\exp[-\frac{1}{2}(\boldsymbol{x}-A_{1}\boldsymbol{\mu_{1}})^\top(A_{1}IA_{1}^\top)^{-1}(\boldsymbol{x}-A_{1}\boldsymbol{\mu_{1}})]}{\sqrt{(2\pi)^d|A_{1}IA_{1}^\top|}} \\
&\implies -\frac{1}{2}\ln\frac{|A_{-1}^2|}{|A_1^2|} - \frac{1}{2}[\boldsymbol{x}^\top(A_{-1}^{-2} - A_1^{-2})\boldsymbol{x} \\
&- 2(\boldsymbol{\mu_{-1}}^\top A_{-1}^{-1} - \boldsymbol{\mu_1}^\top A_1^{-1})\boldsymbol{x} \\
&+ (\boldsymbol{\mu_{-1}}^\top\boldsymbol{\mu_{-1}} - \boldsymbol{\mu_1}^\top\boldsymbol{\mu_1})] = 0 \\
&\implies \tilde{P}(\boldsymbol{x}) \equiv \boldsymbol{x}^\top(A_{-1}^{-2} - A_1^{-2})\boldsymbol{x} \\
&- 2(\boldsymbol{\mu_{-1}}^\top A_{-1}^{-1} - \boldsymbol{\mu_1}^\top A_1^{-1})\boldsymbol{x} + (\|\boldsymbol{\mu_{-1}}\|_2^2 - \|\boldsymbol{\mu_1}\|_2^2) \\
&+ \sum_{i=1}^{d}\ln\left(\frac{1+2a_{-1}\cos(\frac{i\pi}{d+1})}{1+2a_1\cos(\frac{i\pi}{d+1})}\right)^2 = 0 \\
&\implies \tilde{P}(\boldsymbol{x}) \equiv \boldsymbol{x}^\top(A_{-1}^{-2} - A_1^{-2})\boldsymbol{x} \\
&+ 2[(A_{-1}^{-1} + A_1^{-1})\boldsymbol{\mu}]^\top\boldsymbol{x} \\
&+ \sum_{i=1}^{d}\ln\left(\frac{1+2a_{-1}\cos(\frac{i\pi}{d+1})}{1+2a_1\cos(\frac{i\pi}{d+1})}\right)^2 = 0 \\
&\implies \tilde{P}(\boldsymbol{x}) \equiv \boldsymbol{x}^\top A\boldsymbol{x} + \boldsymbol{b}^\top\boldsymbol{x} + c = 0.
\end{aligned}
$$

□

Note that here if $\tilde{P}(\boldsymbol{x}) < 0$, the Bayes optimal classification is class -1, else the classification is class 1. Here, for shorthand notations we denote $A = (A_{-1}^{-2} - A_1^{-2})$, $\boldsymbol{b} = 2(A_{-1}^{-1} + A_1^{-1})\boldsymbol{\mu}$, $c = \sum_{i=1}^{d} \ln\left(\frac{1 + 2a_{-1}\cos(\frac{i\pi}{d+1})}{1 + 2a_1\cos(\frac{i\pi}{d+1})}\right)^2$.

A.5. Proof for Lemma 5

Let $Z = \boldsymbol{z}^\top A\boldsymbol{z} + \boldsymbol{z}^\top \boldsymbol{b} + c$ and $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, I) \subset \mathbb{R}^d$ where $A = Q\Lambda Q^\top$. Also,

$$Z = \boldsymbol{z}^\top A\boldsymbol{z} + \boldsymbol{z}^\top \boldsymbol{b} + c$$
$$= \left(\boldsymbol{z} + \frac{1}{2}A^{-1}\boldsymbol{b}\right)^\top A\left(\boldsymbol{z} + \frac{1}{2}A^{-1}\boldsymbol{b}\right) + c - \frac{1}{4}\boldsymbol{b}^\top A^{-1}\boldsymbol{b}.$$

For any $t \geq 0$ and $\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{0}, I)$, we write the moment generating function for a quadratic random variable $Y = \boldsymbol{x}^\top A\boldsymbol{x}$ as [2]

$$\mathbb{E}[\exp(tY)] = \frac{1}{(2\pi)^{d/2}}\int_{\mathbb{R}^d} \exp\{t\boldsymbol{x}^\top A\boldsymbol{x}\}$$
$$\exp\Big\{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top(\boldsymbol{x}-\boldsymbol{\mu})\Big\}d\boldsymbol{x}$$
$$= \frac{\exp\{-\boldsymbol{\mu}^\top\boldsymbol{\mu}/2\}}{(2\pi)^{d/2}}\int_{\mathbb{R}^d}\exp\Big\{-\frac{1}{2}\boldsymbol{x}^\top(I-2tA)\boldsymbol{x} + \boldsymbol{\mu}^\top\boldsymbol{x}\Big\}d\boldsymbol{x}$$
$$= \frac{\exp\{-\boldsymbol{\mu}^\top\boldsymbol{\mu}/2\}}{(2\pi)^{d/2}}\frac{(2\pi)^{d/2}\exp\Big\{\frac{1}{2}\boldsymbol{\mu}^\top(I-2tA)^{-1}\boldsymbol{\mu}\Big\}}{|I-2tA|^{1/2}}$$
$$= \frac{\exp\Big\{-\frac{1}{2}\boldsymbol{\mu}^\top[I-(I-2tA)^{-1}]\boldsymbol{\mu}\Big\}}{|I-2tA|^{1/2}}.$$
$$\implies \mathbb{E}[\exp(tZ)] =$$
$$\frac{\exp\{-\frac{\boldsymbol{b}^\top}{8}A^{-1}[I-(I-2tA)^{-1}]A^{-1}\boldsymbol{b} + t[c - \frac{\boldsymbol{b}^\top}{4}A^{-1}\boldsymbol{b}]\}}{|I-2tA|^{\frac{1}{2}}}.$$

Using the Chernoff bound and $\mathbb{E}\ \boldsymbol{z}^\top A\boldsymbol{z} = \mathrm{Tr}(A\mathbb{E}[zz^\top]) = \mathrm{Tr}(A)$, for some γ,

$$\mathbb{P}\{Z \geq \mathbb{E}[Z] + \gamma\} \leq \frac{\mathbb{E}[\exp(tZ)]}{\exp\{t[\gamma + \mathbb{E}(Z)]\}} =$$
$$\frac{\exp\{-\frac{\boldsymbol{b}^\top}{8}A^{-1}[I-(I-2tA)^{-1}]A^{-1}\boldsymbol{b} + t[c - \frac{\boldsymbol{b}^\top}{4}A^{-1}\boldsymbol{b}]\}}{\exp\{t([\gamma + \mathrm{Tr}(A) + \|b\|_2 + c]\}|I-2tA|^{\frac{1}{2}}}.$$

Let us take $\boldsymbol{u} = Q^\top\boldsymbol{b}$. Also, $-\Lambda^{-1}[I-(I-2t\Lambda)^{-1}]\Lambda^{-1} = 2t\Lambda^{-1}(I-2t\Lambda)^{-1}$ since Λ is a diagonal matrix. Using Woodbury matrix identity, we get $(I-2t\Lambda)^{-1} = I - (I - \frac{1}{2t}\Lambda^{-1})^{-1}$. This gives us

[2] [46]

$$\mathbb{P}\{Z \geq \mathbb{E}[Z] + \gamma\} \leq$$
$$\exp\{-\frac{1}{8}\boldsymbol{b}^\top A^{-1}[I-(I-2tA)^{-1}]A^{-1}\boldsymbol{b}$$
$$+ t[c - \frac{1}{4}\boldsymbol{b}^\top A^{-1}\boldsymbol{b}]\}\exp\{-t[\gamma + \mathrm{Tr}(A) + \|b\|_2 + c]\}$$
$$|I-2tA|^{\frac{-1}{2}}$$
$$= \exp\{-\frac{1}{8}\boldsymbol{u}^\top\Lambda^{-1}[I-(I-2t\Lambda)^{-1}]\Lambda^{-1}\boldsymbol{u}$$
$$+ t[c - \frac{1}{4}\boldsymbol{u}^\top\Lambda^{-1}\boldsymbol{u}]\}\exp\{-t[\gamma + \mathrm{Tr}(\Lambda) + \|b\|_2 + c]\}$$
$$|I-2t\Lambda|^{\frac{-1}{2}}$$
$$= \exp\{\frac{t}{4}\boldsymbol{u}^\top\Lambda^{-1}(I-2t\Lambda)^{-1}\boldsymbol{u}$$
$$+ t[c - \frac{1}{4}\boldsymbol{u}^\top\Lambda^{-1}\boldsymbol{u}]\}\exp\{-t[\gamma + \mathrm{Tr}(\Lambda) + \|b\|_2 + c]\}$$
$$|I-2t\Lambda|^{\frac{-1}{2}}$$
$$= \exp\{\frac{t}{4}\boldsymbol{u}^\top\Lambda^{-1}[I-(I-\frac{1}{2t}\Lambda^{-1})^{-1}]\boldsymbol{u}$$
$$+ t[c - \frac{1}{4}\boldsymbol{u}^\top\Lambda^{-1}\boldsymbol{u}]\}\exp\{-t[\gamma + \mathrm{Tr}(\Lambda) + \|b\|_2 + c]\}$$
$$|I-2t\Lambda|^{\frac{-1}{2}}$$
$$= \frac{\exp\{\frac{-t}{4}\boldsymbol{u}^\top\Lambda^{-1}(I-\frac{1}{2t}\Lambda^{-1})^{-1}\boldsymbol{u} + tc\}}{\exp\{t[\gamma + \mathrm{Tr}(\Lambda) + \|b\|_2 + c]\}|I-2t\Lambda|^{\frac{1}{2}}} \leq$$
$$\exp\{\frac{-t}{4\|\boldsymbol{b}\|_2^2}\lambda_{\min}(\Lambda^{-1}(I-\frac{1}{2t}\Lambda^{-1})^{-1})$$
$$- t(\gamma + \mathrm{Tr}(\Lambda) + \|b\|_2)\}|I-2t\Lambda|^{\frac{-1}{2}}$$
$$= \exp\{\frac{-t}{4\|\boldsymbol{b}\|_2^2}\frac{1}{\|\Lambda\| - 1/(2t)} - t(\gamma + \mathrm{Tr}(\Lambda) + \|b\|_2)\}$$
$$|I-2t\Lambda|^{\frac{-1}{2}} \leq \frac{\exp\{\frac{-t}{4\|\boldsymbol{b}\|_2^2\|\Lambda\|} - t(\gamma + \mathrm{Tr}(\Lambda) + \|b\|_2)\}}{|I-2t\Lambda|^{\frac{1}{2}}}.$$

□

A.6. Proof for Theorem 2

Note that if $\tilde{P}(\boldsymbol{x}) < 0$, the classifier predicts a label for class -1, else the predicted label would be 1. Here, $\boldsymbol{x} = y\boldsymbol{\mu} + \boldsymbol{z}$ where $\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{0}, I)$ and $y \in \{\pm 1\}$ since $(\boldsymbol{x}, y) \sim \mathcal{D}$.

$$\tau_{\mathcal{D}}(\tilde{P}) = \mathbb{E}\{\mathbb{1}(y(\boldsymbol{x}^\top A\boldsymbol{x} + \boldsymbol{b}^\top \boldsymbol{x} + c) > 0)\}$$
$$= \mathbb{P}\{y(\boldsymbol{\mu}^\top A\boldsymbol{\mu} + \boldsymbol{z}^\top A\boldsymbol{z} + 2y\boldsymbol{\mu}^\top A\boldsymbol{z} + y\boldsymbol{b}^\top \boldsymbol{\mu} + \boldsymbol{b}^\top \boldsymbol{z} + c) > 0\}$$
$$= \mathbb{P}(y=1)\,\mathbb{P}\{y(\boldsymbol{\mu}^\top A\boldsymbol{\mu} + \boldsymbol{z}^\top A\boldsymbol{z} + 2y\boldsymbol{\mu}^\top A\boldsymbol{z} + y\boldsymbol{b}^\top \boldsymbol{\mu} + \boldsymbol{b}^\top \boldsymbol{z} + c) > 0 \mid y=1\} +$$
$$\mathbb{P}(y=-1)\,\mathbb{P}\{y(\boldsymbol{\mu}^\top A\boldsymbol{\mu} + \boldsymbol{z}^\top A\boldsymbol{z} + 2y\boldsymbol{\mu}^\top A\boldsymbol{z} + y\boldsymbol{b}^\top \boldsymbol{\mu} + \boldsymbol{b}^\top \boldsymbol{z} + c) > 0 \mid y=-1\} =$$
$$\frac{1}{2}\mathbb{P}\{\boldsymbol{z}^\top A\boldsymbol{z} + (\boldsymbol{b}+2A\boldsymbol{\mu})^\top \boldsymbol{z} + \boldsymbol{\mu}^\top A\boldsymbol{\mu} + \boldsymbol{b}^\top \boldsymbol{\mu} + c > 0\} +$$
$$\frac{1}{2}\mathbb{P}\{-\boldsymbol{z}^\top A\boldsymbol{z} - (\boldsymbol{b}-2A\boldsymbol{\mu})^\top \boldsymbol{z} - \boldsymbol{\mu}^\top A\boldsymbol{\mu} + \boldsymbol{b}^\top \boldsymbol{\mu} - c > 0\}$$
$$:= p_1 + p_2$$

We can see that

$$-\gamma_1 := \mathbb{E}\{\boldsymbol{z}^\top A\boldsymbol{z} + (\boldsymbol{b}+2A\boldsymbol{\mu})^\top \boldsymbol{z} + \boldsymbol{\mu}^\top A\boldsymbol{\mu} + \boldsymbol{b}^\top \boldsymbol{\mu} + c\}$$
$$= \operatorname{Tr}(\Lambda) + \|\boldsymbol{b}+2A\boldsymbol{\mu}\|_2 + \boldsymbol{\mu}^\top A\boldsymbol{\mu} + \boldsymbol{b}^\top \boldsymbol{\mu} + c, \text{ and}$$
$$-\gamma_2 := \mathbb{E}\{-\boldsymbol{z}^\top A\boldsymbol{z} - (\boldsymbol{b}-2A\boldsymbol{\mu})^\top \boldsymbol{z} - \boldsymbol{\mu}^\top A\boldsymbol{\mu} + \boldsymbol{b}^\top \boldsymbol{\mu} - c\}$$
$$= -\operatorname{Tr}(\Lambda) + \|\boldsymbol{b}-2A\boldsymbol{\mu}\|_2 - \boldsymbol{\mu}^\top A\boldsymbol{\mu} + \boldsymbol{b}^\top \boldsymbol{\mu} - c.$$

Using Lemma 5, with $\gamma = \gamma_1, t = t_1$ for computing p_1 and $\gamma = \gamma_2, t = t_2$ for computing p_2 where t_1, t_2 are some non-negative constants, we get

$$p_1 = \frac{1}{2|I - 2t_1\Lambda|^{1/2}} \exp\left[t_1\left(\boldsymbol{\mu}^\top A\boldsymbol{\mu} + \boldsymbol{b}^\top \boldsymbol{\mu} + c - \frac{1}{4\|2A\boldsymbol{\mu}+\boldsymbol{b}\|_2\|\Lambda\|}\right)\right], \text{ and}$$
$$p_2 = \frac{1}{2|I - 2t_2\Lambda|^{1/2}} \exp\left[t_2\left(-\boldsymbol{\mu}^\top A\boldsymbol{\mu} + \boldsymbol{b}^\top \boldsymbol{\mu} - c - \frac{1}{4\|2A\boldsymbol{\mu}-\boldsymbol{b}\|_2\|\Lambda\|}\right)\right].$$

This gives us the upper bound for $\tau_{\mathcal{D}}(\tilde{P})$. However, to make sure that this upper bound is smaller than 1, we need to assert more conditions. p_1 and p_2 become smaller as γ_1 and γ_2 are larger positive numbers. However, $\gamma_1 + \gamma_2 = -(\|2A\boldsymbol{\mu}+\boldsymbol{b}\|_2 + \|2A\boldsymbol{\mu}-\boldsymbol{b}\|_2 + 4\boldsymbol{\mu}^\top Q^\top(D_1^{-1} + D_{-1}^{-1})Q\boldsymbol{\mu}) \leq 0$ since $(D_1^{-1} + D_{-1}^{-1}) \succcurlyeq 0$. Hence, we look at separately at cases when either $\gamma_1 > 0$ or $\gamma_2 > 0$.

If $\gamma_1 > 0$, then $\tau_{\mathcal{D}}(\tilde{P}) = \frac{1}{2}(p_1+1) < 1$. Else, if $\gamma_2 > 0$, then $\tau_{\mathcal{D}}(\tilde{P}) = \frac{1}{2}(p_2+1) < 1$. We know that for $\boldsymbol{\mu} \neq \mathbf{0}$, $\tau_{\mathcal{D}}(P) = \phi(\boldsymbol{\mu}) > \frac{1}{2}$. Moreover, for any $a_{-1} \in [0, 0.5]$, $\exists a_1$ such that $\tau_{\mathcal{D}}(\tilde{P}) < \tau_{\mathcal{D}}(P)$. This can be satisfied by picking a_1 such that either γ_1 or γ_2 is very large, i.e., $\frac{1}{2} < \tau_{\mathcal{D}}(\tilde{P}) = \frac{1}{2}[1 + \min(p_1, p_2)] < \tau_{\mathcal{D}}(P)$. We note that the conditions $-\gamma_1 = \boldsymbol{\mu}^\top A\boldsymbol{\mu} + \boldsymbol{b}^\top \boldsymbol{\mu} + c + \operatorname{Tr}(A) + \|2A\boldsymbol{\mu}+\boldsymbol{b}\|_2 < 0$ and $-\gamma_2 = -\boldsymbol{\mu}^\top A\boldsymbol{\mu} + \boldsymbol{b}^\top \boldsymbol{\mu} - c - \operatorname{Tr}(A) + \|2A\boldsymbol{\mu}-\boldsymbol{b}\|_2 < 0$ can always be satisfied by picking a sufficiently large $\boldsymbol{\mu}$ in the direction of an eigenvector corresponding to a negative eigenvalue of A (note that A has negative eigenvalues). □

A.7. Details on generating Figure 3

We use $\boldsymbol{\mu} \in \mathbb{R}^d$, $d = 100$ to generate clean dataset with 1000 data points. They are randomly split into training and testing partitions of equal size. All the assumptions are consistent with the details provided in the main body. We use 30×30 mesh-grid to plot the contour plots. While plotting the theoretical upper bounds, we choose the best t_1, t_2 with grid search from a search space $[2^1, 2^0, 2^{-1}, 2^{-2}, 2^{-3}, 2^{-4}, 2^{-5}]$.

A.8. Experimental details

This subsection provides the details for experiments in Section 5.

Hardware. We use NVIDIA® RTX A4000 GPU with 16GB memory with 16 AMD® EPYC 7302P CPU cores.

Data augmentations. For CIFAR-10 and CIFAR-100, we use random flipping, 4 pixel padding, and random 32×32 size cropping. For ImageNet-100, we use random flipping and random cropping with resizing to 224×224 size. All the images are rescaled to have pixel values in the range $[0, 1]$.

Baselines. We compare CUDA against error-minimizing noise [17], targeted adversarial poisoning [10], neural tangent generalization attack [55], and robust error-minimizing noise [11]. We use the experimental outputs reported in [11] for our comparisons. For REM we choose $\rho_u = 8/255$ and $\rho_a = 4/255$ since REM works the best when $\rho_u = 2\rho_a$ [11]. We perform experiments on REM not present in their work using their code available publicly on GitHub [3] (MIT License).

Networks. For consistency, we use the same architectures used in [11]. We use their GitHub script[4] for this purpose.

Training. We train all the networks for 100 epochs. The initial learning rate is 0.1. Learning rate decays to 0.01 at epoch 40 and to 0.001 at epoch 80. We use a stochastic gradient descent optimizer with a momentum factor of 0.9, weight decay factor of 0.0005, and batch size of 128. For adversarial training, we follow the procedure in [34]. We use 10 steps of projected gradient descent with a step size of $0.15\rho_a$.

[3] https://github.com/fshp971/robust-unlearnable-examples

[4] https://github.com/fshp971/robust-unlearnable-examples/tree/main/models

Analysis of CUDA. For grayscaling experiments, we use images with their average channel values as the input to the network. Test accuracy is computed on the grayscaled test datasets. For smoothing, we use the GitHub codes[5] from [6] (MIT License). For mixup [58], we use the default value of $\alpha = 1.0$.

Deconvolution-based adversarial training (DAT). We experiment with various filter sizes of 3,5, and 7 for the transpose convolution filters. For each batch of data, we use 10 steps of projected gradient descent with a learning rate of 0.1 to learn transpose convolution filters for each class. The weights and biases of the transpose convolution filters are constrained to be within $[-C, C]$. We choose $C = 5$. After 10 steps of inner maximizing optimization, the resulting image is rescaled such that the pixel values lie in $[0, 1]$. See Figure 4 for clean test accuracy vs. epochs plot for DAT with varying transpose filter sizes. As seen in Figure 1, DAT can break CUDA CIFAR-10 with a low blur parameter value of $p_b = 0.1$ to get a clean test accuracy $\sim$78%. However, with higher p_b values DAT can not achieve more than 50% clean test accuracy. DAT solves the following optimization problem:

$$\arg\min_{\theta} \frac{1}{n} \sum_{k \in [K]} \max_{\|s_k\|_\infty \leq C} \sum_{i: y_i = k} \ell(f_\theta(\mathbf{x}_i \star s_k), k) \quad (3)$$

where $\star$ denotes the transpose convolution operator, s_{y_i} denotes the transpose convolution filter for class y_i, and ℓ is the soft-max cross-entropy loss function.

CUDA with augmentations. We use mixup with the default $\alpha = 1.0$ [58]. See Figure 5 for the training curve. For random blurring augmentations, we use $p_b' = 0.1, 0.3$ and $k = 3$. With both these parameters, CUDA is seen to be effective. See Figure 5 for the training curve with $p_b' = 0.3$.

A.9. More experimental results

Figure 6 shows the CUDA CIFAR-10 data generated using $k = 3$ and different p_b blur parameters. Figure 7 shows the CUDA CIFAR-100 and CUDA ImageNet-100 data generated using $k = 3, p_b = 0.3$ and $k = 9, p_b = 0.06$, respectively. Figure 8 shows the clean test accuracy of ResNet-18 with CUDA CIFAR-10 generated using different blur parameters. As we see in the plots, higher the blur parameter, better the effectiveness of CUDA is. However, we choose $p_b = 0.3$ for our experiments since the the images generated using this hyperparameter look perceptibly more similar to the clean images (when compared to $p_b = 0.5$) while giving a very low clean test accuracy. A lower value of $p_b = 0.1$ gives better unlearnability. However, CUDA CIFAR-10 generated using $p_b = 0.1$ is not robust with our Deconvolution-based Adversarial Training, as shown in Figure 1. Figure 9 shows the clean test accuracy of ResNet-18 with CUDA ImageNet-100 dataset generated using different filter sizes. Figure 10 shows the adversarial training curves for ResNet-18 with different CUDA datasets.

We show the effectiveness of CUDA with Tiny-ImageNet [26], DeIT [50], EfficientNetV2 [48], and MobileNetV2 [41] below.

Model	ERM	L_2 AT ($\epsilon = 0.5$)
DeIT [50]	24.85 %	38.90 %
EfficientNetV2-S [48]	20.47 %	42.19 %
MobileNetV2 [41]	21.10 %	32.00 %

Table 6. Effectiveness of CUDA on CIFAR-10.

Training method	Clean	CUDA
ERM	48.14 %	5.98 %
L_2 AT ($\epsilon = 0.5$)	42.72 %	14.54 %

Table 7. Effectiveness of Tiny-ImageNet CUDA with ResNet-18. We use the same hyperparameters as our CIFAR experiments.

A.10. Effects of blurring

Here, we study the effects of blurring. We investigate if class-wise blurring is required for achieving unlearnability. For this, we use a universal filter (generated using the same $p_b = 0.3$ and $k = 3$ hyperparameters) to blur all the training images in the dataset. A ResNet-18 trained on this dataset achieves a clean test accuracy of 90.47%. Essentially, the blurring that is performed only degrades the clean test accuracy by $\sim$4%. This means that class-wise blurring (CUDA) is required for achieving the unlearnability effect (see Figure 11). This experiment also demonstrates that the blurring we perform does not make the dataset useless or destroy its semantics. For this experiment, we use models with fixed initialization and random seeds.

A.11. Why does CUDA work?

In this section, we perform experiments that show that a model trained on CUDA dataset learns the relation between the class-wise filters and the labels. We train ResNet-18 using the CUDA CIFAR-10 dataset for the experiments. We perform three independent trials for each of the experiments and report the mean performance scores. Trained models achieve a mean clean test accuracy of 21.34%. Now, we use the class-wise filters to perturb the images in the test set based on their corresponding labels. Trained models achieve a very high mean accuracy of 99.91% on this perturbed test set. This shows that the trained models learned the relation between the filters and their corresponding labels. Next, we permute the filters to perturb the test set such that test set images with label i are perturbed with the filters

[5] https://github.com/Hadisalman/smoothing-adversarial

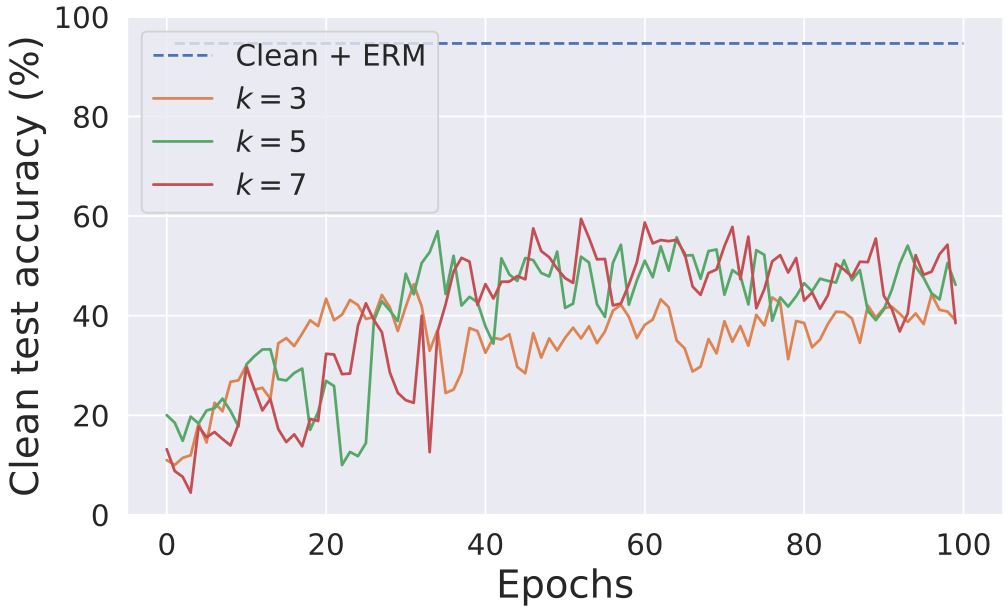


Figure 4. CUDA CIFAR-10 images ($k = 3, p_b = 0.3$) trained using ResNet-18 with the Deconvolution-based Adversarial Training framework with varying transpose convolution filter sizes k.

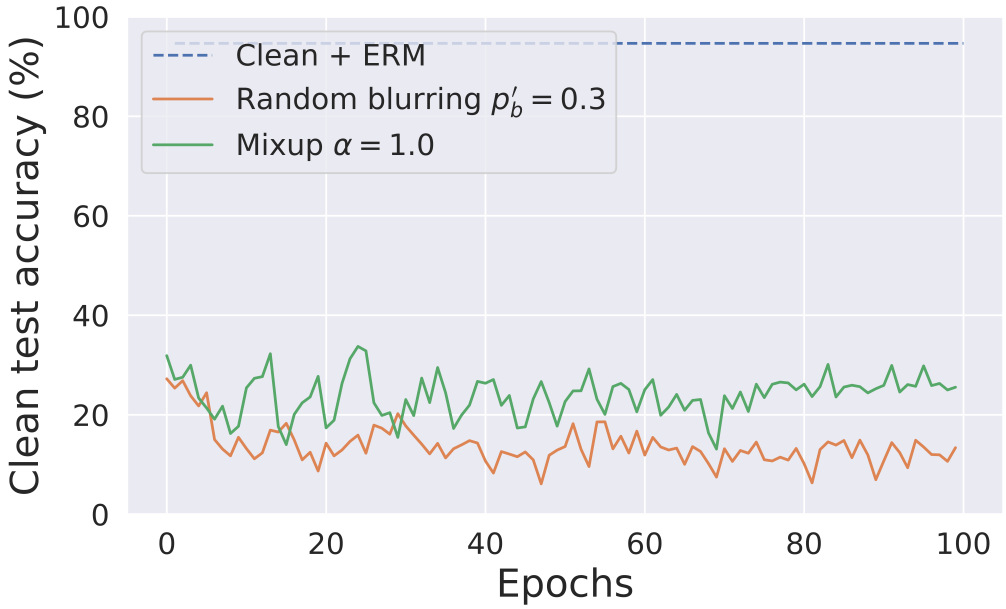
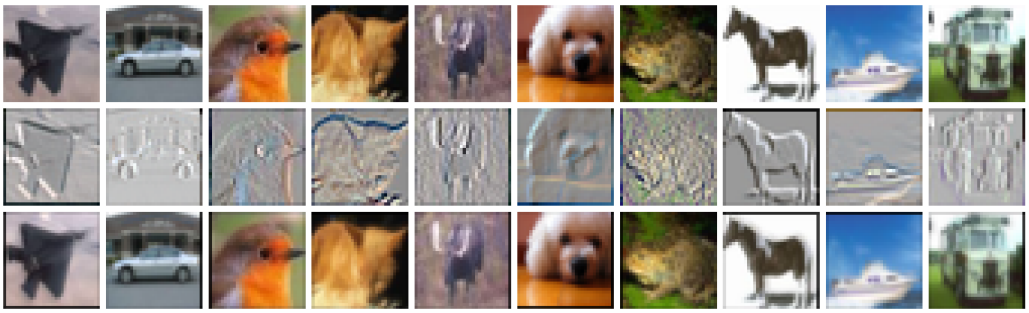


Figure 5. CUDA CIFAR-10 images ($k = 3, p_b = 0.3$) trained using ResNet-18 with mixup and random blurring augmentations.

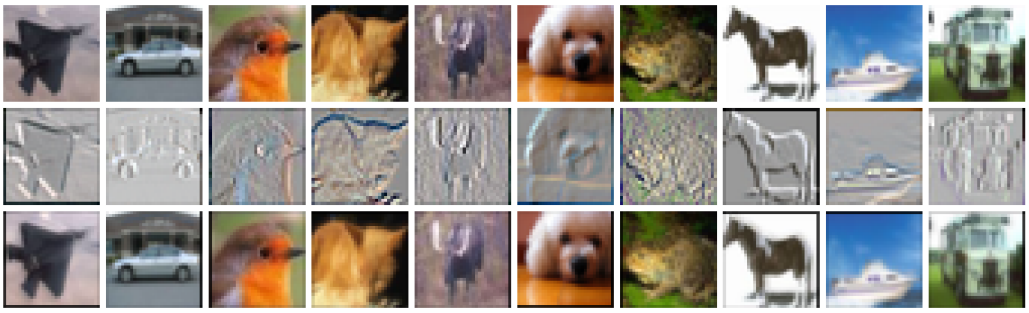
of class $(i + 1)\%10$. Trained models achieve a very low mean accuracy of 2.53% on this perturbed test set. This is evidence that CUDA can also be used for backdoor attacks.

A.12. Effect of transfer learning

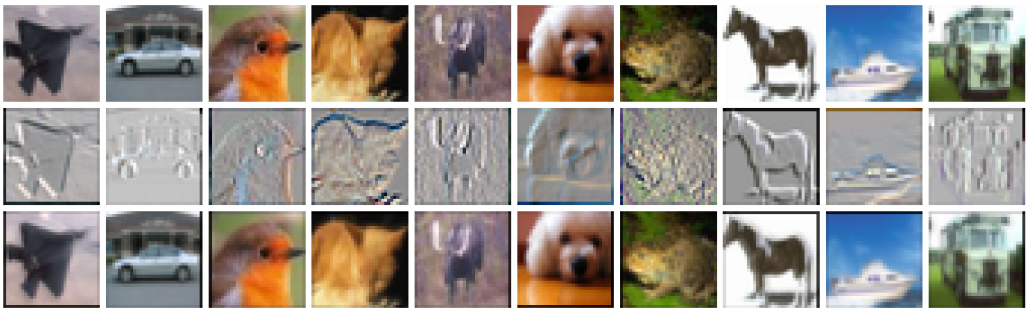
In this section, we experiment the effect of using a pre-trained ResNet-18 with PyTorch [38]. We train it on the CUDA CIFAR-10 dataset in two different ways. First, we



(a) $p_b = 0.1$



(b) $p_b = 0.3$



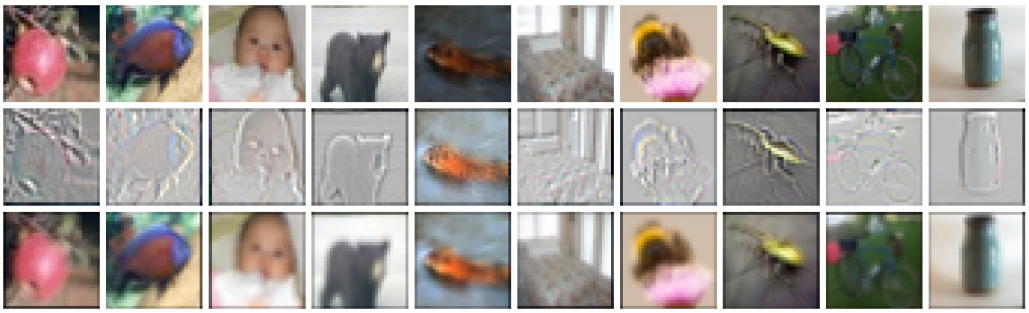
(c) $p_b = 0.5$

Figure 6. CUDA CIFAR-10 images generated using different blur parameters p_b. The top row shows the clean images, the bottom row shows the corresponding CUDA image, and the middle row shows the normalized difference between the clean and the CUDA image.

fine-tune the whole network on the CUDA dataset with a learning rate of 0.001 for 15 epochs. This achieves a clean test accuracy of 42.42%. Fine-tuning the network with clean training data gives 94.19% clean test accuracy. Next, we freeze all the layers except the final layer to train a linear classifier with the pre-trained weights using the CUDA CIFAR-10 dataset. We call this "Freeze and learn". We use a SGD optimizer to train the linear layer for 15 epochs with an initial learning rate of 0.1. The learning rate is decayed by a factor of 10 after every 5 epochs. This achieves a clean test accuracy of 48.22%. The results are shown in Figure 12. This experiment shows that pre-trained network with CUDA data training does not help achieve good generalization on the clean data distribution.

A.13. Effect of CUDA with regularization techniques

In this section, we study the effect of training a ResNet-18 with CUDA CIFAR-10 dataset using various regularization techniques such as mixup [58], cutout [9], cutmix [56], autoaugment [7], and orthogonal regularization [3]. We perform mixup, cutout, cutmix, autoaugment, and orthogonal regularization to achieve 25.53%, 25.80%, 26.93%, 34.09%, and 50.72%. Even though these regular-



(a) CIFAR-100

(b) ImageNet-100

Figure 7. CUDA CIFAR-100 and ImageNet-100 images generated using $k = 3, p_b = 0.3$ and $k = 9, p_b = 0.06$, respectively. The top row shows the clean images, the bottom row shows the corresponding CUDA image, and the middle row shows the normalized difference between the clean and the CUDA image.

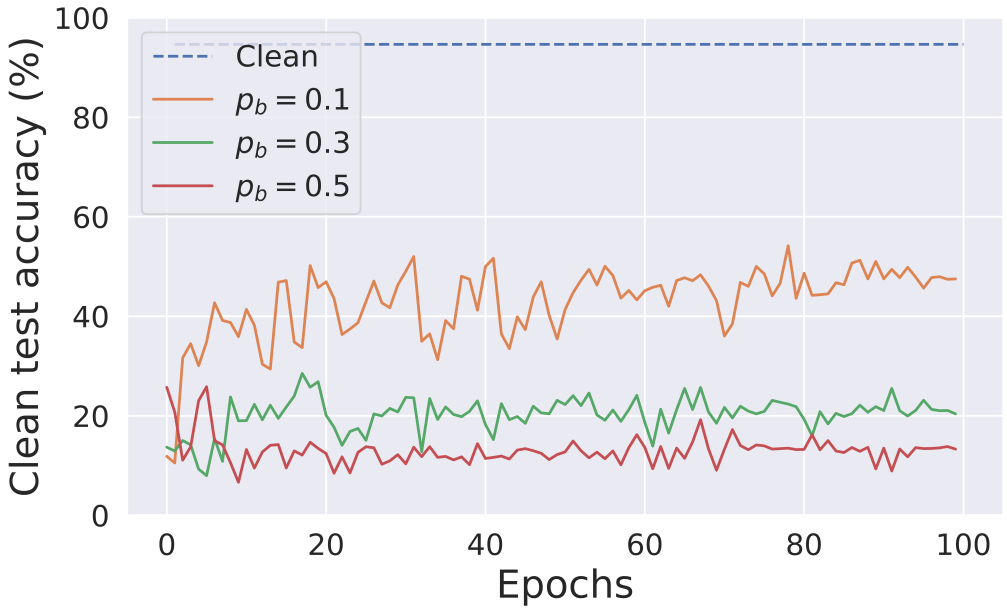


Figure 8. ResNet-18 trained using CUDA CIFAR-10 data generated using different blur parameters p_b.

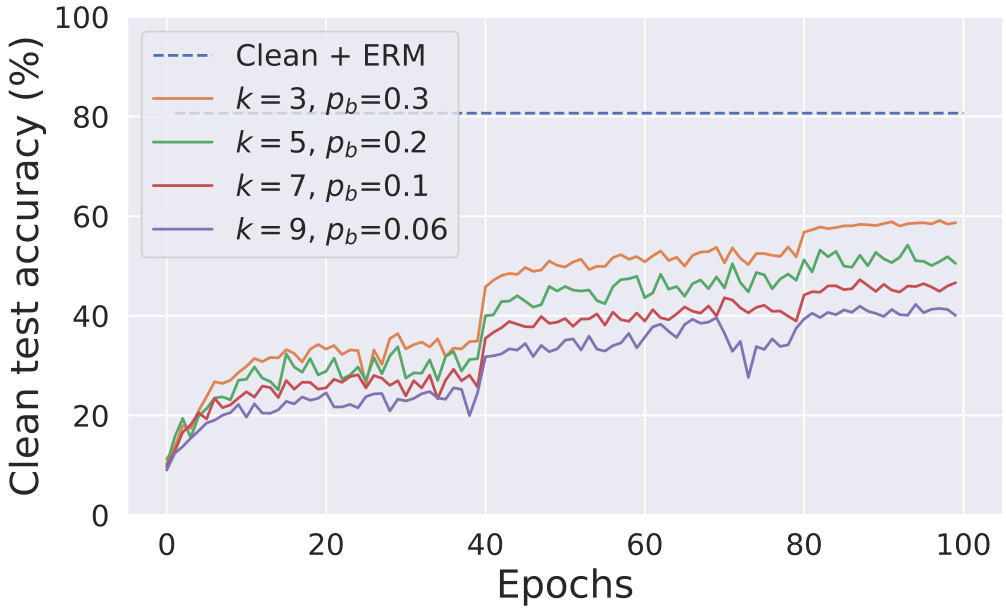
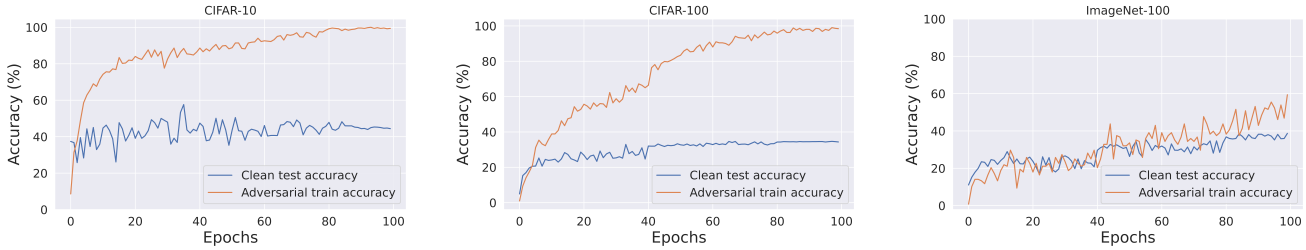


Figure 9. ResNet-18 trained using CUDA ImageNet-100 dataset generated using different filter sizes k.



(a) CIFAR-10 (b) CIFAR-100 (c) ImageNet-100

Figure 10. Adversarial training curves for ResNet-18 with CIFAR-10, CIFAR-100, and ImageNet-100 CUDA datasets.

izations help in improving the vanilla ERM training, these networks still do not achieve good generalization on the clean data distribution. We use cutout using GitHub codes[6] with `length=16` and `n_holes=1`, cutmix using GitHub codes[7] with $\alpha = 1$, autoaugment using PyTorch [38], mixup using GitHub codes[8] with $\alpha = 1$, and orthogonal regularization using GitHub codes[9] with `reg=1e-6` (all MIT licenses).

A.14. Network parameter distribution

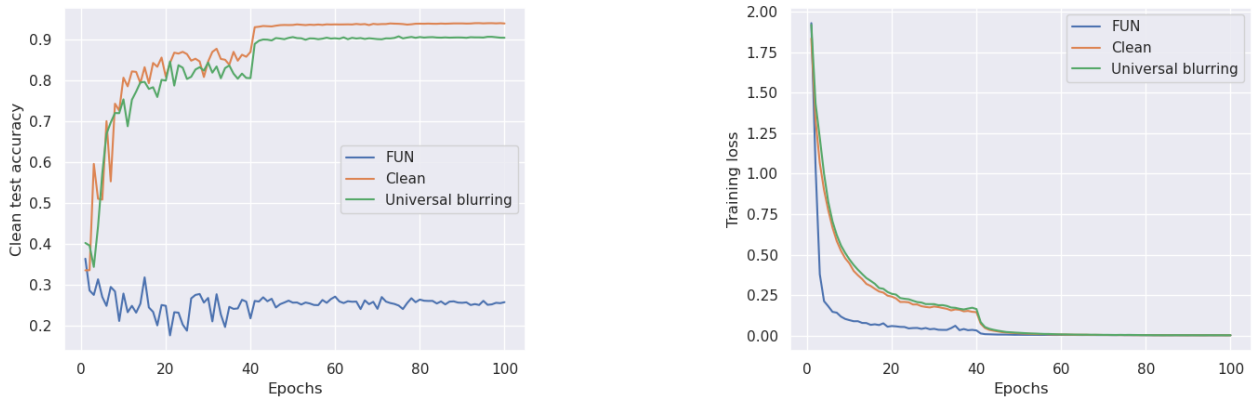
In this section, we compare the network parameter distributions of ResNet-18 trained on clean and CUDA CIFAR-10 datasets (see Figure 13). Both the distributions are similar to normal distributions with a mean of 0. However, the parameter distribution of the clean model has a higher standard deviation than the CUDA-based model's parameter distribution.

[6] https://github.com/uoguelph-mlrg/Cutout/blob/master/util/cutout.py

[7] https://github.com/hysts/pytorch_cutmix/blob/master/cutmix.py

[8] https://github.com/facebookresearch/mixup-cifar10/blob/main/train.py

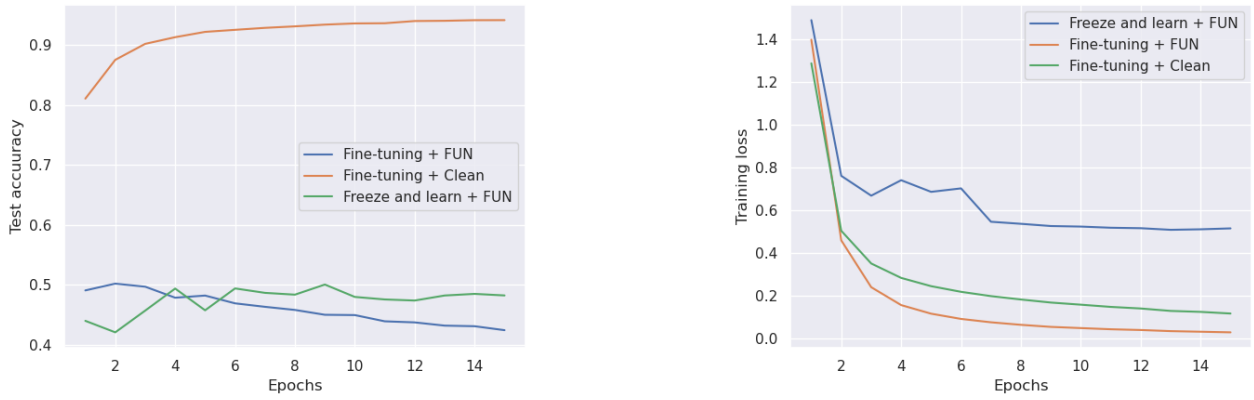
[9] https://github.com/kevinzakka/pytorch-goodies



(a) Test accuracy

(b) Training loss

Figure 11. ResNet-18 trained using clean, CUDA, and universally blurred CIFAR-10 datasets.



(a) Test accuracy

(b) Training loss

Figure 12. Pre-trained ResNet-18 with fine-tuning and training the linear layer using CUDA and clean CIFAR-10 datasets.

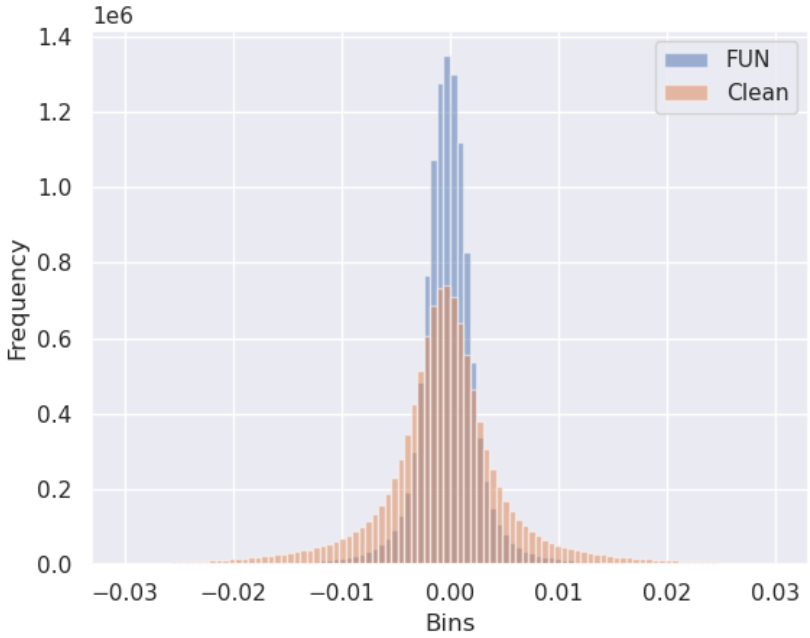


Figure 13. Network parameter distributions of ResNet-18 trained on clean and CUDA CIFAR-10 datasets.