A. Proofs

Theorem 1 (Lee et al. (2019))**.** *For any $f : \mathbb{R}^d \to [0,1]$ and parameter $\lambda \in \mathbb{R}^+$, define:*

$$p(\boldsymbol{x}) := \mathop{\mathbb{E}}_{\epsilon \sim \mathcal{U}^d(-\lambda,\lambda)} \left[f(\boldsymbol{x}+\epsilon) \right]. \tag{30}$$

Then, $p(.)$ is $1/(2\lambda)$-Lipschitz with respect to the ℓ_1 norm.

Proof. Consider two arbitrary points $\boldsymbol{x}, \boldsymbol{x}'$ where $\delta := \boldsymbol{x}' - \boldsymbol{x}$. We consider two cases.

- Case 1: $\|\delta\|_1 \geq 2\lambda$: Then, because $f(\cdot) \in [0,1]$, and therefore $p(\cdot) \in [0,1]$, we have:

$$|p(\boldsymbol{x}) - p(\boldsymbol{x}')| \leq 1 \leq \frac{\|\delta\|_1}{2\lambda} \tag{31}$$

- Case 2: $\|\delta\|_1 < 2\lambda$: In this case, for each i, $|\delta_i| < 2\lambda$. Define $\mathcal{B}(\boldsymbol{x})$ as the ℓ_∞ ball of radius λ around $\boldsymbol{x}$, and $\mathcal{U}(\mathcal{B}(\boldsymbol{x}))$ as the uniform distribution on this ball (and, similarly $\mathcal{U}(\cdot)$, on any other set). In other words:

$$p(\boldsymbol{x}) = \mathop{\mathbb{E}}_{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}))} f(\boldsymbol{z}) \tag{32}$$

Then,

$$\begin{aligned}
&|p(\boldsymbol{x}) - p(\boldsymbol{x}')| \\
&= |\mathop{\mathbb{E}}_{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}))} f(\boldsymbol{z}) - \mathop{\mathbb{E}}_{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}'))} f(\boldsymbol{z})| \\
&= \Big| \Big(\Pr_{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}) \setminus \mathcal{B}(\boldsymbol{x}') \mathop{\mathbb{E}}_{\substack{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}) \\ \setminus \mathcal{B}(\boldsymbol{x}'))}} f(\boldsymbol{z}) \\
&+ \Pr_{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}) \cap \mathcal{B}(\boldsymbol{x}') \mathop{\mathbb{E}}_{\substack{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}) \\ \cap \mathcal{B}(\boldsymbol{x}'))}} f(\boldsymbol{z}) \Big) \\
&- \Big(\Pr_{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}'))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}') \setminus \mathcal{B}(\boldsymbol{x}) \mathop{\mathbb{E}}_{\substack{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}') \\ \setminus \mathcal{B}(\boldsymbol{x}))}} f(\boldsymbol{z}) \\
&+ \Pr_{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}'))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}) \cap \mathcal{B}(\boldsymbol{x}') \mathop{\mathbb{E}}_{\substack{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}) \\ \cap \mathcal{B}(\boldsymbol{x}'))}} f(\boldsymbol{z}) \Big) \Big| \\
&= \Big| \Pr_{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}) \setminus \mathcal{B}(\boldsymbol{x}') \mathop{\mathbb{E}}_{\substack{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}) \\ \setminus \mathcal{B}(\boldsymbol{x}'))}} f(\boldsymbol{z}) \\
&- \Pr_{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}'))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}') \setminus \mathcal{B}(\boldsymbol{x}) \mathop{\mathbb{E}}_{\substack{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}') \\ \setminus \mathcal{B}(\boldsymbol{x}))}} f(\boldsymbol{z}) \Big|
\end{aligned} \tag{33}$$

Note that:

$$\begin{aligned}
&\Pr_{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}'))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}') \setminus \mathcal{B}(\boldsymbol{x}) \\
&= \Pr_{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}) \setminus \mathcal{B}(\boldsymbol{x}')
\end{aligned} \tag{34}$$

Because both represent the probability of a uniform random variable on an ℓ_∞ ball of radius λ taking a value outside of the region $\mathcal{B}(\boldsymbol{x}) \cap \mathcal{B}(\boldsymbol{x}')$ (which is entirely contained within both balls.) Then:

$$\begin{aligned}
&|p(\boldsymbol{x}) - p(\boldsymbol{x}')| \\
&= \Pr_{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}) \setminus \mathcal{B}(\boldsymbol{x}') \\
&\times \Big| \mathop{\mathbb{E}}_{\substack{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}) \setminus \\ \mathcal{B}(\boldsymbol{x}'))}} f(\boldsymbol{z}) - \mathop{\mathbb{E}}_{\substack{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}') \\ \setminus \mathcal{B}(\boldsymbol{x}))}} f(\boldsymbol{z}) \Big| \\
&\leq \Pr_{\boldsymbol{z} \sim \mathcal{U}(\mathcal{B}(\boldsymbol{x}))} \boldsymbol{z} \in \mathcal{B}(\boldsymbol{x}) \setminus \mathcal{B}(\boldsymbol{x}').
\end{aligned} \tag{35}$$

Where, in the last line, we used the fact that $f(\cdot) \in [0,1]$. Let $\mathcal{V}(\mathcal{S})$ represent the volume of a set $\mathcal{S}$. Note that $\mathcal{B}(\boldsymbol{x}) \cap \mathcal{B}(\boldsymbol{x}')$ is a d-hyperrectangle, with each edge of length

$$\min(x_i, x_i') + \lambda - (\max(x_i, x_i') - \lambda) = 2\lambda - |\delta_i| \tag{36}$$

Then following Equation 35,

$$\begin{aligned}
&|p(\boldsymbol{x}) - p(\boldsymbol{x}')| \\
&\leq \frac{\mathcal{V}(\mathcal{B}(\boldsymbol{x})) - \mathcal{V}(\mathcal{B}(\boldsymbol{x}) \cap \mathcal{B}(\boldsymbol{x}'))}{\mathcal{V}(\mathcal{B}(\boldsymbol{x}))} \\
&= 1 - \frac{\Pi_{i=1}^d (2\lambda - |\delta_i|)}{(2\lambda)^d} \\
&= 1 - \prod_{i=1}^d \left(1 - \frac{|\delta_i|}{2\lambda} \right)
\end{aligned} \tag{37}$$

Note that, for $1 \leq d' \leq d$:

$$\begin{aligned}
&\prod_{i=1}^{d'} \left(1 - \frac{|\delta_i|}{2\lambda} \right) \\
&= \prod_{i=1}^{d'-1} \left(1 - \frac{|\delta_i|}{2\lambda} \right) - \frac{|\delta_{d'}|}{2\lambda} \prod_{i=1}^{d'-1} \left(1 - \frac{|\delta_i|}{2\lambda} \right) \\
&\geq \prod_{i=1}^{d'-1} \left(1 - \frac{|\delta_i|}{2\lambda} \right) - \frac{|\delta_{d'}|}{2\lambda}
\end{aligned} \tag{38}$$

By induction:

$$\prod_{i=1}^{d} \left(1 - \frac{|\delta_i|}{2\lambda} \right) \geq 1 - \sum_{i=1}^{d} \frac{|\delta_i|}{2\lambda} \tag{39}$$

Therefore,

$$\begin{aligned}
&|p(\boldsymbol{x}) - p(\boldsymbol{x}')| \\
&\leq 1 - \prod_{i=1}^{d} \left(1 - \frac{|\delta_i|}{2\lambda} \right) \\
&\leq 1 - \left(1 - \sum_{i=1}^{d} \frac{|\delta_i|}{2\lambda} \right) \\
&= \frac{\|\delta\|_1}{2\lambda}
\end{aligned} \tag{40}$$

Thus, by the definition of Lipschitz-continuity, p is $1/(2\lambda)$-Lipschitz with respect to the ℓ_1 norm. □

Theorem 2 (General Case). *For any $f : \mathbb{R}^d \to [0,1]$, and $\lambda > 0$ let $\boldsymbol{s} \in [0, 2\lambda]^d$ be a random variable, with a fixed distribution such that:*

$$s_i \sim \mathcal{U}(0, 2\lambda), \quad \forall i. \tag{41}$$

*Note that the components $s_1, ..., s_d$ are **not** required to be distributed independently from each other. Then, define:*

$$\tilde{x}_i := \frac{\min(2\lambda\lceil\frac{x_i - s_i}{2\lambda}\rceil + s_i, 1)}{2} \tag{42}$$

$$+ \frac{\max(2\lambda\lceil\frac{x_i - s_i}{2\lambda} - 1\rceil + s_i, 0)}{2}, \quad \forall i \tag{43}$$

$$p(\boldsymbol{x}) := \mathop{\mathbb{E}}_{\boldsymbol{s}}[f(\tilde{\boldsymbol{x}})]. \tag{44}$$

Then, $p(.)$ is $1/(2\lambda)$-Lipschitz with respect to the ℓ_1 norm.

Proof. Consider two arbitrary points $\boldsymbol{x}, \boldsymbol{x}'$ where $\delta := \boldsymbol{x}' - \boldsymbol{x}$. We consider two cases.

- Case 1: $\|\delta\|_1 \geq 2\lambda$: Then, because $f(\cdot) \in [0,1]$, and therefore $p(\cdot) \in [0,1]$, we have:

$$|p(\boldsymbol{x}) - p(\boldsymbol{x}')| \leq 1 \leq \frac{\|\delta\|_1}{2\lambda} \tag{45}$$

- Case 2: $\|\delta\|_1 < 2\lambda$:

  In this case, for each i, $|\delta_i| < 2\lambda$, and therefore $\lceil\frac{x_i - s_i}{2\lambda}\rceil$ and $\lceil\frac{x_i' - s_i}{2\lambda}\rceil$ differ by at most one. Furthermore, $\lceil\frac{x_i - s_i}{2\lambda}\rceil$ differs from $\lceil\frac{x_i}{2\lambda}\rceil$ by at most one, and similarly for x_i'. Without loss of generality, assume $x_i < x_i'$ (i.e., $\delta_i = |\delta_i| = x_i' - x_i$).

  There are two cases:

  - Case A: $\lceil\frac{x_i}{2\lambda}\rceil = \lceil\frac{x_i'}{2\lambda}\rceil$. Let this integer be n. Then:
    - $\lceil\frac{x_i - s_i}{2\lambda}\rceil = \lceil\frac{x_i' - s_i}{2\lambda}\rceil = n$ iff $\frac{s_i}{2\lambda} < \frac{x_i}{2\lambda} - (n-1)$ (which also implies $\frac{s_i}{2\lambda} < \frac{x_i'}{2\lambda} - (n-1)$).
    - $\lceil\frac{x_i - s_i}{2\lambda}\rceil = \lceil\frac{x_i' - s_i}{2\lambda}\rceil = n - 1$ iff $\frac{s_i}{2\lambda} \geq \frac{x_i'}{2\lambda} - (n-1)$ (which also implies $\frac{s_i}{2\lambda} \geq \frac{x_i}{2\lambda} - (n-1)$).

    Then $\lceil\frac{x_i - s_i}{2\lambda}\rceil$ and $\lceil\frac{x_i' - s_i}{2\lambda}\rceil$ differ only if $\frac{x_i}{2\lambda} - (n-1) \leq \frac{s_i}{2\lambda} < \frac{x_i'}{2\lambda} - (n-1)$, which occurs with probability $\frac{\delta_i}{2\lambda}$.
  - Case B: $\lceil\frac{x_i}{2\lambda}\rceil + 1 = \lceil\frac{x_i'}{2\lambda}\rceil$. Let $n := \lceil\frac{x_i}{2\lambda}\rceil$. Then $\lceil\frac{x_i - s_i}{2\lambda}\rceil$ and $\lceil\frac{x_i' - s_i}{2\lambda}\rceil$ can differ if either:
    - $\lceil\frac{x_i - s_i}{2\lambda}\rceil = n$ and $\lceil\frac{x_i' - s_i}{2\lambda}\rceil = n + 1$. This occurs iff $\frac{s_i}{2\lambda} < \frac{x_i'}{2\lambda} - n$ (which also implies $\frac{s_i}{2\lambda} < \frac{x_i}{2\lambda} - (n-1)$).
    - $\lceil\frac{x_i - s_i}{2\lambda}\rceil = n - 1$ and $\lceil\frac{x_i' - s_i}{2\lambda}\rceil = n$. This occurs iff $\frac{s_i}{2\lambda} \geq \frac{x_i}{2\lambda} - (n-1)$ (which also implies $\frac{s_i}{2\lambda} \geq \frac{x_i'}{2\lambda} - n$).

    In other words, $\lceil\frac{x_i - s_i}{2\lambda}\rceil = \lceil\frac{x_i' - s_i}{2\lambda}\rceil$ iff:

$$\frac{x_i}{2\lambda} - (n-1) > \frac{s_i}{2\lambda} \geq \frac{x_i'}{2\lambda} - n$$

    Or equivalently:

$$\frac{x_i}{2\lambda} - n + 1 > \frac{s_i}{2\lambda} \geq \frac{x_i}{2\lambda} - n + \frac{\delta_i}{2\lambda}$$

    This happens with probability $1 - \frac{\delta_i}{2\lambda}$. Therefore, $\lceil\frac{x_i - s_i}{2\lambda}\rceil$ and $\lceil\frac{x_i' - s_i}{2\lambda}\rceil$ differ with probability $\frac{\delta_i}{2\lambda}$.

  Note that $\lceil\frac{x_i - s_i}{2\lambda} - 1\rceil$ and $\lceil\frac{x_i' - s_i}{2\lambda} - 1\rceil$ differ only when $\lceil\frac{x_i - s_i}{2\lambda}\rceil$ and $\lceil\frac{x_i' - s_i}{2\lambda}\rceil$ differ. Therefore in both cases, $\tilde{x}_i$ and $\tilde{x}_i'$ differ with probability at most $\frac{|\delta_i|}{2\lambda}$. The rest of the proof proceeds as in the $\lambda \geq 0.5$ case in the main text.

□

Corollary 1 (General Case). *For any $f : \mathbb{R}^d \to [0,1]$, and $\lambda \geq 0$ (with 2λ a multiple of $1/q$), let $\boldsymbol{s} \in [0, 2\lambda - 1/q]^d_{(q)} + \mathbb{1}/(2q)$ be a random variable with a fixed distribution such that:*

$$s_i \sim \mathcal{U}_{(q)}(0, 2\lambda - 1/q) + 1/(2q), \quad \forall i. \tag{46}$$

*Note that the components $s_1, ..., s_d$ are **not** required to be distributed independently from each other. Then, define:*

$$\tilde{\mathbf{x}}_i := \frac{\min(2\lambda\lceil\frac{\mathbf{x}_i - s_i}{2\lambda}\rceil + s_i, 1)}{2} \tag{47}$$

$$+ \frac{\max(2\lambda\lceil\frac{\mathbf{x}_i - s_i}{2\lambda} - 1\rceil + s_i, 0)}{2}, \quad \forall i \tag{48}$$

$$p(\mathbf{x}) := \mathop{\mathbb{E}}_{\boldsymbol{s}}[f(\tilde{\mathbf{x}})]. \tag{49}$$

Then, $p(.)$ is $1/(2\lambda)$-Lipschitz with respect to the ℓ_1 norm on the quantized domain $\mathbf{x} \in [0,1]^d_{(q)}$.

Proof. The proof is substantially similar to the proof of the continuous case above. Minor differences occur in Cases 2.A and 2.B (mostly due to inequalities becoming strict, because possible values of s_i are offset from values of $\mathbf{x}_i$) which we show here:

- Case A: $\lceil\frac{\mathbf{x}_i}{2\lambda}\rceil = \lceil\frac{\mathbf{x}_i'}{2\lambda}\rceil$. Let this integer be n. Then:
  - $\lceil\frac{\mathbf{x}_i - s_i}{2\lambda}\rceil = \lceil\frac{\mathbf{x}_i' - s_i}{2\lambda}\rceil = n$ iff $\frac{s_i}{2\lambda} < \frac{\mathbf{x}_i}{2\lambda} - (n-1)$ (which also implies $\frac{s_i}{2\lambda} < \frac{\mathbf{x}_i'}{2\lambda} - (n-1)$).
  - $\lceil\frac{\mathbf{x}_i - s_i}{2\lambda}\rceil = \lceil\frac{\mathbf{x}_i' - s_i}{2\lambda}\rceil = n-1$ iff $\frac{s_i}{2\lambda} > \frac{\mathbf{x}_i'}{2\lambda} - (n-1)$ (which also implies $\frac{s_i}{2\lambda} > \frac{\mathbf{x}_i}{2\lambda} - (n-1)$).

Then $\lceil \frac{\mathbf{x}_i - s_i}{2\lambda} \rceil$ and $\lceil \frac{\mathbf{x}'_i - s_i}{2\lambda} \rceil$ differ only if $\frac{\mathbf{x}_i}{2\lambda} - (n-1) < \frac{s_i}{2\lambda} < \frac{\mathbf{x}'_i}{2\lambda} - (n-1)$. There are exactly $q \cdot \delta_i$ discrete values that s_i can take such that this condition holds. This is out of $2\lambda q$ possible values over which s_i is uniformly distributed. Therefore, the condition holds with probability $\frac{\delta_i}{2\lambda}$.

- Case B: $\lceil \frac{\mathbf{x}_i}{2\lambda} \rceil + 1 = \lceil \frac{\mathbf{x}'_i}{2\lambda} \rceil$. Let $n := \lceil \frac{\mathbf{x}_i}{2\lambda} \rceil$. Then $\lceil \frac{\mathbf{x}_i - s_i}{2\lambda} \rceil$ and $\lceil \frac{\mathbf{x}'_i - s_i}{2\lambda} \rceil$ can differ if either:
  - $\lceil \frac{\mathbf{x}_i - s_i}{2\lambda} \rceil = n$ and $\lceil \frac{\mathbf{x}'_i - s_i}{2\lambda} \rceil = n+1$. This occurs iff $\frac{s_i}{2\lambda} < \frac{\mathbf{x}'_i}{2\lambda} - n$ (which also implies $\frac{s_i}{2\lambda} < \frac{\mathbf{x}_i}{2\lambda} - (n-1)$).
  - $\lceil \frac{\mathbf{x}_i - s_i}{2\lambda} \rceil = n-1$ and $\lceil \frac{\mathbf{x}'_i - s_i}{2\lambda} \rceil = n$. This occurs iff $\frac{s_i}{2\lambda} > \frac{\mathbf{x}_i}{2\lambda} - (n-1)$ (which also implies $\frac{s_i}{2\lambda} > \frac{\mathbf{x}'_i}{2\lambda} - n$).

  In other words, $\lceil \frac{\mathbf{x}_i - s_i}{2\lambda} \rceil = \lceil \frac{\mathbf{x}'_i - s_i}{2\lambda} \rceil$ iff:

$$\frac{\mathbf{x}_i}{2\lambda} - (n-1) > \frac{s_i}{2\lambda} > \frac{\mathbf{x}'_i}{2\lambda} - n$$

  Or equivalently:

$$\frac{\mathbf{x}_i}{2\lambda} - n + 1 > \frac{s_i}{2\lambda} > \frac{\mathbf{x}_i}{2\lambda} - n + \frac{\delta_i}{2\lambda}$$

  There are exactly $q \cdot (1 - \delta_i)$ discrete values that s_i can take such that this condition holds. This is out of $2\lambda q$ possible values over which s_i is uniformly distributed. Therefore, the condition holds with probability $\frac{1-\delta_i}{2\lambda}$. Thus, $\lceil \frac{\mathbf{x}_i - s_i}{2\lambda} \rceil$ and $\lceil \frac{\mathbf{x}'_i - s_i}{2\lambda} \rceil$ differ with probability $\frac{\delta_i}{2\lambda}$.

□

B. Experimental Details

For uniform additive noise, we reproduced Yang et al. (2020)'s results directly, using their released code. Note that we also reproduced the training of all models, rather than using released models. For Independent SSN and DSSN, we followed the same training procedure as in Yang et al. (2020), but instead used the noise distribution of our methods during training. For DSSN, we used the same vector $\boldsymbol{v}$ to generate noise during training and test time: note that our certificate requires $\boldsymbol{v}$ to be the same fixed vector whenever the classifier is used. In particular, we used a pseudorandom array generated using the Mersenne Twister algorithm with seed 0, as implemented in NumPy as numpy.random.RandomState. This is guaranteed to produce identical results on all platforms and for all future versions of NumPy, given the same seed, so in practice we only store the seed (0). In Section C, we explore the sensitivity of our method to different choices of pseudorandom seeds.

In a slight deviation from Cohen et al. (2019), Yang et al. (2020) uses different noise vectors for each sample in a batch when training (Cohen et al. (2019) uses the same ϵ for all samples in a training batch to improve speed). We follow Yang et al. (2020)'s method: this means that when training DSSN, we train the classifier on each sample only once per epoch, with a single, randomly-chosen value of s_{base}, which varies between samples in a batch.

Training parameters (taken from Yang et al. (2020)) were as follows (Table 2):

	CIFAR-10	ImageNet
Architecture	WideResNet-40	ResNet-50
Number of Epochs	120	30
Batch Size	64 [5]	64
Initial Learning Rate	0.1	0.1
LR Scheduler	Cosine Annealing	Cosine Annealing

Table 2. Training parameters for experiments.

For all training and certification results in the main text, we used a single NVIDIA 2080 Ti GPU. (Some experiments with denoisers, in Section D, used two GPUs.)

For testing, we used the entire CIFAR-10 test set (10,000 images) and a subset of 500 images of ImageNet (the same subset used by Cohen et al. (2019)).

When reporting clean accuracies for randomized techniques (uniform additive noise and Independent SSN), we followed (Yang et al., 2020) by simply reporting the percent of samples for which the $N_0 = 64$ initial noise perturbations, used to pick the top class during certification, actually selected the correct class. (Notably, (Yang et al., 2020) does not use an "abstain" option for prediction, as some other randomized smoothing works (Cohen et al., 2019) do.) On the one hand, this is an inexact estimate of the accuracy of the *true* classifier $p(\boldsymbol{x})$, which uses the true expectation. On the other hand, it is the actual, empirical accuracy of a classifier that is being used in practice. This is not an issue when reporting the clean accuracy for DSSN, which is exact.

In DSSN, following Levine & Feizi (2020a) (discussed in

[5] There is a discrepancy between the code and the text of Yang et al. (2020) about the batch size used for training on CIFAR-10: the paper says to use a batch size of 128, while the instructions for reproducing the paper's results released with the code use a batch size of 64. Additionally, inspection of one of Yang et al. (2020)'s released models indicates that a batch size of 64 was in fact used. (In particular, the "num_batches_tracked" field in the saved model, which counts the total number of batches used in training, corresponded with a batch size of 64.) We therefore used a batch size of 64 in our reproduction, assuming that the discrepancy was a result of a typo in that paper.

Section 1.1), if two classes tie in the number of "votes", we predict the first class lexicographically: this means that we can certify robustness up to *and including* the radius ρ, because we are guaranteed consistent behavior in the case of ties. Reported certified radii for DSSN should therefore be interpreted to guarantee robustness even in the $\|\mathbf{x} - \mathbf{x}'\|_1 = \rho$ case. (This is not a meaningful distinction in randomized methods where the space is taken as continuous).

C. Effect of pseudorandom choice of $\boldsymbol{v}$

In Section B, we mention that the vector $\boldsymbol{v}$ used in the derandomization of DSSN, which must be re-used every time the classifier is used, is generated pseudorandomly, using a seed of 0 in all experiments. In this section, we explore the sensitivity of our results to the choice of vector $\boldsymbol{v}$, and in particular to the choice of random seed. To do this, we repeated all standard-training DSSN experiments on CIFAR-10, using two additional choices of random seeds. We performed both training and certification using the assigned $\boldsymbol{v}$ vector for each experiment. Result are summarized in Table 3. We report a tabular summary, rather than certification curves, because the curves are too similar to distinguish. In general, the choice of random seed to select $\boldsymbol{v}$ does not seem to impact the certified accuracies: all best certified accuracies were within 0.65 percentage points of each other. This suggests that our method is robust to the choice of this hyperparameter.

D. Effect of a Denoiser

As shown in Figure 8 in the main text, at large λ, there is a substantial benefit to SSN which is unrelated to derandomization, due to the differences in noise distributions discussed in Section 4.2.1. However, Equation 22 shows that the difference between uniform additive noise and Independent SSN is a simple, deterministic transformation on each pixel. We therefore wondered whether training a denoiser network, to learn the relationship between $\boldsymbol{x}$ and the noisy sample ($\boldsymbol{x} + \epsilon$ or $\tilde{\boldsymbol{x}}$), would eliminate the differences between the methods. Salman et al. (2020) proposes methods of training denoisers for randomized smoothing, in the context of using smoothing on pre-trained classifiers. In this context, the noisy image first passes through a denoiser network, before being passed into a classification network trained on clean images. We used their code (and all default parameters), in three variations:

1. **Stability Denoising**: In this method, the pre-trained classifier network is required for training the denoiser. The loss when training the denoiser is based on the consistency between the logit outputs of the classifier on the clean input $\boldsymbol{x}$ and on the denoised version of the noisy input. This is the best-performing method in (Salman et al., 2020). However, note that it does not directly use the pixel values of $\boldsymbol{x}$ when training the denoiser, and therefore might not "learn" the correspondence between clean and noisy samples (Figure 2 in the main text) as easily.
2. **MSE Denoising**: This trains the denoiser via direct supervised training, with the objective of reducing the mean squared error difference between the pixel values of the clean and denoised samples. Then, classification is done using a classifier that is pre-trained only on clean samples. This performs relatively poorly in (Salman et al., 2020), but should directly learn the correspondence between clean and noisy samples.
3. **MSE Denoising with Retraining**: For this experiment, we trained an MSE denoiser as above, but *then* trained the entire classification pipeline (the denoiser + the classifier) on noisy samples. Note that the classifier is trained from scratch in this case, with the pre-trained denoiser already in place (but being fine-tuned as the classifier is trained).

We tested on CIFAR-10, at three different noise levels, without stability training. See Figure 9 for results. Overall, we find that at high noise, there is still a significant gap in performance between Independent SSN and (Yang et al., 2020)'s method, using all of the denoising techniques. One possible explanation is that it is also more difficult *for the denoiser* to learn the noise distribution of (Yang et al., 2020), compared to our distributions.

E. Additive and splitting noise allow for different types of joint noise distributions

In Section 4.2 in the main text, we showed that, in the $\lambda = 0.5$ case, SSN leads to marginal distributions which are simple affine transformations of the marginal distributions of the uniform additive smoothing noise (Equation 23). However, we also showed (Proposition 1) that, even in this case, certification is not possible using arbitrary joint distributions of ϵ with uniform additive noise, as it is with SSN. This difference is explained by the fact that, even for $\lambda = 0.5$, the joint distributions of $(\boldsymbol{x} + \epsilon)$ which can be generated by uniform additive noise and the joint distributions of $\tilde{\boldsymbol{x}}$ which can be generated by SSN respectively are in fact quite different.

To quantify this, consider a pair of two joint distributions: $\mathcal{D}$, with marginals uniform on $[-0.5, 0.5]$, and $\mathcal{S}$, with marginals uniform on $[0, 1]$. Let $\mathcal{D}$ and $\mathcal{S}$ be considered *equivalent* if, for $\epsilon \sim \mathcal{D}$ and $\boldsymbol{s} \sim \mathcal{S}$:

$$\tilde{\boldsymbol{x}} \sim (1/2)(\boldsymbol{x} + \epsilon) + \mathbb{1}/4 \quad \forall \boldsymbol{x} \tag{50}$$

where $\tilde{\boldsymbol{x}}$ is generated using the SSN noise $\boldsymbol{s}$ (compare to Equation 23 in the main text).

	$\rho = 0.5$	$\rho = 1.0$	$\rho = 1.5$	$\rho = 2.0$	$\rho = 2.5$	$\rho = 3.0$	$\rho = 3.5$	$\rho = 4.0$
Seed = 0	72.25% (81.50% @ σ=0.75)	63.07% (77.85% @ σ=1.25)	56.21% (71.17% @ σ=2.25)	51.33% (67.98% @ σ=3.0)	46.76% (65.40% @ σ=3.5)	42.66% (65.40% @ σ=3.5)	38.26% (65.40% @ σ=3.5)	33.64% (65.40% @ σ=3.5)
Seed = 1	72.01% (81.85% @ σ=0.75)	62.73% (75.64% @ σ=1.5)	56.03% (72.19% @ σ=2.0)	51.20% (67.65% @ σ=3.0)	46.71% (66.93% @ σ=3.25)	42.45% (66.19% @ σ=3.5)	37.87% (66.19% @ σ=3.5)	33.08% (66.19% @ σ=3.5)
Seed = 2	72.62% (81.19% @ σ=0.75)	62.79% (74.26% @ σ=1.75)	56.06% (70.13% @ σ=2.5)	51.02% (70.13% @ σ=2.5)	46.85% (65.33% @ σ=3.5)	42.52% (65.33% @ σ=3.5)	38.22% (65.33% @ σ=3.5)	33.53% (65.33% @ σ=3.5)

Table 3. Comparison of DSSN using different random seeds to generate $\boldsymbol{v}$ on CIFAR-10. Matching Yang et al. (2020), we test on 15 noise levels ($\sigma \in \{0.15, 0.25n \text{ for } 1 \leq n \leq 14\}$). We report the best certified accuracy at a selection of radii ρ, as well as the clean accuracy and noise level of the associated classifier. We find very little difference between the different seed values, with all certified accuracies within ± 0.65 percentage points of each other.

Proposition 2. *The only pair of equivalent joint distributions* $(\mathcal{D}, \mathcal{S})$ *is* $\mathcal{D} \sim \mathcal{U}^d(-0.5, 0.5)$, $\mathcal{S} \sim \mathcal{U}^d(0, 1)$.

Proof. We first describe a special property of SSN (with λ = 0.5):

Fix a smoothed value $\tilde{\boldsymbol{x}}'$, and let $\mathcal{X}'$ be the set of all inputs $\boldsymbol{x}$ such that $\tilde{\boldsymbol{x}}'$ can be generated from $\boldsymbol{x}$ under *any* joint splitting distribution $\mathcal{S}$. From Figure 2-a in the main text, we can see that this is simply

$$\mathcal{X}' = \{\boldsymbol{x} | \tilde{x}'_i \leq x_i/2 + (1/2) \leq \tilde{x}'_i + (1/2) \quad \forall i\}. \tag{51}$$

Notice that to generate $\tilde{\boldsymbol{x}}'$, *regardless of the value of* $\boldsymbol{x} \in \mathcal{X}'$, the splitting vector $\boldsymbol{s}$ must be exactly the following:

$$s_i = \begin{cases} 2\tilde{x}'_i & \text{if } \tilde{x}'_i < 1/2 \\ 2\tilde{x}'_i - 1 & \text{if } \tilde{x}'_i \geq 1/2 \end{cases} \tag{52}$$

(This is made clear by Figure 1 in the main text.)

If $\boldsymbol{x} \in \mathcal{X}'$, then $\tilde{\boldsymbol{x}}'$ will be generated iff this value of $\boldsymbol{s}$ is chosen. Therefore, given a fixed splitting distribution $\mathcal{S}$, the probability of generating $\tilde{\boldsymbol{x}}'$ must be *constant* for all points in $\mathcal{X}'$.

Now, we compare to uniform additive noise. In order for $\mathcal{D}$ and $\mathcal{S}$ to be equivalent, for the fixed noised point $(\boldsymbol{x} + \epsilon)' = 2\tilde{\boldsymbol{x}'} - \mathbb{1}/2$, it must be the case that all points in $\mathcal{X}'$ are equally likely to generate $(\boldsymbol{x} + \epsilon)'$. But note from Equation 51 that $\mathcal{X}'$ is simply the uniform ℓ_∞ ball of radius 0.5 around $(\boldsymbol{x} + \epsilon)'$. This implies that $\mathcal{D}$ must be the uniform distribution $\mathcal{D} \sim \mathcal{U}^d(-0.5, 0.5)$, which is equivalent to the splitting distribution $\mathcal{S} \sim \mathcal{U}^d(0, 1)$. □

The *only* case when SSN and uniform additive noise can produce similar distributions of noisy samples is when all noise components are independent. This helps us understand how SSN can work with *any* joint distribution of splitting noise, while uniform additive noise has only been shown to produce accurate certificates when all components of ϵ are independent.

F. Tightness of Theorem 2

Here, we discuss the tightness of our certification result. Theorem 2 is tight in the following sense:

Proposition 3. *For any* $\lambda > 0$ *and a random variable* $\boldsymbol{s} \in [0, 2\lambda]^d$*, with any fixed distribution such that:*

$$s_i \sim \mathcal{U}(0, 2\lambda), \quad \forall i, \tag{53}$$

there exists a $f : \mathbb{R}^d \to [0, 1]$*, such that if we define:*

$$\tilde{x}_i := \frac{\min(2\lambda\lceil \frac{x_i - s_i}{2\lambda} \rceil + s_i, 1)}{2} \tag{54}$$

$$+ \frac{\max(2\lambda\lceil \frac{x_i - s_i}{2\lambda} - 1 \rceil + s_i, 0)}{2}, \quad \forall i \tag{55}$$

$$p(\boldsymbol{x}) := \mathop{\mathbb{E}}_{\boldsymbol{s}} [f(\tilde{\boldsymbol{x}})]. \tag{56}$$

then, $p(.)$ *is not* c*-Lipschitz with respect to the* ℓ_1 *norm for any* $c < 1/(2\lambda)$.

In other words, we cannot make the Lipschitz constraint any tighter without some base classifier f providing a counterexample. Note that this result holds for any legal choice of joint distribution of $\boldsymbol{s}$.

Proof. We consider two cases, on λ:

- **Case 1:** $\lambda < 0.5$**:** Consider the following base classifier:

$$f(\tilde{\boldsymbol{x}}) = \begin{cases} 1 & \text{if } \tilde{x}_1 > \lambda \\ 0 & \text{otherwise.} \end{cases} \tag{57}$$

Now, consider the points $\boldsymbol{x} = [0, 0, 0, 0, ...]$ and $\boldsymbol{x}' = [2\lambda, 0, 0, 0, ...]$. Note that $\|\boldsymbol{x}' - \boldsymbol{x}\|_1 = 2\lambda$. From the

definition of $\tilde{\mathbf{v}}$, we have (with probability 1):

$$\tilde{x}_1 = \frac{s_1}{2}$$
$$\tilde{x}'_1 = \min\left(\lambda + \frac{s_1}{2}, \frac{1}{2}\right) + \frac{s_1}{2}$$

Note that this means that, with probability 1, we have $\tilde{x}_1 \leq \lambda$, and therefore $f(\tilde{\boldsymbol{x}}) = 0$. Similarly, with probability 1, $\tilde{x}'_1 > \lambda$, so $f(\tilde{\boldsymbol{x}}') = 1$. Then, for all $c < 1/(2\lambda)$:

$$|p(\boldsymbol{x}) - p(\boldsymbol{x}')| =$$
$$|\mathop{\mathbb{E}}_{\boldsymbol{s}}[f(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{s}}[f(\tilde{\boldsymbol{x}}')]| =$$
$$|0 - 1| = 1 > c \cdot 2\lambda = c\|\boldsymbol{x}' - \boldsymbol{x}\|_1$$

So $p(\cdot)$ is not c-Lipschitz.

- **Case 2:** $\lambda \geq 0.5$**:** Consider the following base classifier:

$$f(\tilde{\boldsymbol{x}}) = \begin{cases} 1 & \text{if } \tilde{x}_1 > 0.5 \\ 0 & \text{otherwise.} \end{cases} \tag{58}$$

Now, consider the points $\boldsymbol{x} = [0, 0, 0, 0, ...]$ and $\boldsymbol{x}' = [1, 0, 0, 0, ...]$. Note that $\|\boldsymbol{x}' - \boldsymbol{x}\|_1 = 1$. From the definition of $\tilde{\mathbf{x}}$, we have (with probability 1):

$$\tilde{x}_1 = \frac{\min(s_1, 1)}{2}$$
$$\tilde{x}'_1 = \frac{\min(s_i, 1) + \mathbf{1}_{s_i<1}}{2}$$

Note that this means that, with probability 1, we have $\tilde{x}_1 \leq 0.5$, and therefore $f(\tilde{\boldsymbol{x}}) = 0$. On the other hand, $\tilde{x}'_1 > 0.5$ iff $s \in (0, 1)$ which occurs with probability $1/(2\lambda)$. Therefore, for all $c < 1/(2\lambda)$:

$$|p(\boldsymbol{x}) - p(\boldsymbol{x}')| =$$
$$|\mathop{\mathbb{E}}_{\boldsymbol{s}}[f(\tilde{\boldsymbol{x}})] - \mathop{\mathbb{E}}_{\boldsymbol{s}}[f(\tilde{\boldsymbol{x}}')]| =$$
$$|0 - 1/(2\lambda)| = 1/(2\lambda) > c = c\|\boldsymbol{x}' - \boldsymbol{x}\|_1$$

So $p(\cdot)$ is not c-Lipschitz.

□

However, the tightness of the global Lipschitz bound on p does not imply that the final *certificate* result, on the minimum possible distance from $\boldsymbol{x}$ to the decision boundary given $p(\boldsymbol{x})$, is necessarily a tight bound.

For simplicity, consider a binary classifier, so that the decision boundary is at $p(\boldsymbol{x}) = 0.5$. The certificate given by our method can be formalized as a function

$$\text{cert}(z) := 2\lambda(z - 0.5), \tag{59}$$

which maps the value of $p(\boldsymbol{x})$ to the certified lower bound on the distance to the decision boundary.

A certificate function can be considered tight if, for all $z \in (0.5, 1.0]$ there exists an $f, \boldsymbol{x}, \boldsymbol{x}'$ such that:

$$\begin{aligned} p(\boldsymbol{x}) &= z \\ \|\boldsymbol{x} - \boldsymbol{x}'\|_1 &= \text{cert}(z) \\ p(\boldsymbol{x}') &= 0.5 \end{aligned} \tag{60}$$

Note that, for example, the well-known smoothing-based ℓ_2 robustness certificate proposed by (Cohen et al., 2019) is tight by the analogous definition.

It turns out that our certificate function is not necessarily tight by this definition. In particular, one can show for some valid choice of λ and joint distribution of $\boldsymbol{s}$ that this definition of tightness does not hold.

For example, consider the case where $s_1 = s_2 = ... = s_d$, and $\lambda > 1$. We discuss this scenario briefly in Section 4.1[6]; recall (Equation 19) that the smoothed classifier must take the form:

$$p(\boldsymbol{x}) = \frac{2\lambda - 1}{2\lambda} f(0.5 \cdot \mathbb{1}) + \frac{1}{2\lambda} \mathop{\mathbb{E}}_{\boldsymbol{s}<1}[f(\tilde{\boldsymbol{x}})], \quad \forall \boldsymbol{x}, \tag{61}$$

which is the sum of a constant, and a function bounded in $[0, 1/(2\lambda)]$. If $z > 0.5 + 1/(2\lambda)$, this implies that $\frac{2\lambda-1}{2\lambda} f(0.5 \cdot \mathbb{1}) > 0.5$, which implies that $p(\cdot) > 0.5$ everywhere. This means that the tightness condition cannot hold for $z \in (0.5 + 1/(2\lambda), 1]$.

G. Complete Certification Data on CIFAR-10 and ImageNet

We provide complete certification results for uniform additive noise, randomized SSN with independent noise, and DSSN, at all tested noise levels on both CIFAR-10 and ImageNet, using both standard and stability training. For CIFAR-10, see Figures 10, 11, 12, and 13. For ImageNet, see Figure 14.

[6] In that section, we discuss this distribution in the quantized case, but the differences is not relevant to our argument here

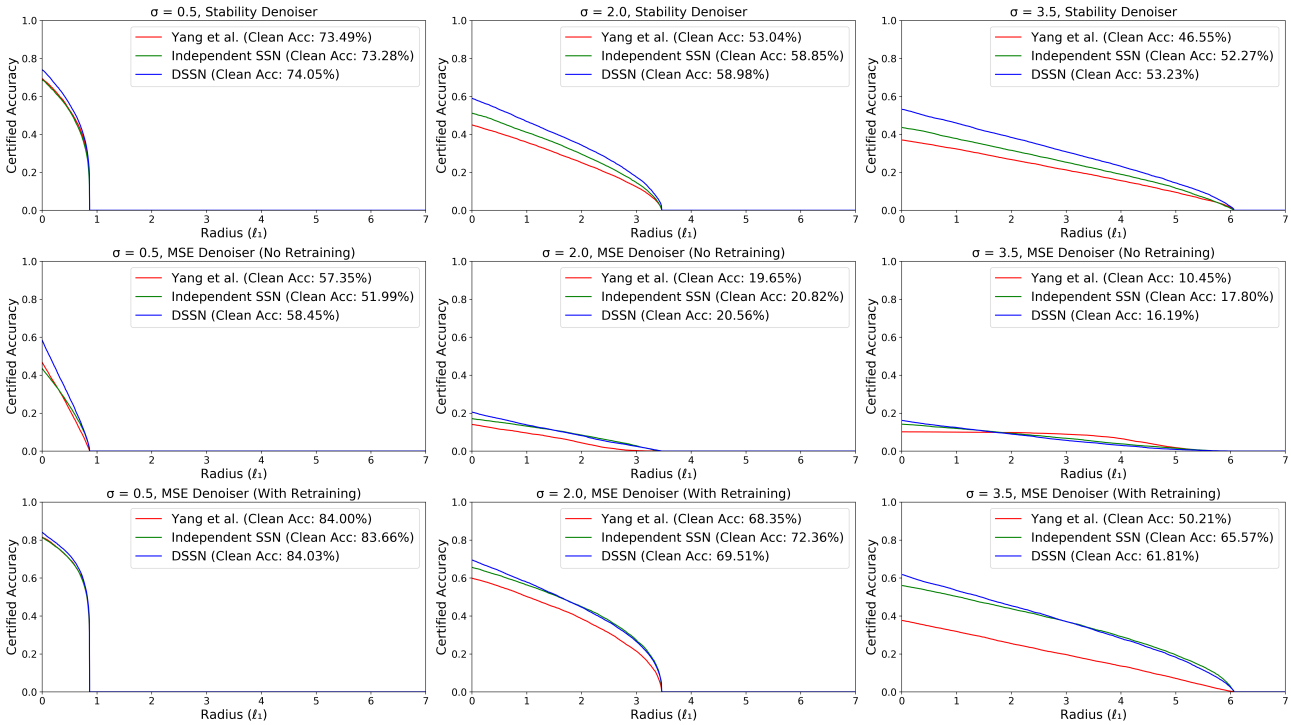

Figure 9. Certified accuracies of models trained with denoisers, for additive uniform noise, SSN with independent noise, and DSSN. See text of Section D for further details on the denoisers used. For $\sigma \geq 2.0$, Independent SSN outperfroms (Yang et al., 2020)'s method, suggesting that the difference in noise representations can not be resolved by using a denoiser. (It may appear as if (Yang et al., 2020)'s method is more robust at large radii for $\sigma = 3.5$ with an MSE denoiser without retraining: however, this is for a classifier with *clean accuracy* $\approx 10\%$, so this is vacuous: similar results can be achieved by simply always returning the same class.)

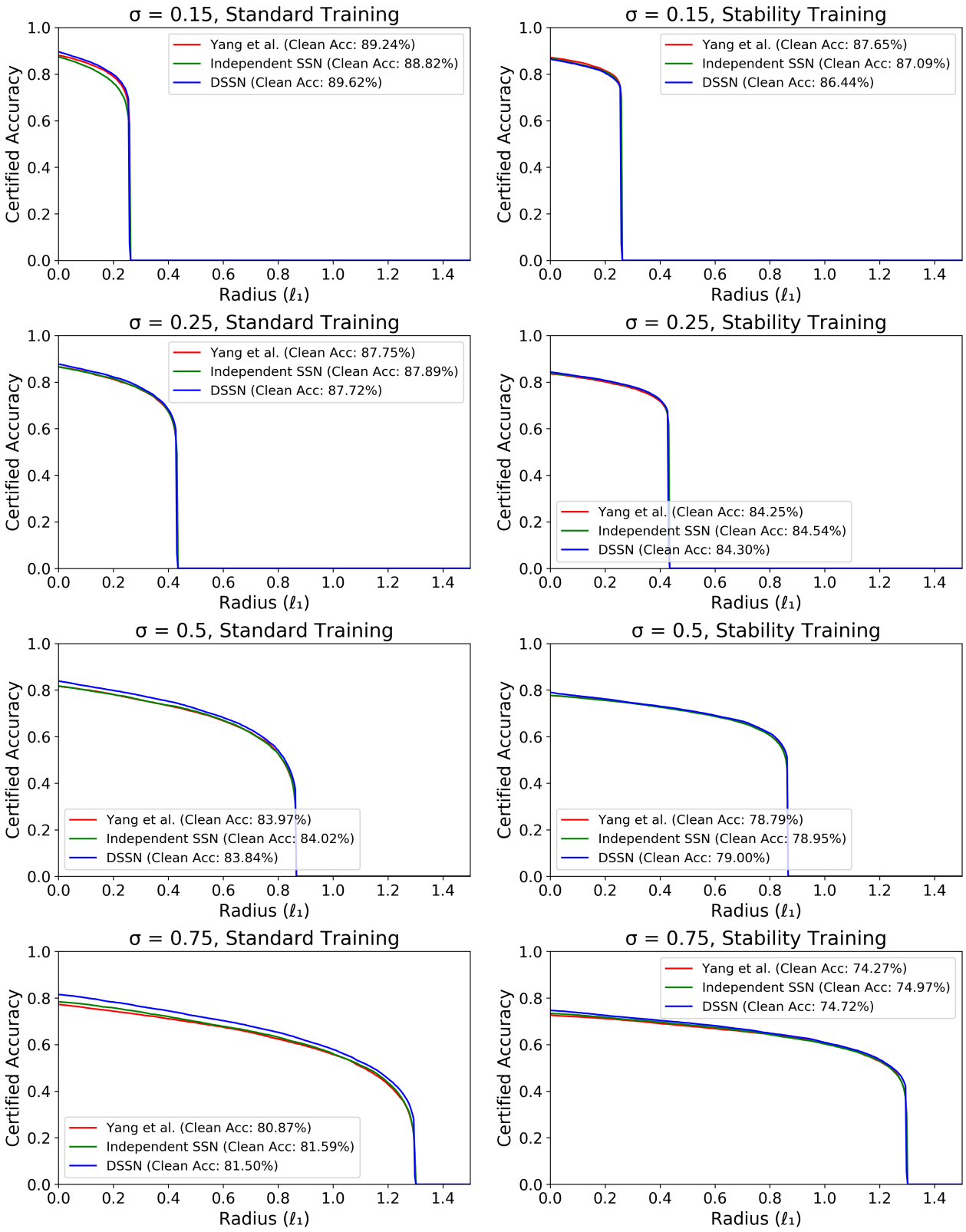

Figure 10. Certification results for CIFAR-10, comparing uniform additive noise, randomized SSN with independent noise, and DSSN, for $\sigma \in \{0.15, 0.25, 0.5, 0.75\}$

.

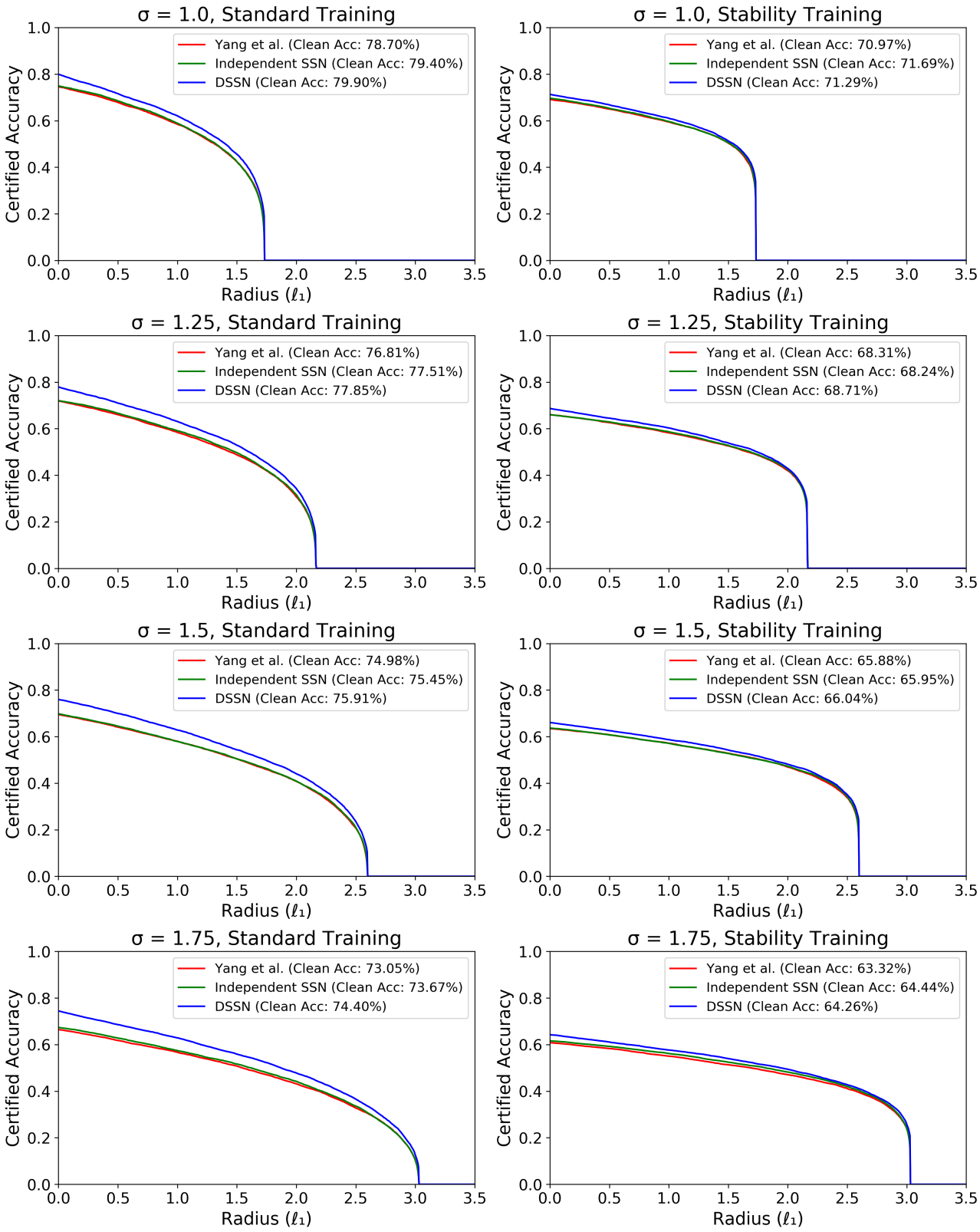

Figure 11. Certification results for CIFAR-10, comparing uniform additive noise, randomized SSN with independent noise, and DSSN, for $\sigma \in \{1.0, 1.25, 1.5, 1.75\}$

.

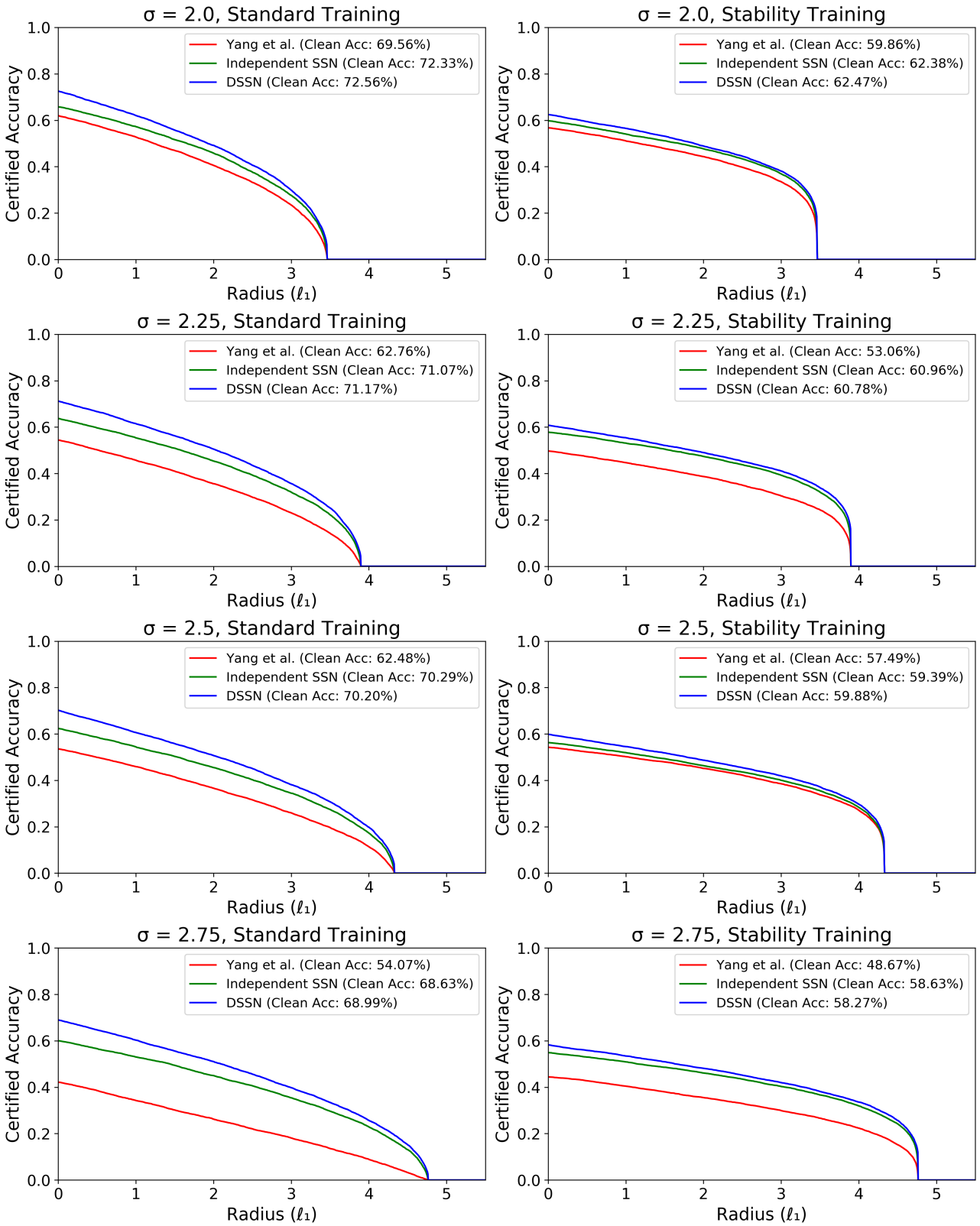

Figure 12. Certification results for CIFAR-10, comparing uniform additive noise, randomized SSN with independent noise, and DSSN, for $\sigma \in \{2.0, 2.25, 2.5, 2.75\}$

.

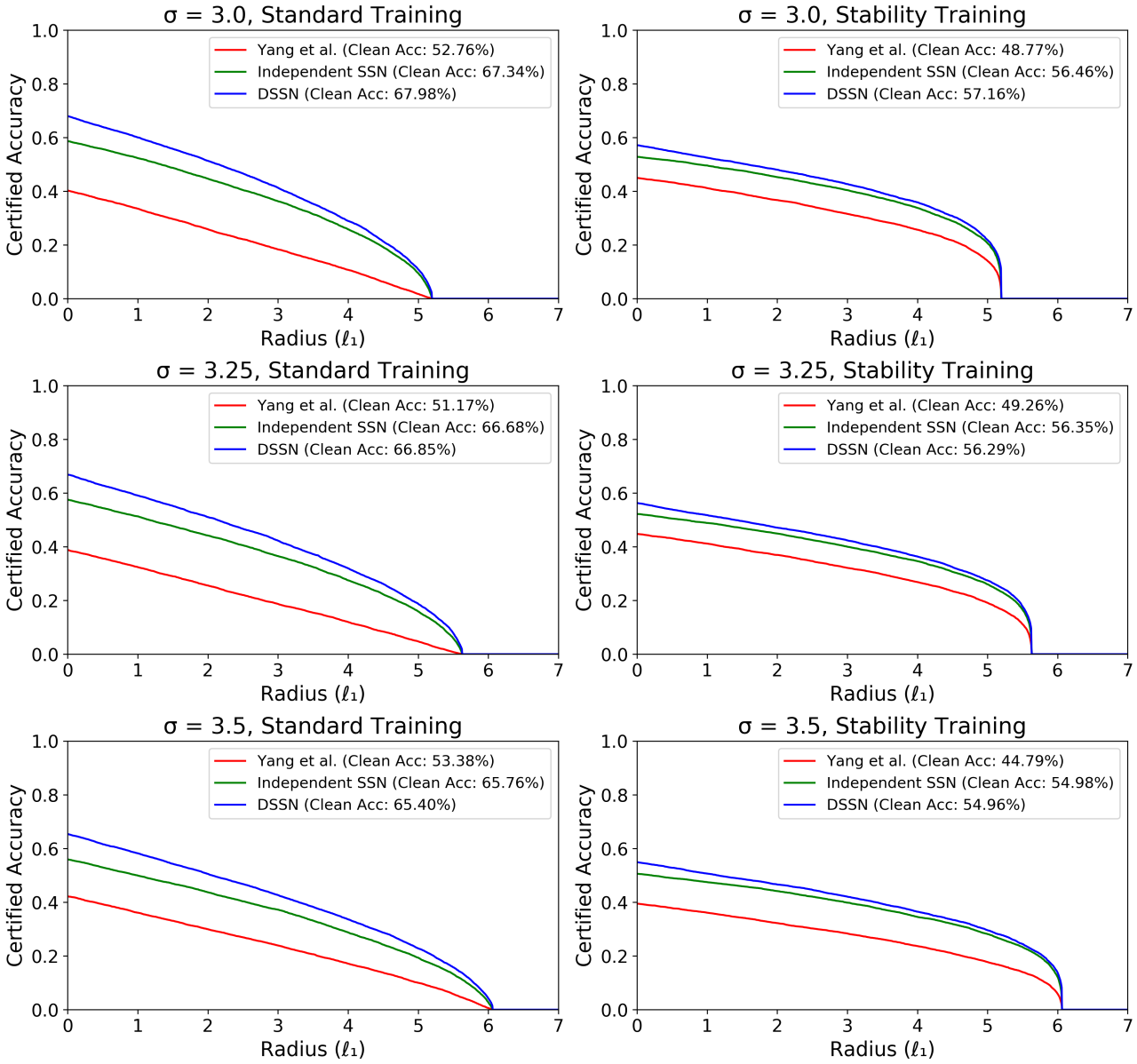

Figure 13. Certification results for CIFAR-10, comparing uniform additive noise, randomized SSN with independent noise, and DSSN, for $\sigma \in \{3.0, 3.25, 3.5\}$

.

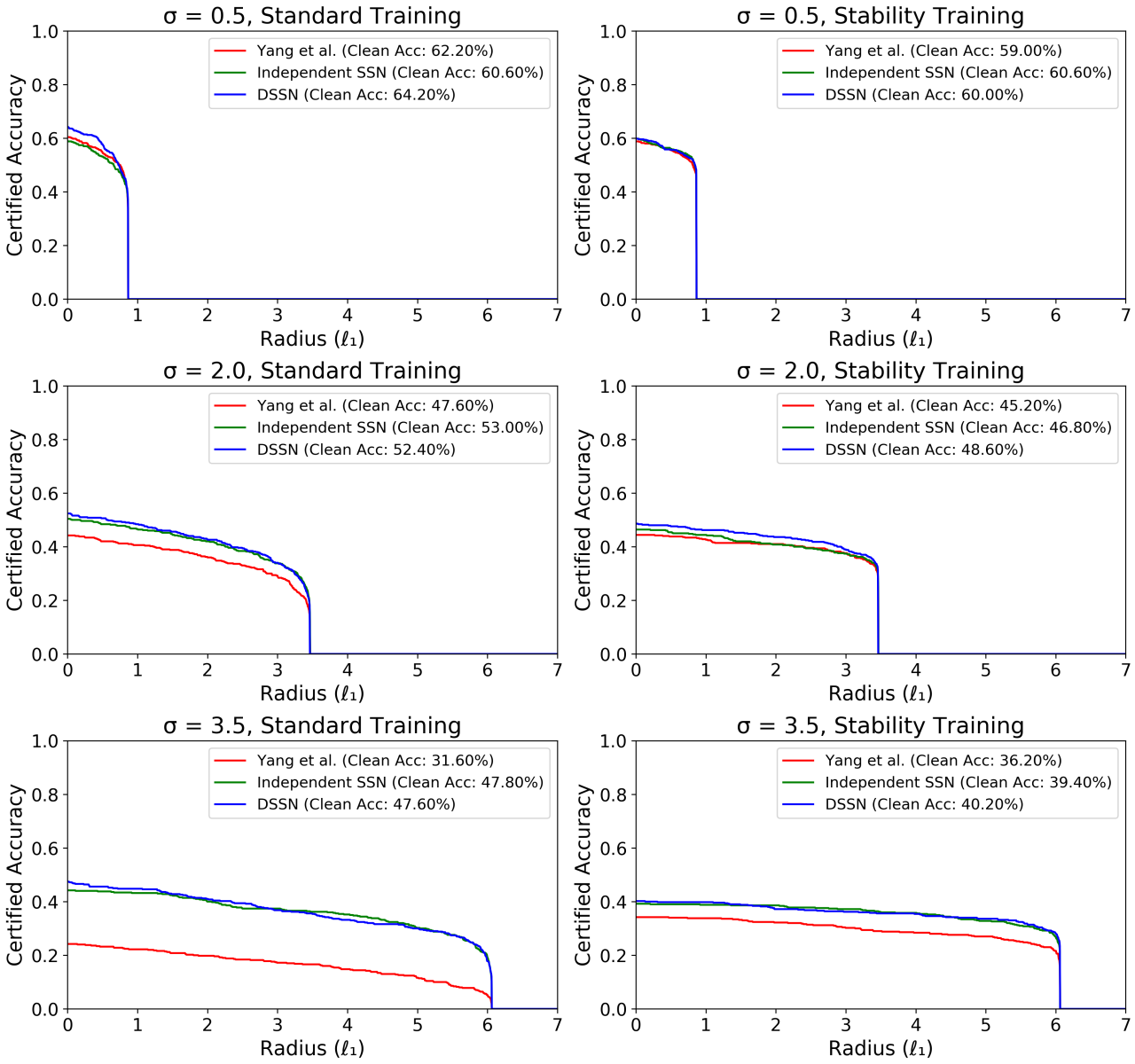

Figure 14. Certification results for ImageNet, comparing uniform additive noise, randomized SSN with independent noise, and DSSN, for $\sigma \in \{0.5, 2.0, 3.5\}$. Note that we see less improvement in reported certified accuracies due to derandomization (i.e., less difference between Independent SSN and DSSN) in ImageNet compared to in CIFAR-10, particularly at large noise levels.

.