SUPPLEMENTARY MATERIALS: ABBA Neural Networks: Coping with Positivity, Expressivity, and Robustness *

Ana Neacșu †, Jean-Christophe Pesquet ‡, Vlad Vasilescu †, and Corneliu Burileanu †

SM1. Symmetric activation functions. In practice, the activation operator R_i is often separable, that is it operates componentwise:

$$(\forall x = (\xi_k)_{1\leqslant k\leqslant N_i} \in \mathbb{R}^{N_i}) \quad R_i x = \big(\varrho_i(\xi_k)\big)_{1\leqslant k\leqslant N_i}, \tag{SM1.1}$$

where, for every $k \in \{1,\dots,N_i\}$, $\varrho_i\colon \mathbb{R}\to\mathbb{R}$. Examples of odd functions allowing us to define a symmetric separable activation operators R_i with $c_i = d_i = 0$ are

- the hyperbolic tangent activation function $\rho_i = \tanh$
- the arctangent activation function $\rho_i = (2/\pi)\arctan$
- the inverse square root linear unit function $\varrho_i\colon \mathbb{R}\to\mathbb{R}\colon \xi \mapsto \xi/\sqrt{1+\xi^2}$
- the Elliot activation function $\varrho_i\colon \mathbb{R}\to\mathbb{R}\colon \xi\mapsto \xi/(1+|\xi|)$.

Some examples of separable activation operators which are non-odd are described below. The capped ReLU function is given by

$$(\forall \xi\in\mathbb{R}) \quad \rho_i(\xi) = \begin{cases} 0 & \text{if } \xi < 0 \\ \xi & \text{if } 0 \leqslant \xi < \chi \\ \chi & \text{otherwise,} \end{cases} \tag{SM1.2}$$

where $\chi \in]0,+\infty[$. We have then $c_i = d_i = \chi 1_{N_i}$ with $1_{N_i} = [1,\dots,1]^\top \in \mathbb{R}^{N_i}$. We can also define a leaky version of this function as

$$(\forall \xi\in\mathbb{R}) \qquad \rho_i(\xi) = \begin{cases} \alpha\xi & \text{if} \quad \xi < 0, \\ \xi & \text{if} \quad 0 \leqslant \xi < \chi, \\ \alpha(\xi-\chi)+\chi & \text{otherwise,} \end{cases} \tag{SM1.3}$$

where $\alpha \in]0,1[$ and $\chi \in]0,+\infty[$ are hyper-parameters.

SM2. Proof of the properties of ABBA matrices. (i)-(iii): These properties follow from basic algebra. We will just detail the proof of the third one. Let

$$M_1 = \begin{bmatrix} A_1 & B_1 \\ B_1 & A_1 \end{bmatrix} \quad \text{and} \quad M_2 = \begin{bmatrix} A_2 & B_2 \\ B_2 & A_2 \end{bmatrix}, \tag{SM2.1}$$

where $(A_1, B_1) \in \mathbb{R}^{N_2\times N_1}$ and $(A_2, B_2) \in \mathbb{R}^{N_3\times N_2}$. Then

$$M_2 M_1 = \begin{bmatrix} A_2A_1 + B_2B_1 & A_2B_1 + B_2A_1 \\ A_2B_1 + B_2A_1 & A_2A_1 + B_2B_1 \end{bmatrix} \in \mathcal{A}_{N_3,N_1}. \tag{SM2.2}$$

*Part of this work was supported by the French ANR Research and Teaching Chair in Artificial Intelligence BRIDGEABLE.

†Speech and Dialogue Laboratory, University Politehnica of Bucharest, Romania (ana_antonia.neacsu@upb.ro).
‡Centre de Vision Numérique, Inria, CentraleSupélec, Gif-sur-Yvette, France.

In addition,

$$\begin{aligned}\mathfrak{S}(M_2M_1) &= A_2A_1 + B_2B_1 + A_2B_1 + B_2A_1\\ &= (A_2+B_2)(A_1+B_1)\\ &= \mathfrak{S}(M_2)\mathfrak{S}(M_1).\end{aligned} \tag{SM2.3}$$

(iv): This property is a direct consequence of (ii) and (iii).

(v): Let $M = \begin{bmatrix} A & B \\ B & A \end{bmatrix} \in \mathbb{R}^{(2N_1)\times(2N_1)}$. $\lambda \in \mathbb{C}$ is an eigenvalue of M if and only if

$$\det(M - \lambda \operatorname{Id}) = 0 \quad \Leftrightarrow \quad \det\left(\begin{bmatrix} A-\lambda\operatorname{Id} & B \\ B & A-\lambda\operatorname{Id}\end{bmatrix}\right) = 0. \tag{SM2.4}$$

We have

$$\begin{bmatrix} A-\lambda\operatorname{Id} & B \\ B & A-\lambda\operatorname{Id}\end{bmatrix}\begin{bmatrix} \operatorname{Id} & -\operatorname{Id} \\ \operatorname{Id} & \operatorname{Id}\end{bmatrix} = \begin{bmatrix} A+B-\lambda\operatorname{Id} & -A+B+\lambda\operatorname{Id} \\ A+B-\lambda\operatorname{Id} & A-B-\lambda\operatorname{Id}\end{bmatrix}. \tag{SM2.5}$$

Since $A-B-\lambda\operatorname{Id}$ and $-A+B+\lambda\operatorname{Id}$ commute, we have [SM5]

$$\det\left(\begin{bmatrix} A+B-\lambda\operatorname{Id} & -A+B+\lambda\operatorname{Id} \\ A+B-\lambda\operatorname{Id} & A-B-\lambda\operatorname{Id}\end{bmatrix}\right) = 2^N \det\big((A+B-\lambda\operatorname{Id})(A-B-\lambda\operatorname{Id})\big). \tag{SM2.6}$$

Similarly

$$\det\left(\begin{bmatrix} \operatorname{Id} & -\operatorname{Id} \\ \operatorname{Id} & \operatorname{Id}\end{bmatrix}\right) = 2^N. \tag{SM2.7}$$

We deduce from (SM2.5) that

$$\begin{aligned}&\det\left(\begin{bmatrix} A-\lambda\operatorname{Id} & B \\ B & A-\lambda\operatorname{Id}\end{bmatrix}\right) = \det\big((A+B-\lambda\operatorname{Id})(A-B-\lambda\operatorname{Id})\big)\\ \Leftrightarrow\quad &\det(M-\lambda\operatorname{Id}) = \det(A+B-\lambda\operatorname{Id})\,\det(A-B-\lambda\operatorname{Id}).\end{aligned} \tag{SM2.8}$$

So λ is an eigenvalue of M if and only if $\det(A+B-\lambda\operatorname{Id}) = 0$ or $\det(A-B-\lambda\operatorname{Id}) = 0$, i.e., λ is an eigenvalue of $A+B$ or $A-B$.

(vi) Let M be defined similarly to previously with $(A,B) \in (\mathbb{R}^{N_2\times N_1})^2$. We have

$$\|M\| = \|MM^\top\|^{1/2} = \left\|\begin{bmatrix} AA^\top+BB^\top & AB^\top+BA^\top \\ AB^\top+BA^\top & AA^\top+BB^\top\end{bmatrix}\right\|^{1/2} \tag{SM2.9}$$

According to (v), the eigenvalues of $MM^\top \in \mathcal{A}_{N_2,N_2}$ are those of $AA^\top+BB^\top+AB^\top+BA^\top = (A+B)(A+B)^\top$ and $AA^\top+BB^\top-AB^\top-BA^\top = (A-B)(A-B)^\top$. The maximum eigenvalues of the two latter matrices are $\|A+B\|^2$ and $\|A-B\|^2$, respectively. Therefore

$\|M\| = \max\{\|A+B\|, \|A-B\|\}$.
(vii): In addition, if A and B have nonnegative elements,

$$\begin{aligned}\|A-B\| &= \sup_{x\in\mathbb{R}^N\setminus\{0\}} \frac{\|Ax-Bx\|}{\|x\|}\\ &\leqslant \sup_{x\in\mathbb{R}^N\setminus\{0\}} \frac{\||A|x|+B|x|\|}{\|x\|}\\ &= \sup_{a\in[0,+\infty[^N\setminus\{0\}} \frac{\|Aa+Ba\|}{\|a\|}\\ &\leqslant \|A+B\|, \end{aligned} \tag{SM2.10}$$

where $|x|$ denotes the vector whose components are the absolute values of those of vector x.
We deduce from (vi) that $\|M\| = \|A+B\| = \|\mathfrak{S}(M)\|$.
(viii): We have

$$A+B = \sum_{k=1}^{K} \lambda_k u_k v_k^\top \tag{SM2.11}$$

$$A-B = \sum_{k=1}^{K} \mu_k t_k w_k^\top. \tag{SM2.12}$$

Thus

$$A = \frac{1}{2}\sum_{k=1}^{K} (\lambda_k u_k v_k^\top + \mu_k t_k w_k^\top) \tag{SM2.13}$$

$$B = \frac{1}{2}\sum_{k=1}^{K} (\lambda_k u_k v_k^\top - \mu_k t_k w_k^\top) \tag{SM2.14}$$

and we deduce that

$$\begin{aligned}\begin{bmatrix} A & B \\ B & A \end{bmatrix} &= \sum_{k=1}^{K} \frac{1}{2}\left(\lambda_k \begin{bmatrix} u_k v_k^\top & u_k v_k^\top \\ u_k v_k^\top & u_k v_k^\top \end{bmatrix} + \mu_k \begin{bmatrix} t_k w_k^\top & -t_k w_k^\top \\ -t_k w_k^\top & t_k w_k^\top \end{bmatrix}\right)\\ &= \sum_{k=1}^{K} \frac{1}{2}\left(\lambda_k \begin{bmatrix} u_k \\ u_k \end{bmatrix}\begin{bmatrix} v_k \\ v_k \end{bmatrix}^\top + \mu_k \begin{bmatrix} t_k \\ -t_k \end{bmatrix}\begin{bmatrix} w_k \\ -w_k \end{bmatrix}^\top\right). \end{aligned} \tag{SM2.15}$$

On the other hand, for every $(k,\ell) \in \{1,\dots,K\}^2$,

$$\begin{bmatrix} u_k \\ u_k \end{bmatrix}^\top \begin{bmatrix} u_\ell \\ u_\ell \end{bmatrix} = 2u_k^\top u_\ell = \begin{cases} 2 & \text{if } k=\ell \\ 0 & \text{otherwise,} \end{cases}$$

$$\begin{bmatrix} t_k \\ -t_k \end{bmatrix}^\top \begin{bmatrix} t_\ell \\ -t_\ell \end{bmatrix} = 2t_k^\top t_\ell = \begin{cases} 2 & \text{if } k=\ell \\ 0 & \text{otherwise,} \end{cases}$$

$$\begin{bmatrix} u_k \\ u_k \end{bmatrix}^\top \begin{bmatrix} t_\ell \\ -t_\ell \end{bmatrix} = 0, \tag{SM2.16}$$

which shows that $\left\{ \frac{1}{\sqrt{2}} \begin{bmatrix} u_k \\ u_k \end{bmatrix}, \frac{1}{\sqrt{2}} \begin{bmatrix} t_k \\ -t_k \end{bmatrix} \right\}_{1 \leqslant k \leqslant K}$ is an orthonormal family of $\mathbb{R}^{2N_2}$. For similar reasons, $\left\{ \frac{1}{\sqrt{2}} \begin{bmatrix} v_k \\ v_k \end{bmatrix}, \frac{1}{\sqrt{2}} \begin{bmatrix} w_k \\ -w_k \end{bmatrix} \right\}_{1 \leqslant k \leqslant K}$ is an orthonormal family of $\mathbb{R}^{2N_1}$. This allows us to conclude that (SM2.15) provides a singular value decomposition of $\begin{bmatrix} A & B \\ B & A \end{bmatrix}$.

(ix): The rank of $\begin{bmatrix} A & B \\ B & A \end{bmatrix}$ is equal to the number of its nonzero singular values. From the previous result, it is thus equal to the sum of the nonzero values of $A+B$ and those of $A-B$, that is the sum of the ranks of matrices $A+B$ and $A-B$.

(x): The fact that the ABBA structure is kept by matrix mappings operating elementwise is obvious. Let us thus focus on the case of spectral functions. By using the same notation as in (viii), it follows from (SM2.15) that

(SM2.17)
$$f\Big(\begin{bmatrix} A & B \\ B & A \end{bmatrix} \Big) = \sum_{k=1}^{K} \frac{1}{2} \left(\varphi(\lambda_k) \begin{bmatrix} u_k v_k^\top & u_k v_k^\top \\ u_k v_k^\top & u_k v_k^\top \end{bmatrix} + \varphi(\mu_k) \begin{bmatrix} t_k w_k^\top & -t_k w_k^\top \\ -t_k w_k^\top & t_k w_k^\top \end{bmatrix} \right) = \begin{bmatrix} \tilde{A} & \tilde{B} \\ \tilde{B} & \tilde{A} \end{bmatrix},$$

where

$$\tilde{A} + \tilde{B} = \sum_{k=1}^{N} \varphi(\lambda_k) u_k v_k^\top$$

(SM2.18)
$$\tilde{A} - \tilde{B} = \sum_{k=1}^{N} \varphi(\mu_k) t_k w_k^\top.$$

(xi): By using the same notation as in (3.6), The best approximation of rank less than or equal to R to a matrix M_0 in $\mathbb{R}^{(2N_2)\times(2N_1)}$ is $f(M_0)$ where f is given by (3.6) with

(SM2.19)
$$(\forall \lambda \in \mathbb{R}_+) \quad \varphi(\lambda) = \begin{cases} \lambda & \text{if } \lambda \leqslant \tilde{\lambda}_{0,[R]} \\ 0 & \text{otherwise,} \end{cases}$$

and $\tilde{\lambda}_{0,[R]}$ is the R-th eigenvalue of M_0 when these are ordered by decreasing value: $\tilde{\lambda}_{0,1} \geqslant \ldots \geqslant \tilde{\lambda}_{0,K}$. It thus follows from (x) that if $M_0 \in \mathcal{A}_{N_2,N_1}$, then $f(M_0) \in \mathcal{A}_{N_2,N_1}$.

(xii): The projection onto the spectral ball of center 0 and and radius $\rho \in]0, +\infty[$ of a matrix $M \in \mathcal{A}_{N_2,N_1}$ is given by (3.6) where

$$(\forall \xi \in \mathbb{R}) \qquad \varphi(\xi) = \min\{\xi, \rho\}.$$

The result then follows from Property (x).

Remark SM2.1. The last result can be generalized as follows. Let $\psi \colon \mathbb{R} \to]-\infty, +\infty]$ be a lower-semicontinuous function, which is proper, even, and convex, and let

$$g \colon \mathbb{R}^{(2N_2)\times(2N_1)} \to]-\infty, +\infty]$$

(SM2.20)
$$M \mapsto \sum_{i=1}^{2K} \psi(\tilde{\lambda}_k)$$

where $K = \min\{N_1, N_2\}$ and $(\tilde{\lambda}_k)_{1\leqslant k\leqslant 2K}$ are the singular values of M. The proximity operator of g at $M \in \mathbb{R}^{(2N_2)\times(2N_1)}$ is [SM1, Proposition 24.68]:

$$\begin{aligned}\operatorname{prox}_g \colon M \mapsto & \operatorname*{argmin}_{P\in\mathbb{R}^{(2N_2)\times(2N_1)}} \frac{1}{2}\|P-M\|_{\mathrm{F}}^2 + g(P) \\ & = \sum_{k=1}^{2K} \operatorname{prox}_{\psi}(\tilde{\lambda}_k)\tilde{u}_k\tilde{v}_k^\top, \end{aligned} \tag{SM2.21}$$

where $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm. It then follows from Property (x) that, if $M \in \mathcal{A}_{N_2,N_1}$, then $\operatorname{prox}_g(M) \in \mathcal{A}_{N_2,N_1}$.

SM3. Link between Conv layers and MIMO systems. To be rigorous, let us first define the space $\mathcal{H}_{i-1}$ (resp. $\mathcal{H}_i$) in which signals $(x_p)_{1\leqslant p\leqslant\zeta_{i-1}}$ (resp. $(y_q)_{1\leqslant q\leqslant\zeta_i}$) used in (4.1) live. Typically, $\mathcal{H}_i$ is some finite-dimensional subspace of $(\ell^2(\mathbb{Z}^d))^{\zeta_i}$ where $\ell^2(\mathbb{Z}^d)$ denotes the space of square summable discrete d-dimensional fields. For the discrete convolution $*$ to be properly defined, kernels $(w_{i,q,p})_{1\leqslant p\leqslant\zeta_{i-1},1\leqslant q\leqslant\zeta_i}$ are then assumed to be summable. In practice, this assumption is satisfied since these kernels are chosen with finite size.

For $x = \big(x(\mathbf{n})\big)_{\mathbf{n}\in\mathbb{Z}^d} \in \ell^2(\mathbb{Z}^d)$, the decimation operation $(\cdot)\downarrow_{s_i}$ returns the output signal

$$(\forall \mathbf{n}\in\mathbb{Z}^d) \quad y(\mathbf{n}) = u(s_i\mathbf{n}). \tag{SM3.1}$$

Eq. (4.1) defines a MIMO (multi-input multi-output) filter that can be rexpressed in a matrix form as

$$\begin{aligned}(\forall \mathbf{n}\in\mathbb{Z}^d) \quad \mathbf{u}(\mathbf{n}) &= \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{W}_i(\mathbf{n}')\,\mathbf{x}(\mathbf{n}-\mathbf{n}') \\ &= (\boldsymbol{W}_i * \mathbf{x})(\mathbf{n}), \end{aligned} \tag{SM3.2}$$

where

$$\mathbf{u}(\mathbf{n}) = \begin{bmatrix} u_1(\mathbf{n}) \\ \vdots \\ u_{\zeta_i}(\mathbf{n}) \end{bmatrix} \in \mathbb{R}^{\zeta_j}, \quad \mathbf{x}(\mathbf{n}) = \begin{bmatrix} x_1(\mathbf{n}) \\ \vdots \\ x_{\zeta_{i-1}}(\mathbf{n}) \end{bmatrix} \in \mathbb{R}^{\zeta_{i-1}}, \tag{SM3.3}$$

and $\boldsymbol{W}_i(\mathbf{n})$ is given by (4.4). $(\boldsymbol{W}_i(\mathbf{n}))_{\mathbf{n}\in\mathbb{Z}^d}$ defines the so-called MIMO impulse response of $\mathcal{W}_i$. The MIMO impulse response of an ABBA layer is similarly given by (4.5).

These relations can also be written more concisely in the d-dimensional frequency domain[1] as

$$(\forall \boldsymbol{\nu}\in[0,1]^d) \quad \widehat{\mathbf{u}}(\boldsymbol{\nu}) = \widehat{\boldsymbol{W}}_i(\boldsymbol{\nu})\,\widehat{\mathbf{x}}(\boldsymbol{\nu}), \tag{SM3.4}$$

where

$$\widehat{\mathbf{x}}(\boldsymbol{\nu}) = \sum_{\mathbf{n}\in\mathbb{Z}^d} \mathbf{x}(\mathbf{n}) \exp\big(-\imath 2\pi \mathbf{n}^\top\boldsymbol{\nu}\big) \in \mathbb{C}^{\zeta_{j-1}}, \tag{SM3.5}$$

[1] Alternatively, we could use the d-dimensional z-transform since we are dealing with discrete-space signals.

(SM3.6) $$\widehat{\boldsymbol{W}}_i(\boldsymbol{\nu}) = \sum_{\mathbf{n}\in\mathbb{Z}^d} \boldsymbol{W}_i(\mathbf{n}) \exp\big(-\imath 2\pi \mathbf{n}^\top \boldsymbol{\nu}\big) \in \mathbb{C}^{\zeta_{j-1}\times\zeta_j},$$

and $\widehat{\boldsymbol{W}}_i$ is the frequency response of the associated MIMO filter.

Note that $\int_{[0,1]^d} \|\widehat{\mathbf{x}}(\boldsymbol{\nu})\|^2 d\boldsymbol{\nu} < +\infty$, whereas $\widehat{\boldsymbol{W}}_i$ is a continuous (hence bounded) function on $[0,1]^d$. Another useful result from sampling theory [SM6] is that the Fourier transform of $\mathbf{y} = (y_q)_{1\leqslant q\leqslant\zeta_j}$ in (4.1) is deduced from the Fourier transform of $\mathbf{u}$ by the relation

(SM3.7) $$(\forall \boldsymbol{\nu}\in[0,1]^d) \quad \widehat{\mathbf{y}}(\boldsymbol{\nu}) = \frac{1}{s_i^d} \sum_{\mathbf{j}\in\mathbb{S}(s_i)} \widehat{\mathbf{u}}\left(\frac{\boldsymbol{\nu}+\mathbf{j}}{s_i}\right).$$

where

(SM3.8) $$(\forall \sigma\in\mathbb{N}\setminus\{0\}) \quad \mathbb{S}(\sigma) = \{0,\dots,\sigma-1\}^d.$$

It is also worth noting that the interpolation by a factor s of $\mathbf{y}$

(SM3.9) $$\mathbf{v} = \mathbf{y}_{\uparrow_s} \quad \Leftrightarrow \quad (\forall \mathbf{n}\in\mathbb{Z}^d) \quad \mathbf{v}(\mathbf{n}) = \begin{cases} \mathbf{y}\left(\dfrac{\mathbf{n}}{s}\right) & \text{if } \mathbf{n}\in s\,\mathbb{Z}^d \\ 0 & \text{otherwise,} \end{cases}$$

translates into

(SM3.10) $$(\forall \nu\in[0,1]) \quad \widehat{\mathbf{y}}_{\uparrow_s}(\boldsymbol{\nu}) = \widehat{\mathbf{y}}(s\boldsymbol{\nu}),$$

in the frequency domain.

SM4. Frequency expressions of Lipschitz bounds. In this appendix, we establish frequency-based bounds of the Lipschitz constant of an m-layer convolutional neural network T.

Based on the MIMO concepts introduced in Appendix SM3, we will introduce the following global frequency response of the network:

(SM4.1) $$(\forall \boldsymbol{\nu}\in[0,1]^d) \quad \widehat{\boldsymbol{W}}(\boldsymbol{\nu}) = \widehat{\boldsymbol{W}}_m(\sigma_{m-1}\boldsymbol{\nu})\cdots\widehat{\boldsymbol{W}}_2(\sigma_1\boldsymbol{\nu})\widehat{\boldsymbol{W}}_1(\boldsymbol{\nu}) \in \mathbb{C}^{\zeta_m\times\zeta_0},$$

where $\widehat{\boldsymbol{W}}_i$ is the frequency response associated to filter $\boldsymbol{W}_i$ (see (SM3.6)).

We have then the following result providing a frequency formula for evaluating the Lipschitz constant of a convolutional network.

Proposition SM4.1. *The quantity*

(SM4.2) $$\theta_m = \frac{1}{\sigma_m^{d/2}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\| \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \widehat{\boldsymbol{W}}\left(\boldsymbol{\nu}+\frac{\mathbf{j}}{\sigma_m}\right) \widehat{\boldsymbol{W}}\left(\boldsymbol{\nu}+\frac{\mathbf{j}}{\sigma_m}\right)^{\mathsf{H}} \right\|^{1/2}.$$

provides a lower bound on the Lipschitz constant estimate of network T [2]*. In addition, if for every $i \in \{1,\dots,m\}$, $p \in \{1,\dots,\zeta_{i-1}\}$, and $q \in \{1,\dots,\zeta_i\}$, $w_{i,q,p} = (w_{i,q,p}(\mathbf{n}))_{\mathbf{n}\in\mathbb{Z}^d}$ is a nonnegative kernel i.e.,*

$$(\forall \mathbf{n} \in \mathbb{Z}^d) \quad w_{i,p,q}(\mathbf{n}) \geqslant 0, \tag{SM4.3}$$

then θ_m is a Lipschitz constant of T.

Proof. In the considered case all activation operators are nonexpansive and they are assumed separable, except maybe at the last layer. Thus T is a special case of the networks investigated in [SM3, Section 5] for which a tight estimate of the Lipschitz constant was provided. It then follows from [SM3, Theorem 5.2] that a lower bound on this Lipschitz constant estimate is

$$\theta_m = \|\mathcal{W}_m \circ \cdots \circ \mathcal{W}_1\|. \tag{SM4.4}$$

In addition, under the additional assumption that all the kernels are nonnegative, T is an instance of the positively weighted networks investigated in [SM3, Section 5.3] and it follows from [SM3, Proposition 5.10] that θ_m is then a Lipschitz constant of T.
So the problem is to calculate the norm of the linear operator $\mathcal{W} = \mathcal{W}_m \circ \cdots \circ \mathcal{W}_1$. Each operator $\mathcal{W}_i$ with $i \in \{1,\dots,m\}$ is the composition of a d-dimensional MIMO filter with a decimator. It follows from Noble identities [SM6] that $\mathcal{W}$ reduces to cascading a $\zeta_m \times \zeta_0$ MIMO filter with frequency response $\widetilde{\boldsymbol{W}}$ with a decimation of each output by a factor σ_m. More precisely, if $\mathbf{x} \in \mathcal{H}_0$ is the input of this linear system and $\mathbf{y}$ its output, we have in the frequency domain:

$$\begin{aligned}(\forall \boldsymbol{\nu} \in [0,1]^d) \quad \widehat{\mathbf{y}}(\boldsymbol{\nu}) &= \frac{1}{\sigma_m^d} \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \widehat{\boldsymbol{W}}\left(\frac{\boldsymbol{\nu}+\mathbf{j}}{\sigma_m}\right) \widehat{\mathbf{x}}\left(\frac{\boldsymbol{\nu}+\mathbf{j}}{\sigma_m}\right) \\ &= \frac{1}{\sigma_m^d} \widetilde{\boldsymbol{W}}\left(\frac{\boldsymbol{\nu}}{\sigma_m}\right) \widetilde{\mathbf{x}}\left(\frac{\boldsymbol{\nu}}{\sigma_m}\right), \end{aligned} \tag{SM4.5}$$

where $\widetilde{\mathbf{x}}\left(\frac{\boldsymbol{\nu}}{\sigma_m}\right)$ is a vector of dimension $\mathbb{C}^{\sigma_m^d \zeta_0}$ where the vectors $\left(\widehat{\mathbf{x}}((\boldsymbol{\nu}+\mathbf{j})/\sigma_m)\right)_{\mathbf{j}\in\mathbb{S}(\sigma_m)}$ are stacked columnwise and $\widetilde{\boldsymbol{W}}\left(\frac{\boldsymbol{\nu}}{\sigma_m}\right)$ is a $c_m \times \sigma_m^d \zeta_0$ matrix where the matrices $\left(\widehat{\boldsymbol{W}}((\boldsymbol{\nu}+\mathbf{j})/\sigma_m)\right)_{\mathbf{j}\in\mathbb{S}(\sigma_m)}$ are stacked rowwise. For example, when $d = 2$, we have, for every $\boldsymbol{\nu} =$

[2] $(\cdot)^{\mathsf{H}}$ denotes the Hermitian transpose operation.

$(\nu_1,\nu_2)\in[0,1]^2$,

$$\widetilde{\mathbf{x}}(\boldsymbol{\nu}) = \begin{bmatrix} \check{\mathbf{x}}(\nu_1,\nu_2) \\ \check{\mathbf{x}}\Big(\nu_1,\nu_2+\frac{1}{\sigma_m}\Big) \\ \vdots \\ \check{\mathbf{x}}\Big(\nu_1,\nu_2+\frac{\sigma_m-1}{\sigma_m}\Big) \end{bmatrix} \in \mathbb{C}^{\sigma_m^2\zeta_0} \tag{SM4.6}$$

$$\check{\mathbf{x}}(\boldsymbol{\nu}) = \begin{bmatrix} \widehat{\mathbf{x}}(\nu_1,\nu_2) \\ \widehat{\mathbf{x}}\Big(\nu_1+\frac{1}{\sigma_m},\nu_2\Big) \\ \vdots \\ \widehat{\mathbf{x}}\Big(\nu_1+\frac{\sigma_m-1}{\sigma_m},\nu_2\Big) \end{bmatrix} \in \mathbb{C}^{\sigma_m\zeta_0} \tag{SM4.7}$$

$$\widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) = \left[\check{\boldsymbol{W}}(\nu_1,\nu_2)\ \ \check{\boldsymbol{W}}\Big(\nu_1,\nu_2+\frac{1}{\sigma_m}\Big)\ \ \ldots\ \ \check{\boldsymbol{W}}\Big(\nu_1,\nu_2+\frac{\sigma_m-1}{\sigma_m}\Big)\right] \in \mathbb{C}^{\zeta_m\times\sigma_m^2\zeta_0} \tag{SM4.8}$$

$$\check{\boldsymbol{W}}(\boldsymbol{\nu}) = \left[\widehat{\boldsymbol{W}}(\nu_1,\nu_2)\ \ \widehat{\boldsymbol{W}}\Big(\nu_1+\frac{1}{\sigma_m},\nu_2\Big)\ \ \ldots\ \ \widehat{\boldsymbol{W}}\Big(\nu_1+\frac{\sigma_m-1}{\sigma_m},\nu_2\Big)\right] \in \mathbb{C}^{\zeta_m\times\sigma_m\zeta_0}. \tag{SM4.9}$$

By using now Parseval's formula,

$$\begin{aligned}
\|\mathbf{y}\|^2 &= \int_{[0,1]^d} \|\widehat{\mathbf{y}}(\boldsymbol{\nu})\|^2 d\boldsymbol{\nu} \\
&= \frac{1}{\sigma_m^{2d}} \int_{[0,1]^d} \left\|\widetilde{\boldsymbol{W}}\left(\frac{\boldsymbol{\nu}}{\sigma_m}\right)\widetilde{\mathbf{x}}\left(\frac{\boldsymbol{\nu}}{\sigma_m}\right)\right\|^2 d\boldsymbol{\nu} \\
&\leqslant \frac{1}{\sigma_m^{d}} \int_{[0,1/\sigma_m]^d} \left\|\widetilde{\boldsymbol{W}}(\boldsymbol{\nu})\right\|^2 \|\widetilde{\mathbf{x}}(\boldsymbol{\nu})\|^2 d\nu_1 d\nu_2 \\
&\leqslant \frac{1}{\sigma_m^{d}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\|\widetilde{\boldsymbol{W}}(\boldsymbol{\nu})\right\|^2 \int_{[0,1/\sigma_m]^d} \|\widetilde{\mathbf{x}}(\boldsymbol{\nu})\|^2 d\boldsymbol{\nu} \\
&= \frac{1}{\sigma_m^{2}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\|\widetilde{\boldsymbol{W}}(\boldsymbol{\nu})\right\|^2 \sum_{\mathbf{j}\in\mathbb{S}(\sigma_m)} \int_{[0,1/\sigma_m]^d} \left\|\widehat{\mathbf{x}}\left(\boldsymbol{\nu}+\frac{\mathbf{j}}{\sigma_m}\right)\right\|^2 d\boldsymbol{\nu} \\
&= \frac{1}{\sigma_m^{d}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\|\widetilde{\boldsymbol{W}}(\boldsymbol{\nu})\right\|^2 \int_{[0,1]^d} \|\widehat{\mathbf{x}}(\boldsymbol{\nu})\|^2 d\boldsymbol{\nu} \\
&= \frac{1}{\sigma_m^{d}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\|\widetilde{\boldsymbol{W}}(\boldsymbol{\nu})\right\|^2 \|\mathbf{x}\|^2. \qquad \text{(SM4.10)}
\end{aligned}$$

This shows that

$$\theta_m^2 \leqslant \frac{1}{\sigma_m^{d}} \sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\|\widetilde{\boldsymbol{W}}(\boldsymbol{\nu})\right\|^2. \tag{SM4.11}$$

On the other hand since $\widehat{\boldsymbol{W}}$ is continuous, $\widetilde{\boldsymbol{W}}$ is also continuous, and there exists $\widehat{\boldsymbol{\nu}}\in[0,1/\sigma_m]^d$ such that

$$\sup_{\boldsymbol{\nu}\in[0,1/\sigma_m]^d} \left\|\widetilde{\boldsymbol{W}}(\boldsymbol{\nu})\right\| = \left\|\widetilde{\boldsymbol{W}}(\widehat{\boldsymbol{\nu}})\right\|. \tag{SM4.12}$$

Let us now choose, for every $\boldsymbol{\nu} \in [0, 1/\sigma_m]^d$, $\widetilde{\mathbf{x}}(\boldsymbol{\nu}) = \alpha_\epsilon(\boldsymbol{\nu})\mathbf{u}(\boldsymbol{\nu})$ where $\mathbf{u}(\boldsymbol{\nu})$ is a unit norm eigenvector associated with the maximum eigenvalue of $\widetilde{\boldsymbol{W}}(\boldsymbol{\nu})^{\mathsf{H}}\widetilde{\boldsymbol{W}}(\boldsymbol{\nu})$, $\epsilon \in]0, +\infty[$, and

$$\alpha_\epsilon(\boldsymbol{\nu}) = \begin{cases} \frac{1}{\epsilon^{d/2}} & \text{if } (\exists \mathbf{j} \in \{-1, 0, 1\}^d)\ \|\boldsymbol{\nu} + \frac{\mathbf{j}}{\sigma_m} - \widehat{\boldsymbol{\nu}}\|_\infty \leqslant \frac{\epsilon}{2} \\ 0 & \text{otherwise.} \end{cases} \tag{SM4.13}$$

Then we see that when $\epsilon \to 0$, the upper bound in (SM4.10) is reached. We conclude that

$$\theta_m = \frac{1}{\sigma_m^{d/2}} \sup_{\boldsymbol{\nu} \in [0, 1/\sigma_m]^d} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|. \tag{SM4.14}$$

In addition, by using the relation between $\widetilde{\boldsymbol{W}}$ and $\widehat{\boldsymbol{W}}$ (i.e., (SM4.8) and (SM4.9) in the 2D case),

$$\begin{aligned} \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \right\|^2 &= \left\| \widetilde{\boldsymbol{W}}(\boldsymbol{\nu}) \widetilde{\boldsymbol{W}}(\boldsymbol{\nu})^{\mathsf{H}} \right\| \\ &= \left\| \sum_{\mathbf{j} \in \mathbb{S}(\sigma_m)} \widehat{\boldsymbol{W}}\left(\boldsymbol{\nu} + \frac{\mathbf{j}}{\sigma_m}\right) \widehat{\boldsymbol{W}}\left(\boldsymbol{\nu} + \frac{\mathbf{j}}{\sigma_m}\right)^{\mathsf{H}} \right\|. \end{aligned} \tag{SM4.15}$$

Gathering the last two equalities yields (SM4.2). ∎

When there is no decimation, i.e. the strides $(s_i)_{1 \leqslant i \leqslant m}$ are all equal to 1, (SM4.2) reduces to

$$\theta_m = \sup_{\boldsymbol{\nu} \in [0,1]^d} \|\widehat{\boldsymbol{W}}_m(\boldsymbol{\nu}) \cdots \widehat{\boldsymbol{W}}_2(\boldsymbol{\nu}) \widehat{\boldsymbol{W}}_1(\boldsymbol{\nu})\|. \tag{SM4.16}$$

We recall that the following upper bound holds [SM49]:

$$\theta_m \leqslant \overline{\theta}_m = \prod_{i=1}^{m} \|\mathcal{W}_i\|. \tag{SM4.17}$$

Applying our result to the one-layer case shows that, for every $i \in \{1, \ldots, m\}$,

$$\|\mathcal{W}_i\| = \frac{1}{s_i^{d/2}} \sup_{\boldsymbol{\nu} \in [0, 1/s_i]^2} \left\| \sum_{\mathbf{j} \in \mathbb{S}(s_i)} \widehat{\boldsymbol{W}}_i\left(\boldsymbol{\nu} + \frac{\mathbf{j}}{s_i}\right) \widehat{\boldsymbol{W}}_i\left(\boldsymbol{\nu} + \frac{\mathbf{j}}{s_i}\right)^{\mathsf{H}} \right\|^{1/2}. \tag{SM4.18}$$

Note that the resulting upper bound in (SM4.17) gives a loose estimate of the Lipschitz constant, which has however the merit to be valid for convolutional networks having kernels with an arbitrary sign.

SM5. Numerical evaluation of the Lipschitz constant of nonnegative convolutional networks.

We compare the tight bound θ_m in Theorem 4.1 with the separable one $\overline{\theta}_m$ given by (4.12) for a classic convolutional network using non-negative kernels. The results provided in Table SM1 correspond to the convolutive part of LeNet-5 [SM4]. In our experiments, we initialized

the networks with randomly sampled weights drawn from a uniform distribution on $[0,1]$. Table SM1 shows the relative difference

$$\epsilon_{\mathrm{r}} = \frac{\overline{\theta}_m - \theta_m}{\overline{\theta}_m},$$

for 10 distinct noise realizations. We thus observe that the difference between the two bounds is small. Similar observations can be made on various convolutive architectures. In contrast, for fully connected networks, a separable bound is usually overpessimistic.

	LeNet-5									
	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10
θ_m	30302.73	27734.91	30298.73	29374.35	30180.16	28632.60	30615.02	30395.67	34828.90	30097.62
$\overline{\theta}_m$	30696.07	28114.29	30860.56	29821.62	30670.05	29298.64	31152.06	30866.87	35220.36	30367.71
ϵ_{r} [%]	1.28	1.35	1.82	1.50	1.60	2.27	1.72	1.53	1.11	0.89

Table SM1: Lipschitz bounds obtained for 10 independent realizations of random positive initialization for LeNet-5.

SM6. Lipschitz constant of average pooling. We consider the case when the i-th layer is an *average pooling* where the average is computed on patches of length L_i in each dimension and with stride s_i. For simplicity, we suppose that L_i is a multiple of s_i. The number of input and output channels is then equal, i.e. $\zeta_i = \zeta_{i-1}$. The average is calculated on each channel independently, this operation is a special case of a nonnegative convolutional layer where, for every $\mathbf{n} \in \mathbb{Z}^d$, $\boldsymbol{W}_i(\mathbf{n})$ is a diagonal matrix. The diagonal elements of this matrix are

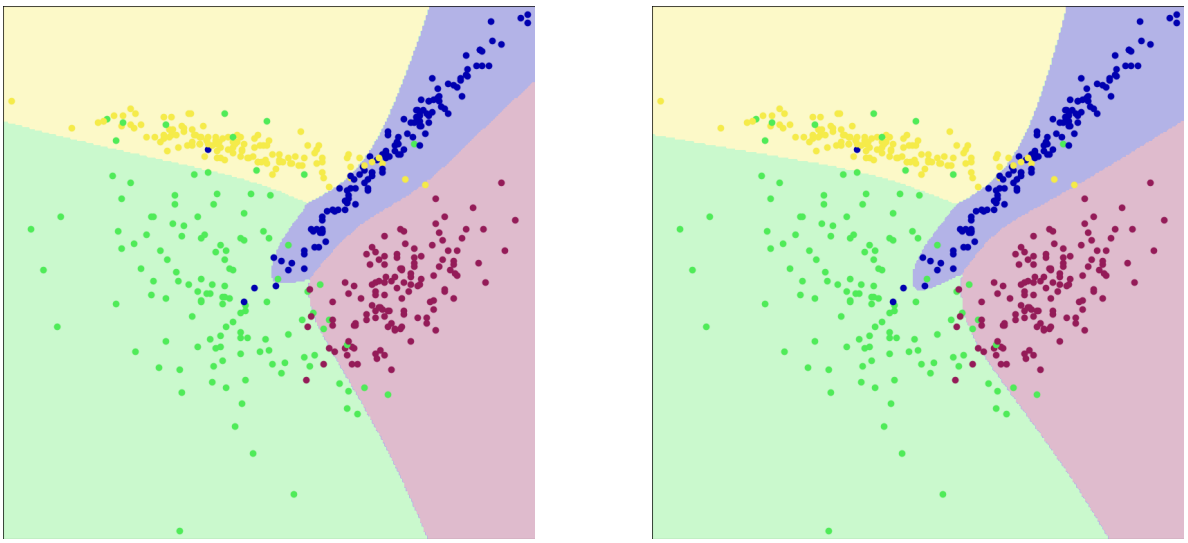
$$(\forall p \in \{1,\ldots,\zeta_i\})(\forall \mathbf{n} \in \mathbb{Z}^d) \quad w_{i,p,p}(\mathbf{n}) = \begin{cases} \frac{1}{L_i^d} & \text{if } \mathbf{n} \in [0, L_i - 1]^d \\ 0 & \text{otherwise.} \end{cases} \tag{SM6.1}$$

We deduce that, for every $\mathbf{j} \in \mathbb{S}(s_i)$, the matrix $\overline{\boldsymbol{W}}_i^{(\mathbf{j})}$ is also a diagonal matrix. More precisely, the sum in (4.13) can be restricted to values of $\mathbf{n} \in \{0,\ldots,L_i/s_i - 1\}^d$ and $\overline{\boldsymbol{W}}_i^{(\mathbf{j})} = \frac{1}{s_i^d}\mathbf{Id}$. We deduce that the Lipschitz constant of the average pooling layer is

$$\|\mathcal{W}_i\| = \Big\| \sum_{\mathbf{j} \in \mathbb{S}(s_i)} \overline{\boldsymbol{W}}_i^{(\mathbf{j})} \big(\overline{\boldsymbol{W}}_i^{(\mathbf{j})}\big)^\top \Big\|^{1/2} = \frac{1}{s_i^{d/2}}. \tag{SM6.2}$$

We see that this constant is independent of the patch size and is a decreasing function of the stride.

SM7. Expressivity of ABBA networks – simulations. For this experiment, we randomly sampled points from four distinct 2D Gaussian distributions, with different means and covariance matrices, totaling 125 2-dimensional points per class. Figure SM1 shows a comparison between decision boundaries resulting from training two models: a standard one trained conventionally and its non-negative ABBA equivalent. The two models reach a similar solution, showing that the theoretical properties proved in this paper are also observed in practical simulations.



(a) Standard model (b) ABBA model

Figure SM1: Decision space comparison between fitting an ABBA network and a standard arbitrary-signed one.

SM8. Constrained training of signed convolutional layers. The first and the last layers of an ABBA convolutional network have signed kernels. The norm of these layers is computed by using (SM4.18) and constrained to be less than $\overline{\theta}_{m,i,t}$ with $i \in \{0, m+1\}$. Note that (SM4.18) makes use of the frequency response $\widehat{\boldsymbol{W}}_i$ of filter $\mathcal{W}_i$. A discrete Fourier transform (DFT) is actually implemented (using 128×128 discrete frequencies). In the discrete frequency domain, the upper bound constraint is thus decomposed into 128^2 matrix norm bounds obtained by summing over s_i^2 frequencies. The projection onto each of these elementary constraint sets is computed by truncating a singular value decomposition. An additional constraint, however, is to be addressed, which is related to the fact that the kernels are of finite size. This implicitly defines a linear constraint. Projecting onto the associated vector space is simply obtained by truncating the kernel (after inverse DFT) to the desired size. The set $\mathcal{S}_{i,t}$ is thus defined as the intersection of the former matrix norm constraint set and the latter vector space. Projecting onto this intersection can be achieved by an iterative convex optimization approach. In our case, we use a Douglas-Rachford algorithm [SM2].

SM9. ABBA architectures. Table SM3 details the ABBA Dense and ABBA Conv architectures used for MNIST and FMNIST datasets, while Table SM2 shows our choices for RPS and CelebA datasets. As the ABBA layers have a specific form, their output size will be twice the number of filters. The used activation operator is the Capped Leaky ReLu (CLR) function defined in (SM1.3) for all Dense layers. For convolutional operators we employed a 3×3 kernel, using the same activation.

We used the official train-test split provided by the Tensorflow framework for both MNIST

Layer type	RPS	stride	CelebA	stride
Input	$150 \times 150 \times 3$		$128 \times 128 \times 3$	
Conv2D	$150 \times 150 \times 8$	1	$128 \times 128 \times 8$	1
ABBA Conv2D + CLR	–	–	$64 \times 64 \times 8(\times 2)$	2
ABBA Conv2D + CLR	$75 \times 75 \times 32(\times 2)$	2	$32 \times 32 \times 16(\times 2)$	2
ABBA Conv2D + CLR	$37 \times 37 \times 64(\times 2)$	2	$16 \times 16 \times 32(\times 2)$	2
ABBA Conv2D + CLR	$18 \times 18 \times 128(\times 2)$	2	$8 \times 8 \times 64(\times 2)$	2
Conv2D	$18 \times 18 \times 128$	1	$8 \times 8 \times 64$	1
Global Max-Pooling2D	$128(\times 2)$		$64(\times 2)$	
ABBA Dense + CLR	$128(\times 2)$		–	
ABBA Dense + CLR	$64(\times 2)$		–	
ABBA Dense + CLR	$32(\times 2)$		–	
Dense	3		2	

Table SM2: ABBA Conv architectures details for RPS and CelebA datasets.

Layer type	MNIST/FMNIST
Input	$28 \times 28 \times 1$
Conv2D	$28 \times 28 \times 32$
ABBA Conv2D + CLR	$28 \times 28 \times 16$
ABBA Conv2D + CLR	$28 \times 28 \times 16$
Conv2D	$28 \times 28 \times 1$
Dense	256
ABBA Dense + CLR	128
ABBA Dense + CLR	64
Dense	10

Layer type	MNIST	FMNIST
Input	784	784
Dense	256	256
ABBA Dense + CLR	128	128
ABBA Dense + CLR	64	64
ABBA Dense + CLR	–	32
Dense	10	10

Table SM3: ABBA Dense and ABBA Conv architecture details for MNIST and FMNIST datasets. For convolutional layers the stride is set to 1.

and FMNIST datasets and did not employ any augmentation strategy during training. For RPS and CelebA models, we resized the input images to 150×150, resp. 128×128, before feeding them to the network. In the case of CelebA dataset, we opted for a binary classification task on the *bald* feature. We extracted all the images containing the *bald* attribute, and we randomly select the same number of examples from the *non-bald* class, in order to avoid class imbalance. Additional information regarding the optimization parameters used during training is provided in Table SM4.

SM10. Adversarial examples. For all datasets, adversarial examples created by using DDN attack are displayed in Figures SM2, SM3, SM4, and SM5. We generated adversarial samples using untargeted DDN attacks, with a budget of 300 iterations and initial parameters as proposed by the authors. We did not limit the maximum perturbation ϵ, in order to find the minimum one allowing us to fool the model. It can be easily seen that for DeelLip and ABBA

Figure SM2: Adversarial examples with DDN attack for Conv-Dense models, on MNIST dataset. ℓ_2 perturbation magnitude is given in the top-left corner.

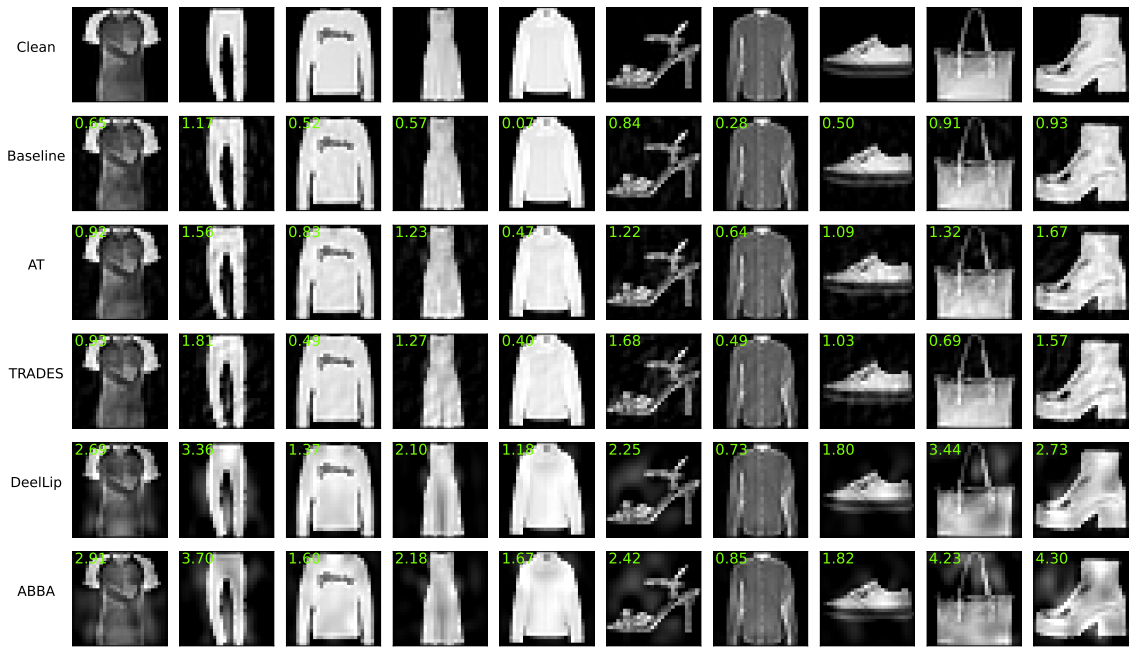


Figure SM3: Adversarial examples with DDN attack for Conv-Dense models, on FMNIST dataset. ℓ_2 perturbation magnitude is given in the top-left corner.

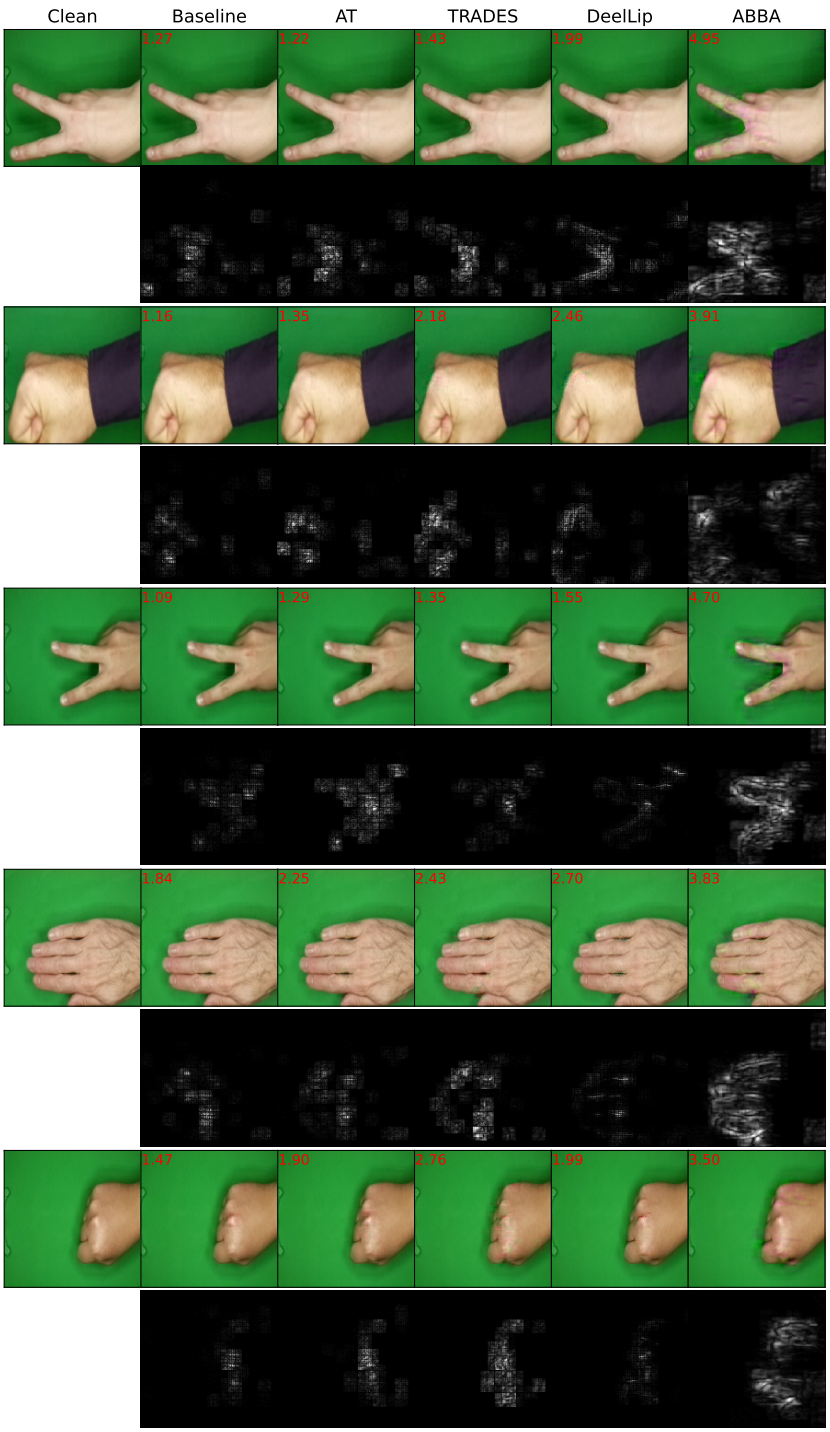


Figure SM4: Adversarial examples generated with DDN, on RPS dataset. For each example: first row – adversarial images; second row – pixel differences between adversarial and clean sample.

Figure SM5: Adversarial examples with DeepFool attack for CelebA. ℓ_2 perturbation magnitude is given in the top-left corner.

Dataset	Optimizer	No. Epochs	Learning rate	Batch size
MNIST	projected ADAM	150	10^{-3}	1024
FMNIST	projected ADAM	200	10^{-3}	1024
RPS	projected ADAM	250	10^{-4}	64
CelebA	projected ADAM	100	10^{-4}	128

Table SM4: Training hyperparameters.

networks the required perturbations for misclassification are higher. In particular, we observe that the perturbations needed to fool ABBA networks lead to severe artifacts in the images.

SM11. Training time. We first compared the average time/epoch for training a standard network and its ABBA equivalent. Table SM5 reports the average seconds per epoch for both cases, for different feed-forward architectures. On average, training an ABBA neural network for 200 epochs on MNIST introduces less than 10% additional training time.

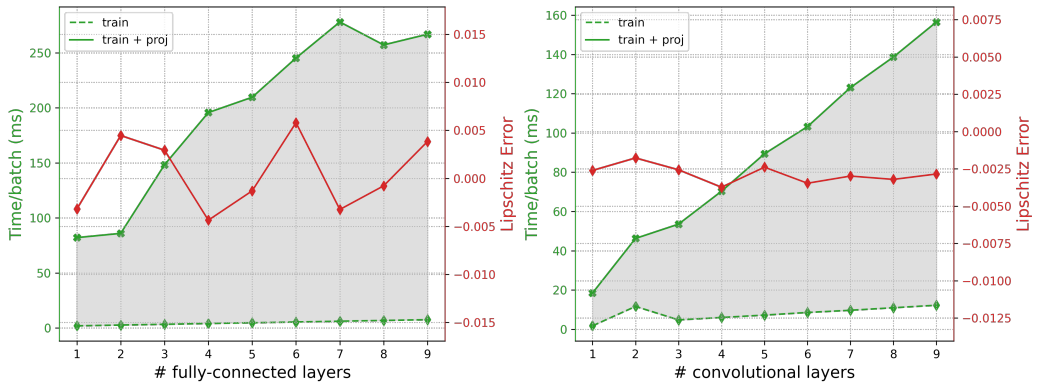
The projection is a costly step, and it is the main contributor to the training overhead. A comparison of the training time (per-batch) with (*green line*) and without (*dotted green line*) projection is featured in Figures SM6a and SM6b for architectures with an increasing number of fully connected and convolutional layers, respectively. The average deviation from the imposed global bound (*red*), which was set to 1 in all cases, is also reported. This shows that we are able to maintain the imposed bounds, given the same number of iterations, irrespective of the network depth.

Model		Architecture					
		2C2F	2C3F	3C2F	4C2F	4C3F	5C1F
Standard	Acc [%]	95.28	95.80	99.18	99.30	99.26	99.10
	Sec./Epoch	4.25	4.29	4.31	4.28	4.37	4.31
	Size (MB)	0.09	0.14	0.39	0.53	0.59	0.97
ABBA	Acc [%]	95.54	95.34	98.62	99.12	99.14	98.72
	Sec./Epoch	4.52	4.61	4.67	4.68	4.79	4.74
	Size (MB)	0.18	0.28	0.78	1.06	1.18	1.94

Table SM5: Comparison of per-epoch training times for various Standard and ABBA architectures, on MNIST. *XCYF* corresponds to an architecture with *X* Convolution layers, followed by *Y* fully-connected layers.

REFERENCES

[1] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert spaces*, 2nd ed., corrected printing. New York: Springer, (2019).



(a) (b)

Figure SM6: Computation time for the projection step for a variable-length sequence of ABBA SM6a fully-connected and SM6b convolutional layers. All projections wer computed with a number of 10 iterations, and the results were averaged over 50 independent simulations.

[2] P. L. Combettes and J.-C. Pesquet, *Proximal splitting methods in signal processing*, Fixed-Point Algorithms for Inverse Probl. Sci. Eng., (2011), pp. 185–212.
[3] ———, *Lipschitz certificates for layered network structures driven by averaged activation operators*, J. Math. Data Sci., 2 (2020), pp. 529–557.
[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proc. IEEE, 86 (1998), pp. 2278–2324.
[5] J. R. Silvester, *Determinants of block matrices*, The Math. Gazette, 84 (2000), pp. 460–467.
[6] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*, Prentice Hall (NJ), (1993).