A Appendix

A.1 Proof of Lemma 2

We will proof the Lemma 2 here. Recall

Lemma 2 (Convolutional AOL Layers) *Let $P \in \mathbb{R}^{k\times k\times c_I\times c_O}$, be a convolution kernel matrix, where $k \times k$ is the* kernel size *and c_I and c_O are the number of input and output channels, respectively. Then, the convolutional layer*

$$f(x) = P * R(x) + b \tag{7}$$

is guaranteed to be 1-Lipschitz, where $R(x)$ is a channel-wise rescaling that multiplies each channel $c \in \{1, \dots, c_I\}$ of the input by

$$d_c = \Big(\sum_{i,j} \sum_{a=1}^{c_I} \Big| \sum_{b=1}^{c_O} P^{(a,b)} * P^{(c,b)} \Big|_{i,j} \Big)^{-1/2}. \tag{8}$$

We can equivalently write f as $f(x) = W * x + b$, where $W = P * D$ with $D \in \mathbb{R}^{1\times 1\times c_I\times c_I}$ given by $D^{(c,c)}_{1,1} = d_c$, and $D^{(c_1,c_2)}_{1,1} = 0$ for $c_1 \neq c_2$.

Proof. For the proof we will assume *maximal* padding of the input: We pad an input $x \in \mathbb{R}^{n\times n\times c_\mathrm{I}}$ (with values independent of x) to a size of $(n + 2k - 2) \times (n + 2k - 2) \times c_\mathrm{I}$, and then apply the convolution to the padded input. Then we obtain an output of size $(n+k-1)\times(n+k-1)\times c_\mathrm{O}$. We will derive a rescaling of the input that ensures this convolution has a Lipschitz constant of 1. Then, any convolution with a different kind of padding (such as *same* size or *valid*) can be considered as first doing a maximally padded convolution, followed by a center cropping operation. Since a cropping operation also has a Lipschitz constant of 1, this also shows that convolutional layers with a different kind of padding have a Lipschitz constant of 1.

Denote by $\tilde{x}$ the padded version of input x, with $\tilde{x}_{i+k-1,j+k-1} = x_{i,j}$. Then, the multi-channel, maximally padded convolution with a convolutional kernel $P \in \mathbb{R}^{k\times k\times c_I\times c_O}$ is given by

$$[P * x]^b_{i,j} = \sum_{p=0}^{k-1} \sum_{q=0}^{k-1} \sum_{a=1}^{c_I} P^{(a,b)}_{p,q} \tilde{x}^a_{i+p,j+q}, \tag{11}$$

for $1 \leq i, j \leq n + k - 1$ and $1 \leq b \leq c_\mathrm{O}$.

We now consider the Jacobian J of the linear map (from unpadded input to output) defined by Equation 11. It is a matrix of size $(n+k-1)^2 c_\mathrm{O} \times n^2 c_\mathrm{I}$, with entries given by

$$J^{(b,a)}_{(i_2,j_2),(i_1,j_1)} = P^{(a,b)}_{(i_1-i_2+k-1),(j_1-j_2+k-1)}, \tag{12}$$

for $1 \leq i_1, i_2 \leq n$, $1 \leq i_2, j_2 \leq n + k - 1$, $1 \leq a \leq c_\mathrm{I}$ and $1 \leq b \leq c_\mathrm{O}$. Here, we define $P^{(a,b)}_{p,q} = 0$ unless $0 \leq p < n$ and $0 \leq q < n$.

We can use that to obtain an expression for $J^\top J$ (writing $m = n + k - 1$):

$$
\begin{aligned}
&[J^\top J]^{(a_1,a_2)}_{(i_1,j_1),(i_2,j_2)} && (13)\\
&\quad= \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{b=1}^{c_\mathrm{O}} J^{(b,a_1)}_{(i,j),(i_1,j_1)} J^{(b,a_2)}_{(i,j),(i_2,j_2)} && (14)\\
&\quad= \sum_{i=1}^{m}\sum_{j=1}^{m}\sum_{b=1}^{c_\mathrm{O}} P^{(a_1,b)}_{(i_1-i+k-1),(j_1-j+k-1)} P^{(a_2,b)}_{(i_2-i+k-1),(j_2-j+k-1)} && (15)\\
&\quad= \sum_{i=0}^{k-1}\sum_{j=0}^{k-1}\sum_{b=1}^{c_\mathrm{O}} P^{(a_1,b)}_{i,j} P^{(a_2,b)}_{(i+i_2-i_1),(j+j_2-j_1)} && (16)\\
&\quad= \left(\sum_{b=1}^{c_\mathrm{O}} P^{(a_1,b)} * P^{(a_2,b)}\right)_{i_2-i_1,j_2-j_1} . && (17)
\end{aligned}
$$

We can now apply Theorem 1 (from the main paper) with $P = J$ together with Equation (17) in order to obtain the necessary rescaling: In order to guarantee the convolution to have Lipschitz constant 1, we need to multiply input $x^{(c)}_{i_2,j_2}$ by

$$\left(\sum_{i_1=1}^{n}\sum_{j_1=1}^{n}\sum_{a_1=1}^{c_\mathrm{I}}\left|\sum_{b=1}^{c_\mathrm{O}} P^{(a_1,b)} * P^{(c,b)}\right|_{i_2-i_1,j_2-j_1}\right)^{-1/2} . \tag{18}$$

A lower bound of this expression (that is tight for most values of i_2 and j_2) is given by

$$\left(\sum_{i=-k+1}^{k}\sum_{j=-k+1}^{k}\sum_{a=1}^{c_\mathrm{I}}\left|\sum_{b=1}^{c_\mathrm{O}} P^{(a,b)} * P^{(c,b)}\right|_{i,j}\right)^{-1/2} . \tag{19}$$

This value is independent of both i_2 and j_2, so this completes our proof for maximally padded convolutions. This also implies that convolutions with less padding have a Lipschitz constant of 1 when the input is rescaled as described. □

Note that this proof requires padding independent of the input. However, for example for cyclic padding, the Jacobian is a doubly circuland matrix, and a very similar proof shows that our rescaling still works.

Also note that a result similar to equation (17), relating $J^\top J$ to a self-convolution, has been observed before by [25].

A.2 Comparison of the orthogonality

We will present a comparison of $J^\top J$ for J the Jacobian of different layers. We consider three architectures: A standard convolutional architecture with stan-

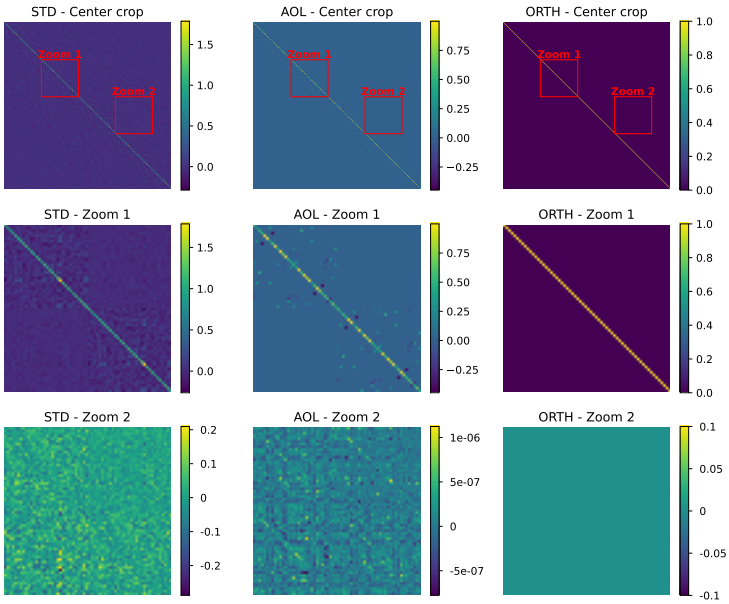

Fig. 2. Evaluation of orthogonality of trained models. The first row shows a center crop of $J^\top J$, where J is the Jacobian of the layer, as well as the location of the other two crops. Those are shown in the second and third row. **Left column:** Standard convolutional layer (STD). **Center column:** AOL-STD (AOL), **Right column:** (Perfectly) orthogonal layer (ORTH). Note the different color scales of different subplots. Best viewed in color and zoomed in.

dard convolutions (see Section A.4), the same architectures with AOL convolutions and (theoretically) an architecture with perfectly orthogonal Jacobian. We pick the third layer of this architecture. It is a convolution with kernel of size $3 \times 3 \times 32 \times 32$, and input as well as output of size $32 \times 32 \times 32$. This results in a Jacobian J of size $32\,768 \times 32\,768$, and we calculate and visualize the values of a center crop of size 288×288 of the matrix $J^\top J$. (See Figure 2.)

One can see that for our AOL-STD architecture most off-diagonal elements are very close to 0, whereas for the standard architecture they are not. Interestingly, for our architecture there are some off-diagonal elements that are clearly non-zero. This shows that the learning resulted in a few column pairs not being orthogonal, in line with our claim of almost-orthogonality.

A.3 Accuracies of different architectures

The results for different architectures using our proposed AOL layers are in Table 6.

Table 6. Experimental results for different architectures using AOL on CIFAR-10. We report the standard accuracy on the test set as well as the certified robust accuracy under input perturbations up to size ϵ for different values of ϵ.

Method	Standard Accuracy	Certified Robust Accuracy			
		$\epsilon = \frac{36}{255}$	$\epsilon = \frac{72}{255}$	$\epsilon = \frac{108}{255}$	$\epsilon = 1$
AOL-FC	67.9%	59.1%	51.1%	43.1%	17.6%
AOL-STD	65.0%	56.4%	48.2%	39.9%	15.8%
AOL-ALT	68.4%	60.3%	52.4%	44.8%	19.9%
AOL-DIL	62.7%	54.2%	46.0%	38.3%	14.9%

A.4 Further Architectures

We present the other architectures here that were used in the ablation studies.

Standard CNN: In this Architecture the number of channels doubles whenever the spatial size is decreased. For details see Table 7.

Table 7. A relatively standard convolutional architecture for CIFAR-10. For all layers we use zero padding to keep the size the same. *First channels* just selects the first channels and ignores the rest.

Layer name	Filters	Kernel size	Stride	Activation	Output size	Amount
Conv	32	3×3	1×1	MaxMin	$32 \times 32 \times 32$	4
Conv	64	2×2	2×2	None	$16 \times 16 \times 64$	1
Conv	64	3×3	1×1	MaxMin	$16 \times 16 \times 64$	4
Conv	128	2×2	2×2	None	$8 \times 8 \times 128$	1
Conv	128	3×3	1×1	MaxMin	$8 \times 8 \times 128$	4
Conv	256	2×2	2×2	None	$4 \times 4 \times 256$	1
Conv	256	3×3	1×1	MaxMin	$4 \times 4 \times 256$	4
Conv	512	2×2	2×2	None	$2 \times 2 \times 512$	1
Conv	512	3×3	1×1	MaxMin	$2 \times 2 \times 512$	4
Conv	1024	2×2	2×2	None	$1 \times 1 \times 1024$	1
Conv	1024	1×1	1×1	None	$1 \times 1 \times 1024$	1
First Channels	-	-	-	-	$1 \times 1 \times 10$	1
Flatten	-	-	-	-	10	1

AOL-FC: Architecture consisting of 9 fully connected layers. For details see Table 8.

Table 8. Fully Connected Architecture. *First channels* just selects the first channels and ignores the rest.

Layer name	Activation	Output size	Amount
Flatten	-	3072	1
AOL FC	MaxMin	4096	8
AOL FC	None	4096	1
First Channels	-	10	1

AOL-STD: This architecture is identical to the one in Table 7, only with standard convolutions replaced by AOL convolutions.

AOL-ALT: Architecture that quadruples the number of channels whenever the spatial size is decreased in order to keep the number of activations constant for the first few layers. This is done until there are 1024 channels, then the number of channels is kept at this value. For details see Table 9.

Table 9. Convolutional architecture. For all layers we use zero padding to keep the size the same. *First channels* just selects the first channels and ignores the rest.

Layer name	Filters	Kernel size	Stride	Activation	Output size	Amount
AOL Conv	16	2×2	2×2	MaxMin	$16 \times 16 \times 16$	1
AOL Conv	16	3×3	1×1	MaxMin	$16 \times 16 \times 16$	4
AOL Conv	64	2×2	2×2	MaxMin	$8 \times 8 \times 64$	1
AOL Conv	64	3×3	1×1	MaxMin	$8 \times 8 \times 64$	4
AOL Conv	256	2×2	2×2	MaxMin	$4 \times 4 \times 256$	1
AOL Conv	256	3×3	1×1	MaxMin	$4 \times 4 \times 256$	4
AOL Conv	1024	2×2	2×2	MaxMin	$2 \times 2 \times 1024$	1
AOL Conv	1024	1×1	1×1	MaxMin	$2 \times 2 \times 1024$	4
AOL Conv	1024	1×1	1×1	None	$2 \times 2 \times 1024$	1
First Channels	256	-	-	-	$2 \times 2 \times 256$	1
AOL Conv	1024	2×2	2×2	MaxMin	$1 \times 1 \times 1024$	1
AOL Conv	1024	1×1	1×1	MaxMin	$1 \times 1 \times 1024$	4
AOL Conv	1024	1×1	1×1	None	$1 \times 1 \times 1024$	1
First Channels	10	-	-	-	$1 \times 1 \times 10$	1
Flatten	-	-	-	-	10	1

AOL-DIL We use an architecture similar to the one used by ECO. For that, we just replace each 3×3 convolution in the architecture in Table 7 with a strided AOL convolution.

A.5 Data augmentation

We use data augmentation in all our experiments. We first do some color augmentation of the images, followed by some spatial transformations. For the color augmentation, we first adjust the hue of the image (by a random factor with delta in $[-0.02, 0.02]$), then we adjust the saturation of the image (by a factor in $[.3, 2]$), then we adjust the brightness of the image (by a random factor with delta in $[-0.1, 0.1]$), and finally we adjust the contrast of the image (by a factor in $[.5, 2]$). After this we clip the pixel values so they are in $[0., 1.]$. For the spatial transformations, we first apply a random rotation, with a maximal rotation of 5 degrees. Then we apply a random shift of up to 10% of the image size, and finally we flip the image with a probability of 50%. We rely on the tensorflow layers *RandomRotation* and *RandomTranslation* for the spatial transformations, and leave all hyperparameters such as the fill mode as the default values. All hyperparameters specifying the amount of augmentation were chosen based on visual inspection of the augmented training images.

A.6 Computational Complexity of AOL Layers

In addition to performing a standard convolution, in AOL Layers the rescaling factors have to be computed. For fully connected layers, the most expensive operation required for computing the rescaling factors is calculating $P^\top P$. Implemented in a standard way this has complexity $O(s^2 t)$ for a parameter matrix $P \in \mathbb{R}^{t \times s}$. However, it only has to be performed once per batch, so it is comparable to the speed of the forward pass itself, which is $O(bst)$ for batch size b ($b = 250$ in our experiments).

For convolutional AOL layers with kernel size $k \times k$ and c channels calculating the rescaling factors has a complexity of $O(k^4 c^3)$ per batch. The complexity of a forward pass (of a convolution) with input and output sizes $n \times n$ and batch size b is $O(bn^2 k^2 c^2)$. Usually, we expect the forward pass to require many more computations, but when the number of channels c is large, and both the spatial size of the input as well as the batch size b are small, calculating the rescaling factor can cause a non-trivial computational overhead, however the computations should still be compareable in complexity.

In order to update the parameter matrix, we also need to differentiate through the calculation of the rescaling factors (also only once per batch). It turns out that the backward pass has the same complexity as the forward pass, e.g., for fully-connected layers, the backpropagation through $P \mapsto P^\top P$ is most costly and takes $O(s^2 t)$ in explicit form.