A. Additional experiments

Out-of-distribution robustness

While in our model shift-invariance is guaranteed, it is merely a learned property in other models, and thus may only be partially generalized to out-of-distribution images [2]. To evaluate this hypothesis, we measured robustness to fractional translations on ImageNet-C [14], which contains common corruptions of ImageNet images in ascending severity levels. We used fractional grid attack with a minimal translation of $1/7$ a pixel *i.e.*

$$T_{\text{frac},k=7} = \left\{ \left(\frac{m_1}{n_1}, \frac{m_2}{n_2} \right) \mid 1 \leq m_{1,2} \leq n_{1,2} \leq 7 \right\} . \tag{10}$$

The results are visualized in Figure 5. As our model's robustness to translations is guaranteed, it has no accuracy reduction caused by translations. In contrast, the other models' vulnerability to translation attacks increases with the severity of the corruption; ConvneXt-AFC relative accuracy degradation due to fractional translations increases from 5% to as much as 23% in the highest corruption severity. This indicates that the generalization of learned shift-invariance is limited in comparison to architectural shift-invariance.

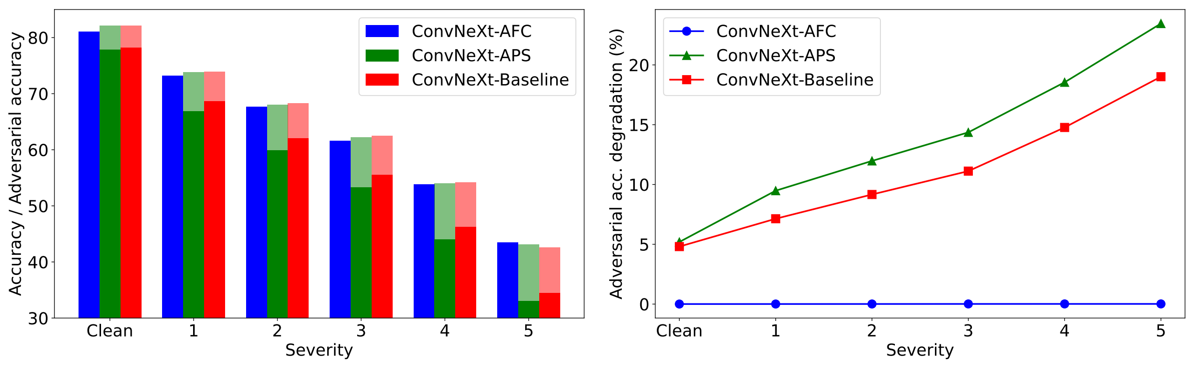


Figure 5. **Adversarial accuracy with image corruptions. Top**: ImageNet-C accuracy (solid) vs. adversarial fractional grid accuracy (transparent). **Bottom**: Accuracy vs. adversarial accuracy difference (percentage). ConvNeXt-AFC (ours) ImageNet-C accuracy is not affected by translations, while in ConvNeXt-APS and ConvNeXt-Basline the relative accuracy degradation as a result of translations increases with the corruption severity.

B. Limitations

B.1. Accuracy

Although our model has improved robustness in comparison to the baseline model and other shift-invariance methods, it has a lower accuracy on the original test set. This result makes sense since robustness often comes at the cost of accuracy. Specifically, the property of perfect shift-invariance is architecturally forced in our model, in contrast to other CNNs where it can be violated. This may seem as a reduction of the hypothesis set. However, in Table 1 of the paper we observe that the highest drop in accuracy does not occur at the last modification, where the model becomes shift-invariant, but rather as a result of modifying the Normalization Layer to be alias-free. Although the proposed alias-free normalization layer was designed to remain similar to the original model LayerNorm, other alias-free normalization methods may exist and lead to a higher accuracy than ours, without hurting robustness. Another drop in accuracy occurs as a result of replacing GeLU in polynomial activation, which surprisingly does not happen in the regular setting (models without cyclic convolutions); a polynomial activation in a cyclic setting leads to a reduction of 0.4% in accuracy (see Tab. 1 of the paper) while in a non-cyclic setting it leads to a reduction of 0.1% only (see Table 4). It implies that additional hyper-parameter tuning might help in the recovery of this accuracy drop. In addition, we note again that the modifications in our model effectively remove non-linearities from the baseline model, and that ConvNeXt-AFC is roughly (up to the layer normalizations) a polynomial of the input with a degree that is an exponent of the convnet's depth. Thus, using a wider or deeper convnet might help in closing the accuracy gap.

B.2. Runtime performance

Although the AFC model has only a small amount of additional parameters in the polynomial activation function, it has a higher computation cost than the baseline (see Table 3). The main reason for that is that while the activation function in the baseline model is a single pointwise operation, our activation requires upsampling and downsampling which are rather expensive. This issue has been addressed in a previous study [17], where a similar scheme for partially alias-free activation has been used. They combined all the required operations to a single CUDA kernel which (according to them) led to a speed-up of at least x20 over native PyTorch implementation, and in total to x10 speed-up in training time. Our model training time is only 5 times higher than the baseline training time, therefore it is reasonable to assume such efficient implementation may significantly reduce the training time gap.

Model	Train time [hours]	Eval time [ms per sample]
ConvNeXt-Baseline [19]	84	1.39
ConvNeXt-APS [4]	93	1.56
ConvNeXt-AFC (ours)	418	9.16

Table 3. **Training and evaluation performance.** Train time was measured on Nvidia A6000 x 8 using the maximal possible batch-size per model due to memory constraints. Evaluation time was measured on a single A6000 with batch-size 256.

B.3. Image translations

B.3.1 Circular translations

The guaranteed robustness in the AFC model is limited to circular shifts, similarly to previous work [4]. Applying this kind of translation on a finite image causes edge artifacts and creates an unrealistic image (*e.g*., see Fig. 8). Although our model has improved robustness even in translations of the frame with respect to the scene (see Fig. 4 of the paper) which may seem as a more practical setting, information-loss makes guaranteed robustness impossible — for example, consider an image in which a translation cause the classified object to get out of the frame. Although the certified robustness may seem not applicable, circular shifts can actually be practically relevant. For example, shifting an object over a constant (*i.e*., uniform) background will seem identical to a circular translation of the entire frame. This setting may be relevant in face recognition tasks and medical imaging (see Fig. 10). In addition, horizontal circular shifts are relevant for panoramic (360°) cameras, *e.g*. in autonomous cars (see Fig. 11).

B.3.2 Interpolation kernel

In Section 2 of the paper we proved that AFC is robust even to sub-pixel shifts. Our robustness guarantees assume the digital image processed by the network corresponds to point-wise samples of a continuous-space image that had been convolved with a perfect anti-aliasing filter prior to sampling (though, empirically, our method performs well with other types of interpolation kernels, *e.g*. see Fig. 4 of the paper). Although this may seem like a serious limitation, it is in fact the standard setting in any imaging system. Indeed, in any optical imaging system, the image impinging on the detector corresponds to the continuous scene convolved with a low-pass filter. This low-pass filter completely zeros out all frequencies above a cutoff frequency that is inversely proportional to the diameter of the aperture [10]. Thus, while in many domains perfect low-pass filtering is challenging to achieve, in Optics, this diffraction limit is in fact impossible to avoid. Cameras are designed such that the cutoff frequency of the aperture corresponds to the Nyquist frequency of the CCD array. In systems where the aperture can be modified by the user, the CCD array is typically adjusted to the minimal aperture width and an additional anti-aliasing filter element is inserted in front of the CCD [26], so that the Nyquist condition is still met for any chosen aperture diameter within the allowed range. It should be noted, however, that the digital image captured by the sensor typically undergoes a series of nonlinear operations within the image signal processor (ISP) of the camera. This implies that shift equivariance may be lost already at the camera level, before the image reaches our convnet. Addressing these effects is beyond the scope of our work.

C. Future work

This study shows how an aliasing-free convnet can be used to build an image classifier with certified shift robustness. Yet, the applications of such convnet are not limited to that purpose; our method can be applied in other domains in which aliasing has been shown to be damaging, such as generative models [17]. Furthermore, our method guarantees shift-equivariant internal representation, a stronger property than shift-invariance. Future work may examine the importance of this property in other tasks. For example, our method can be naturally expanded to construct a shift-equivariant convnets for segmentation.

D. Polynomial activation function

D.1. Coefficients initialization

The Polynomial activation function is a point-wise polynomial:

$$\text{Poly}_2(x) = a_0 + a_1 x + a_2 x^2 \,, \tag{11}$$

where the coefficients $\{a_0, a_1, a_2\}$ are trainable parameters, which are shared per-channel. They were initialized by fitting this function to the GeLU, as proposed by Gottemukkula [11], to function as an approximation to the original activation function ConvNeXt works well with. This initialization gives the function presented in Figure 6 (left). Yet, the activation function may converge to a completely different function by the end of the training. Moreover, it may differ significantly between different layers and between different channels in the same layer. Figure 6 (right) shows the final activation function for five different channels in the first block of a trained model.

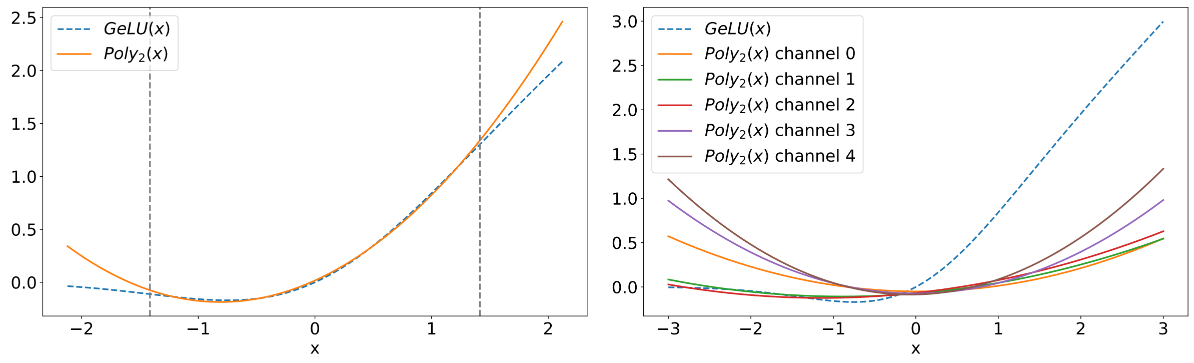


Figure 6. **The** $\text{Poly}_2(x)$ **activation function.** In the left panel, we see how the coefficients are initialized to fit GeLU in range $\left[\sqrt{2},\ -\sqrt{2}\right]$ (dashed lines). In the right panel we see how, in the trained model, the activation function may change significantly and converge to a different function in each channel.

D.2. Activation scaling

During our experiments, we found out that scaling the inputs and outputs of the activation function, regardless of the coefficients themselves, may change the final model results. Thus, we used the activation function:

$$\text{Poly}_c(x) = c\text{Poly}_2(cx) \tag{12}$$

and were looking for the optimal scale c. This scaling factor can effectively be seen as a scaling factor of the weights initialization of the pointwise convolution layers before and after the activation function. In Figure 7 we ran a scan over different scale factors and compared the results of the non-cyclic convolution polynomial model, without additional alias-free modifications (ConvNeXt-Tiny), and on the final model (ConvNeXt-AFC). The scan was run on “ImageNet200”, a subset of ImageNet consisting of 200 classes. We compare the results to the results of non-cyclic convolution GeLU model. In the scope of our search, the best result was achieved with scale $c = 7$ for ConvNeXt-Tiny and scale $c = 4$ for ConvNeXt-AFC. We used scale $c = 7$ for the rest of the polynomial models, as it achieved slightly better results on the full dataset.

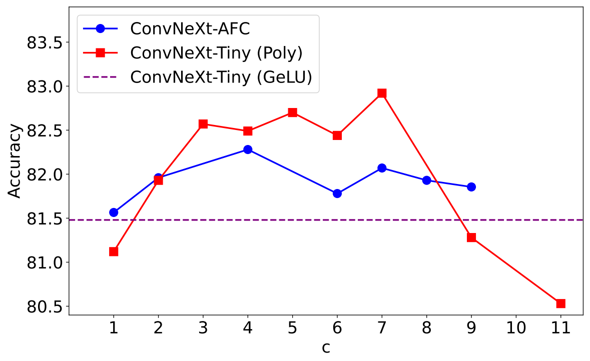


Figure 7. **Test accuracy on ImageNet200, non-cyclic convolutions, GeLU and polynomial ConvNeXt with different scales.** Scaling with $c = 7$ gave the best results in the scope of our search.

D.3. Polynomial activations in other architectures

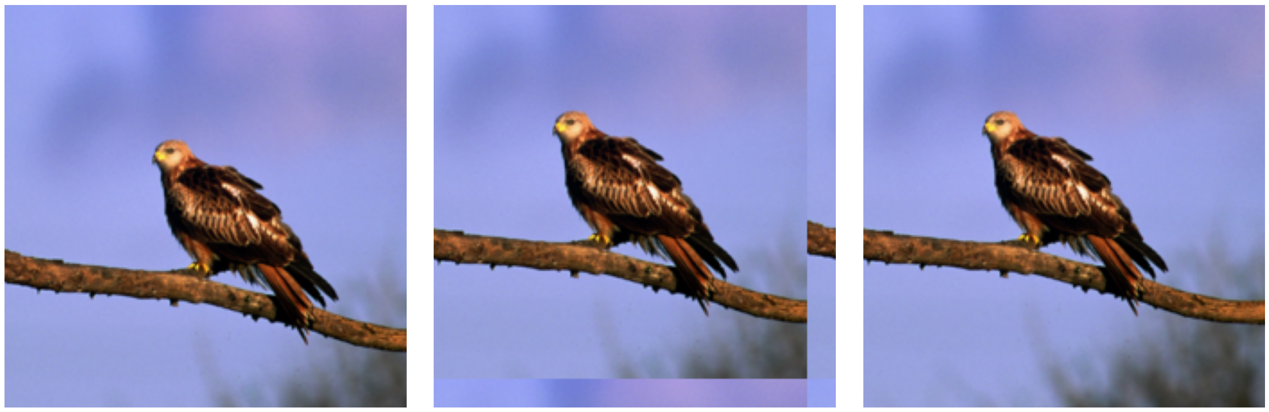
Polynomial activation functions are relatively cheap to compute, which might motivate their use in the future, regardless of their advantage in the context of aliasing-free convnets. Despite this, the high curvature of the polynomial activation seems a barrier to its usage in a wider variety of architectures. To achieve good performance on CIFAR, Gottemukkula [11] implemented in ResNet [13] a stable version of polynomial activation function, by scaling the pre-activation by the L_1 of the layer, which bounds the maximal input element (note that this scaling technique is not aliasing-free and thus was not useful for our purposes). We observe this scaling is unnecessary in architectures with sparser usage of activation functions, such as ConvNeXt and ViT; in Table 4, we show preliminary results suggesting a polynomial activation may be a reasonable substitute for the widely spread GeLU in Transformer-based architectures as well as convnets.

Task	Model	Test acc.
ImageNet	ConvNeXt-tiny (GeLU)	82.1
ImageNet	ConvNeXt-tiny (Poly. deg 2)	81.98
CIFAR10	ViT (GeLU)	97.04
CIFAR10	ViT (Poly. deg 2)	97.08
ImageNet	Deit-tiny (GeLU)	72.29
ImageNet	Deit-tiny (Poly. deg 4)	71.96

Table 4. **Test accuracy in models with polynomial activations.**

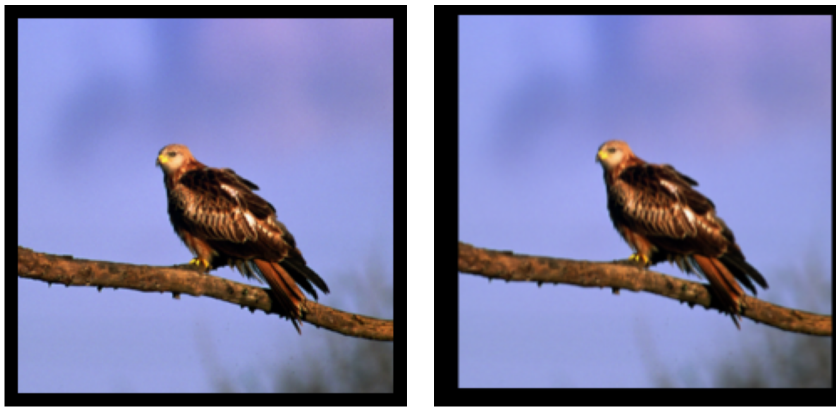
E. Attacks visualization

We provide visual examples of the attacks described in the paper in Figures 8 to 11.



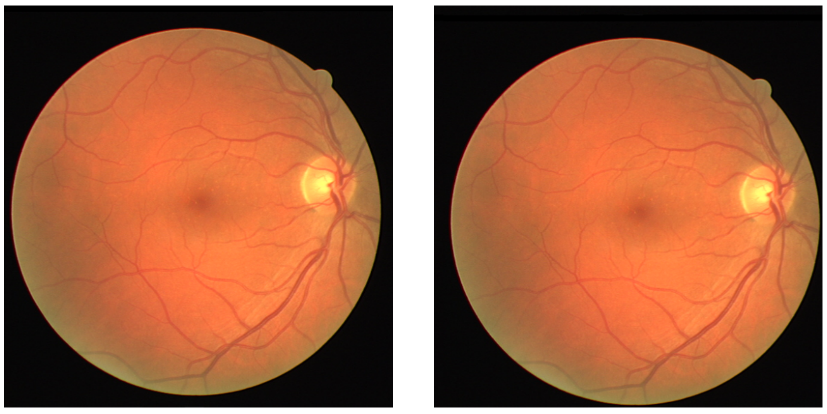
(a) Original image (b) Circular shifted image (c) Crop-shifted image

Figure 8. **Visualization of used attacks.** (a) Original ImageNet [6] validation-set image — 224 × 224 center crop of the original 256 × 256 image. (b): Circular shift of 16 pixels in x and y axes. (c): "Crop-shift" of the original image of 16 pixels in x and y axes; the cropped area is shifted, modeling moving the camera with respect to the scene in the bottom-right direction. The top-left part of the circular shifted and crop-shifted images are equal to the bottom-right part of the original image. The bottom and right edges of the circular shifted image consist of the top and left edges of the original image, causing unrealistic artifacts, while in the crop-shifted change we change the information from the scene.



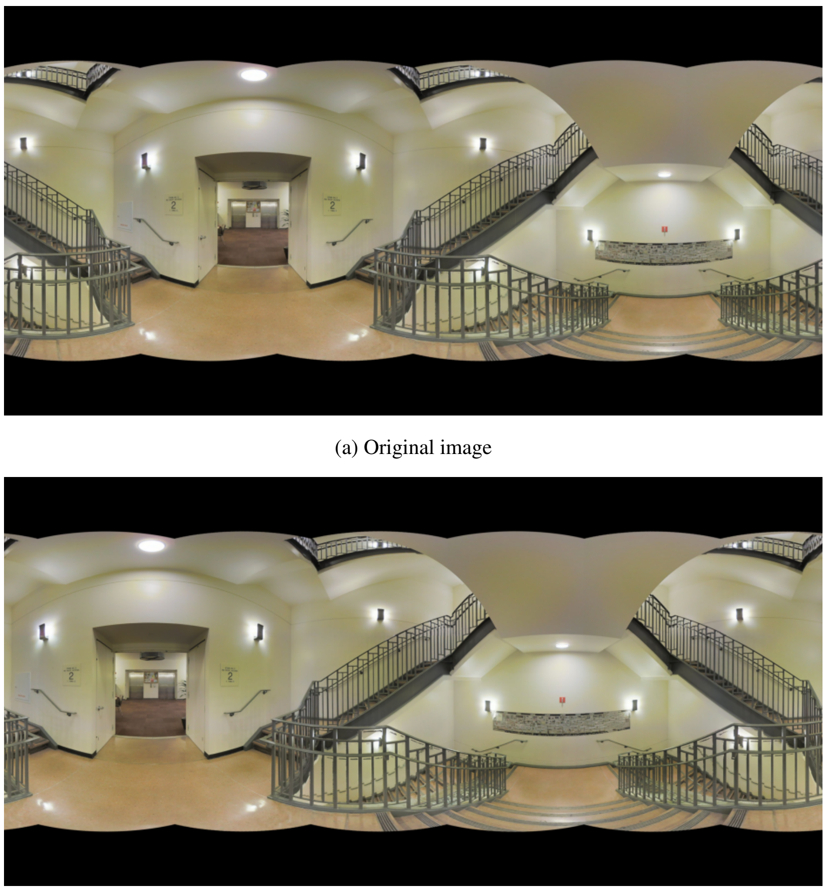
(a) Original image (b) Shifted image

Figure 9. **Visualization of shift attacks similar to the framework of Engstrom *et al*. [8].** (a) The original image is zero-padded in 8 pixels in each direction. The attack is a translation of up to 8 pixels in each direction, *e.g*. (b) is a translation of 6 and -2.5 pixels in x and y axes respectively. Sub-pixel translations are done using bilinear interpolation.



(a) Original image (b) Circular shifted image

Figure 10. **Circular translation of a retinal image.** (a) Original Image [28]. (b) Circular translated image. The retinal image has a uniform background, hence circular shift is equivalent to a translation of the object within the image (*i.e.* "crop-shift").



(a) Original image

(b) Circular shifted image

Figure 11. **Circular translation of a panoramic image.** (a) Original Image [1]. (b) Circular translated image, representing a translation of the camera with respect to the scene, without any edge artifacts.

F. Formal shift-invariance proofs

F.1. Alias-Free polynomial activation function

In the paper, we presented a new alias-free activation function (described in Algorithm 1 in the paper) that, together with alias-free downsampling layers, can completely solve the aliasing problem, and lead to perfectly shift-invariant CNNs. The validity of this solution relies on the following facts:

1. Proposition 2 of the paper (which we formally prove below) ensures the activations are shift-equivariant w.r.t. continuous domain.

2. Convolution and alias-free downsampling layers are indeed shift-equivariant w.r.t. continuous domain (*e.g.*, see proof in [17]).

3. A composition of functions which are shift-equivariant w.r.t. continuous domain remains shift-equivariant w.r.t. continuous domain.

4. Shift-invariance w.r.t. continuous domain is implied from shift-equivariance w.r.t. continuous domain, as was shown in Proposition 1 of the paper.

Therefore, all that remains is to prove Proposition 2. For convenience (to help visualize the proof), we show Figure 2 of the paper again (see Fig. 12).

Proof (Proposition 2) We assume that the input x is sampled from x_c, a $\frac{1}{T}$-band-limited signal at sample rate T, *i.e.*

$$x\left[n\right] = x_c\left(nT\right). \tag{13}$$

We denote the DTFT of $x\left[n\right]$ as:

$$X^f\left(\theta\right) = \sum_{n=-\infty}^{\infty} x\left[n\right] e^{-j\theta n}, \tag{14}$$

and the CTFT of x_c as:

$$X^F\left(\omega\right) = \int_{-\infty}^{\infty} x\left(t\right) e^{-j2\pi\omega t}\, dt. \tag{15}$$

In addition, we define a reconstruction operator as a sinc interpolation of the discrete signal:

$$\text{Recon}\left(x\left[n\right]\right)\left(t\right) = \sum_{n\in\mathbb{Z}} x\left[n\right] \text{sinc}\left(\frac{t-nT}{T}\right). \tag{16}$$

It implies that:

$$x_c\left(t\right) = \text{Recon}\left(x\left[n\right]\right)\left(t\right). \tag{17}$$

For easing the proof notation, we denote applying Algorithm 1 of the paper as a whole on $x\left[n\right]$ as $f\left(x\left[n\right]\right)$.

A well-known relation between $X^f\left(\theta\right)$ and $X^F\left(\omega\right)$ for any continuous signal and its discrete representation is:

$$X^f\left(\theta\right) = \frac{1}{T}\sum_{k=-\infty}^{\infty} X^F\left(\frac{\theta+2\pi k}{T}\right). \tag{18}$$

This relation is represented in Figure 12 (a). Since the support of X^F is limited by $\frac{1}{T}$, we can express $X^f\left(\theta\right)$ in the frequency range $\theta\in\left[-\pi,\ \pi\right]$ as :

$$X^f\left(\theta\right) = X^F\left(\frac{\theta}{T}\right). \tag{19}$$

From now on we will look at the DTFT domain in the range $\theta\in\left[-\pi,\ \pi\right]$, since the effects on the rest of the replications are equal, *i.e.*:

$$\forall k\in\mathbb{Z},\ \forall\theta\in\left[-\pi,\ \pi\right]:\ X^f\left(\theta+2\pi k\right) = X^f\left(\theta\right)\ ; \tag{20}$$

this expression is true also for the DTFT of every signal from now on.

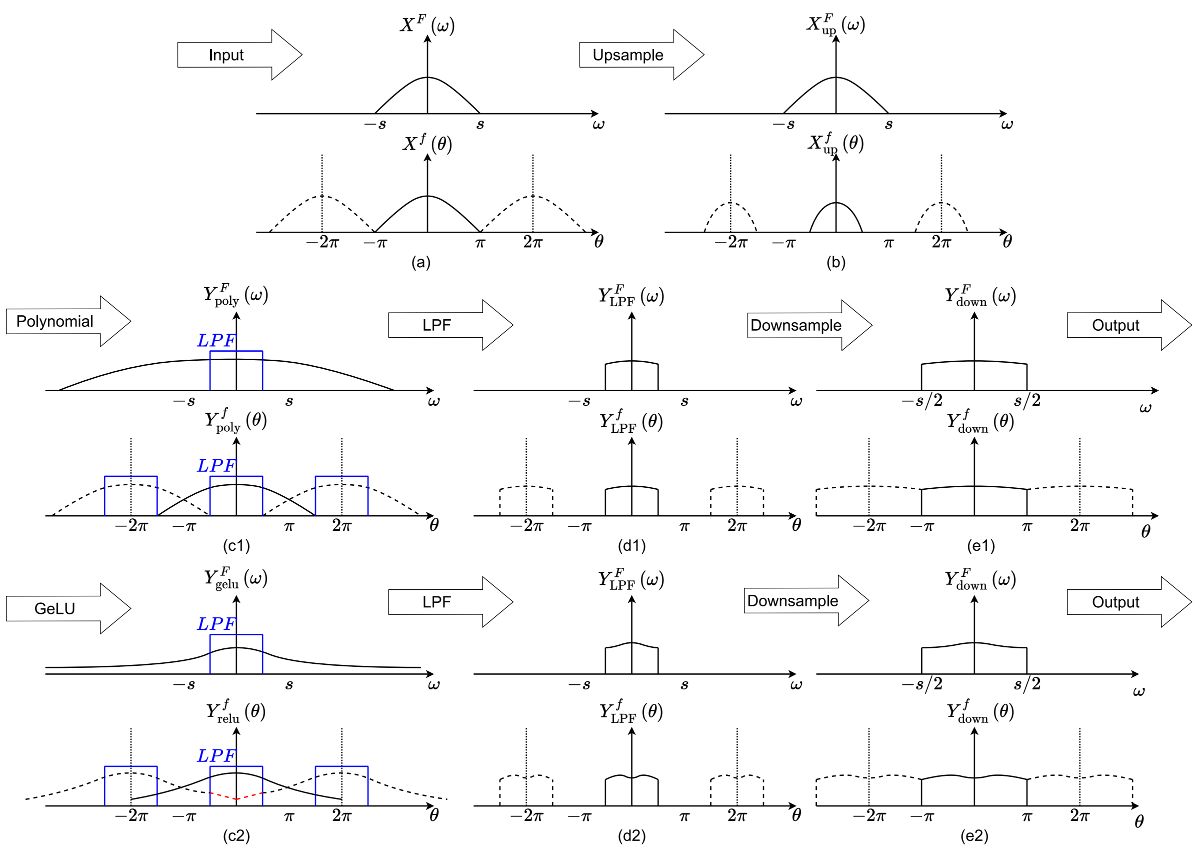


Figure 12. **A demonstration of the proposed non-linearities in the frequency domain.** The top plot at each panel represents the signal in the continuous domain, and the bottom represents the discrete domain. Where the input (a) is upsampled it shrinks its frequency response, expanding the allowed frequencies (b). Applying the polynomial activation expands the frequency response support by as factor d, without causing aliasing in the relevant frequencies (c1). Thus, the discrete signal remains a faithful representation of the continuous signal after applying LPF (d1) and downsample back to the same spatial size (d2). However, applying GeLU expands the support infinitely (c2). This leads to an aliasing effect — interference in the relevant frequencies marked in red in (c2). This causes the discrete signal not to be a correct representation of the continuous one, after LPF (d2) and downsampling (e2).

We will show that the operation presented in Algorithm 1 on $x[n]$ is equivalent to applying polynomial activation in the continuous domain, following an LPF, *i.e.*

$$f(x[n])[n] = \mathrm{LPF}_{\frac{2}{d+1}}\left(\mathrm{Poly}_d\left(x_c(t)\right)\right)(nT) \tag{21}$$

$$= \mathrm{LPF}_{\frac{2}{d+1}}\left(\mathrm{Poly}_d\left(\mathrm{Recon}\left(x[n]\right)\right)\right)(nT)\ . \tag{22}$$

At step 1 of the algorithm, the signal is upsampled using sinc interpolation ($x_{\mathrm{up}} \leftarrow \mathrm{Upsample}_{\frac{d+1}{2}}(x)$), giving the expression

$$x_{\mathrm{up}}[m] = \sum_{n\in\mathbb{Z}} x[n]\,\mathrm{sinc}\left(\frac{m-nI}{I}\right)\ , \tag{23}$$

where $I = \frac{d+1}{2}$.

The frequency response of upsampling is a contraction in the frequency axis:

$$X^f_{\mathrm{up}}(\theta) = X^f(\theta I) = \begin{cases} X^f(\theta I) & |\theta| \le \pi/I \\ 0 & |\theta| > \pi/I \end{cases}\ , \tag{24}$$

which indeed represents a sample of the continuous signal at the rate IT, as can be seen in Fig. 12(b).

At step 2 of the algorithm, Poly_d is applied on x_{up}, giving y_{poly}. From the duality of multiplication and convolution in spatial and Fourier domains, we get that:

$$Y^f_{\mathrm{poly}}(\theta) = a_0 + \sum_{k=1}^{d} a_k \left(\frac{1}{2\pi}\right)^k \underbrace{X^f_{\mathrm{up}} * ... * X^f_{\mathrm{up}}}_{k\text{ times}}(\theta) \; . \tag{25}$$

Without loss of generality, we can assume for simplicity that $\mathrm{Poly}_d(x) = x^d$ and omit the constant factor $\left(\frac{1}{2\pi}\right)^k$, since the frequency expansion is determined solely by the highest degree of the polynomial. Since the support of $\mathrm{Poly}_d(x) = x^d$ equals to the support of x multiplied by d, and since the input support was contracted at factor $I = \frac{d+1}{2}$, we get that the support of $\underbrace{X^f_{\mathrm{up}} * ... * X^f_{\mathrm{up}}}_{d\text{ times}}$ is $\frac{\pi d}{I}$.

We get that the polynomial output support is:

$$\frac{\pi d}{I} = \frac{2\pi d}{d+1} > \pi \, , \tag{26}$$

therefore aliasing occurs. The extension of the support beyond the range $\theta \in [-\pi, \; \pi]$ is :

$$\frac{2\pi d}{d+1} - \pi = \frac{\pi d - \pi}{d+1} = \pi - \frac{2\pi}{d+1} = \pi - \frac{\pi}{I} \tag{27}$$

Hence, the replications due to the aliasing do not affect the frequency domain of $|\theta| \leq \pi/I$, *i.e.*:

$$Y^f_{\mathrm{poly}}(\theta) = \begin{cases} \underbrace{X^f_{\mathrm{up}} * ... * X^f_{\mathrm{up}}}_{d\text{ times}}(\theta I) & |\theta| \leq \pi/I \\ \underbrace{X^f_{\mathrm{up}} * ... * X^f_{\mathrm{up}}}_{d\text{ times}}(\theta I) + \underbrace{X^f_{\mathrm{up}} * ... * X^f_{\mathrm{up}}}_{d\text{ times}}(\theta I - 2\pi) & \theta > \pi/I \\ \underbrace{X^f_{\mathrm{up}} * ... * X^f_{\mathrm{up}}}_{d\text{ times}}(\theta I) + \underbrace{X^f_{\mathrm{up}} * ... * X^f_{\mathrm{up}}}_{d\text{ times}}(\theta I + 2\pi) & \theta < -\pi/I \end{cases} , \tag{28}$$

where the summations in the two bottom cases represent aliasing caused by the expansion of the near replications. This partial aliasing effect caused by the polynomial is presented in Fig. 12(c1).

At step 3, we use an $\mathrm{LPF}_{1/I}$, thus we eliminate all the aliased frequencies, and get:

$$Y^f_{\mathrm{LPF}}(\theta) = \begin{cases} \underbrace{X^f_{\mathrm{up}} * ... * X^f_{\mathrm{up}}}_{d\text{ times}}(\theta I) & |\theta| \leq \pi/I \\ 0 & |\theta| > \pi/I \end{cases} , \tag{29}$$

which can be seen in Fig. 12(d1).

At step 4, applying $\mathrm{Downsample}_I$ expands the frequency domain, so we get

$$Y^f(\theta) = \underbrace{X^f * ... * X^f}_{d\text{ times}}(\theta) \, , \; \theta \in [-\pi, \pi] \; . \tag{30}$$

We note again that this expression is true for the domain $\theta \in [-\pi, \pi]$, specifically because the actual support of $\underbrace{X^f * ... * X^f}_{d\text{ times}}(\theta)$ is larger. However, the frequencies beyond this range were eliminated by the LPF, as can be seen in Fig. 12(e1).

Recalling again that $X^f(\theta) = X^F\left(\frac{\theta}{T}\right)$, we get that the CTFT of the continuous signal of the final expression y is

$$Y^F(\omega) = \begin{cases} \underbrace{X^F * ... * X^F}_{d\text{ times}}(\omega) & |\omega| \leq \frac{1}{T} \\ 0 & |\omega| > \frac{1}{T} \, , \end{cases} \tag{31}$$

which is equivalent to the signal we would get by applying $\mathrm{LPF}_{1/I}\left(\mathrm{Poly}_d\left(\cdot\right)\right)$ on x_c.

Shift-equivariance w.r.t. continuous domain stems from this equivalence because we get that

$$f\left(x\left[n\right]\right)\left[n\right] = \mathrm{LPF}_{\frac{2}{d+1}}\left(\mathrm{Poly}_d\left(\mathrm{Recon}\left(x\left[n\right]\right)\right)\right)\left(nT\right) \tag{32}$$

$$\begin{aligned} \Rightarrow f\left(\tau x\left[n\right]\right)\left[n\right] &= \mathrm{LPF}_{\frac{2}{d+1}}\left(\mathrm{Poly}_d\left(\mathrm{Recon}\left(\tau x\left[n\right]\right)\right)\right)\left(nT\right) && (33)\\ &= \mathrm{LPF}_{\frac{2}{d+1}}\left(\mathrm{Poly}_d\left(\mathrm{Recon}\left(x\left[n\right]\right)\right)\right)\left(nT+\Delta\right) && (34)\\ &= \tau f\left(x\left[n\right]\right)\left[n\right]\ . && (35)\end{aligned}$$

The transition in Eq. (34) is justified due to shift-equivariance w.r.t. continuous domain of reconstruction and alias-free downsample operators, and shift-equivariance of point-wise operations in the continuous domain.

F.2. LPF-Poly

In Appendix F.1 we showed that our polynomial activation function, which is derived in Algorithm 1 of the paper, is alias-free for any polynomial. Specifically, as can be seen in Algorithm 1, the required upsample rate to avoid aliasing is dependent on the polynomial degree and equals to $\frac{d+1}{2}$, where d is the polynomial degree. In this section, we generalize this concept to cases where we would like to avoid upsampling, *e.g.* in layers where the channels have large spatial extents, where it is too computationally expensive. In addition, we show that $\mathrm{LPFPoly}_2$ which was presented in the paper is indeed alias-free and shift-equivariant w.r.t. continuous domain, using the proof concept regarding Algorithm 1.

We defined LPF-Poly as:

$$\mathrm{LPFPoly}_2\left(x\left[n\right]\right)\left[n\right] = a_0 + a_1 x\left[n\right] + x\left[n\right]\cdot \mathrm{LPF}_c\left(x\left[n\right]\right)\left[n\right]\ . \tag{36}$$

Note that c is a real number in the range $(0, 1)$, representing the LPF's cutoff ratio. In addition, note that in the paper we omitted some of the $[n]$ in the notation for more compact writing. The output support is implied by the component of $x \cdot \mathrm{LPF}_c\left(x\right)$, and, similarly to the computation in Eq. (26), is equal to $\left(1+c\right)\pi$.

We get that

$$Y^f_{\mathrm{poly}}\left(\theta\right) = \begin{cases} X^f * X^f_{\mathrm{LPF}}\left(\theta\right) & |\theta| \le \pi - \pi c\\ X^f * X^f_{\mathrm{LPF}}\left(\theta\right) + X^f * X^f_{\mathrm{LPF}}\left(\theta - 2\pi\right) & \theta > \pi - \pi c\\ X^f * X^f_{\mathrm{LPF}}\left(\theta\right) + X^f * X^f_{\mathrm{LPF}}\left(\theta + 2\pi\right) & \theta < -\left(\pi - \pi c\right)\end{cases}, \tag{37}$$

which means that the range $|\theta| \le \pi(1-c)$ is alias-free. This was achieved without the need of upsampling. Next, applying LPF_{1-c} gives:

$$Y^f_{\mathrm{LPF}}\left(\theta\right) = \begin{cases} X^f * X^f_{\mathrm{LPF}}\left(\theta\right) & |\theta| \le \pi - \pi c\\ 0 & |\theta| > \pi - \pi c\end{cases}, \tag{38}$$

hence, the final output is alias-free. Shift-equivariance w.r.t. continuous domain is derived from this, similarly to Eq. (33).

G. Implementation

Our theoretical results regarding discrete representation of continuous signals are based on infinite signals, which may seem impractical to real models which work on finite images. However, the results apply in a setting in which we assume that the continuous signals are periodic, and we finitely sample a single period. These assumptions practically limit our discussion to robustness to circular translations, which is the same setting that was considered in previous works [4, 33]. Next, we explain our implementation for the "ideal LPF", which was used in BlurPool and phase 3 of Algorithm 1 of the paper, and for the "reconstruction filter", which was used in Upsample in step 1 of Algorithm 1.

Both of these filters can be implemented using multiplication in the Fourier domain, working with DFT, which is defined for a finite signal with length N as

$$X^D[k] = \mathrm{DFT}\left(x\left[n\right]\right)[k] = \sum_{n=0}^{N-1} x\left[n\right] e^{-\frac{j2\pi}{N}kn}\ . \tag{39}$$

Similarly, the inverse of DFT is defined as:

$$x\left[n\right]=\mathrm{IDFT}\left(X^{D}\left[k\right]\right)\left[n\right]=\frac{1}{N}\sum_{k=0}^{N-1}X^{D}\left[k\right]e^{\frac{j2\pi}{N}kn}\,. \tag{40}$$

For simplicity, all our derivations are for 1-D signals. Our derivations trivially apply to the 2-D case by applying the filters separately on rows and on columns.

LPF We used a low-pass filter wherever it was necessary to prevent aliasing due to downsampling, namely in BlurPool layers that replace strided convolutions, and alias-free polynomial activations, before subsampling the polynomial results (Algorithm 1). We used an "ideal filter", *i.e.* a filter that eliminates all the frequencies above the cutoff ratio. Practically, this kind of filter can be implemented using multiplication in DFT domain:

$$\mathrm{LPF}_{\mathrm{cutoff},c}\left(x\left[n\right]\right)\left[n\right]=\mathrm{IDFT}\left(\mathrm{DFT}\left(x\left[n\right]\right)\left[k\right]H^{D}_{\mathrm{cutoff},c}\left[k\right]\right)\left[n\right]\ , \tag{41}$$

where $H^{D}_{\mathrm{cutoff,c}}$ is a "rectangle filter" defined for spatial dimension N and cutoff ratio $c\in[0,\ 1]$ as

$$H^{D}_{\mathrm{cutoff,c}}\left[k\right]=\begin{cases}1 & 0\leq k<\frac{N}{2}c\,,\\ 0 & \frac{N}{2}c\leq k\leq N-\frac{N}{2}c\,,\\ 1 & N-\frac{N}{2}c<k\leq N-1\,.\end{cases}\ . \tag{42}$$

Downsampling As mentioned above, all downsampling operations were performed in an alias-free manner, using low-pass filters before subsampling. For subsampling at factor s, we used LPF with cutoff ratio $c=\frac{1}{s}$. Then, we used subsampling with a fixed grid:

$$x_{\mathrm{down}}\left[n\right]=\mathrm{LPF}_{1/s}\left(x\left[n\right]\right)\left[sn\right] \tag{43}$$

Upsampling In the proof of Algorithm 1, we assume we use "ideal upsample", which can be interpreted as a re-sampling in a higher rate of the continuous signal, which was restored using sinc interpolation:

$$x_{\mathrm{up}_I}\left[m\right]=\sum_{n}x\left[n\right]\mathrm{sinc}\left(\frac{m-nI}{I}\right) \tag{44}$$

In practice, and specifically in a finite signal case, upsampling is performed in two steps: First, we use zero padding and get the intermediate signal

$$x_z\left[m\right]=\begin{cases}x\left[\frac{m}{I}\right] & m=kI\,,\\ 0 & \text{otherwise.}\end{cases} \tag{45}$$

Then, the zero-padded signal is convolved with sinc interpolation kernel. This step is equivalent to multiplication in Fourier domain with a rectangle, similarly to the LPF implementation.

$$\mathrm{Upsample}_I\left(x\left[n\right]\right)\left[n\right]=\mathrm{IDFT}\left(\mathrm{DFT}\left(x_z\left[n\right]\right)\left[k\right]H^{D}_{\mathrm{upsample},I}\left[k\right]\right)\left[n\right]\ , \tag{46}$$

Practically we used the following upsample kernel for a signal with spatial dimension N. For even N:

$$H^{D}_{\mathrm{upsample},I}\left[k\right]=\begin{cases}1 & 0\leq k<\frac{N}{2},\\ 1 & N\left(I-\frac{1}{2}\right)+1\leq k\leq IN-1\\ 0.5 & k=\frac{N}{2},k=N\left(I-\frac{1}{2}\right)\\ 0 & \text{else}\end{cases} \tag{47}$$

For odd N:

$$H^{D}_{\mathrm{upsample},I}\left[k\right]=\begin{cases}1 & 0\leq k<\lfloor\frac{N}{2}\rfloor\\ 1 & \lceil N\left(I-\frac{1}{2}\right)\rceil\leq k\leq 2N-1\\ 0 & \text{else}\end{cases} \tag{48}$$

The reason for that is that in practice, we cannot assume the Nyquist condition holds. Specifically, for a finite signal $x[n]$ with an even size N, we cannot assume that $X^D\left[\frac{N}{2}\right] = X^D\left[\frac{3N}{2}\right] = 0$. Note that for signals with even length, the $\frac{N}{2}$ component in the DFT domain represents the continual frequency of $\frac{\pi}{T}$, and thus, due to aliasing effect, we have $X^D\left[\frac{N}{2}\right] = X^D\left[\frac{3N}{2}\right] = X^F\left(\frac{\pi}{T}\right) + X^F\left(\frac{-\pi}{T}\right)$. For a representation of the continuous signal with a higher sampling rate (*e.g.* the upsampled signal), the overlap in this frequency would not happen, hence we multiply this component by $\frac{1}{2}$ to get $\frac{1}{2}\left(X^F\left(\frac{\pi}{T}\right) + X^F\left(\frac{-\pi}{T}\right)\right) = X^F\left(\frac{\pi}{T}\right)$. In this equation we use the assumption that $X^F\left(\frac{\pi}{T}\right) \in \mathbb{R}$. For a real signal The CTFT is conjugate symmetric, meaning $X^F\left(\frac{\pi}{T}\right) = X^F\left(\frac{-\pi}{T}\right)^*$. Therefore in case $X^F\left(\frac{\pi}{T}\right)$ has an imaginary component, it cannot be retrieved from the sum. A more detailed proof of this is given in Appendix H below.

The upsample method presented above is formally shown in Algorithm 2 with upsampling factor 2. To ease notations we denote below $H_2^D = H^D_{\text{upsample},2}$.

Algorithm 2 Upsample

```
**Input:** $x_N[n] \in \mathbb{R}^N$
$x_z \leftarrow \{x_N[0],\ 0,\ x_d[1],\ 0, ...,\ x_d[N-1],\ 0\}$
$x_{2N} \leftarrow \text{IDFT}\left\{\text{DFT}\{x_z\} H_2^D\right\}$
**Output:** $x_{2N}[n]$
```

H. Implementation proofs

In the following section, we provide formal proofs for the correctness of the filters presented in Appendix G In the setting of finite discrete signals, where we assume the continuous domain signals are periodic. For simplicity to the reader, we prove the correctness for 1-dimensional signal, and for upsampling at factor 2. The proofs can be easily generalized to 2D signals and a higher upsampling rate.

H.1. Definitions

Let $x(t)$ be a band-limited continuous signal, with CTFT $X^F(\omega)$ (as defined in Eq. (15)). $x(t)$ is periodic with period NT, *i.e.* $x(NT+t) = x(t)$. We define discrete sampling as:

$$x[n] = x(nT)\ , \tag{49}$$

and define a finite sampling as taking only one period of the discrete signal, *i.e.*

$$x_N[n] = \{x_0, ..., x_{N-1}\} = \{x(0), x(T), ..., x((N-1)T)\}\ . \tag{50}$$

H.2. Upsample

As noted in Appendix G, we assume that $X^F(\omega) = 0\ \ \forall|\omega| > \frac{\pi}{T}$ and that $X^F\left(\frac{\pi}{T}\right) \in \mathbb{R}$ (This a relaxation of Nyquist condition for which $X^F\left(\frac{\pi}{T}\right) = 0$). We prove the validity of the following method to upsample $x_N[n]$ to retrieve:

$$x_{2N}[n] = x\left(n\frac{T}{2}\right) = \left\{x(0), x\left(\frac{T}{2}\right), ..., x\left((2N-1)\frac{T}{2}\right)\right\}\ . \tag{51}$$

Claim 1 *Let $x(t)$ and $x_N[n]$ be a continuous signal and its finite discrete representation as defined in Appendix H.1. Then the output of Algorithm 2 is*

$$x_{2N}[n] = x\left(n\frac{T}{2}\right) = \left\{x(0), x\left(\frac{T}{2}\right), ..., x\left((2N-1)\frac{T}{2}\right)\right\}\ ,$$

using the described reconstruction filter H_2^D below:
For even N:

$$H_2^D[k] = \begin{cases} 1 & 0 \le k < \frac{N}{2}, \\ 1 & \frac{3N}{2}+1 \le k \le 2N-1 \\ 0.5 & k = \frac{N}{2}, k = \frac{3N}{2} \\ 0 & else \end{cases} \tag{52}$$

For odd N:

$$H_2^D[k] = \begin{cases} 1 & 0 \leq k < \lfloor \frac{N}{2} \rfloor \\ 1 & \lceil \frac{3N}{2} \rceil \leq k \leq 2N-1 \\ 0 & else \end{cases} \tag{53}$$

Proof (Claim 1) In order to proof Claim 1, we will show that the DFT of its output equals to X_{2N}^D, *i.e.* the DFT of the signal x_{2N} that is defined in Eq. (51).

H.2.1 DFT of zero-padded signal

Recall that in the first step of the Algorithm 2 we apply zero padding on the input x_N. For a finite discrete signal with length N, DFT is defined as Eq. (39):

$$X^D[k] = \sum_{n=0}^{N-1} x[n]e^{-\frac{j2\pi}{N}nk} \underbrace{=}_{W_N \triangleq e^{\frac{j2\pi}{N}}} \sum_{n=0}^{N-1} x[n]W_N^{-nk}$$

Claim 2 *Let X_N^D be the DFT of the input x_N of Algorithm 2 and X_z^D be the DFT of the zero-padded signal x_z in step 1 of Algorithm 2. Then:*

$$X_z^D[k] = \begin{cases} X_N^D[k] & k < N \\ X_N^D[k-N] & k \geq N \end{cases} \tag{54}$$

Proof (Claim 2)

$$\begin{aligned} X_z^D[k] &= \sum_{n=0}^{2N-1} x_z[n] W_{2N}^{-nk} && (55) \\ &\underset{(*)}{=} \sum_{n'=0}^{N-1} \left(x_z[2n'] W_{2N}^{-2n'k} + \underbrace{x_z[2n'+1]}_{=0\ \forall n'} W_{2N}^{-(2n'+1)k} \right) && (56) \\ &= \sum_{n'=0}^{N-1} x_z[2n'] W_{2N}^{-2n'k} && (57) \\ &\underset{(**)}{=} \sum_{n'=0}^{N-1} x_N[n'] W_N^{-n'k} && (58) \\ & && (59) \end{aligned}$$

In $(*)$ we separated the summation to even and odd components, and in $(**)$ we used the fact that

$$W_{2N}^{-2n'k} = e^{\frac{j2\pi}{2N}(-2n'k)} = e^{\frac{j2\pi}{N}(-n'k)} = W_N^{-n'k}.$$

Note that we got a sum of N components, yet $X_z^D[k]$ is defined for $k = 0, 1, ..., 2N-1$. For $k = 0, 1, ..., N-1$ we got the definition of DFT, meaning

$$X_z^D[k] = X_N^D[k]. \tag{60}$$

For $k = N, ..., 2N-1$ using the property

$$W_N^{-n'(N+k)} = \underbrace{W_N^{-n'N}}_{=e^{\frac{j2\pi}{N}(-n'N)}=e^{-j2\pi n'}=1} W_N^{-n'k} = W_N^{-n'k}$$

we get:

$$X_z^D[k] = \sum_{n'=0}^{N-1} x_N[n'] W_N^{-n'k} \tag{61}$$
$$= \sum_{n'=0}^{N-1} x_N[n'] W_N^{-n'(N+(k-N))} \tag{62}$$
$$= \sum_{n'=0}^{N-1} x_N[n'] W_N^{-n'(k-N)} \tag{63}$$
$$= X_N^D[k-N]\ . \tag{64}$$

H.2.2 Expressing DFT with CTFT

In the previous section we expressed X_z^D with X_N^D. In the following section we will express X_{2N}^D using X_N^D. In addition, we assume that N is even, and will show the other case afterwards. First, we express the DFT of x_N with the CTFT of x, by using their relations with DTFT:

$$X_N^D[k] = X_N^f\left(\theta = \frac{2\pi k}{N}\right) \tag{65}$$
$$= \frac{1}{T}\sum_{l=-\infty}^{\infty} X^F\left(\frac{\theta+2\pi l}{T}\right) \tag{66}$$
$$= \frac{1}{T}\sum_{l=-\infty}^{\infty} X^F\left(\frac{2\pi k}{NT}+\frac{2\pi l}{T}\right) \tag{67}$$
$$\underbrace{=}_{X^F(\omega)=0\ \ \forall|\omega|>\frac{\pi}{T}} \frac{1}{T}\left(X^F\left(\frac{2\pi k}{NT}\right)+X^F\left(\frac{2\pi k}{NT}-\frac{2\pi}{T}\right)\right) \tag{68}$$

$$\Rightarrow X_N^D[k] = \frac{1}{T}\begin{cases} X^F(\frac{2\pi k}{NT}) & 0 \le k \le \frac{N}{2}-1 \\ X^F(\frac{\pi}{T})+X^F(-\frac{\pi}{T}) & k=\frac{N}{2} \\ X^F(\frac{2\pi k}{NT}-\frac{2\pi}{T}) & \frac{N}{2}+1 \le k \le N-1 \end{cases}\ . \tag{69}$$

Note that the last two transitions hold considering the limited support of X^F :

$$\underbrace{X^F(\frac{2\pi k}{NT})}_{=0\ \ \forall k>\frac{N}{2}} + \underbrace{X^F(\frac{2\pi k}{NT}-\frac{2\pi}{T})}_{=0\ \ \forall k<\frac{N}{2}}\ .$$

Next, we will derive a similar expression for $X_{2N}^D[k]$:

$$X_{2N}^D[k] = X_{2N}^f\left(\theta=\frac{2\pi k}{2N}\right) \tag{70}$$
$$= \frac{1}{T}\sum_{l=-\infty}^{\infty} X^F\left(\frac{\theta+2\pi l}{T/2}\right) \tag{71}$$
$$= \frac{1}{T}\left(\underbrace{X^F\left(\frac{2\pi k}{NT}\right)}_{=0\ \ \forall k>\frac{N}{2}} + \underbrace{X^F\left(\frac{2\pi k}{NT}-\frac{2\pi}{T/2}\right)}_{=0\ \ \forall k<\frac{3N}{2}}\right) \tag{72}$$

(73)

We get:

$$\Rightarrow X_{2N}^D[k] = \frac{1}{T}\begin{cases} X^F(\frac{2\pi k}{NT}) & 0 \le k \le \frac{N}{2} \\ 0 & \frac{N}{2}+1 \le k \le \frac{3N}{2}-1 \\ X^F(\frac{2\pi k}{NT} - \frac{4\pi}{T}) & \frac{3N}{2} \le k \le 2N-1 \end{cases} \tag{74}$$

H.2.3 Expressing X_{2N}^D with X_N^D

Considering the second step of Algorithm 2, we need to show that applying the filter H_2^D on X_z^D yields X_{2N}^D, meaning

$$X_{2N}^D\,[k] = X_z^D\,[k]\,H_2^D\,.$$

By plugging $X_N^D\,[k]$ (Eq. (69)) in $X_z^D\,[k]$ (Eq. (54)) we get:

$$X_z^D\,[k] = \begin{cases} X_N^D\,[k] & k < N \\ X_N^D\,[k-N] & k \ge N \end{cases} \tag{75}$$

$$= \frac{1}{T}\begin{cases} X^F\left(\frac{2\pi k}{NT}\right) & 0 \le k \le \frac{N}{2}-1 \\ X^F\left(\frac{\pi}{T}\right) + X^F\left(-\frac{\pi}{T}\right) & k = \frac{N}{2} \\ X^F\left(\frac{2\pi k}{NT} - \frac{2\pi}{T}\right) & \frac{N}{2}+1 \le k \le N-1 \\ X^F\left(\frac{2\pi(k-N)}{NT}\right) & N \le k \le \frac{3N}{2}-1 \\ X^F\left(\frac{\pi}{T}\right) + X^F\left(-\frac{\pi}{T}\right) & k = \frac{3N}{2} \\ X^F\left(\frac{2\pi(k-N)}{NT} - \frac{2\pi}{T}\right) & \frac{3N}{2}+1 \le k \le 2N-1 \end{cases}. \tag{76}$$

Thus, by applying the filter

$$H_2^D\,[k] = \begin{cases} 1 & 0 \le k < \frac{N}{2}, \\ 1 & \frac{3N}{2}+1 \le k \le 2N-1 \\ 0.5 & k = \frac{N}{2}, k = \frac{3N}{2} \\ 0 & else \end{cases} \tag{77}$$

we get:

$$H_2^D\,[k]\,X_z^D\,[k] = \frac{1}{T}\begin{cases} X^F\left(\frac{2\pi k}{NT}\right) & 0 \le k \le \frac{N}{2}-1 \\ \frac{1}{2}\left(X^F\left(\frac{\pi}{T}\right) + X^F\left(-\frac{\pi}{T}\right)\right) & k = \frac{N}{2} \\ 0 & \frac{N}{2}+1 \le k \le N-1 \\ 0 & N \le k \le \frac{3N}{2}-1 \\ \frac{1}{2}\left(X^F\left(\frac{\pi}{T}\right) + X^F\left(-\frac{\pi}{T}\right)\right) & k = \frac{3N}{2} \\ X^F\left(\frac{2\pi(k-N)}{NT} - \frac{2\pi}{T}\right) & \frac{3N}{2}+1 \le k \le 2N-1 \end{cases}. \tag{78}$$

Note that for real $x(t)$, X^F is conjugate symmetric, and we assumed that $X^F\left(\frac{\pi}{T}\right) \in \mathbb{R}$. Therefore:

- for $k = \frac{N}{2}, \frac{3N}{2}$:

$$X^F\left(\frac{\pi}{T}\right) + X^F\left(-\frac{\pi}{T}\right) = 2X^F\left(\frac{\pi}{T}\right) = 2X^F\left(-\frac{\pi}{T}\right)\,. \tag{79}$$

- for $\frac{3N}{2}+1 \le k \le 2N-1$:

$$X_z^D\,[k] = X^F\left(\frac{2\pi\,(k-N)}{NT} - \frac{2\pi}{T}\right) = X^F\left(\frac{2\pi k}{NT} - \frac{2\pi N}{NT} - \frac{2\pi}{T}\right) = X^F\left(\frac{2\pi k}{NT} - \frac{4\pi}{T}\right)\,. \tag{80}$$

Thus we get:

$$H_2^D[k]\,X_z^D[k] = \frac{1}{T}\begin{cases} X^F\left(\frac{2\pi k}{NT}\right) & 0 \le k \le \frac{N}{2}-1 \\ X^F\left(\frac{\pi}{T}\right) & k = \frac{N}{2} \\ 0 & \frac{N}{2}+1 \le k \le N-1 \\ 0 & N \le k \le \frac{3N}{2}-1 \\ X^F\left(-\frac{\pi}{T}\right) & k = \frac{3N}{2} \\ X^F\left(\frac{2\pi k}{NT}-\frac{4\pi}{T}\right) & \frac{3N}{2}+1 \le k \le 2N-1 \end{cases} \tag{81}$$

$$= \frac{1}{T}\begin{cases} X^F\left(\frac{2\pi k}{NT}\right) & 0 \le k \le \frac{N}{2} \\ 0 & \frac{N}{2}+1 \le k \le \frac{3N}{2}-1 \\ X^F\left(\frac{2\pi k}{NT}-\frac{4\pi}{T}\right) & \frac{3N}{2} \le k \le 2N-1 \end{cases} \tag{82}$$

$$\underset{(74)}{=} X_{2N}^D[k]\,. \tag{83}$$

H.2.4 Odd N

By repeating the derivations of section Appendix H.2.2 for odd N we get:

$$X_N^D[k] = \frac{1}{T}\begin{cases} X^F\left(\frac{2\pi k}{NT}\right) & 0 \le k \le \lfloor\frac{N}{2}\rfloor \\ X^F\left(\frac{2\pi k}{NT}-\frac{2\pi}{T}\right) & \lceil\frac{N}{2}\rceil \le k \le N-1 \end{cases}. \tag{84}$$

$$X_{2N}^D[k] = \frac{1}{T}\begin{cases} X^F\left(\frac{2\pi k}{NT}\right) & 0 \le k \le \lfloor\frac{N}{2}\rfloor \\ 0 & \lceil\frac{N}{2}\rceil \le k \le \lfloor\frac{3N}{2}\rfloor \\ X^F\left(\frac{2\pi k}{NT}-\frac{4\pi}{T}\right) & \lceil\frac{3N}{2}\rceil \le k \le 2N-1 \end{cases}. \tag{85}$$

By plugging $X_N^D[k]$ (84) in $X_0^D[k]$ (54) we get:

$$X_z^D[k] = \begin{cases} X_N^D[k] & k < N \\ X_N^D[k-N] & k \ge N \end{cases} \tag{86}$$

$$= \frac{1}{T}\begin{cases} X^F\left(\frac{2\pi k}{NT}\right) & 0 \le k \le \lfloor\frac{N}{2}\rfloor \\ X^F\left(\frac{2\pi k}{NT}-\frac{2\pi}{T}\right) & \lceil\frac{N}{2}\rceil \le k \le N-1 \\ X^F\left(\frac{2\pi(k-N)}{NT}\right) & N \le k \le \lfloor\frac{3N}{2}\rfloor \\ X^F\left(\frac{2\pi(k-N)}{NT}-\frac{2\pi}{T}\right) & \lceil\frac{3N}{2}\rceil \le k \le 2N-1 \end{cases}. \tag{87}$$

Then, by applying the filter

$$H_2^D[k] = \begin{cases} 1 & 0 \le k < \lfloor\frac{N}{2}\rfloor, \\ 1 & \lceil\frac{3N}{2}\rceil \le k \le 2N-1 \\ 0 & else \end{cases} \tag{88}$$

we get:

$$H_2^D[k]\,X_z^D[k] = \frac{1}{T}\begin{cases} X^F\left(\frac{2\pi k}{NT}\right) & 0 \le k \le \lfloor\frac{N}{2}\rfloor \\ 0 & \lceil\frac{N}{2}\rceil \le k \le \lfloor\frac{3N}{2}\rfloor \\ X^F\left(\frac{2\pi(k-N)}{NT}-\frac{2\pi}{T}\right) & \lceil\frac{3N}{2}\rceil \le k \le 2N-1 \end{cases} \tag{89}$$

$$\underset{(85)}{=} X_{2N}^D[k]\,. \tag{90}$$

H.3. LPF

We use an "ideal LPF" before downsampling in BlurPool layers and in alias-free polynomial activations. As mentioned in Appendix F.1 "ideal LPF" in the context of subsampling of infinite discrete signals is a filter that completely eliminates all the frequencies beyond the Nyquist condition. *E.g.* in case of subsampling in factor I, *i.e.*

$$y\left[n\right] = x\left[In\right] \, ,$$

an "ideal LPF" in DTFT domain is implemented by multiplication with the filter:

$$H^{f}_{1/I}\left(\theta\right) = \begin{cases} 1 & \left|\theta\right| < \frac{\pi}{I} \, , \\ 0 & \left|\theta\right| \geq \frac{\pi}{I} \end{cases}$$

When considering the continuous domain, we expect the results to be equal to the discrete representation of the continuous signal after applying "ideal LPF", that is multiplication in CTFT domain with the filter

$$H^{F}_{1/I}\left(\theta\right) = \begin{cases} 1 & \left|\theta\right| < \frac{\pi}{TI} \\ 0 & \left|\theta\right| \geq \frac{\pi}{TI} \end{cases} \, ,$$

where T is the sample rate of $x\left[n\right]$.

Claim 3 *Let $x\left(t\right)$ and $x_N\left[n\right]$ be a continuous signal and its finite discrete representation as defined in Appendix H.1. In addition, let $x_{\text{LPF}}\left(t\right)$ be the continuous signal received by applying $H^{F}_{1/I}$ on $x\left(t\right)$ in the continuous domain,* i.e.*:*

$$X^{F}_{\text{LPF}}\left(\omega\right) = \begin{cases} X^{F}\left(\omega\right) & \left|\omega\right| < \frac{\pi}{TI} \\ 0 & \left|\omega\right| \geq \frac{\pi}{TI} \end{cases} \tag{91}$$

In addition, define $x_{\text{LPF}}\left(t\right)$ discrete representation as:

$$x_{\text{LPF}}\left[n\right] = \left\{x_{\text{LPF}}\left(0\right), \, x_{\text{LPF}}\left(T\right), \ldots, \, x_{\text{LPF}}\left(\left(N-1\right)T\right)\right\} \, .$$

Then applying $\text{LPF}_{1/I}$ on x_N gives

$$\text{LPF}_{1/I}\left(x_N\right) = x_{\text{LPF}}\left[n\right]$$

where $\text{LPF}_{1/I}$ is defined as multiplication in DFT domain with

$$H^{D}_{1/I}\left[k\right] = \begin{cases} 1 & 0 \leq k < \frac{N}{2I} \, , \\ 0 & \frac{N}{2I} \leq k \leq N - \frac{N}{2I} \, , \\ 1 & N - \frac{N}{2I} < k \leq N-1 \, . \end{cases} \, .$$

proof (Claim 3) Similarly to Appendix H.2.2, using the relations between DFT, DTFT and CTFT we get:

$$\begin{aligned} X^{D}_{\text{LPF}}\left[k\right] &= X^{f}_{\text{LPF}}\left(\theta = \frac{2\pi k}{N}\right) &(92)\\ &= \frac{1}{T}\sum_{l=-\infty}^{\infty} X^{F}_{\text{LPF}}\left(\frac{\theta + 2\pi l}{T}\right) &(93)\\ &= \frac{1}{T}\sum_{l=-\infty}^{\infty} X^{F}_{\text{LPF}}\left(\frac{2\pi k}{NT} + \frac{2\pi l}{T}\right) &(94)\\ &\underbrace{=}_{X^{F}\left(\omega\right)=0 \;\; \forall\left|\omega\right| > \frac{\pi}{T}} \frac{1}{T}\left(X^{F}_{\text{LPF}}\left(\frac{2\pi k}{NT}\right) + X^{F}_{\text{LPF}}\left(\frac{2\pi k}{NT} - \frac{2\pi}{T}\right)\right) \, . &(95) \end{aligned}$$

By plugging Eq. (91) we get:

$$X_{\text{LPF}}^{D}\left[k\right]=\frac{1}{T}\begin{cases}X^{F}\left(\frac{2\pi k}{2T}\right) & 0\leq k<\frac{N}{2I}\\ 0 & \frac{N}{2I}\leq k\leq N-\frac{N}{2I}\\ X^{F}\left(\frac{2\pi k}{NT}-\frac{2\pi}{T}\right) & N-\frac{N}{2I}<k\leq N-1\,.\end{cases} \tag{96}$$

Recall that x_N satisfies (Eq. (69)):

$$X_{\text{N}}^{D}\left[k\right]=\frac{1}{T}\begin{cases}X^{F}\left(\frac{2\pi k}{2T}\right) & 0\leq k<\frac{N}{2}-1\\ X^{F}\left(\frac{\pi}{T}\right)+X^{F}\left(-\frac{\pi}{T}\right) & k=\frac{N}{2}\\ X^{F}\left(\frac{2\pi k}{NT}-\frac{2\pi}{T}\right) & \frac{N}{2}<k\leq N-1\end{cases};$$

thus we get

$$X_{\text{LPF}}^{D}\left[k\right]=H_{1/I}^{D}\left[k\right]X^{D}\left[k\right]\ . \tag{97}$$