Adaptive Resolution Residual Networks — Generalizing Across Resolutions Easily and Efficiently

Anonymous authors
Paper under double-blind review

Abstract

The majority of signal data captured in the real world uses numerous sensors with different resolutions. In practice, however, most deep learning architectures are *fixed-resolution*; they consider a single resolution at training time and inference time. This is convenient to implement but fails to fully take advantage of the diverse signal data that exists. In contrast, other deep learning architectures are *adaptive-resolution*; they directly allow various resolutions to be processed at training time and inference time. This benefits robustness and computational efficiency but introduces difficult design constraints that hinder mainstream use. In this work, we address the shortcomings of both approaches by introducing *Adaptive Resolution Residual Networks* (ARRNs), which inherit the advantages of adaptive-resolution methods and the ease of use of fixed-resolution methods. We construct ARRNs from *Laplacian residuals*, which serve as generic adaptive-resolution adapters for fixed-resolution layers, and which allow instantly casting high-resolution ARRNs into low-resolution ARRNs by simply omitting high-resolution Laplacian residuals, thus reducing computational cost on low-resolution signals while guaranteeing *numerically identical* evaluation when using *perfect* smoothing kernels. We complement this novel component with *Laplacian dropout*, which regularizes for robustness to a distribution of lower resolutions and regularizes for numerical errors that may be induced by *approximate* smoothing kernels. We provide a solid grounding for the advantageous properties of ARRNs through a theoretical analysis based on *neural operators*, and empirically show that ARRNs embrace the challenge posed by diverse resolutions with greater flexibility, robustness, and computational efficiency.

In general, there is no universal resolution for image data or other signal data; there is, instead, a variety of resolutions that are contingent on the sensors used at the time of capture. This poses a challenge to deep learning methods, as adapting and generalizing across diverse resolutions is a complex architecture design problem that is often resolved at the cost of ease of use and compatibility. On one hand, *fixed-resolution* architectures offer a simple paradigm that is attractive from an engineering standpoint, but that falls short when faced with more diverse resolutions. This form of architecture is forced to cast various resolutions to a single resolution using interpolation, which is undesirable as it inflates computational cost when raising resolution; it conversely discards useful information when lowering resolution; it tends to yield brittle generalization overall as this process introduces a mismatch between the training distribution and the inference distribution. This approach nonetheless maintains its popularity as it imposes no special constraints on architectural design. On the other hand, *adaptive-resolution* architectures offer an approach that is generally driven by an analytical study of the link between discrete signals and continuous signals (Sitzmann et al., 2020; Li et al., 2020; Kovachki et al., 2021). This approach can guarantee more robust performance and greater computational efficiency across various resolutions. This approach has nonetheless been sparsely used as it has been broadly incompatible with mainstream layer types (Bartolucci et al., 2023). We propose a class of deep learning architectures that possess both the simplicity of fixed-resolution methods and the computational efficiency and robustness of adaptive-resolution methods. We build these architectures from two simple components: *Laplacian residuals*, which embed standard fixed-resolution layers in architectures that have adaptive-resolution capability, and *Laplacian dropout*, which acts both as a regularizer promoting strong lower resolution performance, and as a regularizer countering numerical errors that may be induced by approximate implementations of our method.

We situate ARRNs relative to prior works and provide background on Laplacian pyramids in section 1 and section 2. We provide an overview of the fundamental notions of signals that allow our work to be formulated and thouroughly define our notation in subsection A.1. We introduce Laplacian residuals and prove that high-resolution ARRNs can shed high-resolution Laplacian residuals to instantly yield low-resolution ARRNs that are computationally cheaper yet *numerically identical* when evaluated on low-resolution signals and when using *perfect* smoothing kernels subsection 3.1. We formulate Laplacian dropout in subsection 3.2 as a training augmentation that randomly omits Laplacian residuals and theoretically motivate its usefulness as a regularizer for low resolution robustness and as a regularizer for tolerance to *approximate* smoothing kernels in Laplacian residuals. We perform a set of experiments showing (subsection 4.1) that our method yields stronger robustness at lower resolutions compared to typical fixed-resolution models; (subsection 4.2) that our method enables significant computational savings through adaptation; (subsection 4.3) that our method is capable of generalizing across layer types in a way that far surpasses prior adaptive-resolution architectures; (subsection 4.4) that our theoretical guarantee for adaptation using *perfect* smoothing kernels holds empirically; (subsection 4.5) that our theoretical interpretation of the dual regularizing effect of Laplacian dropout also holds empirically.

1 Related Works

Here we review related works that allow the formulation of adaptive-resolution architectures. In subsection A.2, we provide a complimentary discussion of adaptive-resolution methods that are not directly motivated by the interplay between continuous signals and discrete signals.

Adaptive-resolution through neural operators. Li et al. (2020); Kovachki et al. (2021); Fanaskov & Oseledets (2022); Bartolucci et al. (2023) are neural architectures that act as *operators* mapping between inputs and outputs that are both *functions*. These methods are uniquely defined by their ability to equivalently operate on *continuous functions* represented as *discrete functions* of any specific resolution. For every layer that composes a neural operator, there must exists a translation process between a *continuous operator form* and a *discrete operator form* that respects an equivalence constraint which guarantees generalization across resolutions (Bartolucci et al., 2023). For many layers such as standard convolutional layers, transformer layers and adaptive pooling layers, the equivalence constraint is violated, making them incompatible with this technique without substantial alteration (Bartolucci et al., 2023). Our method is a form of neural operator, but unlike prior methods, it does not require the layers to conform to this difficult constraint, as Laplacian residuals guarantee its satisfaction independently of the choice of layers.

Adaptive-resolution through implicit neural representations. Park et al. (2019); Mescheder et al. (2019); Sitzmann et al. (2020); Mildenhall et al. (2021); Chen et al. (2021); Yang et al. (2021); Lee & Jin (2022); Xu et al. (2022) are neural architectures whose inputs and outputs are also *functions*. These methods are distinguished by way they manipulate the functions they operate on; they treat *discrete functions* as samples of underlying *continuous functions* that are to be approximately reconstructed as *neurally parameterized functions*. Some of these methods simply leverage neurally parameterized functions to allow evaluation of functions at unseen sample locations, which has proven highly successful for reconstructing volumetric data, image data and light field data (Park et al., 2019; Mescheder et al., 2019; Sitzmann et al., 2020; Mildenhall et al., 2021). Some variants of these methods instead represent and manipulate neurally parameterized functions by partitioning their parameter space into a set that is shared across all data points reconstructed during training, which provides a common structure, and a set that is specific to each individual data point reconstructed during training, which acts as a latent representation. This form of representation has been successfully used in super-resolution methods for image data (Chen et al., 2021; Lee & Jin, 2022; Yang et al., 2021). This form of representation has important shortcomings when it comes to mapping functions to functions in a resolution-agnostic way, as is required in classification, segmentation or diffusion on image data. Since the symmetries of signals are entangled in latent representations, the incorporation of simple inductive biases over signals becomes problematic, and the majority of mainstream layers tend to be unsuitable; even the implementation of convolutional layers is problematic (Xu et al., 2022). In contrast, our method uses the most ubiquitous form of signal representation which is immediately compatible with mainstream layers.

Residual connections Singh et al. (2021) incorporates filtering operations within residuals to separate the frequency content of convolutional networks, although it provides no mechanism for adaptive resolution. Lai et al. (2017) uses Laplacian pyramids to solve super-resolution tasks with adaptive *output* resolution, with residuals ordered by *increasing* resolution. This is unlike our method, which is well suited to tasks with adaptive *input* resolution, with residuals ordered by *decreasing* resolution. Huang et al. (2016) implements a form of dropout where the layers nested within residual blocks may be bypassed randomly. This is somewhat similar to Laplacian dropout, however, this is not equivalent to a form of bandwidth augmentation and does not result in the same improved robustness to various resolutions we show in subsection 4.1.

2 Background

In our overview of background material, we introduce Laplacian pyramids as a stepping stone for the formulation of Laplacian residuals. In addition, we provide a discussion of signals in subsection A.1 that introduces the notation and fundamental concepts behind this work in a way that is broadly accessible to readers unfamiliar with these ideas, and that should also clarify details valued by more experienced readers.

2.1 Laplacian pyramids

In this section, we introduce Laplacian pyramids (Burt & Adelson, 1987), as they closely relate to Laplacian residuals, and as they have often been used in vision techniques to decompose signals across a range of resolutions.

Laplacian pyramids take some signal s, and perform a series of m convolutions with smoothing kernels ϕ_n to generate lower bandwidth signals p_n^{low} that each are incrementally smoother than the previous one. Laplacian pyramids then generate difference signals p_n^{diff} that isolate the part of the signal that was lost at each incremental bandwidth reduction, which intuitively correspond to a certain level of detail of the original signal. The operations that compose a Laplacian pyramid can be captured by a base definition and two simple recursive definitions:

$$p_0^{\text{low}} = s * \phi_1 \qquad \in S_1 \tag{1}$$

$$p_n^{\text{low}} = p_{n-1}^{\text{low}} * \phi_{n+1} \qquad \in S_{n+1} \tag{2}$$

$$p_n^{\text{diff}} = p_{n-1}^{\text{low}} - p_n^{\text{low}} \qquad \in S_n \cap \overline{S_{n+1}} \tag{3}$$

In Figure 1, we summarize the recursive formulation of Laplacian pyramids in a three-block pyramid; this is intended to allow easy comparison with the Laplacian residuals we illustrate in Figure 2.

Laplacian pyramids are convenient to implement, since each reduction in bandwidth allows a reduction in resolution, as hinted by the notation above, which explicitly includes the bandwidth S_n to highlight the coinciding resolution $|X_n|$ that may be used.

Laplacian pyramids also allow reconstructing the original signal up to an arbitrary bandwidth using only the last lower bandwidth signal p_m^{low} and a variable number of difference signals p_n^{diff}:

$$s * \phi_n = p_n^{\text{diff}} + p_{n+1}^{\text{diff}} + \cdots + p_{m-1}^{\text{diff}} + p_m^{\text{diff}} + p_m^{\text{low}} \in S_n \tag{4}$$

Laplacian pyramids are typically constructed using Gaussian smoothing kernels, which violate Equation 33 and introduce errors in the sampling process as a consequence. We require *perfect* Whittaker-Shannon smoothing kernels for the exact computation reduction property we show in Laplacian pyramids just below and in Laplacian residuals later in subsection 3.1. We however show in subsection 4.5 that Laplacian dropout enables ARRNs to learn to correct these errors.

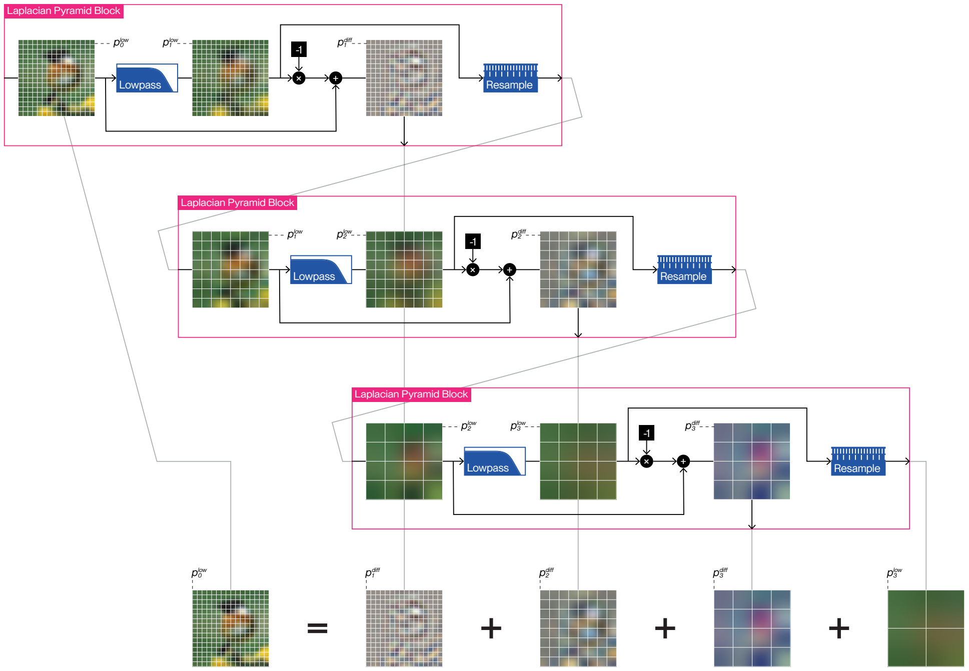

Figure 1: Visualization of a Laplacian pyramid showing blocks that correspond to recursion iterations. The starting signal p_0^{low} is shown at the top left. The blocks are chained one after the other, from top-to-bottom, and produce the difference signals p_1^{diff}, p_2^{diff}, p_3^{diff}. The blocks each start with the signal p_{n-1}^{low}, produce a lower bandwidth signal p_n^{low} moving right, and finally produce a difference signal p_n^{diff} moving right again. Together, p_1^{diff}, p_2^{diff}, p_3^{diff} and p_3^{low} sum to the original signal p_0^{low}; the Laplacian pyramid produces a form of linear decomposition.

Laplacian pyramids have the desirable ability to adapt to low resolutions by skipping computations, as we may intuit from Equation 4 and Figure 1; if we were to start with an 8×8 image of the bird, it would appear reasonable to skip the 16×16 level of the Laplacian pyramid. We formally show this by considering a signal in discrete form $s[\,\cdot\,]_u$ with a low resolution $|X_u|$ and a corresponding low bandwidth constraint S_u. We are interested in what happens up to the level n of the Laplacian pyramid, where n is the highest level that sits just at or above the resolution $|X_n| \geq |X_u|$ and the bandwidth constraint $S_n \supseteq S_u$ of the original signal. We observe that all of the smoothing filters $\phi_1, \cdots, \phi_n$ associated with prior levels of the Laplacian pyramids leave the original signal s unchanged since $s \in S_u \subseteq S_n$ and since $S_n \subset \cdots \subset S_1$, which induces a trail of zero terms in the expansion of the recursive terms Equation 2 and Equation 3, as shown in Equation 14 on the right.

$$
\begin{aligned}
\implies p_0^{\text{low}} &= s * \phi_1 && (5)\\
&= s && (6)\\
\implies p_1^{\text{low}} &= p_0^{\text{low}} * \phi_2 && (7)\\
&= s && (8)\\
\implies p_1^{\text{diff}} &= p_0^{\text{low}} - p_1^{\text{low}} && (9)\\
&= 0 && (10)\\
&\vdots\\
\implies p_{n-1}^{\text{low}} &= p_{n-2}^{\text{low}} * \phi_n && (11)\\
&= s && (12)\\
\implies p_{n-1}^{\text{diff}} &= p_{n-2}^{\text{low}} - p_{n-1}^{\text{low}} && (13)\\
&= 0 && (14)
\end{aligned}
$$

This allows directly setting $p_{n-1}^{\text{low}} = s$ and carrying out computation only to recover difference terms starting at p_n^{diff}, skipping all difference terms $p_1^{\text{diff}}, \ \ldots \ , p_{n-1}^{\text{diff}}$. We later design Laplacian residuals to reproduce exactly this desirable behaviour such that we can adapt deep learning architectures to lower resolutions by skipping computations.

3 Method

In this section, we build towards *Laplacian residuals* (subsection 3.1), which are designed to allow the construction of adaptive-resolution architectures from standard fixed-resolution layers, and *Laplacian dropout* (subsection 3.2), which both serves as a regularizer for robustness at lower resolution, and a regularizer for error-correction when imperfect smoothing kernels are used.

3.1 Laplacian residuals for adaptive-resolution deep learning

Laplacian residuals are alike to Laplacian pyramids in the way they separate a signal into a sum of progressively lower bandwidth signals, and in the way they are able to operate at lower resolution by simply skipping computations. However, Laplacian residuals crucially differ in their ability to incorporate neural architectural blocks that enable deep learning.

Laplacian residuals are formulated as adaptive-resolution layers *(operators on continuous signals)* $r_n : (\mathbf{X} \to \mathbb{R}^{f_n}) \in S_n \to (\mathbf{X} \to \mathbb{R}^{f_{n+1}}) \in S_{n+1}$ that incorporate neural architectural blocks $b_n : (\mathbf{X}_n \to \mathbb{R}^{f_n}) \to (\mathbf{X}_n \to \mathbb{R}^{f_n})$ that are fixed-resolution layers *(operators on discrete signals)*. This formulation enables wide compatibility with mainstream fixed-resolution layers that is unseen in prior adaptive-resolution methods. This compatibility is only conditional on the neural architectural block b_n producing a constant everywhere when its input is zero everywhere Equation 5, which is trivially guaranteed by linear layers, activation layers, convolutional layers, batch normalization layers, some transformer layers, and any composition of layers that individually meet this condition:

$$b_n\{0\} = a \text{ where } a \in \mathbb{R}^{f_n} \tag{5}$$

The base case and recursive cases seen in Laplacian residuals are nearly identical to those of Laplacian pyramids, aside from including a linear projection $\mathbf{A}_0 : \mathbb{R}^{f_1 \times f_0}$ to raise the feature dimensionality from $f_0 \in \mathbb{N}^+$ to $f_1 \in \mathbb{N}^+$ before the first neural architectural block:

$$r_0 = \mathbf{A}_0 s * \phi_1 \qquad \in S_1 \tag{6}$$

$$r_n^{\text{low}} = r_{n-1} * \phi_{n+1} \qquad \in S_{n+1} \tag{7}$$

$$r_n^{\text{diff}} = r_{n-1} - r_n^{\text{low}} \qquad \in S_n \cap \overline{S_{n+1}} \tag{8}$$

The neural architectural block b_n receives the difference signal r_n^{diff} as its input, and then sums its output with the lower bandwidth signal r_n^{low} before passing it to the next Laplacian residual. The signals are combined with some additional processing, which we define below and motivate more concretely next:

$$r_n = \mathbf{A}_n(b_n\{r_n^{\text{diff}}\} * \psi * \phi_{n+1} + r_n^{\text{low}}) \in S_{n+1} \tag{9}$$

In Figure 2, we summarize the recursive formulation of Laplacian residuals into a diagram that illustrates a three-block ARRN, which allows easy comparison with the Laplacian pyramid shown in Figure 1.

We include a constant rejection kernel ψ in Equation 9 to replicate the same computation skipping behaviour seen in Laplacian pyramids. This effectively subtracts the mean and ensures the neural architectural block b_n contributes zero to the residual signal r_n if the difference signal r_n^{diff} is zero, as shown in Equation 10. We obtain this result thanks to the constraint set on the neural architectural block b_n in Equation 5:

$$b_n\{0\} * \psi = 0 \tag{10}$$

We include a smoothing kernel ϕ_{n+1} in Equation 9 to ensure the output bandwidth of r_n coincides with the input bandwidth of r_{n+1}, which is necessary for Laplacian residuals to follow the same general structure as Laplacian pyramids.

We also incorporate a projection matrix $\mathbf{A}_n : \mathbb{R}^{f_{n+1} \times f_n}$ in Equation 9 to allow raising the feature dimensionality from $f_n \in \mathbb{N}^+$ to $f_{n+1} \in \mathbb{N}^+$ at the end of each Laplacian residual, so that more capacity can be allocated to later Laplacian residuals.

We note that the neural architectural block $b_n\{\,\cdot\,\}$ in Equation 9 is written in shorthand, and stands for the more terse expression $\mathfrak{I}_n\{b_n\{\mathfrak{S}_n\{\,\cdot\,\}\}\}$. This is a formal trick that enables our analysis by casting the neural architectural block from an *operator on discrete signals* $(\mathbf{X}_n \to \mathbb{R}^{f_n}) \to (\mathbf{X}_n \to \mathbb{R}^{f_n})$ to an *operator on continuous signals* $(\mathbf{X} \to \mathbb{R}^{f_n}) \in S_n \to (\mathbf{X} \to \mathbb{R}^{f_n}) \in S_n$ that is equivalent in the sense of Equation 34. This does not directly reflect the implementation of the method, as all linear operators are analytically composed then cast to their discrete form to maximize computational efficiency.

We add that we can alter the formulation of Equation 9 to let the neural architectural block perform a parameterized downsampling operation $(\mathbf{X}_n \to \mathbb{R}^{f_n}) \to (\mathbf{X}_{n+1} \to \mathbb{R}^{f_n})$ by changing the interpolation operator discussed above from $\mathfrak{I}_n : (\mathbf{X}_n \to \mathbb{R}^{f_n}) \to (\mathbf{X} \to \mathbb{R}^{f_n}) \in S_n$ to $\mathfrak{I}_{n+1} : (\mathbf{X}_{n+1} \to \mathbb{R}^{f_n}) \to (\mathbf{X} \to \mathbb{R}^{f_n}) \in S_{n+1}$ and by dropping the smoothing kernel ϕ_{n+1} from Equation 9.

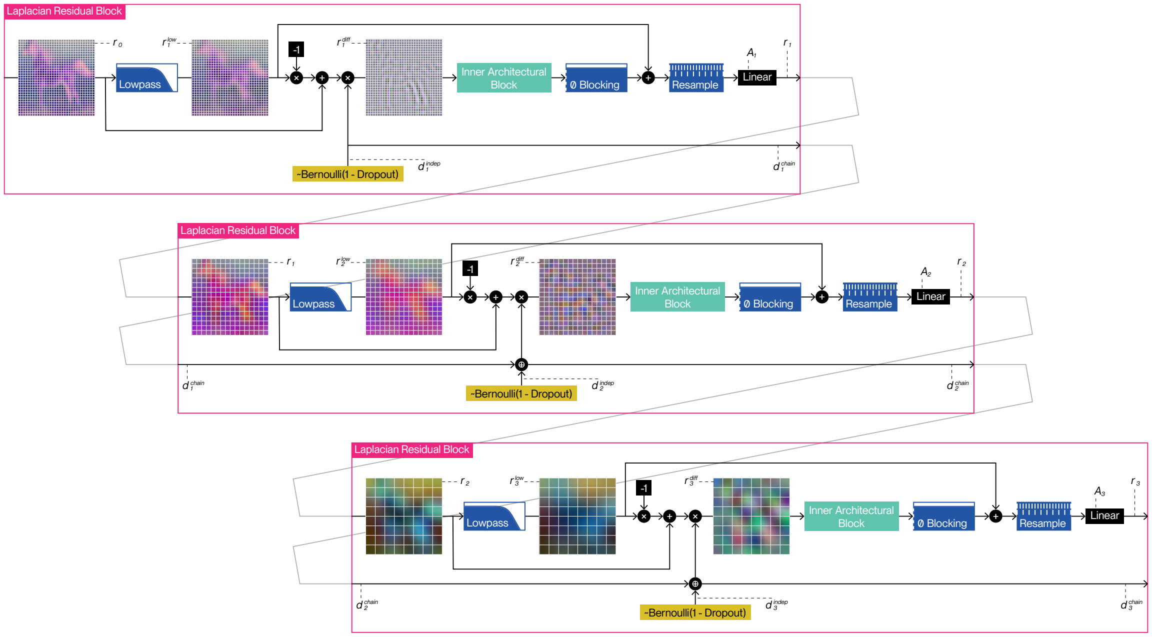
Laplacian Residual Block
r_0
r_1^{low}
Lowpass
-1
r_1^{diff}
Inner Architectural Block
∅ Blocking
Resample
A_1
Linear
r_1
-Bernoulli(1 - Dropout)
d_1^{indep}
d_1^{chain}
Laplacian Residual Block
r_1
r_2^{low}
Lowpass
-1
r_2^{diff}
Inner Architectural Block
∅ Blocking
Resample
A_2
Linear
r_2
d_1^{chain}
-Bernoulli(1 - Dropout)
d_2^{indep}
d_2^{chain}
Laplacian Residual Block
r_2
r_3^{low}
Lowpass
-1
r_3^{diff}
Inner Architectural Block
∅ Blocking
Resample
A_3
Linear
r_3
d_2^{chain}
-Bernoulli(1 - Dropout)
d_3^{indep}
d_3^{chain}

Figure 2: Visualization of an ARRN showing blocks that correspond to Laplacian residuals, displaying a principal component analysis of feature maps. The starting signal r_0 is shown at the top left. The blocks are chained one after the other, from top-to-bottom. The blocks each start with the previous residual signal r_{n-1}, and, moving left-to-right, produce a lower bandwidth signal r_n^{low} and difference signal r_n^{diff} as in Laplacian pyramids, gate the difference signal against the chained Laplacian dropout variable d_n^{chain}, apply the neural architectural block b_n, apply the zero-blocking filter ψ, sum the lower bandwidth signal r_n^{low}, apply the smoothing kernel ϕ_{n+1} and resample, then finally project through $\mathbf{A}_n$ to produce the residual signal r_n.

Adaptation to lower resolution signals with perfect smoothing kernels. We guarantee that architectures built from a series of Laplacian residuals can adapt to lower resolution signals by simply skipping the computation of higher resolution Laplacian residuals *without causing any numerical perturbation* when using *perfect* smoothing kernels. This guarantee provides strong theoretical backing to the validity of our method, and is also supported by empirical evidence in subsection 4.4.

We show this property following the same argument leveraged to derive Equation 14 in the case of Laplacian pyramids. We take a signal in discrete form $s[\,\cdot\,]_u$ with a low resolution $|X_u|$ and a corresponding low bandwidth constraint S_u. We consider Laplacian residuals up to the level n of the architecture, where n is the highest level that sits just at or above the resolution $|X_n| \geq |X_u|$ and the bandwidth constraint $S_n \supseteq S_u$ of the original signal. We observe that all of the smoothing filters $\phi_1, \cdots, \phi_n$ associated with prior levels of the architecture leave the original signal s unchanged since $s \in S_u \subseteq S_n$ and since $S_n \subset \cdots \subset S_1$, which induces a trail of zero terms in the expansion of the recursive terms Equation 7, Equation 8, and Equation 9, as shown in Equation 24:

$$
\begin{aligned}
\implies r_0 &= \mathbf{A}_0 s * \phi_1 && (11)\\
&= \mathbf{A}_0 s && (12)\\
\implies r_1^{\text{low}} &= r_0 * \phi_2 && (13)\\
&= \mathbf{A}_0 s && (14)\\
\implies r_1^{\text{diff}} &= r_0 - r_1^{\text{low}} && (15)\\
&= 0 && (16)\\
\implies r_1 &= \mathbf{A}_1(b_1\{r_1^{\text{diff}}\} * \psi * \phi_2 + r_1^{\text{low}}) && (17)\\
&= \mathbf{A}_1 \mathbf{A}_0 s && (18)\\
&\vdots \\
\implies r_{n-1}^{\text{low}} &= r_{n-2} * \phi_n && (19)\\
&= \mathbf{A}_{n-2} \cdots \mathbf{A}_0 s && (20)\\
\implies r_{n-1}^{\text{diff}} &= r_{n-2} - r_{n-1}^{\text{low}} && (21)\\
&= 0 && (22)\\
\implies r_{n-1} &= \mathbf{A}_{n-1}(b_{n-1}\{r_{n-1}^{\text{diff}}\} * \psi * \phi_n + r_{n-1}^{\text{low}}) && (23)\\
&= \mathbf{A}_{n-1} \cdots \mathbf{A}_0 s && (24)
\end{aligned}
$$

This can be leveraged to evaluate a chain of Laplacian residuals at a lower resolution by simply discarding higher-resolution Laplacian residuals and only considering the chain of linear projections $\mathbf{A}_{n-1} \cdots \mathbf{A}_0$ that is carried over. This provides adaptive-resolution capability to fixed-resolution layers without difficult design constraints, and improves computation efficiency without compromising numerical accuracy in any way.

We can use this result precisely state the equivalence between evaluation using all Laplacian residuals (Equation 25) and evaluation using the strictly necessary Laplacian residuals (Equation 26):

$$
\begin{aligned}
&\mathfrak{S}_m\{r_m\{\,\cdots\, r_0\{\mathfrak{S}_0\{\mathfrak{I}_u\{s[\,\cdot\,]_u\}\}\}\,\cdots\,\}\} && (25)\\
=\ &\mathfrak{S}_m\{r_m\{\,\cdots\, r_n\{\mathbf{A}_{n-1}\cdots\mathbf{A}_0\mathfrak{S}_n\{\mathfrak{I}_u\{s[\,\cdot\,]_u\}\}\}\,\cdots\,\}\} && (26)
\end{aligned}
$$

Adaptation to lower resolution signals with approximate smoothing kernels. We show that using *approximate* smoothing kernels *causes some numerical perturbation* when skipping the computation of higher resolution Laplacian residuals. This observation motivates the use of Laplacian dropout, a training augmentation we introduce in subsection 3.2 that addresses this limitation while also improving robustness.

When using imperfect smoothing kernels $\tilde{\phi}_n \approx \phi_n$, the guarantee we provide does not hold exactly. We consider the case case where ϕ_n would leave a signal s unchanged, and note that $\tilde{\phi}_n$ would disturb the signal s by a small error signal ϵ_n:

$$
s * \phi_n = s \implies s * \tilde{\phi}_n = s + \underbrace{s * (\tilde{\phi}_n - \phi_n)}_{\epsilon_n} \qquad (27)
$$

We highlight that the discrepancy above would induce a small error term ϵ_n in every intermediate zero term that leads to Equation 24, and therefore discarding unnecessary Laplacian residuals (Equation 26) would not be exactly equivalent to retaining all Laplacian residuals (Equation 25). We note that this is not simply constrained to a linear effect, as ϵ_1 will for instance affect b_1, which has nonlinear behavior.

3.2 Laplacian dropout for effective generalization

In this section, we introduce Laplacian dropout, a training augmentation that is specially tailored to improve the performance of our method by taking advantage of the structure of Laplacian residuals, and that comes at effectively no computational cost.

We formulate Laplacian dropout by following the intuition that Laplacian residuals can be randomly disabled during training to improve generalization. We only allow disabling consecutive Laplacian residuals using the logical or operator to ensure that Laplacian dropout does not cut intermediate information flow:

$$d_n^{\text{indep}} \sim B(1 - p_n) \tag{28}$$

$$d_n^{\text{chain}} = d_n^{\text{indep}} \oplus d_{n-1}^{\text{chain}} \tag{29}$$

$$r_n^{\text{diff}} = d_n^{\text{chain}}(r_{n-1} - r_n^{\text{low}}) \tag{30}$$

Next, we provide a theoretical interpretation that identifies two distinct purposes that Laplacian dropout fulfills in our method. We see this dual utility as a highly desirable feature of Laplacian dropout.

Regularization of robustness at lower resolution. Since Laplacian dropout truncates Laplacian residuals in the same way they are truncated when adapted to lower resolutions, Laplacian dropout is identical to randomly lowering resolution when using perfect smoothing kernels. This acts as a training augmentation that promotes robustness over a *distribution* of lower resolutions. We perform a set of classification tasks in subsection 4.1 that show this regularizing effect sometimes *doubling* accuracy over certain lower resolutions without adversely affecting accuracy at the highest resolution.

Regularization of errors introduced by approximate smoothing kernels. Since Laplacian dropout truncates Laplacian residuals in the same way they are truncated when adapted to lower resolutions, Laplacian dropout exactly replicates numerical errors produced by approximate smoothing kernels in Equation 27. This allows learning a form of error compensation that offsets the effect of approximate smoothing kernels. We demonstrate this allows the use of very coarsely approximated smoothing kernels in subsection 4.5 that otherwise impart a significant performance penalty on our method.

4 Experiments

We present a set of experiments that demonstrate our method's robustness across resolutions, its computational efficiency, and its ease of use. We show (subsection 4.1) that our method is highly robust across diverse resolutions; (subsection 4.2) that adaptation provides our method with a significant computational advantage; (subsection 4.3) that our method can generalize across layer types in a way that exceeds the capabilities of prior adaptive-resolution architectures; (subsection 4.4) that our theoretical guarantee for adaptation to lower resolutions with *perfect* smoothing kernels holds empirically; and (subsection 4.5) that our theoretical interpretation of the dual regularization effect of Laplacian dropout coincides with the behaviour we observe empirically when isolating the effect of *approximate* smoothing kernels, which improves the robustness of our method through two distinct mechanisms.

Experiment design. We compare models in terms of their robustness across resolutions, their computational scaling relative to resolution, and their ease of construction. We follow a typical use case for our method, where we train each model at a single resolution and then evaluate over a range of resolutions. We consider the fluctuation of accuracy and inference time over resolution as the metrics of interest for our discussion. We perform a set of classification tasks that require models to effectively leverage the information of low-resolution to medium-resolution images; **CIFAR10** (32×32) (Krizhevsky et al., 2009), **CIFAR100** (32×32) (Krizhevsky et al., 2009), **TinyImageNet** (64×64) (Le & Yang, 2015) and **STL10** (96×96) (Coates et al., 2011).

Model design and selection. We apply our method by constructing adaptive-resolution models that rely on commonly used fixed-resolution layers. For most of our experiments (subsection 4.1, subsection 4.2, subsection 4.4 and subsection 4.5), we take inspiration from MobileNetV2 (Sandler et al., 2018) and EfficientNetV2 (Tan & Le, 2021) to construct ARRNs that are documented in subsection A.3. For the experiment that investigates generalization across layer types (subsection 4.3), we instead construct ARRNs by transplanting layers that are found across a range of mainstream fixed-resolution architectures: **ResNet18**, **ResNet50**, **ResNet101** (11.1M-42.5M) He et al. (2016), **WideResNet50V2**, **WideResNet101V2** (66.8M-124M) (Zagoruyko & Komodakis, 2016), **MobileNetV3Small**, **MobileNetV3Large**(1.52M-4.21M) Howard et al. (2019). We splice the sequence of layers that composes each fixed-resolution architecture at points where resolution changes occur and nest each resulting subsequence of layers in a Laplacian residual with matching resolution. We discard the first two Laplacian residuals for MobileNetV3, as the resolution of the tailing Laplacian residuals otherwise becomes very small. For our choice of baseline methods, we consider mainstream fixed-resolution architectures to show they compromise robustness across diverse resolutions, yet they have no substantial advantage in ease of implementation or compatibility with existing layers compared to our method, as demonstrated by our experiment on generalization across layer types. We include all the fixed-resolution architectures above in this comparison, along with **EfficientNetV2S**, **EfficientNetV2M**, **EfficientNetV2L** (20.2M-117.2M) (Tan & Le, 2021). For the experiments that validate our theoretical analysis (subsection 4.4 and subsection 4.5), we perform an ablation study over the quality of the smoothing kernel, the use of Laplacian dropout at training time, and the use of adaptation at inference time.

Model training and evaluation. All models are trained for 100 epochs at the full dataset resolution with identical hyperparameters that are described in subsection A.3. All models are then evaluated at the full dataset resolution and at a range of lower resolutions that are synthetically generated. In the case of fixed-resolution models, the lower resolution signals are interpolated to the resolution supported by the models. In the case of our models, the lower resolution signals are directly processed if they coincide with the resolution of a Laplacian residual; if they fall in between the resolution of two successive Laplacian residuals, the higher resolution is generally interpolated to, as required by our theoretical proof (Equation 24). However, the lower resolution is sometimes interpolated to instead as it can result in more robust behavior that is computationally cheaper. This is the case with TinyImageNet and STL10 in subsection 4.1 and subsection 4.2, and with TinyImageNet in subsection 4.3. This effect likely results from more consistent statistical properties encountered when only evaluating Laplacian residuals that have full access to the part of the signal they usually address.

4.1 Robustness and the effectiveness of Laplacian dropout

We demonstrate that our method allows for greater low-resolution robustness than mainstream methods without compromise in high-resolution performance. Figure 3 shows four ablations of our method corresponding to permutations of two sets: either with Laplacian dropout (red lines) or without Laplacian dropout (black lines); and either with adaptation (solid lines) or without adaptation (dashed lines). We see that with Laplacian dropout and with adaptation (full red lines), our method outperforms every baseline method across every resolution and every dataset. We also find that, in contrast, without Laplacian dropout (black lines), our method shows much weaker generalization across resolutions, clearly demonstrating Laplacian dropout's effectiveness as a regularizer for robustness to diverse resolutions.

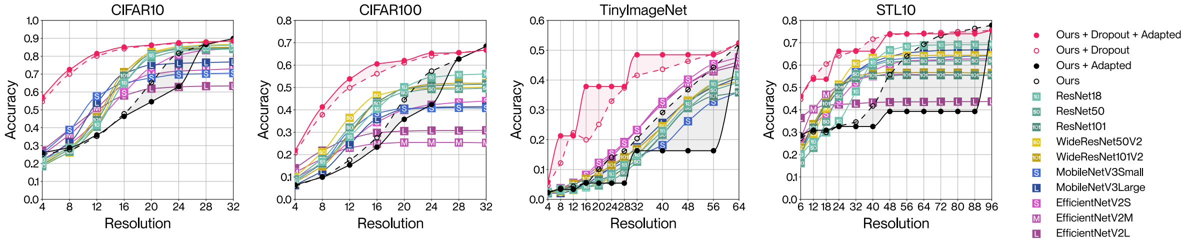

Figure 3: Accuracy of all methods at various resolutions, where each model is trained at the full dataset resolution and tested at a range of lower resolutions. Our method (red full line) displays the best accuracy at the highest resolution and robustly maintains its accuracy at lower resolutions.

4.2 Computational efficiency

We confirm the computational savings granted by adaptation by performing time measurements on the previous experiment. We use CUDA event timers and CUDA synchronization barriers around the forward pass of the network to eliminate other sources of overhead, such as data loading, and sum these time increments over all batches of the full dataset. We repeat this process 10 times and pick the median to reduce the effect of outliers. Figure 4 shows the inference time of ARRNs with adaptation (full red lines) and without adaptation (dashed red lines). Our method significantly reduces its computational cost (highlighted by the shaded area) by requiring the evaluation of a lower number of Laplacian residuals at lower resolutions. Our method also has a reasonable inference time relative to well-engineered standard methods.

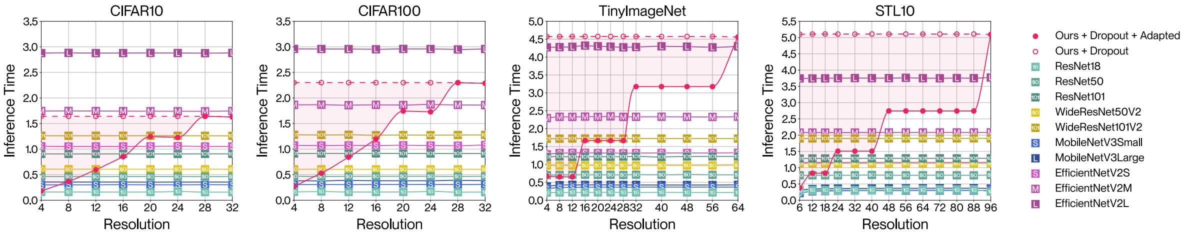

Figure 4: Inference time of all methods at various resolutions, where the inference time for the entire dataset is considered. Our method (red full line) can adapt to lower resolutions by skipping Laplacian residuals, which results in significant computational savings (highlighted by the shaded area) compared to using all Laplacian residuals (red dashed line). Our method also displays a reasonable inference time relative to typical convolutional neural networks despite not having a highly optimized implementation.

4.3 Generalization across layer types

We demonstrate the ease of use of our method and its capability to incorporate various layer types by constructing adaptive-resolution architectures from a range of mainstream fixed-resolution architectures (**ResNet18**, **ResNet50**, **ResNet101**, **WideResNet50V2**, **WideResNet101V2**, **MobileNetV3Small** and **MobileNetV3Large**). Figure 5 compares the accuracy of architectures in adaptive-resolution form (red box plots) and in fixed-resolution form (green box plots). The distribution of accuracies of the seven underlying architectures in the adaptive-resolution group and fixed-resolution group is visually conveyed in a decluttered format by drawing a small box plot at every resolution. Our method consistently delivers better low-resolution performance, and similar or better high-resolution performance. Our method achieves this while generalizing beyond the abilities of prior adaptive-resolution architectures, as it is able to incorporate the fixed-resolution layers used in this experiment.

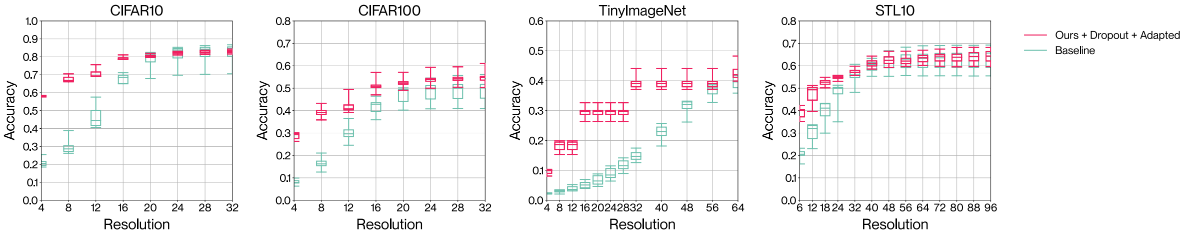

Figure 5: Accuracy of two groups of methods at various resolutions, where 7 adaptive-resolution architectures (in red) are constructed by taking 7 fixed-resolution architectures (in green) and wrapping their layers in Laplacian residuals. Our method yields architectures that have much stronger low-resolution performance, and similar or better high-resolution performance, which demonstrates ease of use and wide compatibility with mainstream layers.

4.4 Adaptation with perfect smoothing kernels

We perform an ablation study to verify our theoretical guarantee for numerically identical adaptation. Figure 6 displays a set of experiments that use *perfect quality* Whittaker-Shannon smoothing kernels (in the upper row of graphs in green) implemented through the Fast Fourier Transform (Cooley & Tukey, 1965). We showcase the usual set of ablations within this group of experiments; with Laplacian dropout (bright green lines) or without Laplacian dropout (dark green lines); with adaptation (full lines) or without adaptation (dashed lines). Our method evaluates practically identically whether unnecessary Laplacian residuals are discarded (with adaptation, full lines, Equation 26), or whether all Laplacian residuals are preserved (without adaptation, dashed lines, Equation 25), with imperceptible discrepancies that are exactly zero, or that are small enough to be attributed to the numerical limitations of floating point computation. Our method is able to skip computations without numerical compromises, as predicted by our theoretical guarantee.

4.5 Adaptation with approximate smoothing kernels and the dual effect of Laplacian dropout

We extend the previous ablation study to verify our theoretical analysis of the dual effect of Laplacian dropout. Figure 6 introduces a set of experiments that relies on *fair quality* approximate Whittaker-Shanon smoothing kernels (in the middle row of graphs in red), and on *poor quality* truncated Gaussian smoothing kernels (in the bottom row of graphs in blue). Figure 7 displays these same results in the form of a decision tree to help recognize the trends that are relevant to our discussion. This decision tree factors the impact of choosing a specific filter quality, choosing whether to use Laplacian dropout or not, and choosing whether to use adaptation or not. This analysis considers accuracy averaged over all resolutions and all datasets as the metric of interest. The numerical values displayed on each node correspond to the average multiplicative change in the metric once a decision is made. We claimed in subsection 3.2 that Laplacian dropout has two distinct purposes: it regularizes for robustness across a wider distribution of resolutions; it also mitigates numerical discrepancies caused by approximate smoothing kernels when using adaptation (Equation 27). We have demonstrated the first effect in subsection 4.1 and can also observe this effect clearly in the decision tree. We can only observe the second effect if we consider the choice of smoothing kernel, which motivates this experimental setup. Our method will conform to our theoretical interpretation if absence of error terms ϵ_n at training time yields better performance in the absence of error terms ϵ_n at inference time, and vice versa; that is to say on the last two levels of the decision tree, on the black and white nodes, tracing a path across two nodes of identical colour should yield a multiplier greater than 1 at the last node, and conversely, tracing a path across two nodes of opposing colour should yield a multiplier smaller than 1 at the last node. Our method displays exactly this behaviour, with the discrepancy at the last level of the decision tree growing monotonically with decreases in filter quality. Our method therefore leverages Laplacian dropout not just to improve robustness across a wider distribution of resolutions, but to compensate numerical errors induced by imperfect smoothing kernels, which enables the use of computationally greedy implementations.

5 Discussion

We have introduced ARRNs, a class of adaptive-resolution architectures that inherits the simplicity of fixed-resolution methods, and the robustness and computational efficiency of adaptive-resolution methods. ARRNs substitute standard residuals with *Laplacian residuals* which allow creating adaptive-resolution architectures using only fixed-resolution layers, and which allow skipping computations at lower resolutions without compromise in numerical accuracy. ARRNs also implement *Laplacian dropout*, which allows training models that perform robustly at a wide range of resolutions.

Future Work. We have provided evidence on classification tasks over low-resolution and medium-resolution image data; our method's ability to generalize is well supported by theoretical justification, but further experiments that include more challenging tasks and high-resolution data are desirable. We have investigated a form of Laplacian residual that *decreases* resolution, but Laplacian residuals may be generalized to a form that *increases* resolution, which would allow constructing a greater variety of architectures. We have applied our method in two dimensions on image data, but it is theoretically valid with any number of dimensions; its application to audio data and volumetric data is of interest.

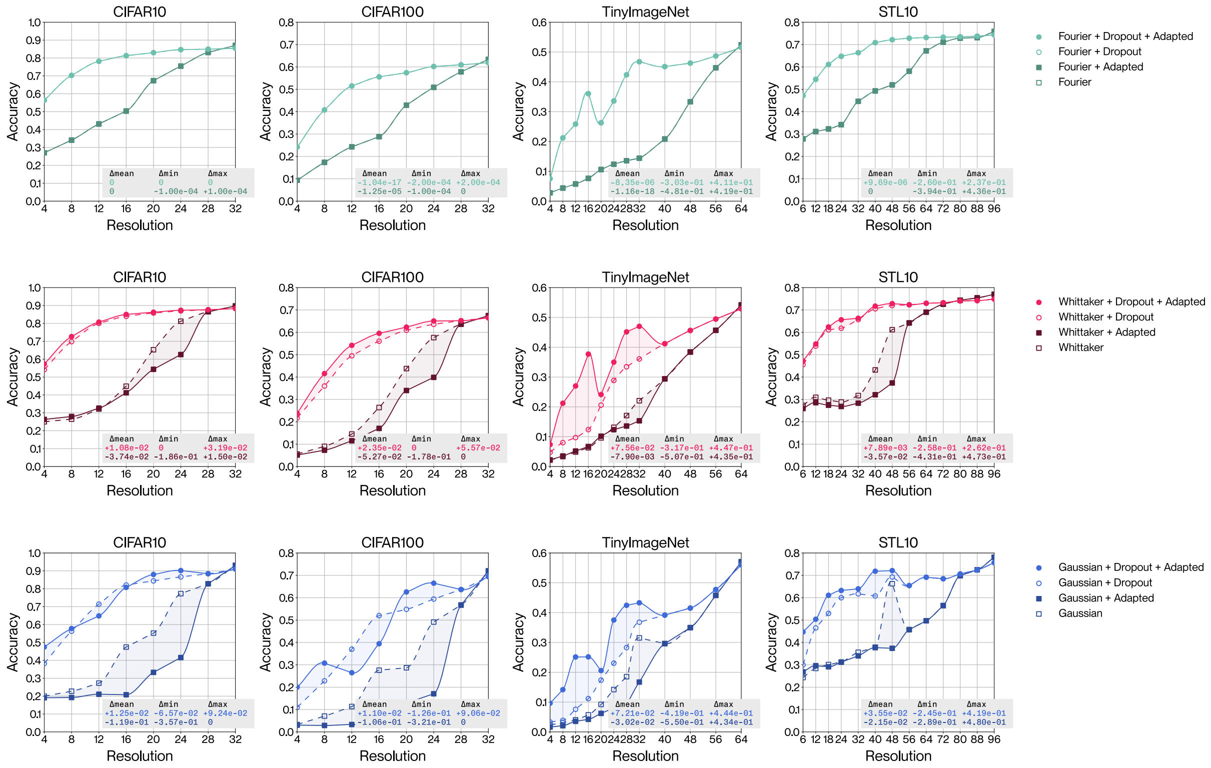

Figure 6: Accuracy of ARRNs at various resolutions, where each ARRN is identical to the architecture used in the main experiments aside from the choice of smoothing kernel. The smoothing kernels each correspond to a different row of graphs and a different hue. The shaded area spanning pairs of curves highlights the difference between the accuracy with adaptation and without adaptation. The overlaid tables display a statistical breakdown of this discrepancy with Laplacian dropout and without Laplacian dropout.

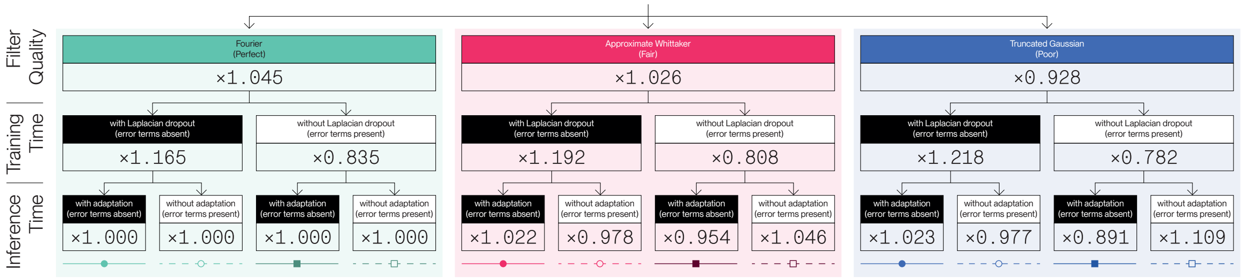
Filter Quality
Training Time
Inference Time
Fourier (Perfect)
×1.045
Approximate Whittaker (Fair)
×1.026
Truncated Gaussian (Poor)
×0.928
with Laplacian dropout (error terms absent)
without Laplacian dropout (error terms present)
×1.165
×0.835
×1.192
×0.808
×1.218
×0.782
with adaptation (error terms absent)
without adaptation (error terms present)
×1.000
×1.000
×1.000
×1.000
×1.022
×0.978
×0.954
×1.046
×1.023
×0.977
×0.891
×1.109

Figure 7: Multiplicative factor analysis showing an ablation over key components of ARRNs. The performance metric considered for analysis is the average accuracy over all resolutions and all datasets. The three levels of the decision tree show which filter is used, whether Laplacian dropout is used at training time, and whether adaptation is used at inference time. The numerical ratios indicate how average accuracy changes conditional to the decision associated with the node. The last two levels of nodes are coloured black or white according to the absence or presence of error terms ϵ_n respectively; at training time, this is determined by use of Laplacian dropout; at inference time, this is determined by use of adaptation. The symbols below the leaf nodes of the decision tree show the line style used in Figure 6 to allow easily referencing the underlying experiments in detail.

Impact Statement. This paper presents work whose goal is to advance fundamental research in the field of deep learning and machine learning. No specific real-world application is concerned, although this contribution may render certain forms of technology more accessible.

References

Francesca Bartolucci, Emmanuel de Bezenac, Bogdan Raonic, Roberto Molinaro, Siddhartha Mishra, and Rima Alaifari. Representation equivalent neural operators: a framework for alias-free operator learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL `https://openreview.net/forum?id=7LSEkvEGCM`.

Peter J Burt and Edward H Adelson. The laplacian pyramid as a compact image code. In *Readings in computer vision*, pp. 671–679. Elsevier, 1987.

Yinbo Chen, Sifei Liu, and Xiaolong Wang. Learning continuous image representation with local implicit image function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8628–8638, 2021.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 215–223, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL `https://proceedings.mlr.press/v15/coates11a.html`.

James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11, 2018.

Vladimir Fanaskov and Ivan Oseledets. Spectral neural operators. *arXiv preprint arXiv:2205.10573*, 2022.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1314–1324, 2019.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pp. 646–661. Springer, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 448–456, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/ioffe15.html`.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 624–632, 2017.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Jaewon Lee and Kyong Hwan Jin. Local texture estimator for implicit representation function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1929–1938, 2022.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4460–4470, 2019.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Samuel G Müller and Frank Hutter. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 774–782, 2021.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 165–174, 2019.

Daniel P. Petersen and David Middleton. Sampling and reconstruction of wave-number-limited functions in n-dimensional euclidean spaces. *Information and Control*, 5(4):279–323, 1962. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(62)90633-2. URL `https://www.sciencedirect.com/science/article/pii/S0019995862906332`.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520, 2018.

Claude E Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37(1):10–21, 1949.

Satya Rajendra Singh, Roshan Reddy Yedla, Shiv Ram Dubey, Rakesh Sanodiya, and Wei-Ta Chu. Frequency disentangled residual network. *arXiv preprint arXiv:2109.12556*, 2021.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33: 7462–7473, 2020.

Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pp. 10096–10106. PMLR, 2021.

ET Whittaker. On the functions which are represented by the expansion of interpolating theory. In *Proc. Roy. Soc. Edinburgh*, volume 35, pp. 181–194, 1915.

John Macnaughten Whittaker. On the cardinal function of interpolation theory. *Proceedings of the Edinburgh Mathematical Society*, 1(1):41–46, 1927.

Dejia Xu, Peihao Wang, Yifan Jiang, Zhiwen Fan, and Zhangyang Wang. Signal processing for implicit neural representations. *Advances in Neural Information Processing Systems*, 35:13404–13418, 2022.

Jingyu Yang, Sheng Shen, Huanjing Yue, and Kun Li. Implicit transformer network for screen content image continuous super-resolution. *Advances in Neural Information Processing Systems*, 34:13304–13315, 2021.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

A Appendix

A.1 Background

We survey fundamental concepts of signal processing and introduce our notation. We aim to provide meaningful intuitions for readers who are not familiar with these principles, and to also rigorously ground our method and satisfy readers who are knowledgeable in this topic.

In Figure 8, we illustrate how the *resolution* and *bandwidth* of signals intuitively relate to each other while also highlighting the notation and indexing scheme we use to designate these characteristics in our analysis. In the paragraphs that follow, we break down these notions in greater detail.

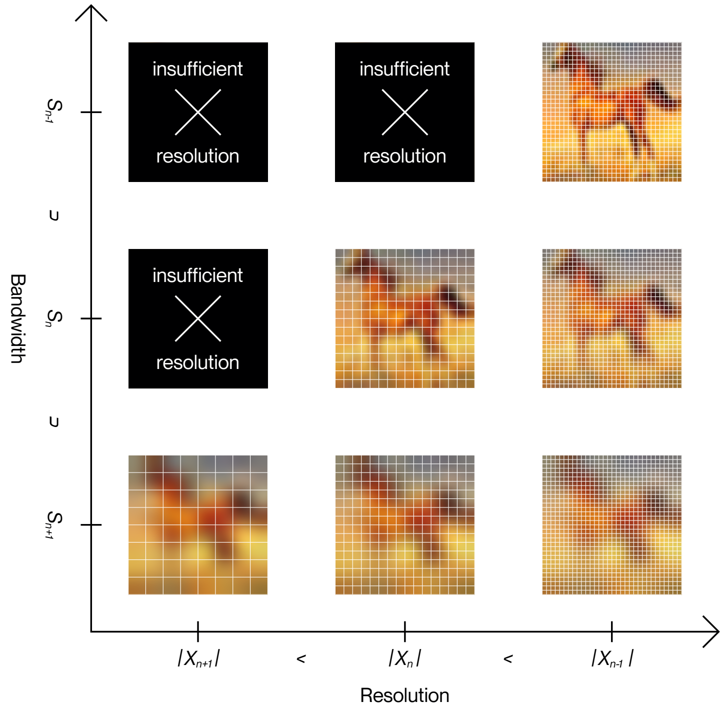

Figure 8: Diagram showing the same signal captured at three distinct resolutions $|X_{n+1}| < |X_n| < |X_{n-1}|$ (laid out horizontally) and three distinct bandwidths $S_{n+1} \subset S_n \subset S_{n-1}$ (laid out vertically). Intuitively, resolution corresponds to the coarseness of the grid that is laid over the continuous signal to sample it into a discrete signal, while bandwidth corresponds to the smoothness of the continuous signal. A continuous signal may only be captured into a discrete signal if resolution is sufficiently large to account for bandwidth; the blacked out signals show cases where this condition is not met.

Signals in continuous form. We can represent signals as functions $s(\,\cdot\,) : \mathbf{X} \to \mathbb{R}^f$ that map a *continuous* spatial domain $\mathbf{X}$ that is a nicely behaved subset of $\mathbb{R}^d$ to a feature domain $\mathbb{R}^f$. This representation is useful because it allows us to leverage notions from functional analysis and calculus, and because it is fully independent from the way a signal is captured. We will often refer to the *bandwidth* of signals in this work, which we can intuitively relate to the smoothness of continuous signals; low bandwidth signals are smooth, while high bandwidth signals are detailed.

Signals in discrete form. We can also represent signals as functions $s[\,\cdot\,]_n : \mathbf{X}_n \to \mathbb{R}^f$ that map from a *discrete* spatial domain $\mathbf{X}_n \subset \mathbf{X}$ to a feature domain $\mathbb{R}^f$. This representation enables us to perform computations on signals as they are fully defined by a finite amount of information given by the $|\mathbf{X}_n|$ individual samples $\mathbf{x}_i \in \mathbf{X}_n$. We commonly point to the quantity of samples as the *resolution* of a discrete signal. We associate indexing by n to distinct resolutions throughout this work, where resolution *decreases* as n *increases*, meaning $|X_n| > |X_{n+1}|$.

Casting continuous signals into discrete signals by sampling. We can easily take a continuous signal $s(\,\cdot\,)$ and create a discrete signal $s[\,\cdot\,]_n$ by sampling values at points $\mathbf{x}_i \in \mathbf{X}_n$. We notate this process as the *sampling operator* $\mathfrak{S}_n : (\mathbf{X} \to \mathbb{R}^f) \to (\mathbf{X}_n \to \mathbb{R}^f)$:

$$\mathfrak{S}_n\{s(\,\cdot\,)\} = \mathbf{x}_i \mapsto s(\mathbf{x}_i) \;\; \forall \mathbf{x}_i \in \mathbf{X}_n \tag{31}$$

Casting discrete signals into continuous signals by interpolation. We can reverse the process above and derive a continuous signal $s(\,\cdot\,)$ from a discrete signal $s[\,\cdot\,]_n$ by applying a convolution with a smoothing kernel ϕ_n, more formally known as a Whittaker-Shannon kernel (Whittaker, 1915; 1927). We note that sampling and interpolation are only inverses of each under certain important conditions we come back to later. We interpret the effect of the convolution against a smoothing kernel as filling in the gaps between the samples. We notate the process outlined here as the *interpolation operator* $\mathfrak{I}_n : (\mathbf{X}_n \to \mathbb{R}^f) \to (\mathbf{X} \to \mathbb{R}^f)$:

$$\mathfrak{I}_n\{s[\,\cdot\,]_n\} = \mathbf{x} \mapsto \sum_{\mathbf{x}_i \in \mathbf{X}_n} s[\mathbf{x}_i]_n \phi_n(\mathbf{x} - \mathbf{x}_i) \;\; \forall \mathbf{x} \in \mathbf{X} \tag{32}$$

Restricting the bandwidth of signals. We need a slightly more formal way of designating signals that respect certain bandwidth constraints in order to better discuss the equivalence between discrete signals and continuous signals. We can use the smoothing kernels ϕ_n we just introduced to define sets of continuous signals $S_n = \{s | s * \phi_n = s\}$ that are already smooth enough to be left unchanged by the action of a smoothing kernel. We underline that applying a smoothing kernel ϕ_n onto a signal s restricts its bandwidth such that it belongs to the corresponding set of signals S_n, which is especially useful as it allows us to guarantee a form of consistency between continuous signals and discrete signals, as we discuss next. We finally note that bandwidth constraints form an ordering $S_n \supset S_{n+1}$, meaning a signal s that respects a low bandwidth constraint S_n also respects any arbitrarily high bandwidth constraint $S_{n-k} \; \forall \; k > 0$.

Equivalence of continuous signals and discrete signals. We can use discrete signals or continuous signals to designate the same underlying information when the Nyquist-Shannon sampling theorem is satisfied (Shannon, 1949; Petersen & Middleton, 1962). This theorem intuitively states that a continuous signal with high *bandwidth* requires a discrete signal with correspondingly high *resolution* for sampling to generally be feasible without error. This theorem more formally states that a discrete signal $s[\,\cdot\,]_n$ can uniquely represent any continuous signal $s(\,\cdot\,)$ that respects the bandwidth constraint encoded by membership to $S_n = \{s' | s' * \phi_n = s'\}$, where the expression for the smoothing kernel ϕ_n is depends on the sampling density $|\mathbf{X}_n|/\text{vol}(\mathbf{X})$ of the discrete spatial domain over the continuous spatial domain and on the assumptions on the boundary conditions of the continuous spatial domain $\mathbf{X}$ (Whittaker, 1915; 1927; Petersen & Middleton, 1962). We summarize the Nyquist-Shannon sampling theorem by stating that the sampling operator and interpolation operator are only guaranteed to be inverses of each other when the bandwidth constraint is satisfied:

$$s \in S_n \implies \mathfrak{I}_n\{\mathfrak{S}_n\{s\}\} = s \tag{33}$$

Equivalence of operators acting upon continuous signals and discrete signals We can extend the notion of equivalence between *continuous signals* $s(\,\cdot\,) : \mathbf{X} \to \mathbb{R}^f$ and *discrete signals* $s[\,\cdot\,]_n : \mathbf{X}_n \to \mathbb{R}^f$ to encompass the actions that can be performed on the same signals using *operators on continuous signals* $\mathfrak{O} : (\mathbf{X} \to \mathbb{R}^f) \to (\mathbf{X} \to \mathbb{R}^f)$ and *operators on discrete signals* $\mathfrak{O}_n : (\mathbf{X}_n \to \mathbb{R}^f) \to (\mathbf{X}_n \to \mathbb{R}^f)$. We are especially interested in this notion as it enables us to think of our neural architecture as a chain of operators that act on continuous signals which can be cast to act on discrete signals of any specific resolution $|X_n|$. We often see this property formally labeled as *discretization invariance* in the neural operator community and highlight this concept is key to other works which allow adaptation to different resolutions (Li et al., 2020; Kovachki et al., 2021; Fanaskov & Oseledets, 2022; Bartolucci et al., 2023). We can formally express the equivalence between the continuous form $\mathfrak{O}$ and discrete form $\mathfrak{O}_n$ of some operator as commutativity over the sampling operator when the bandwidth constraint is satisfied:

$$s \in S_n \implies \mathfrak{S}_n\{\mathfrak{O}\{s\}\} = \mathfrak{O}_n\{\mathfrak{S}_n\{s\}\} \tag{34}$$

A.2 Related works

We cover an additional approach to adaptive-resolution architectures in this complementary section, with the aim of better motivating and situating our approach relative to more mainstream approaches.

Adaptive-resolution by narrowing or widening the domain of evaluation. In principle, many mainstream architectures are able to operate at various resolutions directly, as is the case for fully convolutional architectures. This is because the layers that compose them are invariant or equivariant to translation or permutation, and can therefore evaluate on a wider or narrower domain. In practice, this form of extension to arbitrary resolution is flawed in its ability to guarantee robustness, as it addresses changes in resolution $d|X_n|$ as changes in the bounds of the spatial domain $d\text{vol}(X)$ given fixed spatial sampling density $d(|X_n|/\text{vol}(X)) = 0$. Operating at a smaller resolution, therefore, means operating on a narrower region of space, which erodes the ability to infer global structure and leads to a collapse in performance when pushed beyond a breaking point. Operating at a larger resolution likewise means operating on a wider region of space, which leads to incoherence once pushed beyond the scope of global structure observed during training. This shortcoming can be sidestepped by interpolating the resolution given at inference to the resolution used at training, which ensures the region of space that is operated on remains consistent. This provides stronger robustness to diverse resolutions, however it amounts to treating these architectures as fixed-resolution and fails to address the problem of computational efficiency. In contrast, the adaptive-resolution architecture we introduce in our contribution and the other two forms of adaptive-resolution architectures we discuss in section 2 equate changes in the resolution $d|X_n|$ with changes in spatial sampling density $d(|X_n|/\text{vol}(X))$ given fixed bounds of the spatial domain $d\text{vol}(X) = 0$. Under this more nuanced framing, the region of space that is observed is always constant regardless of resolution, and global structure remains consistently accessible, which guarantees stronger robustness without sacrificing computational efficiency.

A.3 Experiments

We provide further details on our experiments in this section.

Model design. We provide detailed illustrations for the full architecture designs we use with our method according to each dataset: Figure 9 for CIFAR10 (5.33M-8.09M); Figure 10 for CIFAR100 (9.59M-14.5M); Figure 11 for TinyImageNet (15.0M-19.8M); and Figure 12 for STL10 (13.8M-18.4M). We indicate not a single parameter count, but a range of parameter counts for each architecture design, as adaptation enables computation of the forward pass or backward pass using a variable subset of the underlying parameters. We follow a general design pattern inspired by MobileNetV2 (Sandler et al., 2018) and EfficientNetV2 (Tan & Le, 2021) to derive these architecture designs. We nest inner architectural blocks (b_n in Equation 9) within a series of Laplacian residual blocks of decreasing resolution and increasing feature count. We create these inner architectural blocks by composing depthwise 3×3 convolutions and pointwise 1×1 convolutions in alternation. We set all depthwise convolutions to use edge replication padding to satisfy Equation 5 and ensure resolution remains fixed within each Laplacian residual block. We prepend this string of layers with a pointwise convolution that expands the feature channel count. We conversely terminate the sequence of layers with a pointwise convolution that contracts the feature channel count inversely. We separate each convolution with a batch normalization (Ioffe & Szegedy, 2015) and a SiLU activation function (Elfwing et al., 2018), chosen for its tendency to produce fewer aliasing artifacts. We apply different Laplacian dropout rates (p_n in Equation 28) depending on the dataset: 0.6 for CIFAR10; 0.3 for CIFAR100; 0.3 for TinyImageNet; 0.3 for STL10. We use a common classification head that consists of a single linear layer with a dropout set to 0.2, which is applied after global average pooling. The designs were chosen by sweeping over different configurations for inner architectural blocks, and over different resolutions and number of features for the Laplacian residual blocks that contain them. The number of permutations per final sweep ranged between 18 to 216 for each dataset.

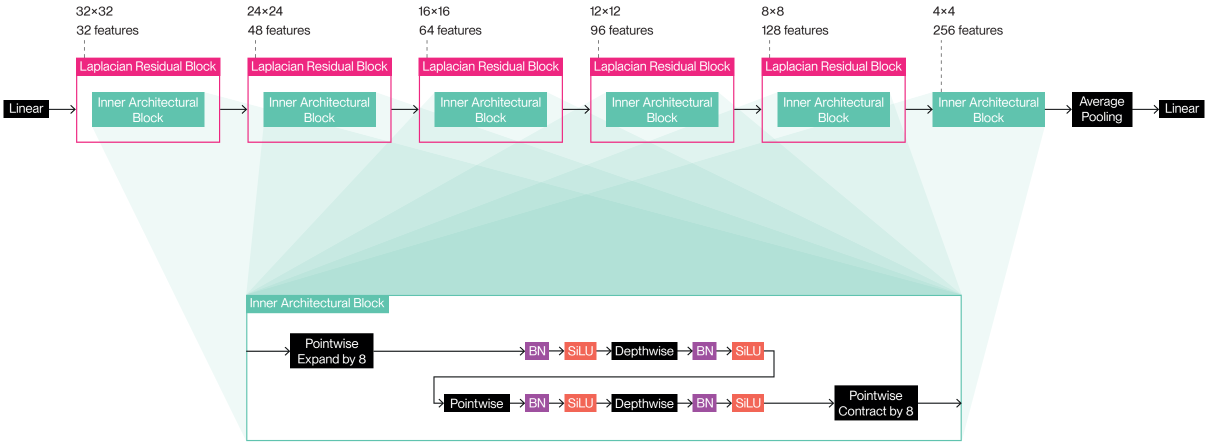
32×32
32 features
24×24
48 features
16×16
64 features
12×12
96 features
8×8
128 features
4×4
256 features
Laplacian Residual Block
Inner Architectural Block
Linear
Average Pooling
Linear
Inner Architectural Block
Pointwise Expand by 8
BN
SiLU
Depthwise
BN
SiLU
Pointwise
BN
SiLU
Depthwise
BN
SiLU
Pointwise Contract by 8

Figure 9: Architecture design for our method on CIFAR10.

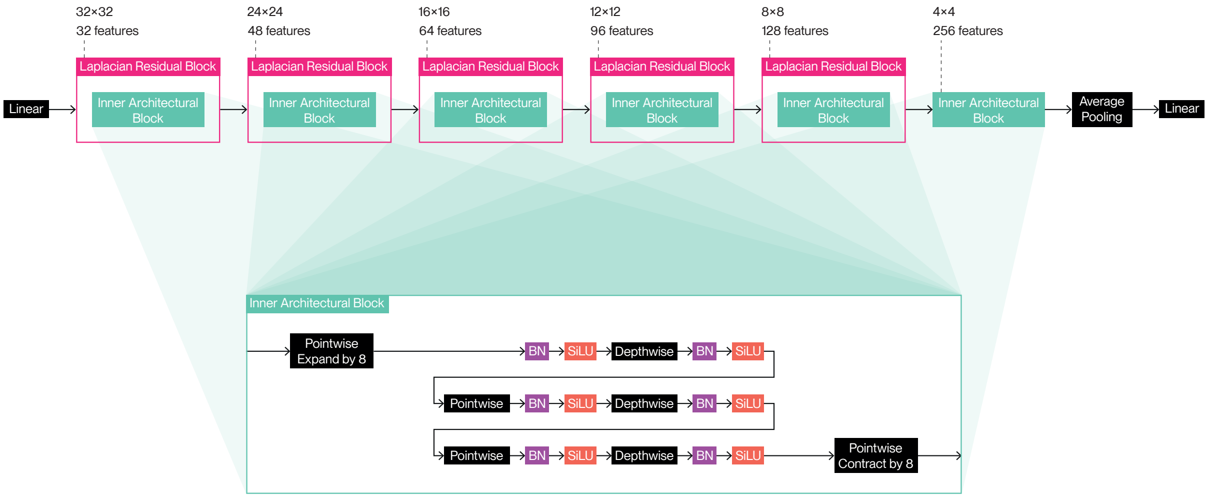
32×32
32 features
24×24
48 features
16×16
64 features
12×12
96 features
8×8
128 features
4×4
256 features
Laplacian Residual Block
Inner Architectural Block
Linear
Average Pooling
Linear
Inner Architectural Block
Pointwise Expand by 8
BN
SiLU
Depthwise
BN
SiLU
Pointwise
BN
SiLU
Depthwise
BN
SiLU
Pointwise
BN
SiLU
Depthwise
BN
SiLU
Pointwise Contract by 8

Figure 10: Architecture design for our method on CIFAR100.

Model training hyperparameters. We provide the specific hyperparameters used during training.

For CIFAR10 and CIFAR100, across all methods, we use AdamW (Loshchilov & Hutter, 2017) with a learning rate of 10^{-3} and $(\beta_1, \beta_2) = (0.9, 0.999)$, cosine annealing (Loshchilov & Hutter, 2016) to a minimum learning rate of 10^{-5} in 100 epochs, weight decay of 10^{-3}, and a batch size of 128. We use a basic data augmentation consisting of normalization, random horizontal flipping with $p = 0.5$, and randomized cropping that applies zero-padding by 4 along each edge to raise the resolution, then crops back to the original resolution.

For TinyImageNet and STL10, across all methods, we use SGD with a learning rate of 10^{-2}, cosine annealing (Loshchilov & Hutter, 2016) to a minimum learning rate of 0 in 100 epochs, weight decay of 10^{-3}, and a batch size of 128. We use TrivialAugmentWide (Müller & Hutter, 2021) to augment training.

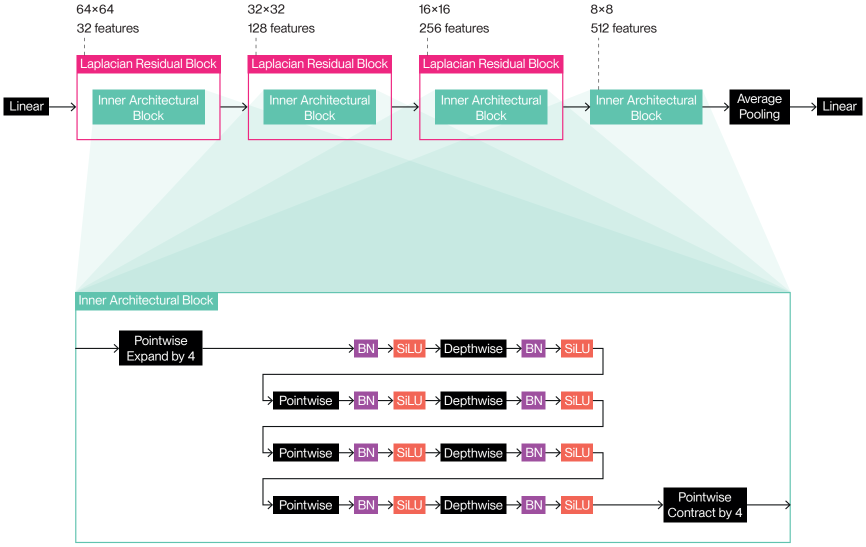
64×64
32 features
32×32
128 features
16×16
256 features
8×8
512 features
Linear
Laplacian Residual Block
Inner Architectural Block
Laplacian Residual Block
Inner Architectural Block
Laplacian Residual Block
Inner Architectural Block
Inner Architectural Block
Average Pooling
Linear
Inner Architectural Block
Pointwise Expand by 4
BN
SiLU
Depthwise
BN
SiLU
Pointwise
BN
SiLU
Depthwise
BN
SiLU
Pointwise
BN
SiLU
Depthwise
BN
SiLU
Pointwise
BN
SiLU
Depthwise
BN
SiLU
Pointwise Contract by 4

Figure 11: Architecture design for our method on TinyImageNet.

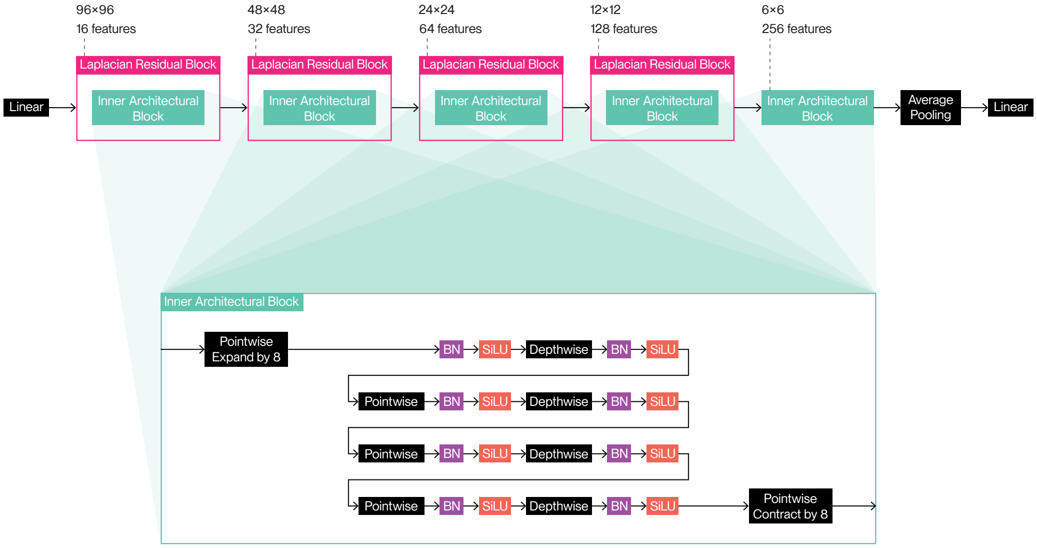
96×96
16 features
48×48
32 features
24×24
64 features
12×12
128 features
6×6
256 features
Linear
Laplacian Residual Block
Inner Architectural Block
Laplacian Residual Block
Inner Architectural Block
Laplacian Residual Block
Inner Architectural Block
Laplacian Residual Block
Inner Architectural Block
Inner Architectural Block
Average Pooling
Linear
Inner Architectural Block
Pointwise Expand by 8
BN
SiLU
Depthwise
BN
SiLU
Pointwise
BN
SiLU
Depthwise
BN
SiLU
Pointwise
BN
SiLU
Depthwise
BN
SiLU
Pointwise
BN
SiLU
Depthwise
BN
SiLU
Pointwise Contract by 8

Figure 12: Architecture design for our method on STL10.