Adaptive Resolution Residual Networks

Léa Demeule, Mahtab Sandhu, Glen Berseth
Mila, Université de Montréal
{lea.demeule, mahtab.sandhu, glen.berseth}@mila.quebec

Abstract

We introduce *Adaptive Resolution Residual Networks* (ARRNs), a form of neural operator that enables the creation of networks for signal-based tasks that can be rediscretized to suit any signal resolution. ARRNs are composed of a chain of *Laplacian residuals* that each contain ordinary layers, which do not need to be rediscretizable for the whole network to be rediscretizable. ARRNs have the property of requiring a lower number of Laplacian residuals for exact evaluation on lower-resolution signals, which greatly reduces computational cost. ARRNs also implement *Laplacian dropout*, which encourages networks to become robust to low-bandwidth signals. ARRNs can thus be trained once at high-resolution and then be rediscretized on the fly at a suitable resolution with great robustness.

1 Introduction

The majority of deep learning methods for signals assume a fixed resolution during training and inference, making it impractical to apply a single network at various resolutions. We address this shortcoming by introducing *Adaptive Resolution Residual Networks* (ARRNs), which are neural operators that can be rediscretized easily and robustly thanks to two components: *Laplacian residuals*, which define the structure of ARRNs and allow rediscretization, and *Laplacian dropout*, which improves the robustness of rediscretized ARRNs through a training augmentation.

We formulate *Laplacian residuals* by combining the properties of standard residuals (He et al., 2016a,b) and Laplacian pyramids (Burt and Adelson, 1987). Thanks to this structure, *rediscretizing an ARRN to a lower resolution simply means evaluating a lower number of Laplacian residuals*. This form of rediscretization has many benefits: it improves computational efficiency at lower resolutions; it can be applied instantaneously at inference suit the given resolution; it imposes very few design constraints on the layers nested within Laplacian residuals, unlike neural operators that allow rediscretization (Kovachki et al., 2021; Li et al., 2020). We find that Lai et al. (2017); Singh et al. (2021) formulate residual connections with similar filtering operations, however we are the first to propose an architecture that leverages this for adaptive input resolution.

We formulate *Laplacian dropout* through the converse idea that *randomly lowering the number of Laplacian residuals is equivalent to randomly rediscretizing an ARRN to a lower resolution*. We leverage this as a training augmentation that has the effect of improving the robustness of the many low-resolution ARRNs that can be derived from a single high-resolution ARRN. We find that Huang et al. (2016) applies dropout similarly to residuals, however Laplacian dropout has an interpretation as a bandwidth augmentation that this prior work lacks.

We provide theoretical analysis for the advantageous properties of ARRNs in section 2, along with a set of experiments that demonstrate these properties in practice in section 3, where we train ARRN and competing methods at a single resolution, then evaluate them at various resolutions.

DLDE-III Workshop in the 37th Conference on Neural Information Processing Systems (NeurIPS 2023).

2 Method

In this section, we formulate the two main components of ARRNs: Laplacian residuals (subsection 2.1) and Laplacian dropout (subsection 2.2). We assume some familiarity with the theory of signals from the reader (Fourier, 1888; Whittaker, 1915, 1927; Shannon, 1949; Petersen and Middleton, 1962). We provide illustrations and complementary background on Laplacian pyramids in section 5 to help better understand the structure of ARRNs. As in most neural operators methods, we follow the perspective of *continuous signals* rather than *discrete signals*. We conceptualize signals as functions $s : \mathbf{X} \rightarrow \mathbb{R}^f$ mapping from a spatial domain $\mathbf{X} \subset \mathbb{R}^d$ to a feature domain $\mathbb{R}^f$.

2.1 Laplacian residuals

Definition. Laplacian residuals $r_n : (\mathbf{X} \rightarrow \mathbb{R}^{f_n}) \rightarrow (\mathbf{X} \rightarrow \mathbb{R}^{f_{n+1}})$ are composed together in a chain of m residuals. Each residual contains some architectural block $b_n : (\mathbf{X} \rightarrow \mathbb{R}^{f_n}) \rightarrow (\mathbf{X} \rightarrow \mathbb{R}^{f_n})$ that can perform any operation as long as its output signal is a constant whenever its input signal is zero (Equation 1, where a is a constant signal); b_n does not need to be conceptualized in the framework of neural operators; b_n does not need the ability to be rediscretized because its discretization is fixed; b_n can be a convolution, vision transformer, normalization, or composition of multiple layers.

$$b_n(0) = a \tag{1}$$

The filter kernels of Laplacian residuals are ideal, and are nested into each other such that deeper filter kernels correspond to lower bandwidth. The base case takes the original signal and performs a linear projection through $\mathbf{A}_0$ to allow a change in feature dimensionality from f_0 to f_1:

$$r_0 = \mathbf{A}_0 s \tag{2}$$

The recursive case takes the preceding residual r_{n-1}, forms a lower bandwidth signal r_n^{low} (Equation 3), and forms a difference signal r_n^{diff} (Equation 4):

$$r_n^{\text{low}} = r_{n-1} * \phi_n^{\text{low}} \tag{3}$$

$$r_n^{\text{diff}} = r_{n-1} - r_n^{\text{low}} \tag{4}$$

The difference signal r_n^{diff} is given to the architectural block b_n contained in the residual. Like in the Laplacian pyramid (Burt and Adelson, 1987), the difference signal r_n^{diff} explains the gap between two representations of the same signal at different resolutions, one higher, and one lower. We are especially interested in what happens when a signal can be *fully* captured at *either* the higher resolution *or* the lower resolution, meaning a higher resolution representation would be wasteful. We can see that in that case, the difference signal r_n^{diff} is zero. We want to leverage this by causing a chain of zero terms that we can use for simplifying rediscretization. We do this by using a zero-blocking filter which subtracts the mean:

$$b_n(0) * \phi^{\text{zero}} = 0 \tag{5}$$

We also must perform further filtering with ϕ_n^{low} so that the output conforms to the lower bandwidth signal that the next residual expects as an input. We then apply a skip connection by adding r_n^{low} to the output, as in standard residuals (He et al., 2016a,b), and apply a linear projection through $\mathbf{A}_n$ to allow a change in feature dimensionality from f_n to f_{n+1} before processing the next residual:

$$r_n = \mathbf{A}_n \left(b_n(r_n^{\text{diff}}) * \phi_n^{\text{low}} * \phi^{\text{zero}} + r_n^{\text{low}} \right) \tag{6}$$

Rediscretization. If the spectrum of the signal s is entirely confined within the spectrum of the first filter kernel ϕ_1^{low}, then the value of the lower bandwidth residual r_1^{low} is given by a linear projection of s (Equation 8), and the difference signal r_1^{diff} is zero (Equation 9). Because the input to the inner architectural block b_1 is zero, its output is a constant (Equation 1). Because we then perform filtering with ϕ^{zero}, the output of the inner architectural block b_1 contributes zero to the residual r_1 (Equation 5). The value of the residual r_1 is therefore entirely defined by a linear projection of s; its exact computation does not need to involve the inner architectural block b_1 (Equation 10). We see that this cascade of zeros persists as long as the spectrum of the input signal s is entirely confined within the spectrum of all the lowpass filters $\phi_{n'}^{\text{low}}$ it encounters, with $n' \in [1, n]$:

$$s * \phi_{n'}^{\text{low}} = s \;\forall\, n' \in [1, n] \tag{7}$$

$$\Longrightarrow r_1^{\text{low}} \quad = \mathbf{A}_0 s * \phi_1^{\text{low}} = \mathbf{A}_0 s \tag{8}$$
$$\Longrightarrow r_1^{\text{diff}} \quad = \mathbf{A}_0 s - \mathbf{A}_0 s = 0 \tag{9}$$
$$\Longrightarrow r_1 \quad = \mathbf{A}_1 \left(b_1(0) * \phi_1^{\text{low}} * \phi^{\text{zero}} + \mathbf{A}_0 s \right) = \mathbf{A}_1 \mathbf{A}_0 s \tag{10}$$
$$\vdots$$
$$\Longrightarrow r_n^{\text{low}} \quad = \mathbf{A}_{n-1} \cdots \mathbf{A}_0 s * \phi_n^{\text{low}} = \mathbf{A}_{n-1} \cdots \mathbf{A}_0 s \tag{11}$$
$$\Longrightarrow r_n^{\text{diff}} \quad = \mathbf{A}_{n-1} \cdots \mathbf{A}_0 s - \mathbf{A}_{n-1} \cdots \mathbf{A}_0 s = 0 \tag{12}$$
$$\Longrightarrow r_n \quad = \mathbf{A}_n \left(b_n(0) * \phi_n^{\text{low}} * \phi^{\text{zero}} + \mathbf{A}_{n-1} \cdots \mathbf{A}_0 s \right) = \mathbf{A}_n \cdots \mathbf{A}_0 s \tag{13}$$

We can therefore exactly evaluate r_n by skipping all filters $\phi_{n'}^{\text{low}}$ and all inner architectural blocks $b_{n'}$ with $n' \in [1, n]$, by instead applying a single linear projection with a precomputed matrix $\mathbf{A}_n^{\text{chain}} = \mathbf{A}_n \cdots \mathbf{A}_0$. This allows us to rediscretize high-resolution ARRNs into low-resolution ARRNs with greater computational efficiency, without performance degradation, and without difficult design constraints.

Implementation. We implement all filtering and rediscretization operations using approximate Whittaker-Shannon interpolation (Whittaker, 1915) based on separable polyphase convolutions that effectively extend Smith (2002); Yang et al. (2021) to higher dimensionality. In Laplacian residuals, the r_n^{diff} terms of Equation 9 are computed while preserving the original resolution, while the r_n^{low} and $b_n(r_n) * \phi_n^{\text{low}}$ terms of Equation 6 are computed while resampling to a lower resolution. By following this process, we always use the lowest resolution that allows appropriate representation of the signals. In the experimental setup, all rediscretization is done through this interpolation method.

2.2 Laplacian dropout

When we show a low-bandwidth signal to a high-resolution ARRN, a set of consecutive early Laplacian residuals may be zero. Conversely, if we show a high-resolution signal to a high-resolution ARRN but randomly zero out a set of consecutive early residuals, this will be equivalent to showing a randomly lowered bandwidth signal to the ARRN. Laplacian dropout simply implements this during training to encourage robustness to low-bandwidth signals. Unlike Huang et al. (2016), we gate the difference signal r_n^{diff} with a Bernoulli random variable *chained through the logical or operator* (Equation 15) to implement the consecutiveness constraint.

$$d_n^{\text{indep}} \sim \mathrm{B}(1 - p_n) \tag{14}$$
$$d_n^{\text{chain}} = d_n^{\text{indep}} \oplus d_{n-1}^{\text{chain}} \tag{15}$$
$$r_n^{\text{diff}} = d_n^{\text{chain}} (r_{n-1} - r_n^{\text{low}}) \tag{16}$$

3 Experiments

Our experiments demonstrate that rediscretized ARRNs have identical performance to non-rediscretized ARRNs; that rediscretized ARRNs have vastly lower inference time than non-rediscretized ARRNs ; and that ARRNs are robust to low bandwidth signals.

We construct a pair of experimental setups each evaluated on the **CIFAR10** and **CIFAR100** (Krizhevsky et al., 2009) image classification datasets. All models are trained once for 100 epochs at the standard 32×32 resolution. All models are then evaluated at various lower resolutions. Since the methods we compare do not have the ability to adapt to lower resolutions, the images are rediscretized to the lower resolutions, then rediscretized back to the original 32×32 resolution during evaluation (see subsection 5.3 for an illustrated explanation). Thus, all methods have access to the same information in a fair manner.

We compare our **ARRN** (subsection 5.3 explains the architecture design in detail; 5.33M-8.09M for CIFAR10 and 9.59M-14.5M for CIFAR100 depending on rediscretization) against a wide range of convolutional network families that are well-suited for the classification task we test: **MobileNetV3** (1.52M-4.21M) Howard et al. (2019), **WideResNetV2** (66.8M-124M) (Zagoruyko and Komodakis, 2016), **ResNet** (11.1M-42.5M) He et al. (2016a), **ConvNeXt** (27.8M-196M) Liu et al. (2022), and **EfficientNetV2** (20.2M-117.2M).

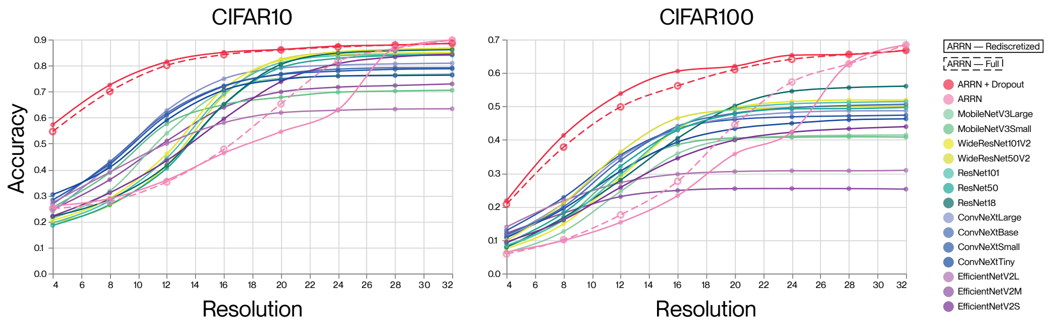

Figure 1: Robustness of all methods to changing resolution. Each model is trained on 32×32 resolution images and tested at various resolutions. Our method ARRN maintains its accuracy at lower resolutions much better than prior methods.

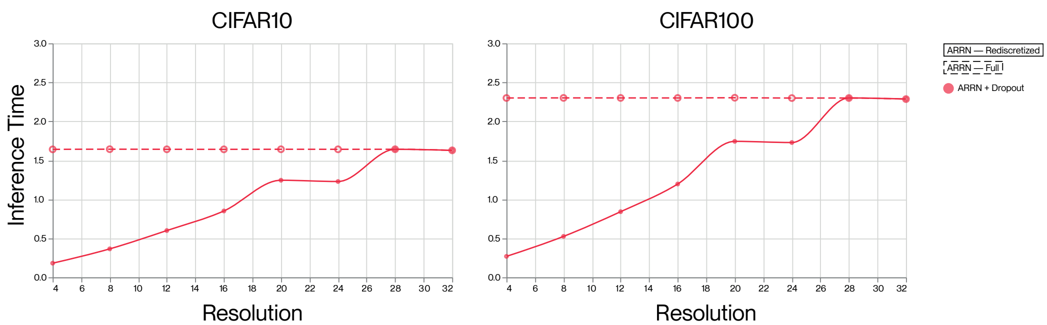

Figure 2: Inference time of ARRN at various resolutions. Our method ARRN lowers its inference time at lower resolutions thanks to rediscretization.

Rediscretization: correctness. We confirm that ARRNs can be rediscretized without degrading performance. Figure 1 shows the performance of ARRNs evaluated with rediscretization (full lines), meaning they discard certain Laplacian residuals, and without rediscretization (dashed lines), meaning they always use all Laplacian residuals. For models with Laplacian dropout (red lines), the performance is identical or better. For models without Laplacian dropout (pink lines), the performance is sometimes worse. This is likely a result of the approximate filters used by the implementation, which allow a small quantity of information to bleed through filters, which ARRNs can learn to depend on. This bleed-through is zeroed out by Laplacian dropout, which is consistent with the observation that ARRNs learn to correct for this error when Laplacian dropout is used.

Rediscretization: inference time. We confirm the computational savings granted by rediscretization, reusing the previous experimental setup. Figure 2 shows the inference time of ARRNs with rediscretization (full lines) and without rediscretization (dashed lines). As expected, rediscretization reduces inference time at lower resolutions, as a lower number of Laplacian residuals need to be evaluated.

Robustness. We validate the ability of Laplacian dropout to increase the robustness of networks to low-resolution signals. Figure 1 shows that ARRNs with Laplacian dropout (red lines) are vastly superior to ARRNs without Laplacian dropout (pink lines), and stronger than all baseline methods.

4 Discussion

We have introduced ARRN, an architecture for deep learning tasks that apply to multidimensional signals which addresses the problem of variation in signal resolution. ARRNs substitute standard residual connections with *Laplacian residual* connections, which allow incorporating a wide variety of architectural blocks into networks that instantly adapt to signal resolution, and that have a drastically lower computational cost at lower signal resolution. ARRNs also implement *Laplacian dropout*, which greatly promotes robustness to low signal bandwidth. These two components allow training high-resolution ARRNs that can then be adapted into low-resolution ARRNs which are robust and computationally efficient.

References

Burt, P. J. and Adelson, E. H. (1987). The laplacian pyramid as a compact image code. In *Readings in computer vision*, pages 671–679. Elsevier.

Elfwing, S., Uchibe, E., and Doya, K. (2018). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural networks*, 107:3–11.

Fourier, J. B. J. (1888). *Théorie analytique de la chaleur*. Gauthier-Villars et fils.

He, K., Zhang, X., Ren, S., and Sun, J. (2016a). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, K., Zhang, X., Ren, S., and Sun, J. (2016b). Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 630–645. Springer.

Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019). Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324.

Huang, G., Sun, Y., Liu, Z., Sedra, D., and Weinberger, K. Q. (2016). Deep networks with stochastic depth. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 646–661. Springer.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Bach, F. and Blei, D., editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France. PMLR.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2021). Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Lai, W.-S., Huang, J.-B., Ahuja, N., and Yang, M.-H. (2017). Deep laplacian pyramid networks for fast and accurate super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 624–632.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. (2020). Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*.

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., and Xie, S. (2022). A convnet for the 2020s. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11976–11986.

Loshchilov, I. and Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Petersen, D. P. and Middleton, D. (1962). Sampling and reconstruction of wave-number-limited functions in n-dimensional euclidean spaces. *Information and Control*, 5(4):279–323.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

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

Singh, S. R., Yedla, R. R., Dubey, S. R., Sanodiya, R., and Chu, W.-T. (2021). Frequency disentangled residual network. *arXiv preprint arXiv:2109.12556*.

Smith, J. O. (2002). Digital audio resampling home page "theory of ideal bandlimited interpolation" section. *Online] http://www-ccrma. stanford. edu/˜ jos/resample*.

Tan, M. and Le, Q. (2021). Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR.

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

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

Yang, Y.-Y., Hira, M., Ni, Z., Chourdia, A., Astafurov, A., Chen, C., Yeh, C.-F., Puhrsch, C., Pollack, D., Genzel, D., Greenberg, D., Yang, E. Z., Lian, J., Mahadeokar, J., Hwang, J., Chen, J., Goldsborough, P., Roy, P., Narenthiran, S., Watanabe, S., Chintala, S., Quenneville-Bélair, V., and Shi, Y. (2021). Torchaudio: Building blocks for audio and speech processing. *arXiv preprint arXiv:2110.15018*.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

5 Supplementary Material

We provide a short discussion of Laplacian pyramids (subsection 5.1) that helps interpret the formulation of our Laplacian residuals; we also include block diagrams (Figure 3, Figure 5) and example images (Figure 4, Figure 6) that highlight the parallel between Laplacian pyramids and Laplacian residuals. We also provide more details on our experimental setup in subsection 5.3.

5.1 Laplacian pyramids

Laplacian pyramids (Burt and Adelson, 1987) are a useful tool for decomposing signals s into m parts according to their frequency content. They are formulated through a recurrence relation that usually relies on Gaussian lowpass filter kernels, but that we may substitute with sine cardinal (Whittaker, 1927) filter kernels ϕ_n^{low}, which are ideal (meaning they act as binary masks in the frequency domain). These filter kernels are also assumed to nest into each other such that deeper filter kernels select for lower bandwidth (the *bandwidth* of *continuous signals* is analogous to the *resolution* of *discrete signals*). The base case (Equation 17) takes the original signal s as the starting point of recurrence p_0^{low}:

$$p_0^{\text{low}} = s \tag{17}$$

The recursive case takes the preceding lower bandwidth signal p_{n-1}^{low}, forms the next lower bandwidth signal p_n^{low} (Equation 18), and forms a difference signal p_n^{diff} (Equation 19) such that both parts sum to the preceding lower bandwidth signal:

$$p_n^{\text{low}} = p_{n-1}^{\text{low}} * \phi_n^{\text{low}} \tag{18}$$

$$p_n^{\text{diff}} = p_{n-1}^{\text{low}} - p_n^{\text{low}} \tag{19}$$

The conditional part (Equation 20) sets what we refer to as the level of the pyramid p_n to the difference signal p_n^{diff} for all levels except the last one, which is instead set to the lower bandwidth signal p_n^{low}. This ensures all pyramid levels sum to the original signal:

$$p_n = \begin{cases} p_n^{\text{diff}} & \text{if } n \neq m \\ p_m^{\text{low}} & \text{otherwise} \end{cases} \tag{20}$$

The Laplacian pyramid can be seen as a form of signal decomposition that allows us to reconstruct the signal with progressively more bandwidth, as we add more difference signals (indexing backwards from the last level of the pyramid):

$$s * \phi_n^{\text{low}} = p_m + p_{m-1} + \cdots + p_{n+1} + p_n \tag{21}$$

In Figure 3, we summarize this recursive formulation into a simple diagram that shows one step of recursion; this is intended to allow an easy comparison with the Laplacian residuals we illustrate in subsection 5.2. In Figure 4, we show an example where we decompose a pair of images using a shallow Laplacian pyramid.

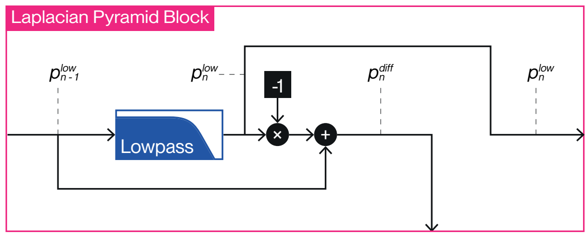

Figure 3: High-level diagram of a single recurrence step of a Laplacian pyramid.

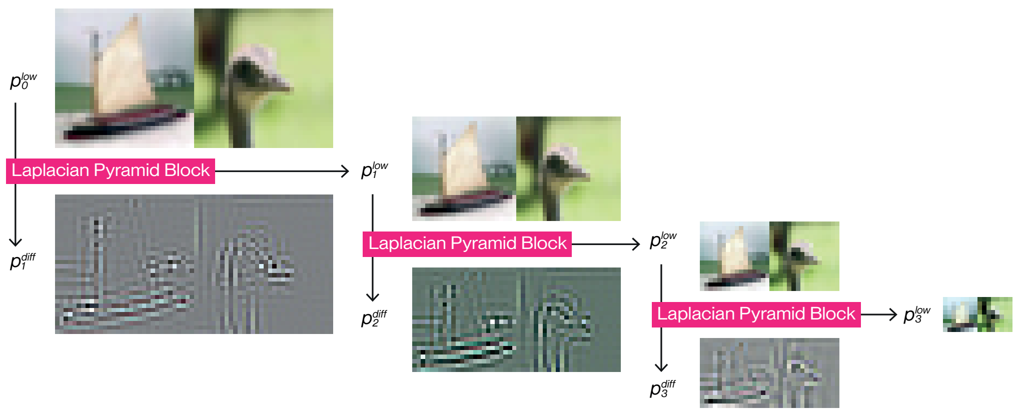

Figure 4: Images decomposed through a Laplacian pyramid. The recursive process starts in the top left with the source image p_0^{low} and progressively creates lower bandwidth signals p_{n+1}^{low} moving right, and difference signals p_{n+1}^{diff} moving down. Together, p_1^{diff}, p_2^{diff}, p_3^{diff} and p_3^{low} sum to the original signal p_0^{low}; they are a form of linear decomposition.

5.2 Laplacian residuals

In Figure 5, we illustrate the recursive formulation of Laplacian residuals in a simple block diagram. We can see on the left the same elements that compose the Laplacian pyramid shown in Figure 3. In Figure 4, we show a visualization of a small ARRN by tapping into r_n^{low}, the input to every Laplacian residual, and r_n^{diff}, the input to every architectural block wrapped within a Laplacian residual.

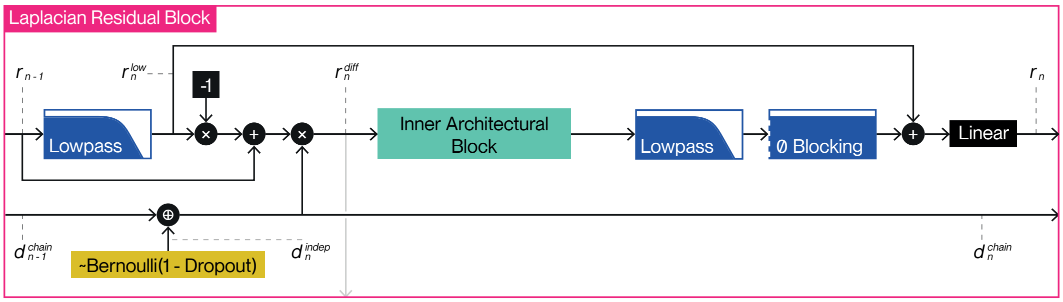

Figure 5: High-level diagram of a Laplacian residual which implements Laplacian dropout.

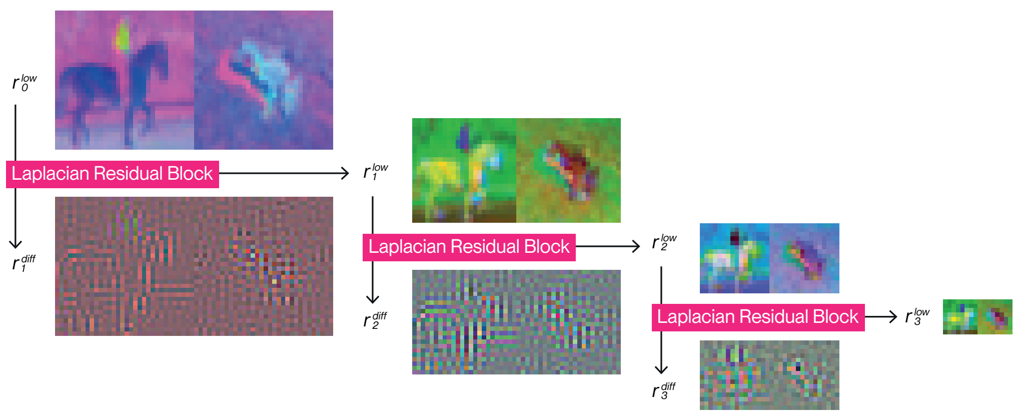

Figure 6: Images of PCA analysis of feature maps created by an ARRN's Laplacian residuals. The process starts in the top left with the source image r_0 that has been mapped through $\mathbf{A}_0$. Moving downwards, we see the difference signal r_{n+1}^{diff} that is later given to the architectural block b_n, which is formed in the same way as with Laplacian pyramids. Moving right, we get a lower bandwidth signal r_{n+1}^{low} based on the output of the last Laplacian residual.

5.3 Experiments

Model evaluation. In Figure 7, we illustrate how we evaluate networks at various resolutions in our experimental setup, after having trained them at a fixed resolution. With standard networks, inference must always take place at the training resolution; lower-resolution input signals must be rediscretized to a higher resolution first to be compatible. With ARRNs evaluated with rediscretization, lower-resolution input signals skip resizing and higher-resolution residuals, and go directly to corresponding lower-resolution residuals, which reduces computational cost. The grey parts of the illustration are skipped. With ARRNs evaluated without rediscretization, we follow the process we usually apply with standard networks, meaning we use all residuals. The grey parts of the illustration are not skipped.

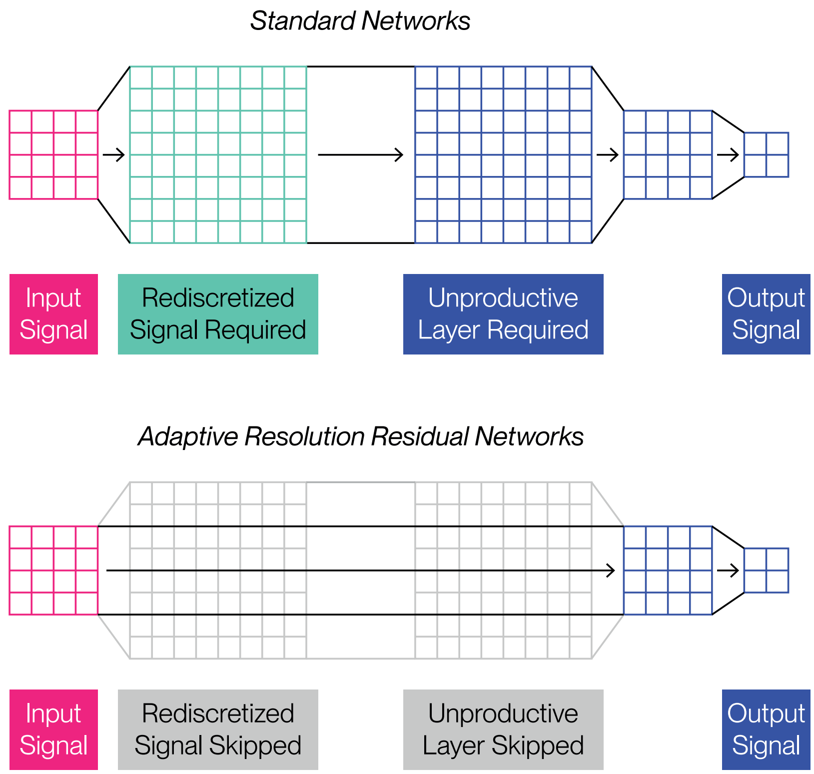

Figure 7: Schematized view showing how standard networks and ARRNs perform inference on lower resolution signals. Each grid shows the discretization of an intermediate signal at some stage in the forward pass of either network; black arrows show the relationship between each intermediate signal in the forward pass; black lines highlight changes in signal resolution. In ARRNs with rediscretization enabled, the intermediate signals faded to grey are skipped.

Model design. The method we propose leaves much freedom for the design of ARRNs; the number of Laplacian residuals, their sizes, their number of features, and their inner architectural block can all be freely picked. The architectural hyperparameters we used in our experiments were found using a series of hand searches and block coordinate searches, maximizing the average accuracy over evaluated resolutions.

We use inner architectural blocks that take inspiration from the parameter-efficient convolutional layers that are used within MobileNetV2 (Sandler et al., 2018) and EfficientNetV2 (Tan and Le, 2021), illustrated in Figure 8 and Figure 9. We use a string of 2 or 3 depthwise 3×3 convolutions for CIFAR10 and CIFAR100 respectively, each separated with pointwise (1×1) convolutions. All depthwise convolutions use edge replication padding in order to satisfy Equation 1. The whole string is preceded by a pointwise convolution that expands the feature channel count by a factor of 8, and is terminated by a pointwise convolution that contracts it inversely by a factor of 8 to restore the original feature count. Each convolution is separated by a batch normalization (Ioffe and Szegedy, 2015) and a SiLU activation function (Elfwing et al., 2018), chosen for its tendency to produce fewer aliasing artifacts.

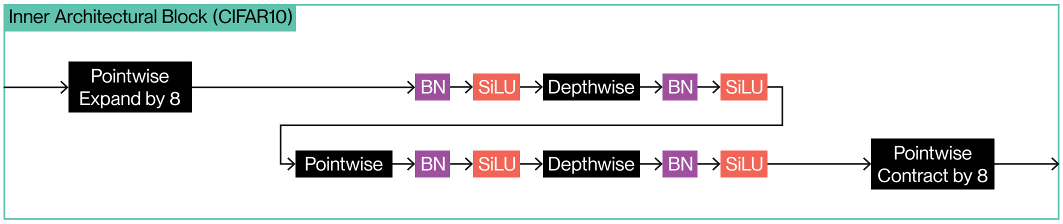

Figure 8: High-level diagram of an inner architectural block nested within a Laplacian residual, in the CIFAR10 ARRN.

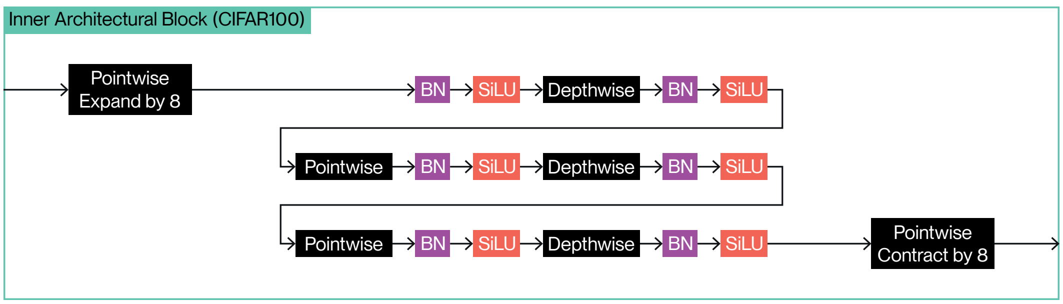

Figure 9: High-level diagram of an inner architectural block nested within a Laplacian residual, in the CIFAR100 ARRN.

We settled on an ARRN with 6 Laplacian residual blocks of size $32 \times 32, 24 \times 24, 16 \times 16, 12 \times 12, 8 \times 8, 4 \times 4$, with feature channel counts of $32, 48, 64, 96, 128, 256$. When enabled, we use a Laplacian dropout rate of 0.6 and 0.3 on CIFAR10 and CIFAR100 respectively.

Model training. For training all methods, we use AdamW (Loshchilov and Hutter, 2017) with a learning rate of 10^{-3} and $(\beta_1, \beta_2) = (0.9, 0.999)$, cosine annealing (Loshchilov and 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.