Using Recurrent Architectures to Generalize From Easy to Hard Problems in Image Denoising

Naveen Raman [1] **Andrew Mao** [1] **Vatsal Agarwal** [1] **Aman Jaiman** [1]

Abstract

Machine learning algorithms should be robust to domain shifts; one type of shift is when the test data is more difficult than the data seen at train time. To tackle this, we propose the problem of easy-to-hard generalization for image denoising, and develop a noisy image dataset with varying levels and types of noise. We find that a recurrent architecture is significantly smaller than state-of-the-art models while achieving only a small drop in performance. We incorporate denoising-specific methods, such as dynamic filters and perceptual loss, and demonstrate that this combination outperforms state-of-the-art models. Our work highlights that recurrent networks can reduce the model size and improve performance when task-specific modules are incorporated.

1. Introduction

A critical feature of intelligent systems is the ability to generalize to unseen environments. Humans can logically extrapolate to solve new and challenging problems by assembling known rules into complex strategies (Son et al., 2008), and such extrapolation to new problems is desirable for machine learning systems. In particular, robust machine learning systems should ideally be able to generalize to more difficult problems.

Previous work has employed techniques such as meta learning (Li et al., 2018) and fine tuning (Sun et al., 2019) for generalization. However, many of these methods require additional data from testing distributions to augment training, which might not always be available. Another approach is the use of recurrent "deep-thinking" networks, which can be run for arbitrary amounts of time, allowing for varying computational power based on task difficulty (Schwarzschild et al., 2021). This approach has proven effective on simple problems, like prefix sums and mazes, but has not yet been applied successfully to more complex tasks.

We propose the task of easy-to-hard generalization for image denoising, a common task in computer vision that's more difficult than prefix sums or mazes, and develop a dataset consisting of images with varying levels and intensities of noise. We apply a general recurrent architecture to denoising, and compare the extrapolation behavior of the architecture with state-of-the-art denoisers. Finally we incorporate denoising-specific features, such as perceptual loss and dynamic filters, to show that general recurrent architectures can be adapted to the task at hand, combining small models with good performance. [1]

Our contributions are the following. We:

1. Formalize the problem of easy-to-hard generalization and apply it to image denoising by developing a dataset consisting of noisy images of varying difficulty.
2. Develop recurrent models for denoising and show that they reduce the model size significantly compared to feedforward baselines, while only suffering a small hit in performance.
3. Incorporate vision-based features into the model training and architecture, such as perceptual loss and dynamic filters, and show that a combination of these features improves upon state-of-the-art models.

2. Related Works

Image denoising includes different types of noise, including salt-and-pepper and Gaussian noise. Common techniques for denoising include the use of filters, such as Gaussian filters (Liang et al., 2021) and median filters (Shrestha, 2014). Other models use deep learning techniques, such as Convolutional Neural Networks (CNNs) (Tian et al., 2020). Our work can be viewed at the intersection of both of these techniques, leveraging neural techniques along with classical filters to perform denoising.

[1]Department of Computer Science, University of Maryland. Correspondence to: Andrew Mao <amao1@umd.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

[1]Our code and data is publicly available at https://github.com/Maosef/deep-thinking

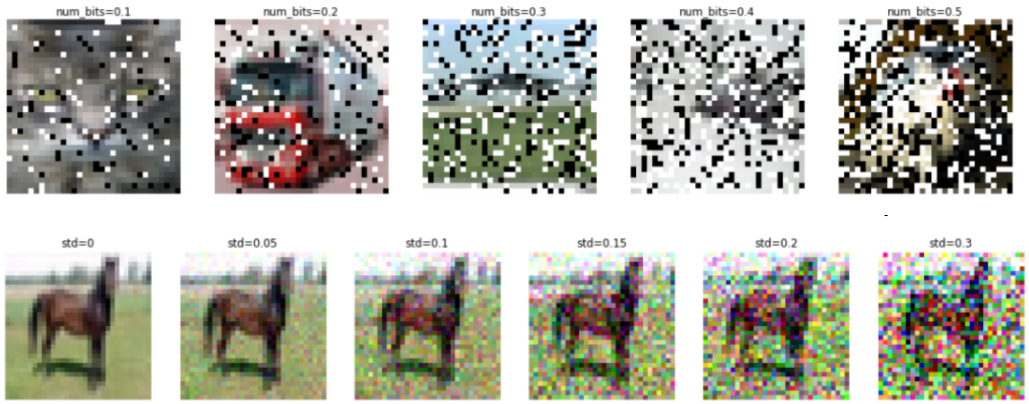

Figure 1. We show images with various amounts of salt-and-pepper (top) and Gaussian noise (bottom). Note how images become less recognizable as the noise level increases, corresponding to the increased difficulty of denoising.

Adaptive computation time (ACT) is a technique that allows a recurrent model to run a variable number of iterations, determining stopping time dynamically. Graves (2016) introduced the technique and used a learned halting unit to determine when to stop, penalizing the model for thinking longer. Improvements include a differentiable form of ACT (Eyzaguirre & Soto, 2020) and the use of an exploration loss term (Banino et al., 2021). These models found improved performance in tasks such as question answering and parity, in addition to faster inference. Our work focuses more on logical extrapolation, instead of early stopping or faster inference. We also use a recurrent convolutional architecture rather than one based on transformers.

Logical extrapolation is the task of generalizing to problems that are more computationally complex than those seen at train time. Schwarzschild et al. (2021); Anonymous (2022) uses recurrent models that process the entire input at each processing step. They find excellent extrapolation performance on solving prefix sums and mazes, but little to no extrapolation on more complex problems, such as chess puzzles. Banino et al. (2021) also achieves logical extrapolation on the parity task. Our work extends these previous methods of logical extrapolation to image denoising, a more complex task.

3. Problem Formulation

We formally define the problem of easy-to-hard generalization, using this notation for the remainder of the paper. Our goal, informally stated, is to train a model, f, on "easy" examples, and evaluate it on "hard" examples by running the model for more iterations. Running for more iterations on hard examples allows for extra computation, which can potentially improve performance. We define easy and hard examples by considering a function, $D(S)$, that outputs a difficulty metric for subsets of data points. For example, suppose we have k difficulty classes, $d_1, d_2, \cdots d_k$, where each difficulty class is a subset of questions. We let $D(d_i) < D(d_j)$ imply that d_i is easier than d_j, and order difficulty classes so d_1 is the easiest and d_k is the hardest. Our goal is to train on an easy subset, $s_e = \{d_1 \cdots d_i\}$ and test on a hard subset $s_h = \{d_{i+1} \cdots d_n\}$, where all difficulty classes in s_e are easier than all difficulty classes in s_h. If we let L denote some loss function and E denote an evaluation function, then we can state our goal as finding a function f that minimizes

$$\frac{1}{n} \sum_{d_i \in s_e} \sum_{x,y \in d_i} L(f(x), y), \tag{1}$$

where n is the number of data points. We evaluate the success of f by computing

$$\frac{1}{n} \sum_{d_i \in s_h} \sum_{x,y \in d_i} E(f(x), y) \tag{2}$$

Note that the evaluation function, E, is not necessarily the loss function, L. We want our loss function to encourage generalization, while the evaluation function is purely an assessment of task performance, and so the loss function can include extra terms to improve generalization.

We focus on the task of image denoising, which involves the replacement of noisy pixels. We decide to stick to one difficulty class for easy data, and test on all classes for hard data, though only report performance on 0.3 bits. We use two definitions of noise, salt-and-pepper noise and Gaussian noise, and detail our data generation procedure in Section 6.1. (Figure 1) shows examples of noisy images for varying levels of noisiness. Our evaluation function is the peak signal-to-noise ratio (PSNR) metric, which measures the difference between two images using MSE. Our evaluation function is

then

$$E(x,y) = 10\log(\frac{255^2}{\text{MSE}(x,y)})$$

4. Deep-Thinking Architecture

To solve the problem of easy-to-hard generalization, we build on prior work that developed a recurrent architecture (Schwarzschild et al., 2021), which we call the "vanilla" deep-thinking model.

The "vanilla" deep-thinking model is a fully convolutional network, with a recurrent block, which is a series of residual layers (He et al., 2016). This residual block is repeated N times, where N can be varied, with larger N for harder problems. Mathematically, we represent this as

$$f_N(x) = h(r^N(p(x), x)) \tag{3}$$

where p is a projection layer that takes the image to feature space, and h is a head that projects the features into image space. Running the recurrent block N times results in N thoughts, $t_1 \cdots t_N$, which are the intermediate images produced by the model. We show this visually in Figure 2.

We refer to the number of convolutional layers applied in a recurrent network as its "depth", and refer to the number of feature maps produced by each convolution as its "width."

We incorporate two features from prior work (Anonymous, 2022): recall and progressive loss. Recall concatenates a copy of the input at each step of the recurrent block. Prior work showed that adding recall was found to be important to the stability of the model across iterations; intuitively, it allows the model to look back at the problem at any time and reproduce features that may have been corrupted over many iterations.

The progressive loss term has the purpose of encouraging the system to learn iteration-agnostic behavior. To compute the progressive loss, the model is first run for N thoughts, producing thoughts $t_1 \cdots t_N$. The model is then re-run, with the starting input being t_N, and the model is run for k iterations, where k is a random number. Backpropagation is then only done using these k iterations.

Our combined loss function is then

$$(1-\alpha)L_{\text{fixed}} + \alpha L_{\text{prog}}, \tag{4}$$

where L_{fixed}, is the loss after N iterations, and α is a hyperparameter which weighs the loss terms. Both the fixed and progressive loss is the MSE of the output and target.

thoughts, $t_1 \cdots t_N$, which are the intermediate images produced by the model. We show this visually in Figure 2.

We refer to the number of convolutional layers applied in a recurrent network as its "depth", and refer to the number of feature maps produced by each convolution as its "width."

We incorporate two features from prior work (Anonymous, 2022): recall and progressive loss. Recall concatenates a copy of the input at each step of the recurrent block. Prior work showed that adding recall was found to be important to the stability of the model across iterations; intuitively, it allows the model to look back at the problem at any time and reproduce features that may have been corrupted over many iterations.

The progressive loss term has the purpose of encouraging the system to learn iteration-agnostic behavior. To compute the progressive loss, the model is first run for N thoughts, producing thoughts $t_1 \cdots t_N$. The model is then re-run, with the starting input being t_N, and the model being run for k iterations, where k is a random number. Backpropagation is then only done using these k iterations.

Our combined loss function is then

$$(1-\alpha)L_{\text{fixed}} + \alpha L_{\text{prog}}, \tag{5}$$

where L_{fixed}, is the loss after N iterations, and α is a hyperparameter which weighs the loss terms. Both the fixed and progressive loss is the MSE of the output and target.

5. Model Additions

We experiment with changes to the loss function and architecture, intending to improve extrapolation or denoising performance.

5.1. Similarity Loss

To enforce gradual progression of thoughts from noisy to clean, we add a similarity loss, which penalizes based on the distance between input and thought, where the weighting of the thought decreases as thoughts progress from input to output. Formally, we define the similarity loss as

$$L_{\text{sim}} = \sum_{i=1}^{n-1} \gamma^i \text{L}(\text{X}, \text{t}_\text{i}), \tag{6}$$

where L is a loss function, such as MSE, t_i are the thoughts, and γ is a hyperparameter that weighs similarity loss for different thoughts. Enforcing this loss constrains thoughts so they don't diverge too far from the original input, while also ensuring gradual progression. Doing so can potentially improve generalization, as models learn to incrementally think rather than solve the problem during early epochs, which can be useful as problem difficulty increases.

5.2. Perceptual Loss

We incorporate perceptual loss functions to more accurately compare image distances. Perceptual loss functions use hidden layers from pre-trained image classification networks,

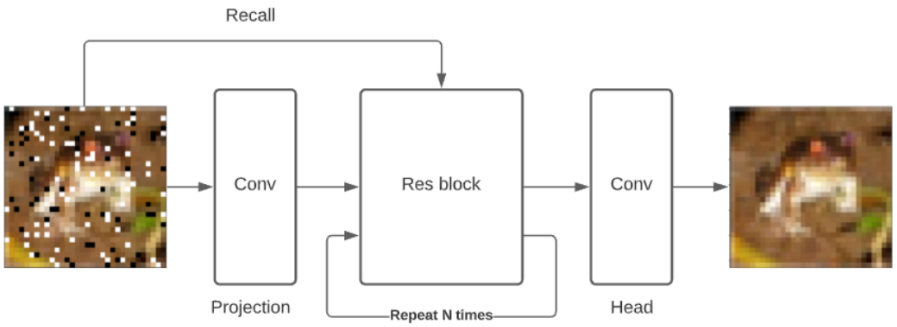

Figure 2. Our "vanilla" deep-thinking models relies on a recurrent block that's repeated N times. Repeated application of the recurrent block reduces noise, with each output from the residual block representing one "thought"

and compare L2 loss between image embeddings in the hidden layer (Johnson et al., 2016). Previous uses for perceptual loss include image sharpening (Zhou et al., 2020) and style transfer (Johnson et al., 2016). For our experiments, we use the hidden layer from a VGG classification network (Simonyan & Zisserman, 2014), and represent our overall loss as

$$(1-\alpha)L_{\text{acc}} + \alpha L_{\text{prog}} + \lambda L_{\text{sim}} + \beta L_{\text{perc}}, \qquad (7)$$

where L_{perc} is the perceptual loss, and α, β, γ are hyperparameters weighting the various loss functions.

5.3. Median Filters

Influenced by prior work which demonstrated the effectiveness of median filters for denoising (Liang et al., 2021), we incorporate median filters into the recurrent block of the deep-thinking model. We develop a median convolution layer (Liang et al., 2021), which combines convolution with median pooling, and incorporate this as the first operation in the recursive block. Medians naturally arise in denoising, as they allow for a replacement of noisy values with information from adjacent pixels, essentially smoothing out the image.

5.4. Dynamic Filters

We attempt to improve upon median filters through dynamic filters, which can learn patch-specific filters for an image. Inspired by Li et al. (2021), we model our dynamic filter generation using the involution architecture. Essentially, we transform k^2 1x1 filters into one $k \times k$ filter. Please refer to Li et al. (2021) for more details. For our purpose, dynamic filters allow the network to learn which filters to apply depending on pertinent image properties; different filters are needed to denoise an image based on the noisiness of the image.

6. Experiments

We describe our experiments which quantify the performance of models on easy-to-hard generalization. Our results show that incorporating new vision-specific features can significantly improve performance, and that general recurrent architectures can be fine-tuned for complex tasks.

6.1. Datasets

We develop two datasets: one based on salt-and-pepper noise, and another based on Gaussian noise. [2] We develop two salt-and-pepper noise datasets, one using CIFAR10 (Krizhevsky et al., 2009), and another using TinyImageNet (Le & Yang, 2015), while for Gaussian Noise, we develop a dataset using CIFAR10. We report all model results using CIFAR10, as we found similar results between CIFAR10 and TinyImageNet.

For the salt-and-pepper noise, we develop images with noise varying from 0.1 to 0.5 in increments of 0.1. For a noise level of 0.1, each pixel has a 10% chance to become corrupted, where pixels are flipped to either white or black uniformly. We develop these images under the assumption that noisier images are harder, which is reflected by decreases in performance by state-of-the-art models as noise level increases (Liang et al., 2021).

For Gaussian Noise, we perturb every pixel according to a normal distribution $N(0, \sigma)$, where σ is in $\{0.05, 0.1, 0.15, 0.2, 0.3\}$ and pixels are clipped between 0 and 1. Gaussian noise allows us to test whether the type of noise added impacts the success of deep-thinking models.

[2] Our salt-and-pepper dataset is available at https://drive.google.com/drive/folders/1MzBZo0ucUeJP1kdupq27ciNsV04nZUg5?usp=sharing

Model	#Parameters	PSNR (easy)	PSNR (hard)
DnCNN (SOTA)	669,312	25.18	23.40
CNN+median filters (SOTA)	3,563,011	23.61	21.91
CNN baseline	84,960	24.92	19.92
DT baseline (ours)	**27,360**	25.46	21.01
DT+similarity loss	27,360	22.53	17.29
DT+perceptual loss	27,360	29.57	21.51
DT+median filter	27,360	19.76	18.30
DT+dynamic filter	27,487	25.27	22.97
DT+dynamic filters+perceptual loss	27,487	**28.98**	**25.30**

Table 1. Comparison of models on CIFAR with salt-and-pepper noise. The recurrent deep-thinking model with dynamic filters and perceptual loss outperforms state-of-the-art with substantially fewer parameters.

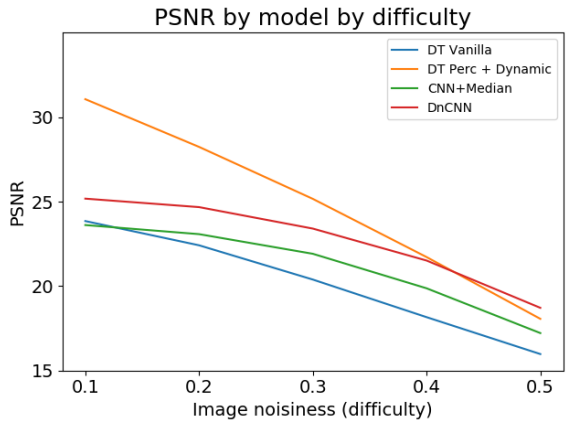

Figure 3. We show how various models perform as the task difficulty increases. Our DT model with perceptual loss and dynamic filters (orange) outperforms a state-of-the-art model (DnCNN) when tested on unseen noisier images, up to 50% pixel corruption.

6.2. Baseline Models

We establish the difficulty of easy-to-hard generalization by evaluating state-of-the-art denoisers on our salt-and-pepper dataset. We list experimental details in Section A. We compare two models: DnCNN (Zhang et al., 2017) and CNN+median filters (Liang et al., 2021). DnCNN was designed for Gaussian denoising and consists of a series of convolution-batch normalization-ReLU layers. CNN+median filters combine CNNs with Median Filters and consist of 16 residual blocks, each with a median layer in between, and then another 16 residual blocks with no median layers. CNN+median filters are currently the state-of-the-art salt-and-pepper denoiser. Finally, we train a baseline CNN model, which is simply a feed-forward CNN not fine-tuned for denoising, as another baseline.

6.3. Deep-thinking models

We compare deep-thinking models to baseline state-of-the-art models on salt-and-pepper noise.

6.3.1. Deep-thinking Performance

We compare performance on easy and hard images in Table 1. We find that state-of-the-art models are significantly larger than deep-thinking models. We also find that, while deep-thinking models perform worse than both DnCNN and CNN+median filters on hard images, their performance is within a couple of points in PSNR. Additionally, deep-thinking models perform better than a baseline CNN on both easy and hard tasks. That is, without any denoising-specific modifications, we find that deep-thinking models perform only slightly worse than state-of-the-art, while significantly reducing model sizes.

6.3.2. Recall and Progressive Loss

Recent work (Anonymous, 2022) found that introducing recall and progressive loss can improve the performance of deep-thinking models (Section 4); we evaluate these claims on denoising, and find that recall can significantly improve denoising models. In particular, we find significant instability when training without recall. This instability manifests itself as the model only being able to learn greyscale images, and in some cases, the model fails to learn anything at all. Introducing recall allows recurrent blocks access to the original input, which reduces instability and improves performance in practice.

We run experiments comparing the impact of progressive loss by varying α in $\{0, 0.5, 1\}$. We compute the PSNR for each thought, and find that large values of α essentially regularize thinking so thoughts converge quickly (Figure 6). However, smaller values of α allow for thinking progression, as PSNR increases even past the training regime, where the training regime refers to the first 5 thoughts.

6.3.3. Model Extrapolation

The intuition behind deep-thinking models is that additional thinking iterations should improve performance, however, in practice, we find this is dependent on α. During training, we run the model for five thinking iterations, and so ideally, we want to see extrapolation beyond this training regime, where performance increases after five iterations. However, in practice, we fail to see such extrapolation for large α, but see some form of convergence for $\alpha = 0$, as PSNR increases even after 5 iterations. However, for $\alpha = 0$, we note that PSNR decreases then increases, with the sharp decrease occurring at the end of the training regime.

6.3.4. Performance on harder images

As expected, we find that model performance decreases with increased image noisiness, and we see a roughly linear relationship between image noisiness and PSNR (Figure 3). We see that DnCNN performs better than all models for very difficult images, with 0.5 noisiness, but our model with perceptual loss+dynamic filters outperforms all other models for difficulties between 0.1 and 0.4. However, we also note that our perceptual loss+dynamic filters model decreases in PSNR faster than all other models; improving this should be investigated by future work.

6.4. Model Addition Performance

We add a variety of filters and loss functions to our baseline deep-thinking model, and note our primary takeaways for each type of addition. We note results for each addition in Table 1 and plot thought progressions for each addition in Figure 7.

1. **Similarity loss may abruptly denoises images** - We see through qualitative analysis that when using similarity loss, thoughts stay similar to the input for the first several inputs, then suddenly switch over to being denoised. This might be because, after a certain point, the weighting for similarity noise, γ^i, is less the weighting for fixed noise, $1 - \alpha$, making denoising performance more important than similarity to input.

2. **Perceptual loss improves performance, especially for easier problems** - We find that incorporating perceptual loss can boost performance across difficulty classes. This effect is especially pronounced for 0.1 noisiness images, where performance is improved by 4 PSNR compared to the deep-thinking baseline.

3. **Median filters fail to perform adequately** - We find that, through both quantitative and qualitative results, that median filter models perform poorly. Part of this arises from the nature of model training; median filter training is time-consuming (Section 5.3), making it difficult to fine-tune models for these filters.

4. **Dynamic filters improve performance marginally while boosting extrapolation** - While overall PSNR for models using dynamic filters is comparable to vanilla deep-thinking models, dynamic filters tend to increase PSNR beyond the training regime. Though the increase is not large, it shows how custom filters can adapt based on the noisiness of the image.

We additionally try combining filters with loss functions, and find that combining perceptual loss with dynamic filters can boost model performance, making it on-par with or better than state-of-the-art models. We note two takeaways from these experiments

1. **Perceptual loss works well with dynamic filters** - Perceptual loss functions improve performance, especially on easy images, while dynamic filters help with generalizations, as extra thinking iterations improve model performance. By combining both of these, we find that model performance increases across the board (Figure 3), and outperforms state-of-the-art by 2 PSNR. This shows the impact that task-specific features can have on recurrent models, allowing them to outperform state-of-the-art.

2. **Model additions exhibit instability between random seeds** - We find that our perceptual loss + dynamic filter model varies across random seeds. For some values of a random seed, our model fails to learn anything, and the reason why is unknown. We report values in Table 1 for a fixed random seed.

6.5. Model Visualizations

To better understand the model results, we visualize the feature activations of our recurrent block for each model configuration. Specifically, we examine both the spatial and channel activations. These visualizations are shown in Figure 8. Furthermore, we also analyze the feature gradients for the recurrent block, which are shown in Figure 9.

Examining the forward feature activations, we find that for all configurations, the model activations do not capture the entire foreground object (specifically the deer). Rather, they seem to focus on only the central and more lit-up area. Moreover, we observe that the use of the recurrent block seems to elicit little change in the spatial and channel feature activations after the first few iterations. This relates to our discussion of model extrapolation as we see that more iterations of the recurrent block do not lead to significant refinement of the deep features, so future work is needed to improve this type of thinking. We do see, however, that there is a large variance between our different model configurations.

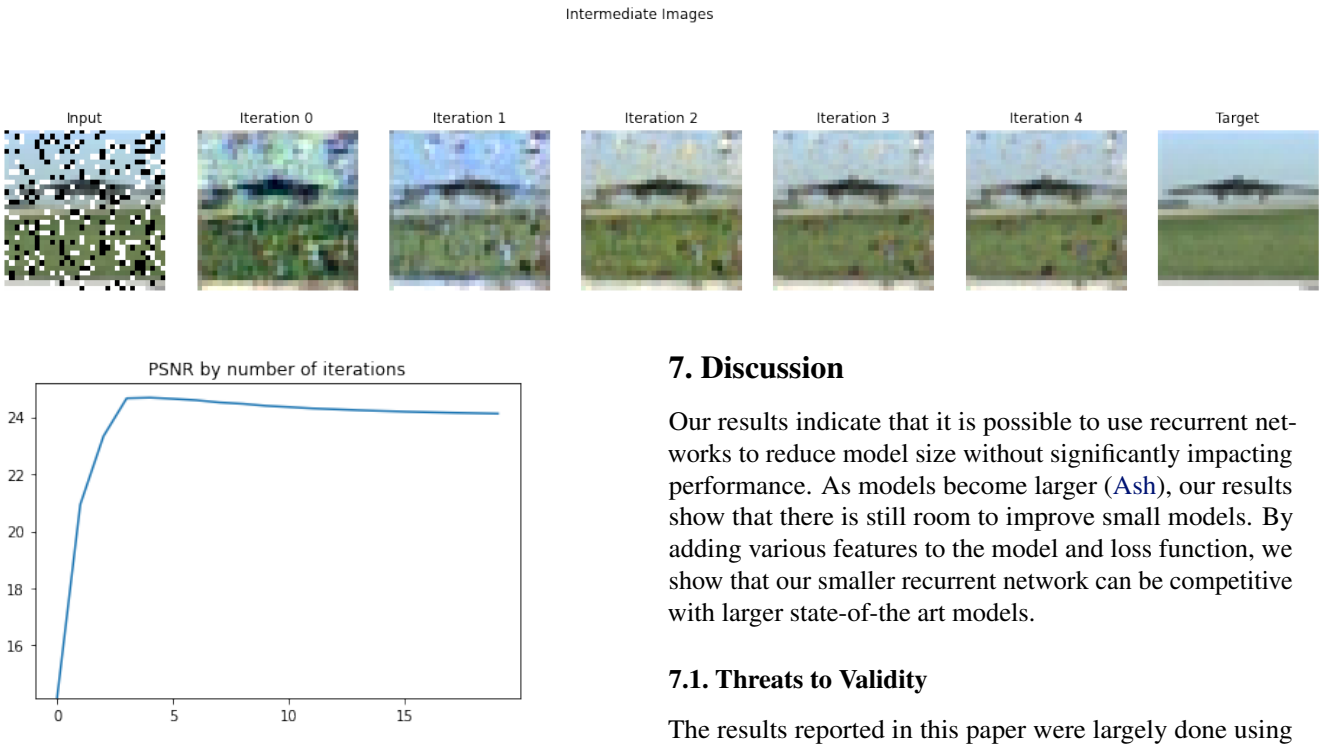

Figure 4. We plot PSNR and intermediate visualizations for our best model, which combines perceptual loss and dynamic filters.

For the gradient activations, we note that most focus is paid towards the background of the image. This could perhaps be due to this region containing more noise, compared to the foreground. While most of the model configurations show similar gradient activations, it is worth noting that the median filter model has extremely dense activations as the map is completely saturated. This could explain the poor results of this configuration.

6.6. Extension to Gaussian Noise

We apply our deep-thinking model to Gaussian noise, to see the impact that other definitions of difficulty have on model performance. We train our vanilla deep-thinking model on a Gaussian noise dataset with $\sigma = 0.1$, and test on all difficulties. We find that, regardless of task difficulty, models trained on Gaussian noise tend to converge within a few iterations, and fail to improve past the training regime (Figure 5). These results explore the performance of deep-thinking for other types of noise, and demonstrate that the inability of deep-thinking models to improve beyond the training regime is a function of the model itself rather than our definition of noise. However, this might change if additions such as perceptual loss or dynamic filters are used with Gaussian filters, which would be interesting future work.

7. Discussion

Our results indicate that it is possible to use recurrent networks to reduce model size without significantly impacting performance. As models become larger (Ash), our results show that there is still room to improve small models. By adding various features to the model and loss function, we show that our smaller recurrent network can be competitive with larger state-of-the art models.

7.1. Threats to Validity

The results reported in this paper were largely done using salt-and-pepper noise with the CIFAR10 dataset. Although we experimented with other datasets and types of noise, and found similar results, it is unknown whether our methods generalize well across all types of noise and datasets.

We were unable to find strong extrapolation behavior beyond the training regime for noisier images, which challenges the ability of deep-thinking to work for more complex tasks. The reasons for this are unclear; there could be a number of reasons, such as insufficient hyperparameter tuning, an improper loss function, or because the problem is not suitable for extrapolation. There is evidence that the model is not learning an iterative process, as the bulk of the gain in PSNR of our models comes after just a few iterations.

There is evidence that dynamic filters enable a small amount of extrapolation outside the train regime; further work is needed to investigate this.

Finally, our perceptual loss and dynamic filter models exhibited instability, so we reported results from the same random seed across runs. When we tried to average runs over multiple seeds, we ran into memory-related issues, which originates from bad model initialization; the way to fix this is unclear and is left for future work.

7.2. Future Work

Future work is needed to study and characterize the types of problems that the deep-thinking paradigm works on. Simple extensions include testing our models on other types of noise, such as patch-based or adversarial noise. The easy-

to-hard paradigm could be applied on other tasks in vision, such as segmentation, inpainting, unscrambling, and classification. More work could be done to examine the scaling behavior of our recurrent models, and whether techniques from model distillation can be applied to further shrink the model. Finally we could examine the impact of different combinations of training and testing difficulties; for example, train on harder data, or including a small amount of hard data in the training set.

8. Conclusion

We develop a dataset to assess the generalization of models from easy to hard examples in denoising. We find that using a recurrent architecture for denoising reduces model sizes while minimally impacting performance. Modifications to the loss function and the incorporation of dynamic filters can further boost performance and generalization, showing how task-specific fine-tuning can improve general recurrent models. Our results show the application of recurrent models for easy-to-hard generalization, which future work can expand to other tasks and problems.

Ethical Impact

We address model complexity by developing a method that reduces complexity significantly while achieving similar performance on denoising. Large models lead to environmental degradation through carbon emissions (Bender et al., 2021). While our model takes steps towards reducing model size, future work is necessary to further reduce sizes to minimize the environmental impact of deep learning.

References

Anonymous. Thinking deeper with recurrent networks: Logical extrapolation without overthinking. In *Submitted to The Tenth International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=kDF4Owotj5j. under review.

Ash, T. Machine learning is getting BIG. https://www.speechmatics.com/blog/machine-learning-is-getting-big-part-i/. Accessed: 2021-12-09.

Banino, A., Balaguer, J., and Blundell, C. Pondernet: Learning to ponder. *arXiv preprint arXiv:2107.05407*, 2021.

Bender, E. M., Gebru, T., McMillan-Major, A., and Shmitchell, S. On the dangers of stochastic parrots: Can language models be too big?. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pp. 610–623, 2021.

Eyzaguirre, C. and Soto, A. Differentiable adaptive computation time for visual reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12817–12825, 2020.

Graves, A. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016.

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

Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pp. 694–711. Springer, 2016.

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

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

Li, D., Yang, Y., Song, Y.-Z., and Hospedales, T. M. Learning to generalize: Meta-learning for domain generalization. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Li, D., Hu, J., Wang, C., Li, X., She, Q., Zhu, L., Zhang, T., and Chen, Q. Involution: Inverting the inherence of convolution for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12321–12330, 2021.

Liang, L., Deng, S., Gueguen, L., Wei, M., Wu, X., and Qin, J. Convolutional neural network with median layers for denoising salt-and-pepper contaminations. *Neurocomputing*, 442:26–35, 2021.

Schwarzschild, A., Borgnia, E., Gupta, A., Huang, F., Vishkin, U., Goldblum, M., and Goldstein, T. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks, 2021.

Shrestha, S. Image denoising using new adaptive based median filters. *arXiv preprint arXiv:1410.2175*, 2014.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Son, J. Y., Smith, L. B., and Goldstone, R. L. Simplicity and generalization: Short-cutting abstraction in children's object categorizations. *Cognition*, 108(3):626–638, 2008.

Sun, C., Qiu, X., Xu, Y., and Huang, X. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pp. 194–206. Springer, 2019.

Tian, C., Fei, L., Zheng, W., Xu, Y., Zuo, W., and Lin, C.-W. Deep learning on image denoising: An overview. *Neural Networks*, 2020.

Zhang, K., Zuo, W., Chen, Y., Meng, D., and Zhang, L. Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. *IEEE transactions on image processing*, 26(7):3142–3155, 2017.

Zhou, C., Zhang, J., Liu, J., Zhang, C., Fei, R., and Xu, S. Perceppan: Towards unsupervised pan-sharpening based on perceptual loss. *Remote Sensing*, 12(14):2318, 2020.

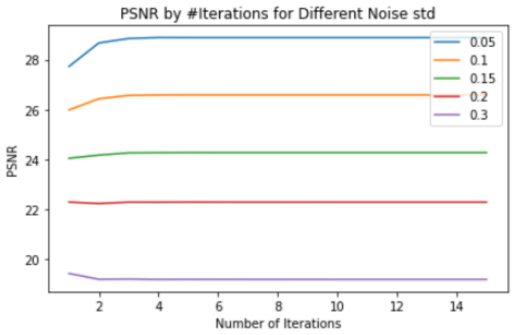

Figure 5. We plot PSNR for various amounts of Gaussian noise, and find that Gaussian noise does not improve extrapolation, as errors do not decrease or change past the training regime.

9. Appendix

A. Experimental Setup

We run baseline models on Google Colab, and run our deep-thinking models using a Red Hat Enterprise 7.9 server with 2 CPUs on an Intel Xeon E3-12xx v2 processor, with 40 GB of RAM and 1 16GB GPU. When using filters and variations of the loss function, we let $\lambda = 0.08, \gamma = 0.8$. Additionally, we train using a learning rate of $3 * 10^{-8}$, use a cosine annealing scheduler, and train models width of 20 and depth 20. We additionally train all models for 20 epochs. We train all models on noise level 0.1, which we define as easy data, and test on data with noise level 0.3, which we define as hard data. During training, deep-thinking models are run for a max of 5 iterations. Each dataset is split into 80% training and 20% validation randomly.

B. Comparison of DT and Feedforward Models

We compare DT and feedforward models for different values of α. We find that a recurrent architecture improves over its feedforward counterpart, while using fewer parameters. We find that adding progressive loss decreases the PSNR after 5 iterations, but maintains that PSNR for more iterations.

Model Type	Alpha	PSNR 0.1	PSNR 0.3
FeedForward	0	24.93	19.92
	0.5	24.65	19.43
	1	23.63	19.03
Deep-thinking	0	25.61	21.03
	0.5	24.89	20.06
	1	24.83	19.93

Table 2. A comparison of DT and feedforward models for different values of α.

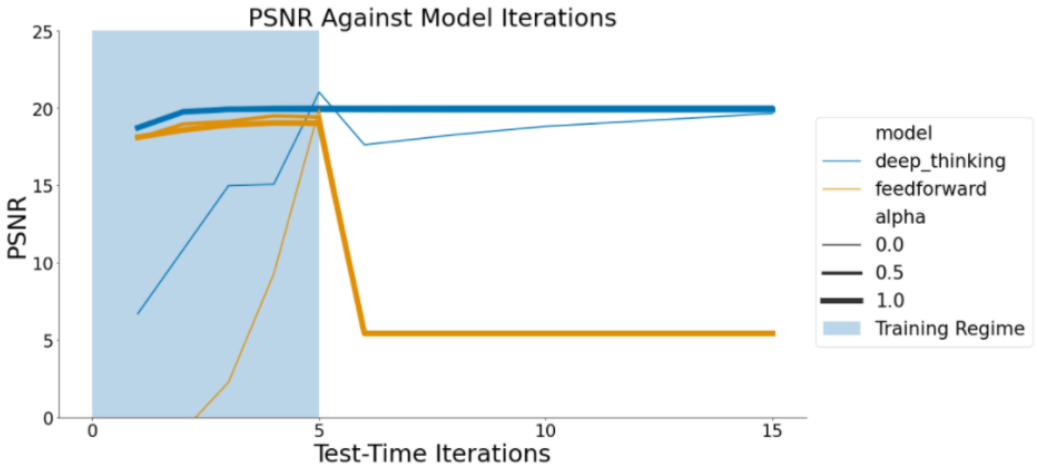

Figure 6. We find that feedforward models perform worse than deep-thinking models, and that adding progressive loss lead to stability outside the train regime.

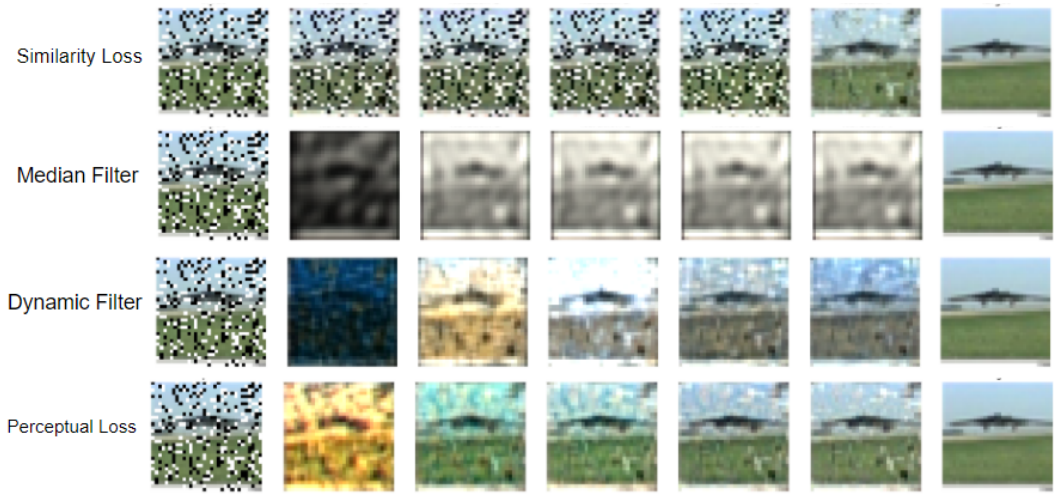

Figure 7. We show thought progressions for various additions to the model. Notice how similarity loss gradually removes noise from the image.

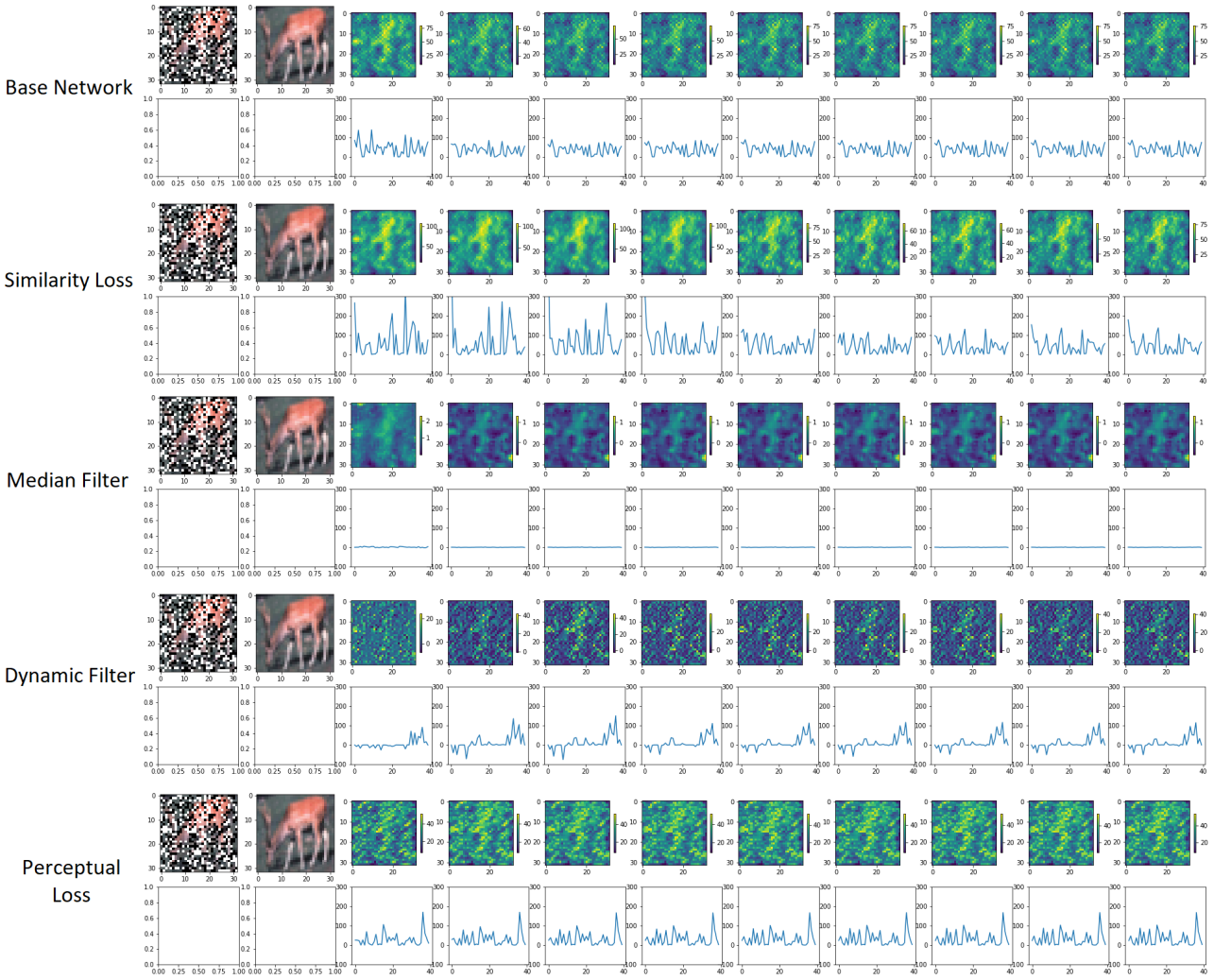

Figure 8. We visualize the feature activations produced by our recurrent block over the course of ten iterations for each model configuration. The top subplot shows the mean spatial activations and the bottom subplot shows the mean channel activations. We observe that the feature activations remain fairly static after the first few iterations of the recurrent block. Furthermore, we note that the feature activations are focused on only a few areas of the foreground region.

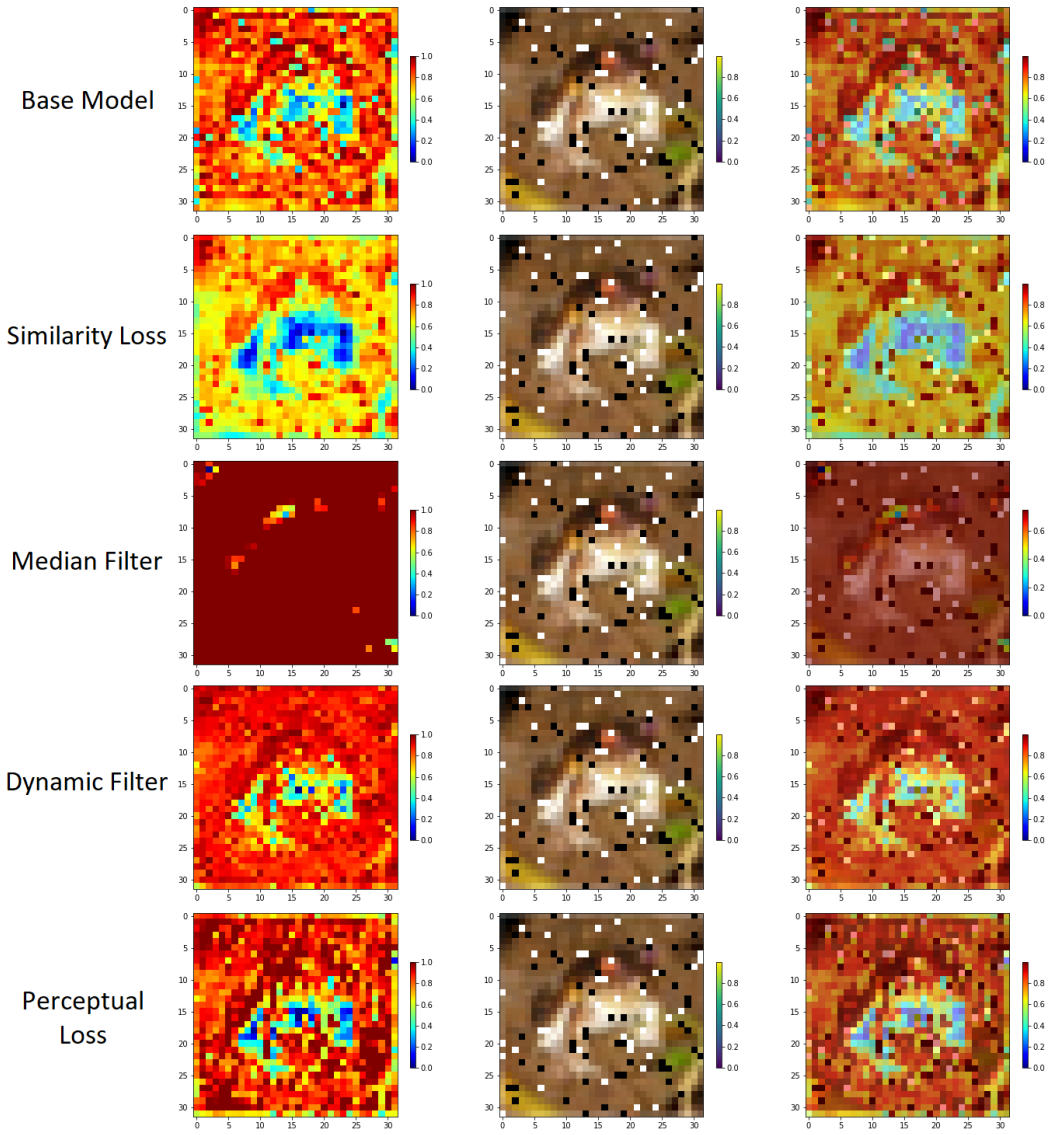

Figure 9. We visualize the feature gradients for the recurrent block for each model configuration and mask them over the original image. We observe that much of the gradient activations are focused on the background regions (perhaps due to more noise there). Furthermore, we note a potential failure of the median filter due to the highly saturated nature of the gradient activation maps.