Triangular Dropout: Variable Network Width without Retraining

Edward W. Staley [1] **Jared Markowitz** [1]

Abstract

One of the most fundamental choices in neural network design is layer width: it affects the capacity of what a network can learn and determines the complexity of the solution. The latter is often exploited when introducing information bottlenecks, forcing a network to learn compressed representations. Unfortunately, network architecture is typically immutable once training begins; switching to a more compressed architecture requires retraining. In this paper we present a new training strategy, Triangular Dropout, that allows effective compression without retraining. It provides for ordered removal of parameters by the user after training, enabling an explicit trade-off between performance and computational efficiency. We demonstrate the construction and utility of the approach through two examples. First, we formulate Triangular Dropout for autoencoders, creating models with configurable compression after training. Second, we apply Triangular Dropout to retrain the fully connected top layer of VGG19 on ImageNet. In both cases, we find only minimal degradation in the performance of the pruned network for even dramatic reductions in its number of parameters.

1. Introduction and Background

Compressing neural network representations without appreciably reducing their performance is a valuable capability in numerous compute-limited applications. In this paper we describe a novel dropout (Srivastava et al., 2014) technique that allows for ordered pruning of fully connected layers after training, maximizing performance at a desired width without requiring additional training. This "triangular" dropout strategy amounts to masking network connections during training with a binary lower triangular matrix, producing order in the relative importance of different connections that can then be leveraged for compression.

The width of a specific network layer is perhaps most critical in the domain of autoencoding (Rumelhart et al., 1986), where the width of the encoding layer (along with its activation) determines the amount of information retained in an encoded sample. The degree to which the latent variables are independent is the subject of disentanglement research, which introduces further properties into the latent variables. β-VAE (Higgins et al., 2017) demonstrates that the gaussian distributions in a variational autoencoder (VAE; (Kingma & Welling, 2014)) can be made increasingly orthogonal in exchange for reconstruction accuracy. The independence and ordering of learned features can also be enforced manually by methods such as Principal Component Analysis Autoencoder (PCAAE; (Pham et al., 2020)). PCAAE learns one latent variable at a time, progressively widening the latent layer to learn additional features via additional training.

Autoencoding makes use of the well-known trade-off between network size and capability, a relationship that holds more generally (Amodei & Hernandez, 2019; Kaplan et al., 2020; McCandlish et al., 2018). This is perhaps most well known with regards to ImageNet (Deng et al., 2009), where a clear correlation is typically observed between network size and performance (summarized nicely in Figure 1 of (Tan & Le, 2020)).

Large neural networks can be very inefficient in their use of parameters, leading to wasted compute. Network pruning is a technique to reduce an overparameterized network to its essential elements, recovering a subnetwork that is similar or even superior in performance (Blalock et al., 2020; Frankle & Carbin, 2018). Our method can be considered a structured pruning technique for MLPs, in which the recovered subnetwork is dense. Many existing works in this area focus on convolutional elements, such as in (Anwar et al., 2015) (channels and kernels), (Li et al., 2016) (filters), and (He et al., 2017) (channels). Critically, unlike the above works and many others (Molchanov et al., 2016;

[1]Johns Hopkins University Applied Physics Laboratory, Laurel, MD 20723, USA. Correspondence to: Edward Staley <edward.staley@jhuapl.edu>, Jared Markowitz <jared.markowitz@jhuapl.edu>.

DyNN workshop at the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, 2022.

Luo et al., 2019), Triangular Dropout does not require fine-tuning and is not iterative. In Section 4, we explore dense prunings recovered by Triangular Dropout on the classification head of VGG19, an early ImageNet solution that is well-known to be overparameterized (as evidenced by subsequent works with fewer parameters (Iandola et al., 2016; Tan & Le, 2020)).

2. Triangular Dropout

Triangular Dropout works by dictating the reliance of a network on particular connections. For a fully-connected layer of width n, processing a minibatch x of B datapoints, the output y has size (B, n). During training, standard dropout multiplies the output element-wise by a random binary mask $\boldsymbol{M}$ with mean $1 - p$ and the same size as the output. Triangular Dropout maintains this form, but the mask $\boldsymbol{M}$ is populated as a binary lower triangular matrix instead of a randomly sampled one:

$$y = \boldsymbol{M}_{Tri} \odot \alpha(\boldsymbol{x}^T \boldsymbol{W} + \boldsymbol{b})$$

where

$$\boldsymbol{M}_{Tri} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{pmatrix}$$

.

Here $\boldsymbol{W}$ and $\boldsymbol{b}$ are the layer weights and biases, respectively, and α is some activation function. The dropout scheme imposed by $\boldsymbol{M}_{Tri}$ imparts progressive levels of dropout across the training batch, and therefore the layer parameters. This is further visualized in Figure 1 (Left). In the case where $B > n$, the rows of $\boldsymbol{M}_{Tri}$ are simply repeated until the desired size is reached. In the case where $B < n$, a triangular block matrix may be used. For any B, an approximate alternative would be to independently sample rows such that a random number of 1s are followed by 0s.

This dropout scheme can be thought of as simultaneous training over a variety of layer widths. Because the mask is triangular, the jth sample in a minibatch results in layer outputs 1 through j being non-zero. Parameters relevant to the jth output are only updated for samples in which the previous $j - 1$ outputs (smaller widths) are also updated. This is not true for subsequent nodes: nodes $j + 1$ and onwards may or may not be masked when node j is utilized. Hence the parameters related to node j can be learned with a guarantee that the preceding $j - 1$ layer parameters are also present. However they cannot rely as heavily on outputs $j + 1$ and beyond.

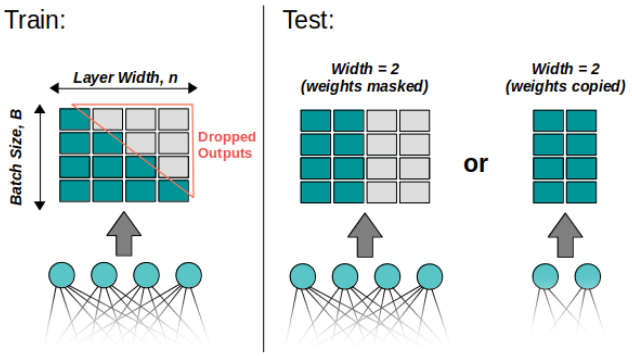

Figure 1. Diagram of Triangular Dropout being applied in training (Left) and testing(Right). At test time, layer width can be controlled by either masking layer output or copying relevant weights into a narrower architecture.

3. Variable Compression in Autoencoders

To further elucidate the properties of Triangular Dropout and compare it to alternative methods, we demonstrate the construction of an autoencoder with selectable size. We train an autoencoder on MNIST (Deng, 2012) with a latent size n, and use Triangular Dropout on the latent layer. After training, we can select any level of compression $\leq n$ and produce both a valid encoding and reconstruction. We also train autoencoders on the CelebA dataset (Liu et al., 2015) (with image size reduced to 64x64), in order to explore Triangular Dropout on a more difficult autoencoding task.

We compare MNIST results quantitatively (reconstruction loss) and qualitatively (visualization of reconstruction) to two baselines: n autoencoders that are trained for specific latent sizes 1 through n, and PCAAE, which trains an autoencoder of size n progressively. PCAAE trains one latent variable at a time and includes a correlation minimization loss term. PCAAE also trains a separate encoder for each latent variable, and a new decoder for each width. This means that, like the set of standard autoencoders, PCAAE requires n training runs.

We examine the reconstruction loss of PCAAE and individual autoencoders as their width is varies, and again as the features from the full size (width 32) model are progressively removed. Triangular Dropout is trained once with maximum size 32 and can be pruned to a desired latent width. These reconstruction results are shown in Figure 2. Full architecture and training details are given in the Appendix.

For a given latent variable size z, the most performant model (without compression) is a standard autoencoder, the least performant is PCAAE, and in-between is Triangular Dropout. The baseline models with compressed en-

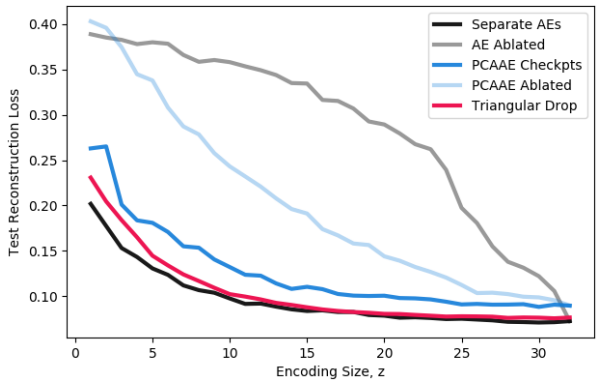

Figure 2. Reconstruction loss versus encoding size for a variety of methods. Triangular Dropout (red) is most similar to independent autoencoders trained at specific encoding sizes (black). PCAAE models produced during progressive training are shown in blue. Light gray and light blue show the reconstruction loss of the full-size standard autoencoder and PCAAE models, respectively, when the latent space is compressed to a given size.

codings do not have comparable reconstruction losses unless the compression is very minor. Triangular Dropout, trained only at size 32, allows layer reduction to very small widths with performance only slightly worse than individual autoencoders trained at those sizes. Triangular Dropout thus approximates the behavior of any one of these autoencoders, while only requiring a single training run.

Figure 3 provides visualizations of the reconstructions of each of these cases, for selected latent widths. Standard autoencoders trained at increasing sizes (Figure 3, A) show increasing reconstruction fidelity with increased latent size. When features from the autoencoder of size 32 are removed (Figure 3, B), the encoding quality is seen to quickly degrade. PCAAE (Figure 3, C and D) has similar reconstruction quality as the standard autoencoders when sufficient features are present, but cannot handle more significant compression. Triangular Dropout (Figure 3, E) has similar reconstruction quality to the individual autoencoders, even when the size of its latent variable is significantly reduced. A user of this model could select a latent size between roughly 6 and 32 and expect it to yield recognizable digits.

We also explored the use of autoencoders on the CelebA dataset, finding similar effects. CelebA training details and reconstructions with various architectures are given in the Appendix.

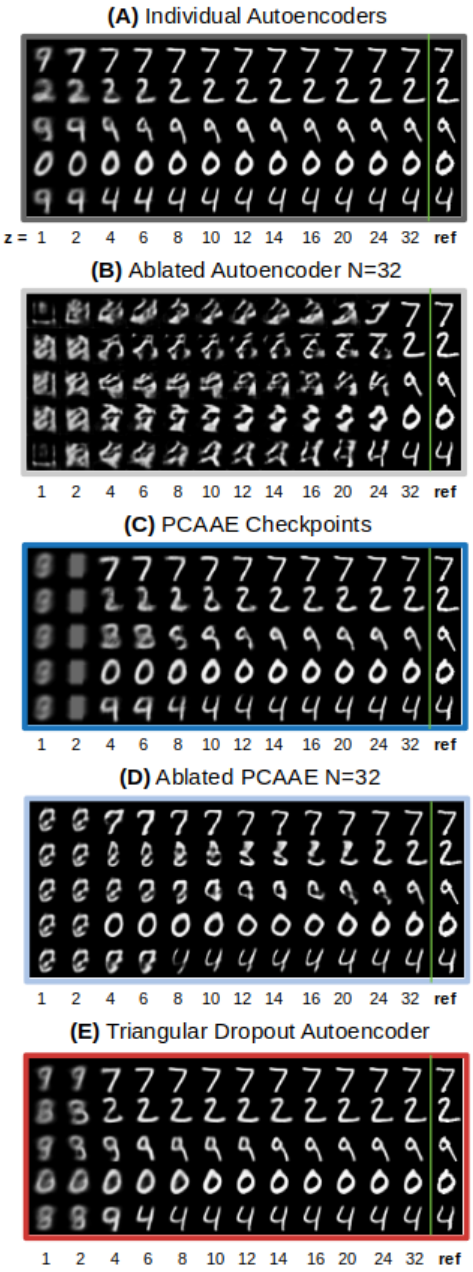

Figure 3. Qualitative analysis of reconstructions. Colored borders correspond to line color in Figure 2. **A:** Reconstructions from individual autoencoders at selected sizes, with original inputs on the right. **B:** Autoencoder trained with width 32 and latent features compressed to desired size. **C:** Reconstructions from PCAAE checkpoint models produced by progressively training latent channels towards a full width of 32. **D:** PCAAE trained with width 32 then compressed to desired size. **E:** Reconstructions from Triangular Dropout trained once at size 32 and with then compressed to given sizes.

4. Parameter Reduction in Large MLPs

The results of our MNIST autoencoding experiments demonstrate the primary property of a Triangular Dropout layer: it can be compressed after training while still providing meaningful output. Hence Triangular Dropout can be considered a pruning technique in which the model is trained in preparation for structured compression. To motivate this view, we retrain the fully connected classification portion of VGG19 on ImageNet, using Triangular Dropout on the two hidden layers of size 4096. We then investigate the fidelity of the model when run at reduced widths. We concentrate on this portion of the model because it is fully-connected and contains the majority of VGG's parameters (roughly 123 million out of 143 million in total). More specific training details are available in the appendix.

When retrained with Triangular Dropout, VGG19 does not achieve quite the top-1 validation accuracy of the original (69.7 percent to 72.4 percent). This is considered a significant difference in a heavily studied benchmark like ImageNet, but may be acceptable in a real-world scenario when weighed with the reduction in deployment compute that Triangular Dropout imparts.

The main takeaway is that Triangular Dropout allows the classification portion of VGG19 to be reduced in width after training, resulting in a dense subnetwork that trades accuracy for narrowness. This a known tradeoff from prior work in network pruning (see pruning results for VGG16 on ImageNet in (Blalock et al., 2020), Figure 3). However, in contrast to prior work, Triangular Dropout enables compression without introducing sparsity or requiring fine-tuning or iterative training.

Our results using full-batch Triangular Dropout (batch size 4096) are summarized in Table 1 and graphically in Figure 4. Figure 4 also shows the effects of using batch sizes that are fractions of the full batches and may be more practical. In this setting, the triangular mask takes the form of a block matrix with blocks of size 1 row by d columns, where $d = 4096/B$. Full-size batches are seen to be the most stable and perhaps most performant, but all batch sizes display similar trends in accuracy versus width.

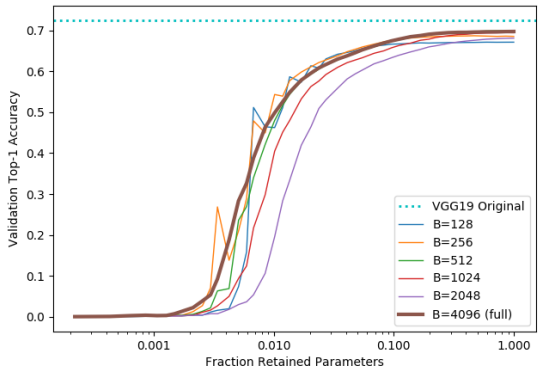

Figure 4. Validation accuracy of VGG19 trained with Triangular Dropout as the fully-connected layers are adjusted in width. Several batch sizes are tested, reduced from the full-sized square batch (bold). The reported validation accuracy of the original VGG19 is show in teal.

Ablated Width	Accuracy	Classif. Params	Param. Reduction
VGG19 Reference	72.4%	123,642,856	N/A
4096 (full)	69.7%	123,642,856	0.0%
2048	69.6%	57,627,624	53.4%
1024	69.2%	27,765,736	77.5%
512	67.8%	13,621,224	89.0%
256	65.3%	6,745,576	94.5%
128	61.7%	3,356,904	97.3%
64	54.9%	1,674,856	98.6%
32	38.8%	836,904	99.3%

Table 1. Accuracy and parameter numbers for VGG19 trained with Triangular Dropout, with various layer widths after training. The first row shows reference values for the original VGG19 architecture.

5. Discussion and Future Directions

Throughout our experimentation, we found that in many cases learned networks meaningfully use only a small fraction of their allocated capacity, as evidenced by their maintained performance after pruning. Triangular Dropout allows this characteristic to be leveraged, enabling structured, often dramatic compression without significant degradation in performance or any required retraining. Such a capability has many practical benefits, particularly for the deployment of large AI models in compute-limited scenarios (e.g. embedded and/or mobile applications). Future studies could further the impact of the technique through application to other types of network layers (for instance covolutional or self-attention).

ACKNOWLEDGMENTS

This work relates to Department of Navy award N00014-20-1-2239 issued by the Office of Naval Research. The United States Government has a royalty-free license throughout the world in all copyrightable material contained herein. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Office of Naval Research.

References

Amodei, D. and Hernandez, D. Ai and compute. 2019. URL https://openai.com/blog/ai-and-compute/.

Anwar, S., Hwang, K., and Sung, W. Structured pruning of deep convolutional neural networks. *CoRR*, abs/1512.08571, 2015. URL http://arxiv.org/abs/1512.08571.

Blalock, D. W., Ortiz, J. J. G., Frankle, J., and Guttag, J. V. What is the state of neural network pruning? *CoRR*, abs/2003.03033, 2020. URL https://arxiv.org/abs/2003.03033.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Training pruned neural networks. *CoRR*, abs/1803.03635, 2018. URL http://arxiv.org/abs/1803.03635.

He, Y., Zhang, X., and Sun, J. Channel pruning for accelerating very deep neural networks. *CoRR*, abs/1707.06168, 2017. URL http://arxiv.org/abs/1707.06168.

Higgins, I., Matthey, L., Pal, A., Burgess, C. P., Glorot, X., Botvinick, M. M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., and Keutzer, K. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size, 2016.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models, 2020.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes, 2014.

Li, H., Kadav, A., Durdanovic, I., Samet, H., and Graf, H. P. Pruning filters for efficient convnets. *CoRR*, abs/1608.08710, 2016. URL http://arxiv.org/abs/1608.08710.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Luo, J.-H., Zhang, H., Zhou, H.-Y., Xie, C.-W., Wu, J., and Lin, W. Thinet: Pruning cnn filters for a thinner net. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(10):2525–2538, 2019. doi: 10.1109/TPAMI.2018.2858232.

McCandlish, S., Kaplan, J., Amodei, D., and Team, O. D. An empirical model of large-batch training, 2018.

Molchanov, P., Tyree, S., Karras, T., Aila, T., and Kautz, J. Pruning convolutional neural networks for resource efficient transfer learning. *CoRR*, abs/1611.06440, 2016. URL http://arxiv.org/abs/1611.06440.

Pham, C.-H., Ladjal, S., and Newson, A. Pcaae: Principal component analysis autoencoder for organising the latent space of generative networks, 2020.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. *Learning Internal Representations by Error Propagation*, pp. 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 026268053X.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

Tan, M. and Le, Q. V. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

A. Appendix

A.1. Autoencoder Experiments

In this section we detail architectures, training routines, and additional experiments for Triangular Dropout applied to MNIST autoencoders.

A.1.1. MNIST Architecture and Training

For all MNIST cases the network architectures were as follows, given some encoder output size z:

Type	Activation	Parameters
Conv	Relu	32 Ch, 9x9 Kern, Stride 1
Conv	Relu	32 Ch, 7x7 Kern, Stride 1
Conv	Relu	32 Ch, 3x3 Kern, Stride 1
Dense	linear	4608 Inputs, z Outputs
Dense	Relu	z Inputs, 4608 Outputs
Trans Conv	Relu	32 Ch, 3x3 Kern, Stride 1
Trans Conv	Relu	32 Ch, 7x7 Kern, Stride 1
Trans Conv	Relu	32 Ch, 9x9 Kern, Stride 1
Conv	sigmoid	1 Ch, 3x3 Kern, Stride 1, Padd 1

Models were trained with batch size 1024 using the Adam optimizer, with a learning rate of 0.005 that decreases by a factor of 10 if the average epoch loss does not decrease by at least 2% percent over the last 15 epochs. This was repeated for 5 decreases of the learning rate. MNIST images were normalized to values between 0 and 1, and binary cross entropy was used as the loss for reconstruction.

A.1.2. Available Width versus Compression

We additionally performed experiments to test if the available encoding size affected the learned representations in Triangular Dropout. That is, we wanted to determine if the model utilized or ignored extra width when available. In Figure 5, we show a plot of reconstruction accuracy from MNIST autoencoders trained using Triangular Dropout with maximum encoding widths of 4, 8, 16, 32, 64, 128, 256, and 512. These models were then compressed back to encoding widths of 1 through 32 (or widest available).

We found that there is a clear pattern of models with more available width being less compressed, but also that the overall difference in performance diminishes as available width increases.

A.1.3. CelebA Architecture and Training

We used the following architecture for our CelebA experiments. The VAE results described below use two parallel encoding layers for the mean and variance, respectively. Models were trained for 100 epochs with batch size 128. We used the Adam optimizer with learning rate 0.001, decreasing by a factor of 10 every 30 epochs.

Type	Activation	Parameters
Conv	Relu	128 Ch, 5x5 Kern, Stride 2
Batch Norm		(Optional)
Conv	Relu	128 Ch, 5x5 Kern, Stride 2
Batch Norm		(Optional)
Conv	Relu	64 Ch, 3x3 Kern, Stride 1
Dense	Relu	7744 Inputs, 2048 Outputs
Dense	linear	2048 Inputs, z Outputs
Dense	Relu	z Inputs, 2048 Outputs
Dense	Relu	2048 Inputs, 7744 Outputs
Trans Conv	Relu	128 Ch, 3x3 Kern, Stride 1
Batch Norm		(Optional)
Trans Conv	Relu	128 Ch, 5x5 Kern, Stride 2, Out Pad 1
Batch Norm		(Optional)
Trans Conv	Relu	128 Ch, 5x5 Kern, Stride 2, Out Pad 1
Conv	Relu	128 Ch, 3x3 Kern, Stride 1, Pad 1
Batch Norm		(Optional)
Conv	Relu	128 Ch, 3x3 Kern, Stride 1, Pad 1
Batch Norm		(Optional)
Conv	Relu	128 Ch, 3x3 Kern, Stride 1, Pad 1
Conv	sigmoid	3 Ch, 3x3 Kern, Stride 1, Pad 1

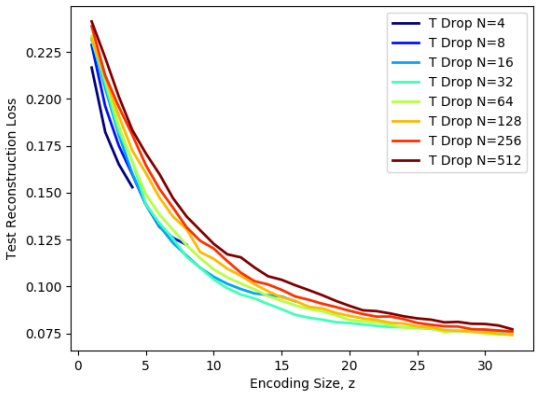

Figure 5. MNIST reconstruction loss for widths 1 through 32 of autoencoders trained with Triangular Dropout and a variety of available latent widths.

A.1.4. CelebA Reconstructions with Alternative Architectures

Please see reconstruction examples on the following page.

Figure 6. Reconstructions of previously unseen CelebA images on various architectures using Triangular Dropout at the encoding layer. In each case (A, B, and C), only a single model is trained, and the encoding width is pruned down to the given size.

A.2. VGG19 Training Details

Our version of VGG19 with Triangular Dropout was created by retraining the classification portion of VGG19. The convolutional layers were frozen (not retrained). The convolutional layers output 25088 features for a given input, which then pass through two hidden layers of size 4096, and finally output 1000 features to be interpreted as classification scores for the image. Our model simply replaced the two hidden layers with Triangular Dropout layers of the same size.

We trained our model for 100 epochs using a batch size of 4096 in order to apply a full triangular mask to the batch. We used stochastic gradient descent (SGD) with an initial learning rate of 0.01, reducing the learning rate by a factor of 10 every 30 epochs.

It is unclear if our model would have behaved differently if the entire architecture were retrained, which may have enabled Triangular Dropout to be applied to the feature descriptors of length 25088. This is of interest because VGG19's 25088 features are often used as a starting point for other image processing tasks.