A Appendix

A.1 Proof of Theorem 1

Proof. Note that the partition $t_i = i$ for all $i \in \mathbb{N}_{\leq N}$ implicitly sets $M = N$ and is equivalent to stating that each transmission batch size must be exactly one. Assume by contradiction that there exists a partition T_* which minimizes the expected prequential code length and has at least one transmission batch with size greater than one. Without loss of generality, let the transmission batch from t_j^* to t_{j+1}^* have size greater than one and note that the expected prequential code length of T_* is

$$-\left(t_{j+1}^* - t_j^*\right) \log_2 \phi\left(t_j^*\right) - \sum_{i=0, i \neq j}^{M-1} \left(t_{i+1}^* - t_i^*\right) \log_2 \phi(t_i^*)$$

However, say that we divide the batch from t_j^* to t_{j+1}^* into smaller transmission batches of size one. Because the function $\phi(d)$ is increasing, we have that

$$-\left(t_{j+1}^* - t_j^*\right) \log_2 \phi\left(t_j^*\right) > -\sum_{k=t_j^*}^{t_{j+1}^*-1} \log_2 \phi(k)$$

Therefore, if we divide the batch from t_j^* to t_{j+1}^* into batches of size one and keep the rest of the intervals from T_* the same, we have a smaller expected prequential code length which is a contradiction. □

A.2 Proof of Theorem 2

In Theorem 2 we note that the optimization problems in Equations 1 and 2 have the same optimal objective function value in the case where $-\log_2 \phi(d)$ is equal to the decreasing convex hull of the Monte Carlo estimate demarcated by the supporting hyperplanes represented by the linear equations formed by A and $\mathbf{w}$. First we briefly make a technical point. Note that in our Monte Carlo estimate it is possible that the convex hull starts increasing before we have reached our final d value of N. In this case since we are only keeping decreasing lines, the lower bound is simply formed by the straight horizontal line h_{min}. We will consider this line as a supporting hyperplane (adding it to the the linear equations formed by A and $\mathbf{w}$) and now refer to the combination of the convex hull and this horizontal line as the non-increasing convex hull. In the following proof we shall assume that $-\log_2 \phi(d)$ is equal to this non-increasing convex hull.

Proof. First, revert the optimization problem in Equation 2 from geometric form by making the substitution $e^{u_i} := b_i$ and $e^{v_i} := c_i$ for all $i = 1, 2, \dots M$. The optimization problem now has the form:

$$\begin{aligned} \min_{\mathbf{b}, \mathbf{c}} \quad & \sum_{i=1}^{M} b_i \cdot c_i \\ \text{s.t.} \quad & A \begin{bmatrix} \sum_{j=1}^{i} b_j \\ c_{i+1} \end{bmatrix} \geq \mathbf{w} \quad \forall i \in \mathbb{N}_{\leq M-1} \\ & \sum_{i=1}^{M} b_i \geq N, \quad h_{\text{min}} \leq c_i \leq h_{\text{max}} \end{aligned} \tag{$\bigstar$}$$

Note that b_i and c_i represent the size and cost rate, respectively, of transmission batch i. It is helpful to use Figure 2 in the main text to visualize this optimization problem. In this figure each pair $\left(\sum_{j=1}^{i} b_j, c_{i+1}\right)$ is represented as a circle, which is allowed to move around in the green zone, demarcated by the supporting hyperplanes and the upper/lower bounds on cost rate.

Let $\mathbf{z}^*$ be any member of G_N^{M-1} that gives the minimal value to the optimization problem in Equation 1:

$$\min_{\mathbf{z} \in G_N^{M-1}} \quad -\sum_{i=0}^{M-1} \left(z_{i+1} - z_i\right) \log_2 \phi(z_i)$$

Substitute $b^*_{i+1} \coloneqq z^*_{i+1} - z^*_i$ for all $i = 0, 1, \ldots, M-1$. Because $z_0 \coloneqq 0$ and, following convention, we define $b^*_0 \coloneqq 0$, we have that $z^*_i = \sum_{j=0}^{i} b^*_j$ for all $i = 0, 1, \ldots M$. Therefore the objective function value for any optimal $\mathbf{z}^*$ can be written as

$$-\sum_{i=0}^{M-1} b^*_{i+1} \log_2 \phi\left(\sum_{j=0}^{i} b^*_j\right)$$

for the corresponding $\mathbf{b}^*$. Now substitute $c^*_{i+1} \coloneqq -\log_2 \phi\left(\sum_{j=0}^{i} b^*_j\right)$ for all $i = 0, 1, \ldots, M-1$ and further rewrite the optimal objective function value as

$$\sum_{i=1}^{M} b^*_i \cdot c^*_i$$

Recall that the non-increasing convex hull, given by the supporting hyperplanes represented by the linear equations formed from A and $\mathbf{w}$, is equivalent to $-\log_2 \phi(d)$. Therefore, $A \begin{bmatrix} \sum_{j=1}^{i} b^*_j \\ c^*_{i+1} \end{bmatrix} = \mathbf{w}$ for all $i \in \mathbb{N}_{\leq M-1}$. It is trivial to note that $h_{\min} \leq c^*_i \leq h_{\max}$ and, since $z_M \coloneqq N$, that $\sum_{i=1}^{M} b^*_i = N$. Thus, the optimal solutions to Equation 1 are valid points in the above optimization problem (Equation ★) and evaluate to the same objective function value as the corresponding points in Equation ★. We will now show that, despite the larger search space, the optimal solutions to Equation ★ are the same as the corresponding $\mathbf{b}^*$ and $\mathbf{c}^*$ for any optimal solution to Equation 1.

First, by contradiction say that we have an optimal solution $\left(\tilde{\mathbf{b}}, \tilde{\mathbf{c}}\right)$ to Equation ★ where every point $\left(\tilde{b_i}, \tilde{c_i}\right)$ does not lie exactly on the supporting hyperplanes. For every $\tilde{b}_i$, the corresponding $\tilde{c}_i$ can be decreased until it touches the supporting hyperplanes. Decreasing these values can clearly only lower the objective function value. Therefore, by contradiction, any optimal solution to Equation ★ must have every point $\left(\tilde{b_i}, \tilde{c_i}\right)$ lie exactly on the supporting hyperplanes (i.e. $A \begin{bmatrix} \sum_{j=1}^{i} \tilde{b}_j \\ \tilde{c}_{i+1} \end{bmatrix} = \mathbf{w}$ for all $i \in \mathbb{N}_{\leq M-1}$).

Next, by contradiction say that we have an optimal solution $\left(\tilde{\mathbf{b}}, \tilde{\mathbf{c}}\right)$ to Equation ★ where $\sum_{i=1}^{M} \tilde{b}_i > N$. Let k be the first index such that $\sum_{i=1}^{k} \tilde{b}_i >= N$. For $j = k, k+1, \ldots M$ set $\tilde{b}_j$ such that $\sum_{i=1}^{j} \tilde{b}_i$ progressively increases from greater than $\sum_{i=1}^{k-1} \tilde{b}_i$ to exactly N when $j = M$. Set the corresponding $\tilde{c}_i$ to be on the supporting hyperplanes and note that because these occur at no higher a rate than $\tilde{c}_{k-1}$ we have decreased the code length. Therefore, by contradiction, any optimal solution to Equation ★ must have $\sum_{i=1}^{M} \tilde{b}_i = N$.

Finally, now that we know any optimal solution $\left(\tilde{\mathbf{b}}, \tilde{\mathbf{c}}\right)$ to Equation ★ must have every point $\left(\tilde{b_i}, \tilde{c_i}\right)$ lie exactly on the supporting hyperplanes and have $\sum_{i=1}^{M} \tilde{b}_i = N$, it is simple to show that this is also an optimal solution to Equation 1. By contradiction, say that we have an optimal solution $\left(\tilde{\mathbf{b}}, \tilde{\mathbf{c}}\right)$ but it is not equal to the corresponding $\mathbf{b}^*$ and $\mathbf{c}^*$ for any optimal solution to Equation 1. Because $\left(\tilde{\mathbf{b}}, \tilde{\mathbf{c}}\right)$ lies on the supporting hyperplanes and has $\sum_{i=1}^{M} \tilde{b}_i = N$, we can find $\tilde{\mathbf{z}} \in G_N^{M-1}$ such that $\tilde{b}_{i+1} = \tilde{z}_{i+1} - \tilde{z}_i$ and $\tilde{c}_{i+1} = -\log_2 \phi(\tilde{z}_i)$. However, because $\tilde{\mathbf{z}}$ is not optimal, there exists a $\mathbf{z}^*$ with a lower objective value in Equation 1 that has corresponding $\mathbf{b}^*$ and $\mathbf{c}^*$. However, because the objective functions for Equations 1 and ★ give equal values for $\mathbf{z}^*$ and $(\mathbf{b}^*, \mathbf{c}^*)$ respectively, we have a contradiction. Therefore the optimal value of Equation 1 and ★ (and thus Equation 2) are equal, proving Theorem 2.

□

A.3 Generalization Function Generation Details

In this section we will describe the generation process for the eight generalization functions seen in Figure 1 that were subsequently used throughout the paper.

A.3.1 Datasets

The following datasets were used: MNIST and CIFAR-10 [1, 2]. These datasets were downloaded from the Torchvision dataset repositories. For more information please see `https://pytorch.org/vision/0.8/datasets.html`. Images were normalized to have a mean of 0.5 and standard deviation of 0.5. MNIST images were resized to 32×32 pixels. For the MLP architecture, CIFAR-10 images were transformed to grayscale. Train and test splits were done according to the standard settings on the torchvision API. The resulting train/test sizes were $60000/10000$ for MNIST and $50000/10000$ for CIFAR-10. Note these are the available train/test sizes - for the creation of the generalization function the actual amount used varies (see Appendix A.3.4). Both datasets are publicly available at their respective authors' websites with no reference to specific licensing.

A.3.2 Networks

Four networks were used: an MLP, a simple convolutional neural network (SCN), a compact VGG like network (VGG-CO) [3], and ResNet18 (RN18) [4].The MLP had two hidden fully connected layers of size 256 and 128 respectively. ReLU activation was used after the first two layers. The SCN used two convolutional layers with output channel sizes of 32 and 16. Both layers had kernel sizes of four and were followed by max pooling layers with a kernel size of two each. There was then a hidden fully connected layer of size 64 activated by a ReLU. VGG-CO is based on VGG style architectures but is compact and uses three convolutional layers, two max pooling layers, ReLU activations, and a classification sequence of ReLU activated fully connected layers with dropout. Both VGG-CO and RN18 were adapted from their PyTorch implementations. For full details on all the networks used please see the accompanying code files `gen_func_creator.py` and `gen_func_resnet18.py`.

A.3.3 Software/Libraries

The code for the generalization functions was written in Python [5] and made use of NumPy [6] and Pytorch [7]. The code for the experiments with time constraints and DC solves additionally used SciPy [8], MOSEK [9], and CVXPY [10, 11] with the DCCP plugin [12]. Figures 1 and 3 were generated using Matplotlib [13].

A.3.4 Generalization Function Monte Carlo Simulation

The following d values (size of training set) were used: from 64 up to 1024 at intervals of 64, then up to 4096 at intervals of 256, then up to 16384 at intervals of 2048, then up to the training size at intervals of 16384. The full training size was also included. To get better averages multiple trials were run for each d value. The first 15 d values were run 128 times, the next 12 were run 16 times and the final 10 were run 12 times.

To get the values for each trial for a given d value, we began by randomly shuffling the training data and then selecting the first d data points for training. After training was completed the generalization function values were computed by running the trained model on 1024 randomly selected data points from the test set. To train the architectures we used backpropagation, cross entropy loss, mini-batch sizes of 64, and the Adam optimizer [14] with default learning rate .001, all for 10 epochs through the data made available. Computation was run on an Intel i7-4790k and an Nvidia Geforce GTX 970 and took, very roughly, half a day to simulate each generalization function. The code to generate the generalization functions (which includes further supporting information such as external library dependencies) can be found in the file `gen_func_creator.py`. The values of the Monte Carlo trials for the networks/datasets used in our paper can be found in the `/gen_funcs/` folder.

A.3.5 Error Bounds on Generalization Functions

The error bars for the generalization functions are visualized using two standard deviations in either direction in Appendix Figures 1 and 2. There is no need to generate error bars for the coding cost functions as these can easily be deduced.

A.4 Time-Constrained Online Coding Experiment Details

The difference of convex programming problem in Equation 2 (with the addition of a positive constraint on u_i for $i = 1, 2, \ldots, M$) was solved using the software of Shen et al. (DCCP) [12]

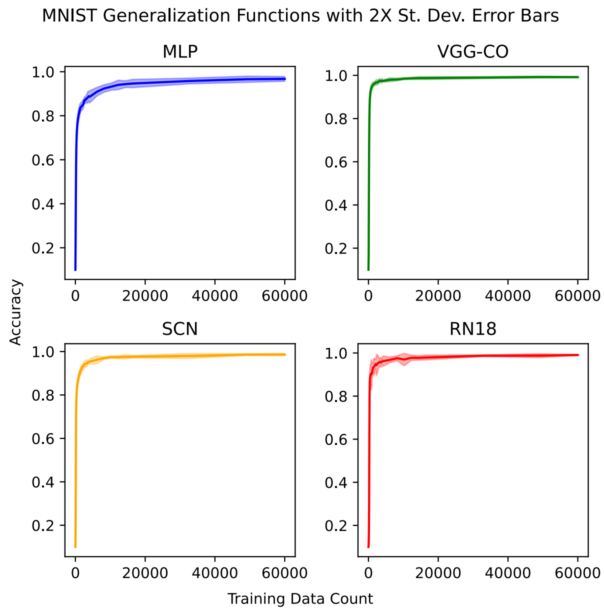

Figure 1: Generalization functions for MNIST dataset with two standard deviation error zones in either direction.

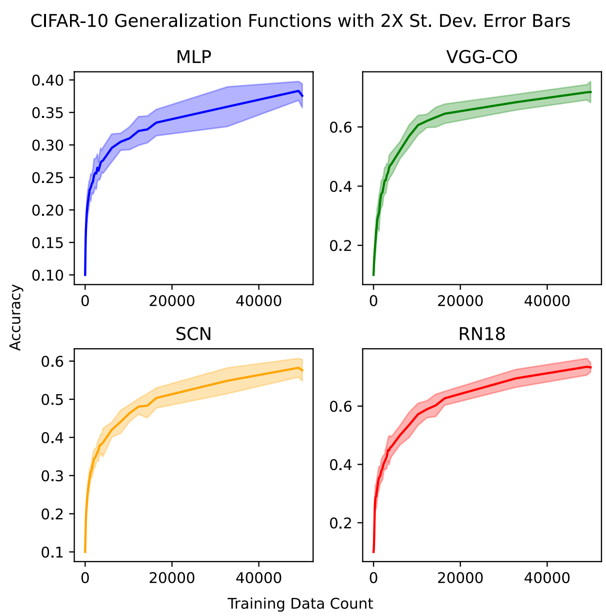

Figure 2: Generalization functions for CIFAR-10 dataset with two standard deviation error zones in either direction.

which was built on top of the popular convex optimization software CVXPY [10, 11]. Although the addition of the positive constraint means the earlier equivalent solution argument (Theorem 2) no longer holds (as there must be spacing in between the points) in practice this constraint helps the optimizer avoid looking for inconsequentially small values. The MOSEK solver [9] was used and DCCP maximum iterations were set to 100. The u_i values were initialized with zero except for the last value which was set equal to $\ln{(N - M + 1)}$. The v_i values were all initialized with $\ln h_{\text{max}}$.

The optimization problem must check the linear inequalities defined by A and $\mathbf{w}$ a total of $M-1$ times in addition to the constraints on total transmission batch size and upper/lower cost bounds. Thus we have $\mathcal{O}(M)$ variables and $\mathcal{O}(sM)$ constraints. The DCCP algorithm essentially solves a series of convex subproblems which, as mentioned, we capped at a maximum iteration count of 100. Therefore, the worst case time complexity cost of using DCCP to solve our optimization problem is the maximum iteration count multiplied by the worst case time complexity cost of our small convex sub-problems. The complexity of solving convex programming is nuanced and beyond the scope of this paper and we suggest [15] as a starting reference. However our experiments solves were fast as a transmission batch count of $M = 48$ took approximately 15 minutes for each generalization function tested on an Intel i7-4790k CPU.

To generate the data for Figure 3, DC solves were run for each of the 8 generalization function at $M = 48$ for time budgets ranging from 65000 to 125000 in increments of 10000; this gave a total of 56 solves. Time constraints were added as defined in Section 4.2 where $\tau = 1$ and λ was set equal to the various time budgets. Note that τ was set to 1 (and was thus not defined by any actual epoch account) in order to standardize the values of λ . The code length values for each of the generalization functions were based on the unrounded batch sizes and were divided by the respective results at 125000 to present a percentage increase over this "baseline". The code to solve the DC problems (and information about external library dependencies) is located in the `dc_solver.py` file which also makes use of the `/gen_funcs/` folder.

References

[1] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[2] Alex Krizhevsky, Geoffrey Hinton, et al. Learning Multiple Layers of Features from Tiny Images. Technical report, 2009.

[3] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2014.

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

[5] Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009. ISBN 1441412697.

[6] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

[7] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[8] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul

van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[9] MOSEK ApS. *MOSEK (Python Interface, Version 9.2.35)*, 2021.

[10] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[11] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[12] Xinyue Shen, Steven Diamond, Yuantao Gu, and Stephen Boyd. Disciplined Convex-Concave Programming. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 1009–1014. IEEE, 2016.

[13] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

[14] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. 2015.

[15] Stephen P Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.