BEYOND L1 & L2: BENCHMARKING ADVANCED REGULARIZATION TECHNIQUES IN NEURAL NETWORKS

Aksel Kretsinger-Walters
Columbia University
adk2164@columbia.edu

Mateo Juliani
Columbia University
msj2164@columbia.edu

December 24, 2025

1 Abstract

Regularization is a common tool in machine learning (ML) to prevent overfitting and improve generalization. From a Bayesian perspective, regularizers correspond to priors, which provide a way to express prior beliefs about model parameters. Yet commonly used techniques such as L1/L2 regularization or dropout correspond to only a narrow set of Bayesian priors. In this paper, we investigate the efficacy of a broader family of priors including 1) Smoothness prior - a prior that penalize the neural net for having large inflection points, 2) Entropic prior - priors that favor higher-entropy predictions, and 3) Automatic relevance determination (ARD) prior which encourages sparse parameters. We test the performance of these priors across a variety of neural network architectures (MLPs, CNNs, BiLSTMs, Transformers, and VAEs) and tasks. Our results show the Entropic prior can outperform traditional L1 / L2 regularization while having a similar computation cost. The ARD and Smoothness priors show less signs of outperformance and are more computationally expensive.

2 Introduction

The practical argument for regularization is that adding a penalty term proportionate to the magnitude of model parameter weights discourages extreme model complexity and curtails the risk of overfitting to training data. In support of this view is the bias-variance tradeoff, which states that as model complexity increases, bias decreases but variance increases. The empirical success of regularization techniques in modern deep learning, coupled with the intuitive mental model for the tradeoff made when scaling regularization, has cemented regularization as a psuedo-standard in machine learning practice.

From a Bayesian perspective, regularization can be interpreted as imposing a prior distribution over model parameters [1]; within this framework, the regularization term corresponds to the negative log-prior probability of the parameters. This Bayesian interpretation provides a principled way to incorporate prior knowledge about the parameters into the learning process, ostesibly leading to improved generalization performance.

The most commonly used regularization priors, L1 and L2, correspond to the Laplace and the Gaussian distribution. The computational efficiency, intuitive justification, and empirical success of these priors have made them the de facto priors for deep learning models [2].

In our work, we searched for alternative priors that could rival (or outperform) the performance of L1/L2 regularizors, while offering different inductive biases. The majority of these priors were proposed in the 1990s and early 2000s, but were ahead of their times: the computational resources and large datasets required to train deep learning models were not yet widely available. We re-examined these priors on a wide variety of popular deep learning architectures and datasets, benchmarking the performance of each prior to the standard L1/L2 regularization.

3 Methods and Experiment Design

3.1 Models and Inference

In this paper, we use neural networks, denoted as $\Omega(x;\theta)$, for a variety of data tasks. In this section, we describe the general model format and inference methods used to estimate the posterior distribution.

The generative process of the parameters θ of the neural network is as follows:

$$\begin{aligned} \theta_{\text{in},k} &\sim \mathcal{P}_p(\lambda_{\text{in}}) && k = 1 \dots \text{K} \\ \theta_{\ell k} &\sim \mathcal{P}_{\text{K}}(\lambda_{\text{mid}}) && \ell = 1 \dots \text{L} - 1 \\ \theta_{\text{out},j} &\sim \mathcal{P}_{\text{K}}(\lambda_{\text{out}}) && j = 1 \dots d. \end{aligned}$$

where p is the input dimension, d is the output dimension, $\mathcal{P}$ is the probability distribution from which each θ is generated from, L is the number of layers in the network, and K is the number of parameters per layer. For any datapoint i, the response is drawn by $y_i|x_i,\theta \sim expfam(\Omega(x_i;\theta))$.

Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_1^n$, we optimize the following log joint probability:

$$\mathcal{L}(\theta) = \log p(\theta) + \sum_{i=1}^{n} \Omega(x_i;\theta) \cdot y_i - a(\Omega(x_i;\theta)) \tag{1}$$

to get a MAP estimate of the network parameters θ using standard neural network optimizers (Adam [3]) and automatic differentiation libraries (PyTorch [4]).

3.2 Priors

In Equation (1), the $logp(\theta)$ refers to the prior on θ. In the section below, we explore different methods to parametrize $p(\theta)$ and explain intuitively how the prior may affect the neural network function. For all priors below, λ refers to the scaling factor of the prior, n is to total number of data points, $\mathbb{H}$ denotes the entropy of a random variable.

Gaussian Prior - The Gaussian prior assumes each weight θ_i of the neural net is sampled from a normal distribution - $\theta_i \sim \mathcal{N}(0, \sigma^2)$, where σ^2 is a fixed value for all weights, typically 1. The

Gaussian prior goes by several different names in machine learning and statistic literature (ridge regularization, L2 regularization, weigh decay) and is one of the most widely used regularization techniques in machine learning [5]. In this paper we refer to it as L2 regularization. L2 regularization can be implemented with the following equation: $\log p(\theta) \propto \lambda \sum_{i=1}^{n} \theta_i^2$, and is used in this paper as a baseline prior, in addition to L1 regularization (see below), and using no prior.

Laplace Prior - The Laplace prior (referred to as L1 regularization) assumes the weights of the neural net are sampled from a Laplace distribution with mean 0 and a fixed scale. This prior typically encourages sparse weights for the neural network [6], and can be implemented with the following equation: $\log p(\theta) \propto \lambda \sum_{i=1}^{n} |\theta_i|$. L1 regularization will also be used a baseline.

Smoothness Prior - The smoothness prior, as proposed by [7, 8], will penalize neural networks with large "curvatures" or changes in magnitudes. Consequently, the neural net will become a smoother function. The smoothness prior can be estimated by the following equation

$$\log p(\theta) \propto -\frac{\lambda}{2n} \sum_{i=1}^{n} d(x_i) \quad \text{where} \quad d(x) = \sum_{k} \left(\frac{\partial^2 \Omega(x;\theta)}{\partial x^{(k)} \partial x^{(k)}} \right)^2 \tag{2}$$

To implement Equation (2), we use Hessian Vector Products [9].

Entropic Prior - The Entropic prior, as proposed in [8], penalizes models for having a low entropy output. In other words, it favors models that distribute the probability mass of its predictions more evenly. Penalizing a model for low entropy outputs has proven successful in certain areas of deep learning (such as reinforcement learning [10] and variational autoencoders [11]), however this paper aims to test the efficacy of the prior in more general neural networks tasks. The Entropic prior is estimated by Equation (3):

$$\log p(\theta) \propto \frac{\lambda}{n} \sum_{i=1}^{n} \mathbb{H}(Y \mid x_i, \theta) \tag{3}$$

Automatic Relevance Determination (ARD) Prior - The ARD prior is similar to the Gaussian prior in that it assumes the weights of the neural net are sampled from a normal distribution with mean zero, but instead of a fixed variance term for all weights, the ARD prior learns an individual variance for each weight. The hierarchical model for each weight θ_i is as follows:

$$\alpha_i \sim \text{Gamma}(a, b)$$
$$\theta_i \sim \mathcal{N}(0, \alpha_i^{-1})$$

where a, b are hyper-parameters (set to $a, b = 1$ for this paper). The MAP estimate of the ARD is shown by Equation (4) (see Section 7.0.1 for full derivation).

$$\log p(\theta) \propto \lambda \sum_{i=1}^{n} \log p(\theta_i) \quad where \quad \log p(\theta_i) \propto -\frac{1}{2}\big[\log(\alpha_i) + \alpha_i \theta_i^2\big] - (a-1)\log(\alpha_i) + b\alpha_i \tag{4}$$

The ARD prior has a similar form to the Gaussian prior. However, each θ_i^2 now has an α_i term associated with it, which allows the optimizer to determine whether or not the weight should be regularized. If the weight θ_i is important, then the value of α_i may be small and therefore have a high variance (and hence more likely to have a larger value). If the weight is deemed irrelevant, the value of α_i will be high, and thus the weight is pushed to zero to minimize the $\log p(\theta)$.

3.3 Data and Neural Network Architectures

In the section below, we detail the datasets and corresponding neural network architectures used to test each prior.

California Housing Dataset - The California housing dataset contains 1990 U.S. Census data on California districts, with 8 numerical features such as median income, house age, and location, used to predict median house values [12]. We use a two-layer MLP to predict the target variable, with mean squared error (MSE) as the primary evaluation metric.

Synthetic Categorical Dataset - We use scikit-learn [13] to create a synthetic categorical dataset with 20 inputs to predict 100 categories, with only 10,000 data points. The goal of this dataset is to understand how the different priors perform with a smaller sample size and a larger class size. We use a two-layer MLP to predict the target variable, and accuracy as the primary evaluation metric.

CIFAR Dataset - CIFAR [14] is an image dataset of 60,000 32x32 color images. CIFAR-10 contains 10 classes and 6,000 images per class, while CIFAR-100 contains 100 classes with 600 images per class, making it a much harder classification task given less data per class. We create a ResNet18-esque architecture (with SiLU activation functions instead of ReLU to ensure a non-zero second derivative, and a smaller convolution size) to predict image classes in the CIFAR-10/100 dataset. We use accuracy as the primary evaluation metric.

MNIST and Oxford 102 Flowers Dataset - The Oxford 102 Flowers dataset consists of over 8,000 images of flowers, containing 102 categories [15]. MNIST [16] contains 70,000 28x28 black and white images of handwritten digits. We use both datasets as the input for a variational autoencoder, where the main evaluation metric is the mean squared error of the reconstructed image.

IMDB Review Sentiment Dataset The IMDB review sentiment dataset contains 50,000 movie reviews labeled as positive or negative [17]. We use a bidirectional LSTM architecture to predict the sentiment of the review, with accuracy as the primary evaluation metric.

Beijing Air Quality Dataset - The Beijing air quality dataset contains hourly air quality readings from 12 monitoring stations in Beijing from 2013 to 2017 [18]. The goal is to predict the PM2.5 value given the previous 24 hours of data. We used both a Transformer and a Unidirectional LSTM to predict the PM2.5 value, with r^2 as the primary evaluation metric.

4 Results

Figure 1 contains the best evaluation metric value on the validation set for each data set and prior [1][2]. Each cell's color corresponds to the normalized performance of the prior relative to the other priors

[1] Because of excessive training time / poor convergence, we were not able to finish the IMDB Smoothness prior test

[2] Entropic prior was not tested on regression based tasks since entropy is fixed under the assumption of fixed output variance

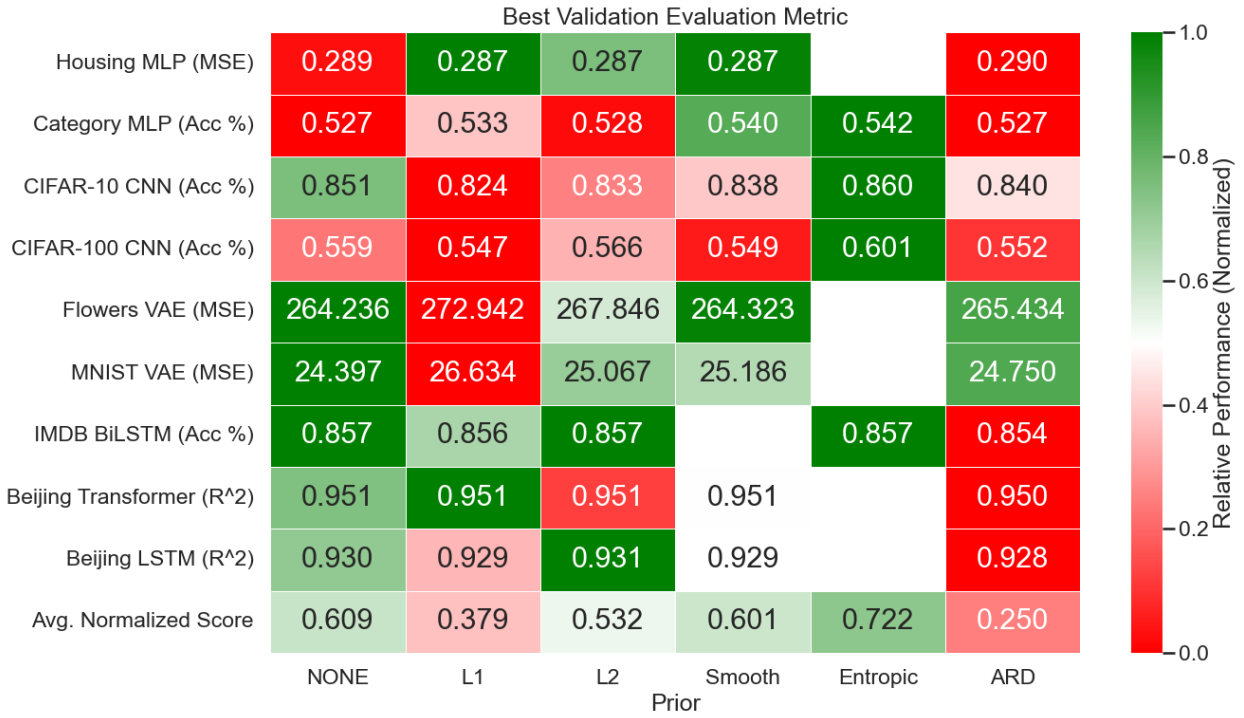
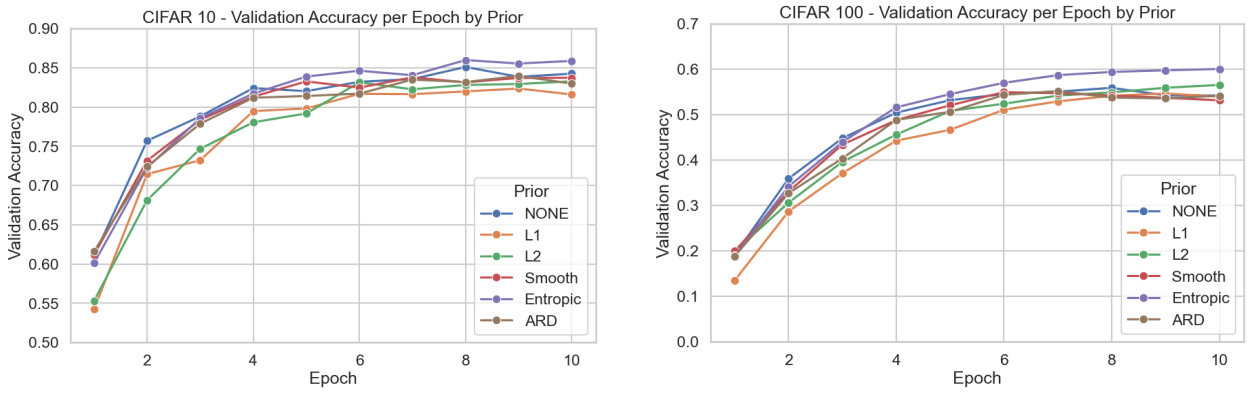


Figure 1: Heatmap of Evaluation Metrics by Dataset and Prior

within the same dataset. Green is the best normalized performance, and red is the worst normalized performance. The final row provides the average normalized score across all datasets for each prior.

Overall, the Entropic Prior achieves the highest average normalized score of 0.722, followed by the Smoothness prior (0.601) and no prior (0.609). On average, the L1 and ARD prior under perform the other priors.



(a) CIFAR 10

(b) CIFAR 100

Figure 2: CIFAR Validation Accuracy per Epoch by Prior

To demonstrate the Entropic's prior effectiveness, Figure 2 provides the validation accuracy per epoch by the different priors for both the CIFAR 10 and CIFAR 100 datasets. The Entropic prior is able to converge to a higher accuracy at a faster rate than the other priors. Further, in Figure 2b,

several of the priors start to show signs of overfitting by achieving peak validation accuracy around epoch 8 and then decreasing. The only priors that do not display this behavior are the L2 and Entropic prior (with the Entropic prior able to achieve a higher accuracy), demonstrating the Entropic's prior ability to avoid overfitting by not being overconfident in its predictions.

4.1 Training Time

Table 1 shows the average training time (seconds) by each prior, with the final row normalized by using no prior. The Entropic prior has the fastest training time (1.06x slower than using no prior - although does not include the LSTM / Transformer architectures which were more computationally intensive to compute priors for in general), followed by the L1 and L2 regularization. The ARD prior is 2.00x slower, while the Smoothness prior is 7.18x slower, highlighting the challenges of implementing it in neural nets.

Table 1: Average Epoch Training Time by Prior (seconds)

Dataset and Model	**No Prior**	**L1**	**L2**	**Smooth**	**Entropic**	**ARD**
Housing MLP	0.56	0.77	0.68	1.26	–	0.97
Cat MLP	0.23	0.31	0.34	0.76	0.26	0.56
CIFAR-10 CNN	45	50	50	355	47	65
CIFAR-100 CNN	47	50	51	361	47	65
Flowers VAE	4	4	4	7	–	4
MNIST VAE	41	43	43	223	–	52
IMDB Review	39	44	43	–	40	64
Beijing LSTM	25	120	124	523	–	104
Beijing Transformer	58	149	185	610	–	172
Avg. Normalized Training Time	1	1.71	1.80	7.18	1.06	2.00

4.2 Robustness Checks

To check the robustness of our results, we perform the following additional checks to understand how different inference techniques and datasets impact our results.

ARD Inference Techniques - We use a MAP estimate to implement the ARD prior. However, other papers [19] have explored implementing ARD priors using variational inference techniques. We compare the validation accuracy of the SiLU Resnet-18 architecture used for the CIFAR datasets using the method proposed by [19] with the PyTorch implementation of [20]. Figure 3 contains the results. We find the MAP estimate out performs the method in [19], however we note the original paper achieves a higher validation accuracy using a larger network, therefore further tests may be needed to understand when the

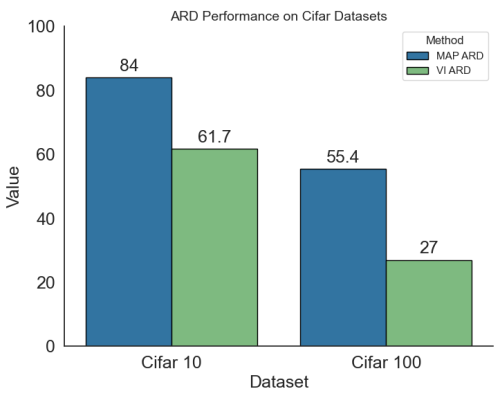


Figure 3: ARD Variational Dropout vs MAP Estimate

ARD prior may improve performance. We also note that in the original paper, using no ARD slightly outperforms using the Variational Inference ARD, which is consistent with our findings in Figure 1.

Class Imbalance - For all tests using the Entropic prior, the datasets have balanced classes. Therefore, the Entropic prior may succeed since all classes apriori are equally likely. To test how the Entropic prior performs in an imbalanced dataset, we create an imbalance version of the CIFAR 100 dataset, such that the class frequencies follow an exponential decay, with the largest class having 10x more samples than the smallest class. Figure 4 shows the results, with the Entropic prior achieving a slightly better F1 scores than the other priors. However, the magnitude of out performance vs the other priors is smaller than when the dataset was balanced.

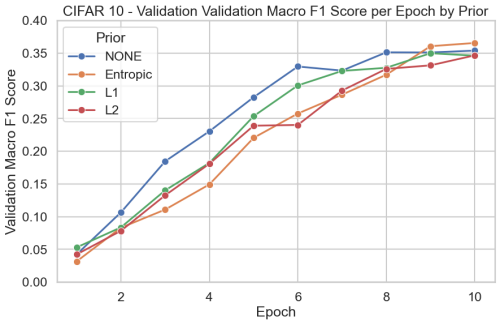


Figure 4: Imbalanced CIFAR 100 Prior Performance

5 Conclusion

In this project, we explored three new priors for neural networks that were previously suggested by the neural network literature, but had little empirical tests to validate their efficacy. We find that the Entropic prior performed the best, outperforming the commonly used L1 and L2 regularizors. Further, we find it does not increase training time significantly, highlighting its potential to be used in modern neural network settings. The ARD and Smoothness prior showed less promise, with less performance improvement compared the L1 and L2 regularizors, and higher computational time. This work is an important step towards understanding how different priors in neural networks impact task performance. We note future directions for this work include 1) testing more challenging datasets that can better differentiate between the priors (in particular for LSTMs / Transformers), 2) exploring different inference techniques beyond MAP, 3) expanding the list of priors tested and 4) testing how task performance changes when combining several priors (i.e L2 and Entropic) together.

6 Acknowledgments

We would like to thank Professor David Blei and our teaching assistants Yuli Slavutsky, Bohan Wu, and Sebastian Salazar for their advice and project inspiration this semester.

References

[1] David Blei. Probabilistic models and machine learning. Unpublished manuscript, 2025.

[2] Felipe Dennis de Resende Oliveira, Eduardo Luiz Ortiz Batista, and Rui Seara. On the compression of neural networks using ℓ_0-norm regularization and weight pruning. *Neural Networks*, 2024. Published online 2024; arXiv preprint: arXiv:2109.05075.

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[4] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach 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, 2019.

[5] Francesco D'Angelo, Maksym Andriushchenko, Aditya Varre, and Nicolas Flammarion. Why do we need weight decay in modern deep learning? In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc.

[6] David P. Wipf, Bhaskar D. Rao, and Srikantan Nagarajan. Latent variable bayesian models for promoting sparsity. *IEEE Transactions on Information Theory*, 57(9):6236–6255, 2011.

[7] C.M. Bishop. Curvature-driven smoothing: a learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, 4(5):882–884, 1993.

[8] Wray L. Buntine and Andreas S. Weigend. Bayesian back-propagation. *Complex Syst.*, 5, 1991.

[9] Mathieu Dagréou, Pierre Ablin, Samuel Vaiter, and Thomas Moreau. How to compute hessian-vector products? In *ICLR Blogposts 2024*, 2024. https://iclr-blogposts.github.io/2024/blog/bench-hvp/.

[10] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[11] Diederik P. Kingma and Max Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[12] R. Kelley Pace and Ronald Barry. Sparse spatial autoregressions. *Statistics Probability Letters*, 33(3):291–297, 1997.

[13] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[14] Alex Krizhevsky. Learning multiple layers of features from tiny images. In *Learning Multiple Layers of Features from Tiny Images*, 2009.

[15] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[16] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010.

[17] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[18] Song Chen. Beijing Multi-Site Air Quality. UCI Machine Learning Repository, 2017. DOI: https://doi.org/10.24432/C5RK5G.

[19] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2498–2507. JMLR.org, 2017.

[20] Artem Ryzhikov. Holybayes/pytorch_ard, 2018.

[21] Emergent Mind. Automatic relevance determination (ard). `https://www.emergentmind.com/topics/automatic-relevance-determination-ard`, 2024. Accessed: 2025-12-18.

[22] David J. C. MacKay and Radford M. Neal. Automatic relevance determination for neural networks. Technical report, Department of Physics, University of Cambridge, 1994. Original ARD prior formulation.

7 Appendix

7.0.1 ARD Prior Derivation

$$\begin{aligned} \alpha_k &\sim \mathrm{Gamma}(a, b) \\ \theta_k \mid \alpha_k &\sim \mathcal{N}(0, \alpha_k^{-1}) \end{aligned}$$

$$\begin{aligned} \mathcal{L}_{\text{ARD}}(\theta, \alpha) &= -\log p(\theta \mid \alpha) - \log p(\alpha) + \text{const} \\ -\log p(\theta \mid \alpha) &= \tfrac{1}{2} \sum_k \big(\alpha_k \theta_k^2 - \log \alpha_k\big) \\ -\log p(\alpha) &= \sum_k (b\,\alpha_k - (a-1)\log \alpha_k) \end{aligned}$$

$$\mathcal{L}_{\text{ARD}}(\theta, \alpha) = \sum_k \big(\tfrac{1}{2}\alpha_k \theta_k^2 \;+\; b\alpha_k \;-\; \big(a - \tfrac{1}{2}\big)\log \alpha_k\big) \;+\; \text{const}$$

7.1 Additional Sample Validation Charts

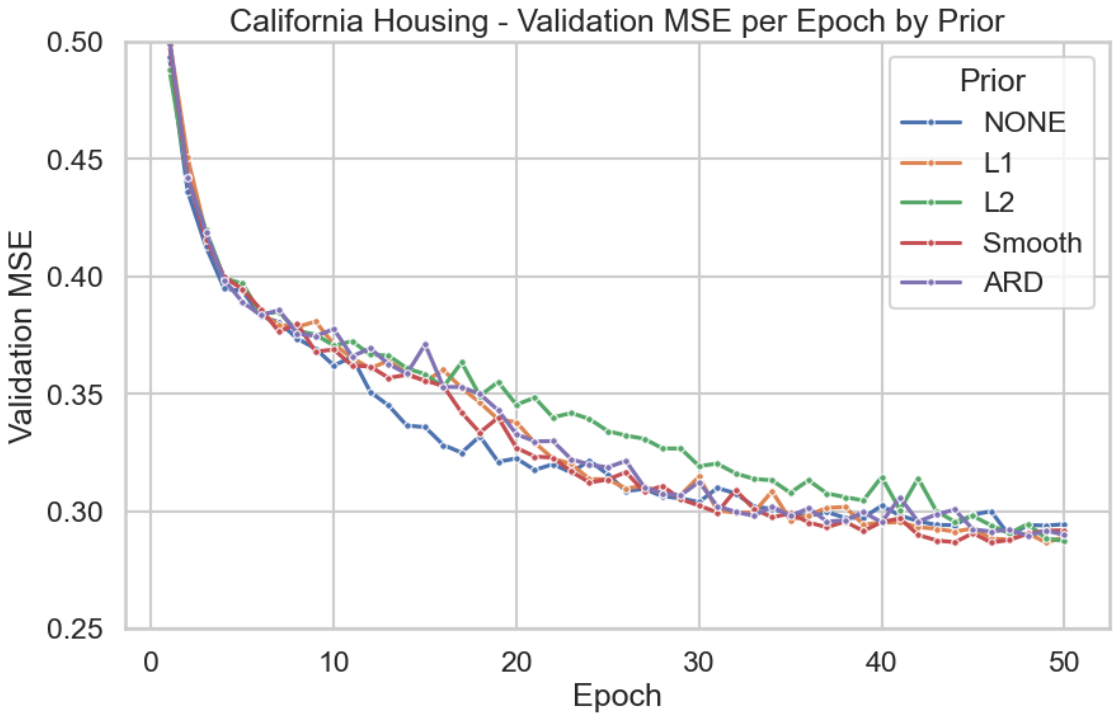


Figure 5: California MLP

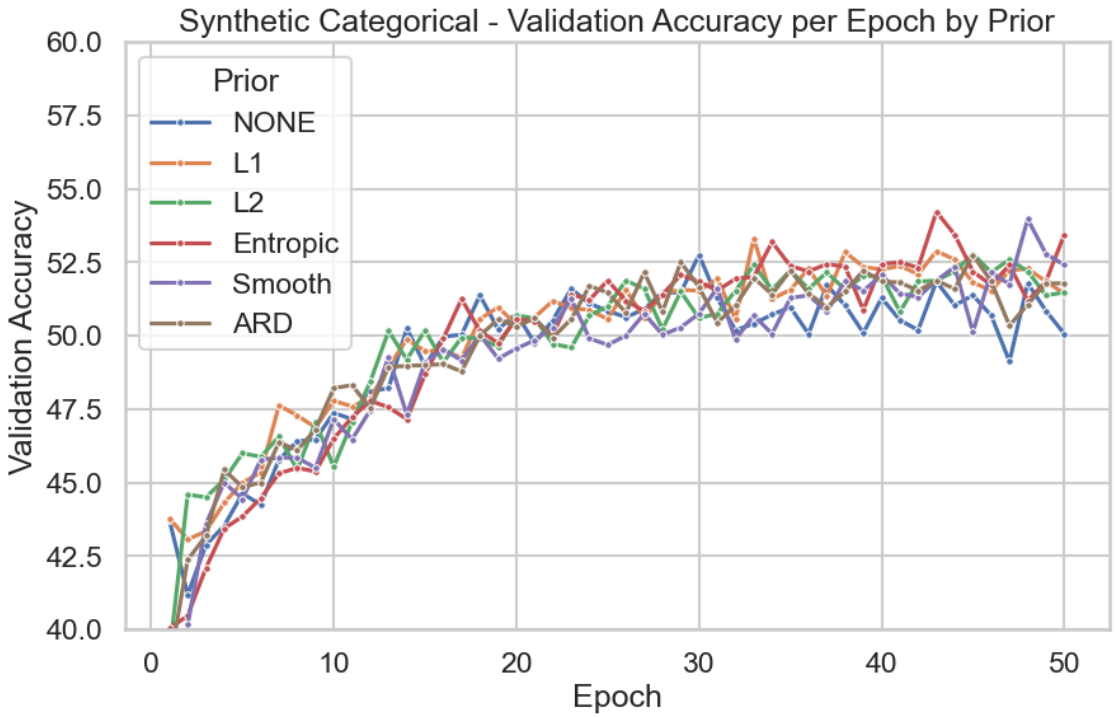


Figure 6: Synthetic Categorical Data

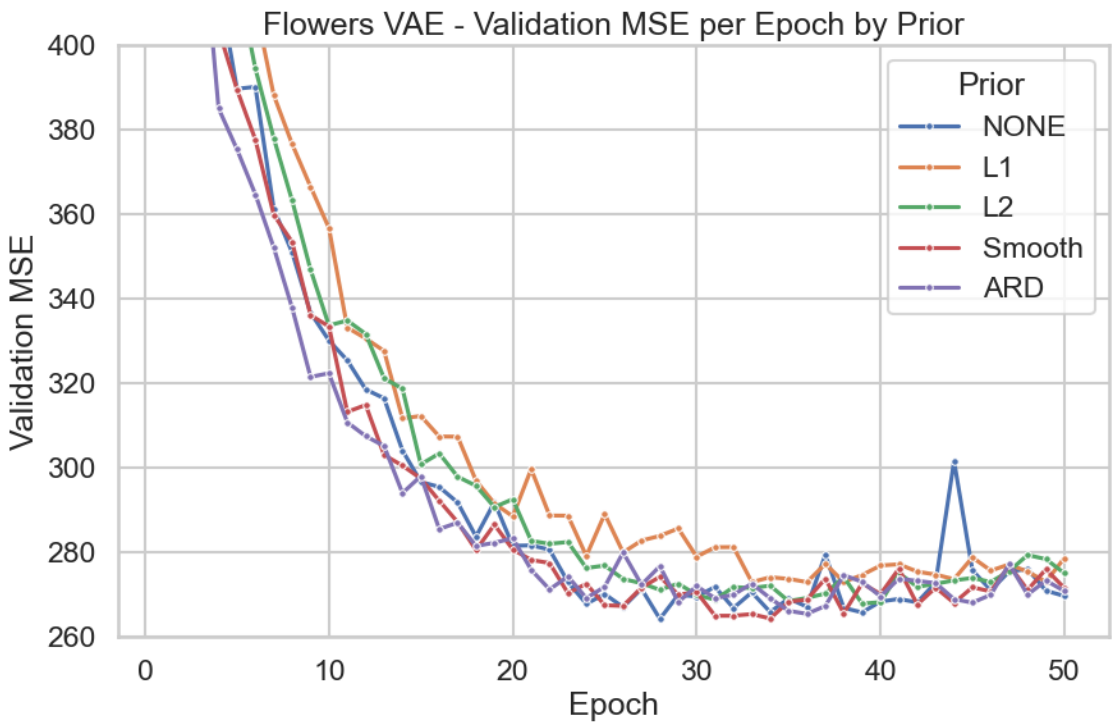


Figure 7: Flowers VAE

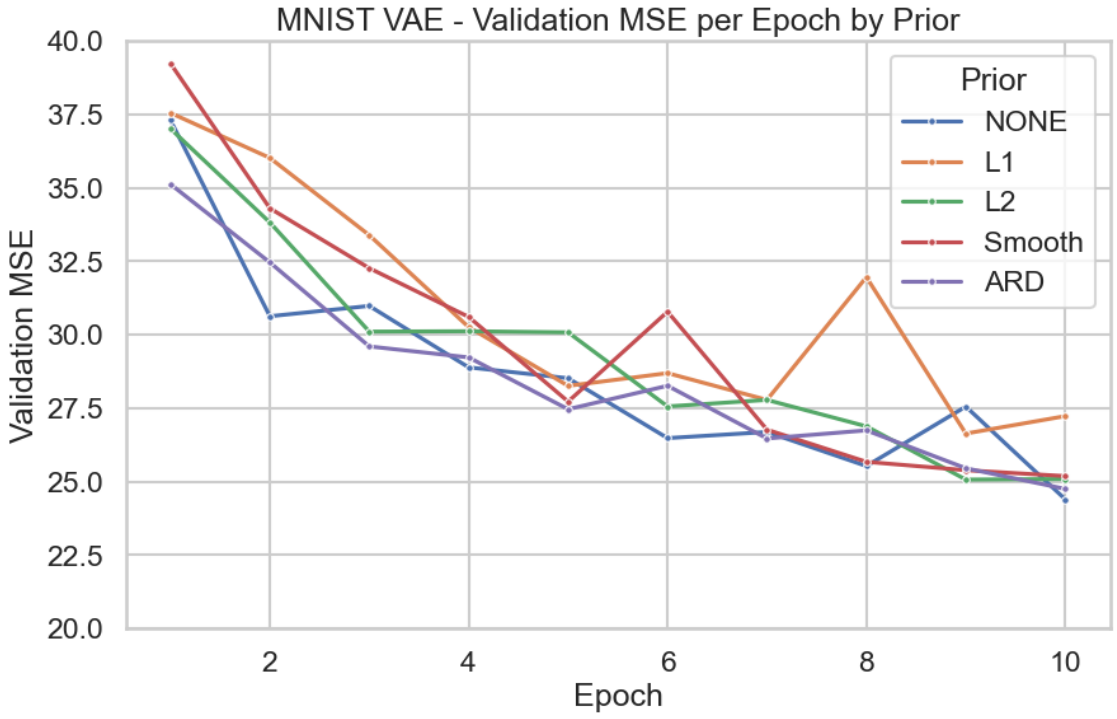


Figure 8: MNIST VAE