ENERGY MINIMIZATION FOR TRAINING DENSE ASSOCIATIVE MEMORY

Bao Pham[*]
RPI
phamb@rpi.edu

Benjamin Hoover[*]
IBM Research
Georgia Tech
benjamin.hoover@ibm.com

Dmitry Krotov
Independent Researcher
krotov.a.dmitry@gmail.com

Parikshit Ram
IBM Research
parikshit.ram@ibm.com

ABSTRACT

Dense Associative Memories (DenseAMs) are modern generalizations of Hopfield networks with high-capacity, energy-based retrieval dynamics, but it remains unclear what the most elegant training principle should be for these models. Contrastive divergence (CD) is theoretically well motivated but requires expensive iterative negative sampling, and backpropagating reconstruction loss through long inference trajectories is also costly while not directly leveraging the explicit energy objective. Inspired by the Hebbian learning rule in classical Hopfield networks, we propose to train DenseAMs by direct energy minimization. For DenseAMs with translation-invariant kernel energies, we show that the partition function is independent of memory parameters, so maximum likelihood estimation (MLE) reduces exactly to minimizing data energy. This yields a sampling-free training rule that preserves an explicit energy formulation. We demonstrate the method in both ambient space and latent space, where a stop-gradient coupling with an autoencoder enables stable joint training and memory synthesis from latent noise.

1 INTRODUCTION

Hopfield networks (Amari, 1972; Hopfield, 1982) are among the earliest Associative Memory (AM) systems and energy-based models (EBMs). Specifically, these systems define an explicit energy landscape where stored patterns are attractors, i.e., low-energy minima. The inference process (memory retrieval) of these systems starts from a noisy or partial query, and iteratively follows energy-decreasing dynamics toward a nearby stable state or *memory*: a local minimum of the energy. The modern revisions of these systems, Dense Associative Memories (DenseAMs) (Krotov & Hopfield, 2016), extend this framework with higher memory capacity and sharper basins around the memories while preserving the AM retrieval principle. From an EBM perspective, lower-energy states have higher probability while implausible or noisy queries remain at higher energy.

This retrieval perspective naturally raises the training question for AMs: how should we shape the energy landscape so inference reliably retrieves data-like patterns at low-energy minima? In general, there are many ways to train EBMs, including AMs. A natural starting point is to borrow from the general EBM literature, where the typical approach is contrastive divergence (CD) (Hinton & Sejnowski, 1986; Ackley et al., 1985; Hinton, 2002), which approximates Maximum Likelihood Estimation (MLE) without directly differentiating an intractable partition function. Though elegant, CD requires iterative negative sampling from an evolving model distribution, introducing both computational overhead and approximation error while training (Du et al., 2020; Song & Kingma, 2021; Dawid & LeCun, 2024). This inefficiency has motivated other EBM training methods (Hyvärinen, 2005; Hyvärinen, 2007; Vincent, 2011; Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Grathwohl et al., 2020; Song et al., 2021), many of which approximate negative energy gradients (scores) directly, which work well since the score function drives the inference dynamics, but comes at the cost of the theoretical power of explicit energy formulations.

[*] denotes equal contribution.

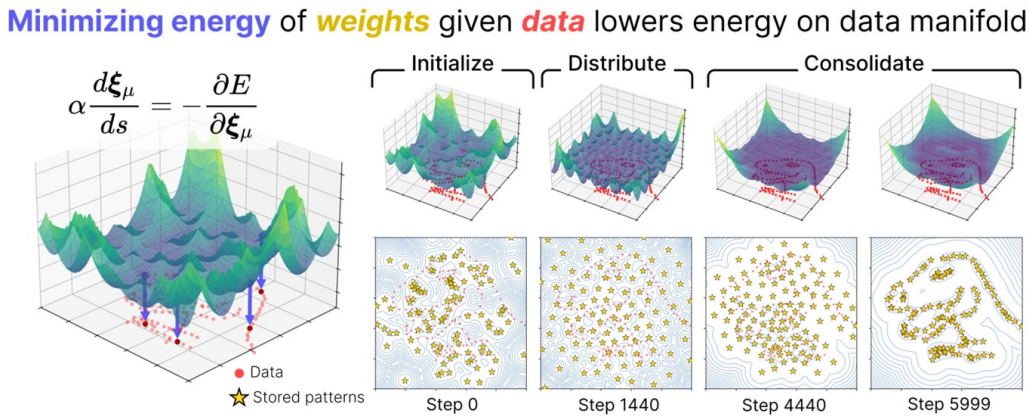

Figure 1: Learning DenseAM memories by direct energy minimization. We evaluate the Gaussian-kernel DenseAM energy function (4) at each data point and minimize it with respect to memory weights, using a form of annealed Langevin dynamics to keep memories active during training (see Sec. (A.1) for full training details). Red dots are dino data points (Matejka & Fitzmaurice, 2017), and yellow stars are the learned memories. The final memories form isotropic Gaussian basins of attraction that line the data manifold. Training this kind of DenseAM in this manner is strictly equivalent to Maximum Likelihood Estimation (see Sec. (3)).

In this work, we exploit a structural property of specific DenseAMs to obtain a *sampling-free training alternative* to CD while keeping an explicit energy formulation. Specifically, the partition function of DenseAMs with translation-invariant kernel energies is independent of the memory parameters, allowing MLE to reduce *exactly* to direct energy minimization on observed data. Training then becomes conceptually aligned with retrieval: both processes minimize the same energy, but with respect to different variables. This rule then exhibits the same elegance of the classical Hopfield Hebbian rule (Hopfield, 1982). See Fig. (1) for an illustration of this training process.

Furthermore, we extend this framework from the ambient space to the latent space by coupling a DenseAM with an autoencoder. A stop-gradient path through the latent code stabilizes the joint optimization of these two systems, allowing the learning of memories through energy minimization while the autoencoder learns meaningful latent representations of the data. In this setup, we also obtain a competitive score on generations of images on the CIFAR-10 dataset (Krizhevsky, 2009), showcasing the viability of this method for training certain types of DenseAMs.

2 Background

In EBMs, the energy and probability of a state $\mathbf{x} \in \mathbb{R}^N$ are linked through the Boltzmann distribution with the inverse temperature $\beta > 0$:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{e^{-\beta E_{\boldsymbol{\theta}}(\mathbf{x})}}{Z_{\boldsymbol{\theta}}}, \tag{1}$$

where $E_{\boldsymbol{\theta}}(\mathbf{x}) : \mathbb{R}^N \to \mathbb{R}$, $\boldsymbol{\theta}$ denotes model parameters, and $Z_{\boldsymbol{\theta}} = \int \exp(-\beta E_{\boldsymbol{\theta}}(\mathbf{y}))\, \mathrm{d}\mathbf{y}$ is the partition function. The most natural learning objective is maximum likelihood estimation (MLE), i.e., minimizing negative log-likelihood on observed data $\mathbf{x} \sim p_{\text{data}}$:

$$\boldsymbol{\theta}^\star_{\text{MLE}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\mathbf{x}\sim p_{\text{data}}} [-\log p_{\boldsymbol{\theta}}(\mathbf{x})] = \operatorname*{argmin}_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\mathbf{x}\sim p_{\text{data}}} [E_{\boldsymbol{\theta}}(\mathbf{x}) + \log Z_{\boldsymbol{\theta}}]\,. \tag{2}$$

Finding $\boldsymbol{\theta}^\star_{\text{MLE}}$ in Eq. (2) requires gradients through the partition term, i.e., $\nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}}$. For general energies, $Z_{\boldsymbol{\theta}}$ is intractable, so directly optimizing Eq. (2) with that term is typically infeasible. Contrastive divergence (CD) (Hinton & Sejnowski, 1986; Ackley et al., 1985; Hinton, 2002) addresses this by replacing the partition-term gradient with sampled "negative" states:

$$\boldsymbol{\theta}^\star_{\text{CD}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \mathop{\mathbb{E}}_{\mathbf{x}\sim p_{\text{data}}} \Big[E_{\boldsymbol{\theta}}(\mathbf{x})\Big] - \mathop{\mathbb{E}}_{\mathbf{x}^-\sim p_{\boldsymbol{\theta}}} \Big[E_{\boldsymbol{\theta}}(\mathbf{x}^-)\Big], \tag{3}$$

which we can view as lowering the energy of data points while raising the energy of model-generated samples. CD is effective, but it is costly since it requires drawing $\mathbf{x}^- \sim p_{\boldsymbol{\theta}}$ from an evolving model at each update step. In practice, gradient-based samplers (like Langevin dynamics (Welling & Teh,

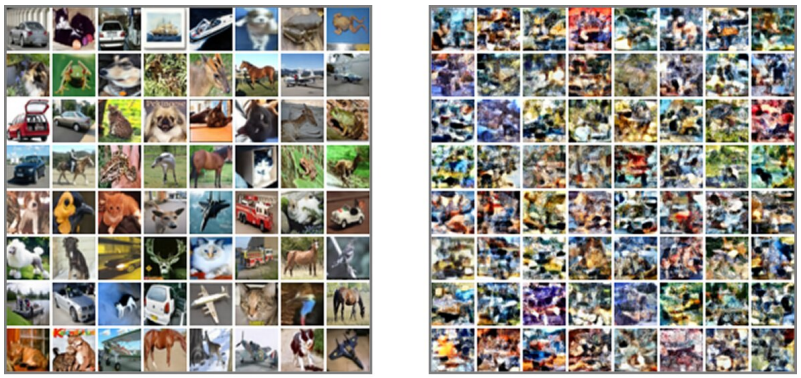

Figure 2: Generation examples **(a)** of our trained model on CIFAR-10 (Krizhevsky, 2009), with $K = 4096$ latent memories, generated from latent noise vectors **(b)** sampled from a Gaussian distribution. The generations are computed by performing Eq. (8) with Langevin noise at each energy descent step for diversity. In these illustrations, generated examples and noise vectors are mapped back to image space via the decoder g_{ϕ}.

2011)) are helpful, but they also introduce approximation error (Du et al., 2020; Song & Kingma, 2021). Another common route is to directly differentiate through unrolling the inference (typically done in training AM) or sampling dynamics. However, this approach can be very memory-intensive and possibly unstable (due to vanishing or exploding gradients) for long unrolls (Metz et al., 2016; Savinov et al., 2021; Saha et al., 2023; Hoover et al., 2023; Dehmamy et al., 2025).

3 Energy Minimization

The bottleneck in the previous section was the partition function. For a broad class of DenseAM energies, that term becomes parameter-independent, and training reduces exactly to direct energy minimization. Intuitively, lowering energy at observed data points should move memories toward the data manifold, as seen in Fig. (1). Below, following the convention of AM literature, we describe energy parameters as memory weights $\boldsymbol{\Xi}$ rather than $\boldsymbol{\theta}$.

In DenseAMs, a common general form of the energy is $E_{\boldsymbol{\Xi}}(\mathbf{x}) = -Q\big[\sum_{\mu=1}^{K} F\big(\beta S(\mathbf{x}, \boldsymbol{\xi}_\mu)\big)\big]$, where $Q[\cdot]$ is monotonic, $S : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$ is a similarity function, and $F : \mathbb{R} \to \mathbb{R}$ controls basin sharpness around each memory $\boldsymbol{\xi}_\mu$ in the bank $\boldsymbol{\Xi} \in \mathbb{R}^{K \times N}$ (Hoover et al., 2025). In this work, we focus on the Gaussian-kernel DenseAM (Millidge et al., 2022; Saha et al., 2023; Ambrogioni, 2023; Pham et al., 2025):

$$E_{\boldsymbol{\Xi}}(\mathbf{x}) = -\frac{1}{\beta} \log \sum_{\mu=1}^{K} \exp\left(-\beta \|\mathbf{x} - \boldsymbol{\xi}_\mu\|^2\right). \tag{4}$$

With fixed β, this yields our proposed energy minimization objective and update rule for memory $\boldsymbol{\Xi} = \{\boldsymbol{\xi}_\mu\}_{\mu=1}^{K}$ as

$$\boldsymbol{\Xi}^{\star}_{\text{EM}} = \operatorname*{argmin}_{\boldsymbol{\Xi}} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[E_{\boldsymbol{\Xi}}(\mathbf{x})\right], \qquad \text{and} \quad \frac{\mathrm{d}\boldsymbol{\xi}_\mu}{\mathrm{d}s} = -\nabla_{\boldsymbol{\xi}_\mu} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \left[E_{\boldsymbol{\Xi}}(\mathbf{x})\right], \tag{5}$$

where s denotes training time (we reserve t for inference-time denoising dynamics). In discrete time, this is standard gradient descent with step size η. Since $\beta > 0$ and constant, minimizing $E_{\boldsymbol{\Xi}}$ is equivalent to minimizing $\beta E_{\boldsymbol{\Xi}}$. When specialized to the classical Hopfield energy, this gradient-based memory update recovers the standard Hebbian outer-product learning rule (up to scaling) (Hopfield, 1982). In practice, we can add langevin dynamics or use other tricks like dropout or noisy data augmentation to keep memories active during training (e.g., see Sec. (A.1)).

For the Gaussian kernel in Eq. (4), *the partition function is independent of the parameters* $\boldsymbol{\Xi}$:

$$Z_{\boldsymbol{\Xi}} = \int \exp(-\beta E_{\boldsymbol{\Xi}}(\mathbf{y}))\, \mathrm{d}\mathbf{y} = \int \sum_{\mu=1}^{K} \exp\big(-\beta \|\mathbf{y} - \boldsymbol{\xi}_\mu\|^2\big)\, \mathrm{d}\mathbf{y} = \sum_{\mu=1}^{K} (\pi/\beta)^{N/2} = C_\beta. \tag{6}$$

Therefore, $\log Z_{\boldsymbol{\Xi}}$ is constant in $\boldsymbol{\Xi}$, and MLE via Eq. (2) reduces exactly to Eq. (5) because the term $\nabla_{\boldsymbol{\Xi}} \log Z_{\boldsymbol{\Xi}}$ vanishes. This same argument holds for translation-invariant kernels more generally, including compactly-supported choices such as the Epanechnikov form of DenseAM in Hoover

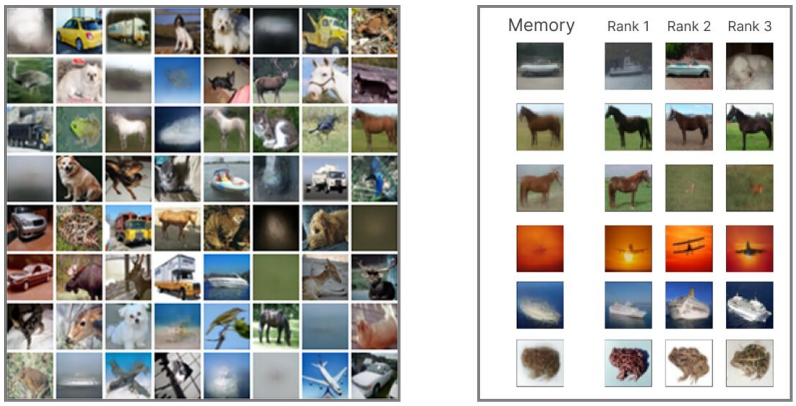

Figure 3: Illustrations of a subset of 64 decoded latent memories in **(a)** belonging to the learned memory bank $\boldsymbol{\Xi}$ for CIFAR-10 (Krizhevsky, 2009), and a hand-selected set of interesting and novel memories with their top-3 nearest neighbors from the training set in **(b)**. The memories in **(b)** are not entirely duplicates of the training points, where some of them can be superpositions of some training points in the latent space.

et al. (2025). See Fig. (1) for an illustration of energy minimization training in the ambient-space, where direct minimization of Eq. (5) yields memories that line the data manifold with isotropic Gaussian basins of attraction. This idea also generalizes to larger data like MNIST (Sec. (A.2)). In the next section, we extend this same training principle to the latent space by jointly training with an autoencoder.

4 Latent Energy Minimization

So far, we have discussed aspects of CD and energy minimization in the input space $\mathbb{R}^N$ instead of a latent space $\mathbb{R}^d$ where $d < N$. Hence, *is it possible to train a DenseAM operating in the latent space alongside an autoencoder?*

In the input space, Eq. (4) can easily be optimized via Eq. (5). However, if one were to jointly train a DenseAM, an encoder $f_{\boldsymbol{\theta}} : \mathbb{R}^N \to \mathbb{R}^d$, and a decoder $g_{\boldsymbol{\phi}} : \mathbb{R}^d \to \mathbb{R}^N$, this cannot easily be done. If the gradients flow through these sets of parameters ($\boldsymbol{\theta}$, $\boldsymbol{\phi}$, and $\boldsymbol{\Xi}$), it is unlikely that this system will converge, as there exists a tug-of-war between them, hindering the latent embeddings from learning properly and causing the energy function to collapse (Dawid & LeCun, 2024).

To make this setup *trainable*, we borrow a trick from VQ-VAE (Van Den Oord et al., 2017), involving the stop-gradient operator $\Omega(\cdot)$. Specifically, we have the following objective:

$$\mathcal{L}\big(\boldsymbol{\theta}, \boldsymbol{\phi}, \boldsymbol{\Xi}\big) = E_{\boldsymbol{\Xi}}\big(\Omega(\mathbf{z})\big) + \mathcal{L}_{\text{recon}}\big(g_{\boldsymbol{\phi}}(\mathbf{z}), \mathbf{x}\big), \qquad \mathbf{z} \sim f_{\boldsymbol{\theta}}(\mathbf{x}), \tag{7}$$

where $\mathbf{z} \in \mathbb{R}^d$ is a latent variable and $\Omega(\mathbf{z})$ stops the gradient flows through $\mathbf{z}$ back to the encoder $f_{\boldsymbol{\theta}}(\mathbf{x})$ from $E_{\boldsymbol{\Xi}}(\cdot)$. The reconstruction loss $\mathcal{L}_{\text{recon}}$ ensures the consistency between the decoded latent embeddings and the data points. Nonetheless, since the operator $\Omega(\cdot)$ decouples the gradient flow between the two objectives in Eq. (7), one could train the autoencoder and DenseAM separately.

Once trained, the DenseAM can act as a generative prior: one can synthesize new data by initializing a noise vector in the latent space $\mathbb{R}^d$, minimizing the energy via gradient descent to find a memory[1],

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} E_{\boldsymbol{\Xi}}(\mathbf{z}) \tag{8}$$

and decoding the result $\mathbf{z}^*$ via the decoder $g_{\boldsymbol{\phi}}$, see Fig. (2) for examples. For better diversity, we rely on Langevin dynamics (Welling & Teh, 2011), to add a small amount of noise at each gradient descent step in Eq. (8).

4.1 Experiment

We quantitatively compare our model, trained using Eq. (7), against established baselines using the Fréchet Inception Distance (FID) (Heusel et al., 2017) on CIFAR-10 (Krizhevsky, 2009). The

[1]The operator $\Omega(\cdot)$ is not applied for this process as we want $\mathbf{z}$ to be differentiable during inference.

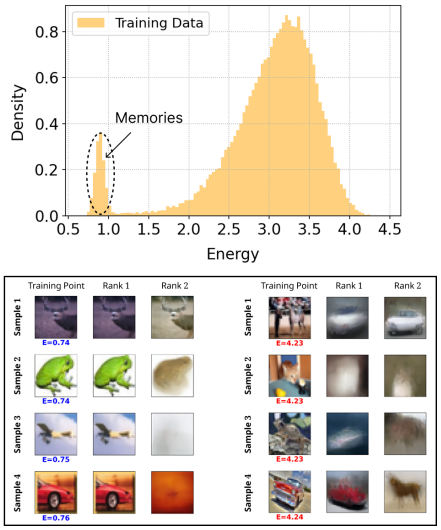

Figure 4: (Top) Density histogram of energy values computed on the embeddings of training data samples. (Bottom) Randomly selected low energy samples and high energy samples alongside their respective top-2 nearest memories identified via ℓ_2-distance in the embedding space. Samples with a low energy value are stored as memories, while those with a high energy value are associated with a novel memory.

Model		**FID**
Deep-GAN	(Radford et al., 2015)	38.2
WGAN-GP	(Gulrajani et al., 2017)	36.4
SN-GAN	(Miyato et al., 2018)	21.7
IGEBM	(Du & Mordatch, 2019)	38.2
f-EBM	(Yu et al., 2020)	30.86
I-IGEBM	(Du et al., 2020)	25.1
NCSN	(Song & Ermon, 2019)	25.32
G-EBM	(Arbel et al., 2020)	19.31
D-EBM	(Gao et al., 2020)	**9.58**
Ours		12.25

Table 1: FID scores on CIFAR-10 (Krizhevsky, 2009). Lower is better. The best baseline is highlighted in gray. Here, our model beats various complicated EBMs, including the one trained with score-matching (Song & Ermon, 2019), but it does not beat the EBM trained with the diffusion-based CD (Gao et al., 2020).

autoencoder architecture is based on the ResNet used in Ho et al. (2020) without time-conditioning. As described above, to generate our samples for the evaluation of FID, we initialized the latent noise vectors from a Gaussian distribution and performed Eq. (8) with Langevin dynamics. For more details, please refer to Appx. (A).

While FID is the standard metric for assessing generalization, it should be noted that it is conventionally evaluated by comparing the distribution of generated samples directly against the entire training set for certain small datasets, like CIFAR-10. Since our model tries to compress the entire dataset into a small set of $K = 4096$ latent memories, we can observe a distinction in energy values within the training samples shown in Fig. (4). The density histogram in Fig. (4) reveals that only a small fraction of the training data resides in the extremely low-energy state corresponding directly to the learned memories. The majority of the training data points yield higher energy values, indicating that they are not local minima. Specifically, from Fig. (3), we can observe *novel memories* that do not correspond to training samples, implying that our model does not trivially memorize.

Nonetheless, our model achieves a highly competitive FID, as shown in Tab. (1). Specifically, it outperforms several sophisticated EBMs, including the score-matching model (Song & Ermon, 2019), and other types of generative models. While our model does not surpass the method from Gao et al. (2020), it is crucial to note that their approach relies on a more sophisticated version of CD based on diffusion. However, from our ablation study in Tab. (4) of Appx. (B), increasing the number of memories K does yield better FID scores. Overall, our result demonstrates that the energy-minimization setup yields promising results.

5 Conclusion

In this work, we introduced a sampling-free approach to training DenseAMs by exploiting the analytic tractability of their partition function. By equipping the DenseAM with a translation-invariant similarity metric, we demonstrated that direct energy minimization circumvents the need to sample negative examples from the model (which is inherent to the CD method). Furthermore, we extended this framework to the latent space by jointly training the DenseAM with an autoencoder. Our empirical results on CIFAR-10 confirm that this approach allows our DenseAM to learn memories in a compressed latent space, synthesize from noise vectors in latent space, and deliver promising FID results. Overall, energy minimization is a viable training method for certain types of DenseAMs.

ACKNOWLEDGMENTS

Bao Pham acknowledges the support of RPI-IBM Future of Computing Research Collaboration program (FCRC).

REFERENCES

David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985.

S-I Amari. Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on computers*, 100(11):1197–1206, 1972.

Luca Ambrogioni. In search of dispersed memories: Generative diffusion models are associative memory networks, September 2023. URL http://arxiv.org/abs/2309.17290.

Michael Arbel, Liang Zhou, and Arthur Gretton. Generalized energy based models. *arXiv preprint arXiv:2003.05033*, 2020.

Anna Dawid and Yann LeCun. Introduction to latent variable energy-based models: a path toward autonomous machine intelligence. *Journal of Statistical Mechanics: Theory and Experiment*, 2024(10):104011, 2024.

Nima Dehmamy, Benjamin Hoover, Bishwajit Saha, Leo Kozachkov, Jean-Jacques Slotine, and Dmitry Krotov. Nrgpt: An energy-based alternative for gpt. *arXiv preprint arXiv:2512.16762*, 2025.

Yilun Du and Igor Mordatch. Implicit generation and generalization in energy-based models. *Advances in Neural Information Processing Systems*, 32, 2019.

Yilun Du, Shuang Li, Joshua Tenenbaum, and Igor Mordatch. Improved contrastive divergence training of energy based models. *arXiv preprint arXiv:2012.01316*, 2020.

Ruiqi Gao, Yang Song, Ben Poole, Ying Nian Wu, and Diederik P Kingma. Learning energy-based models by diffusion recovery likelihood. *arXiv preprint arXiv:2012.08125*, 2020.

Will Grathwohl, Kuan-Chieh Wang, Jörn-Henrik Jacobsen, David Duvenaud, and Richard Zemel. Learning the stein discrepancy for training and evaluating energy-based models without sampling. In *International Conference on Machine Learning*, pp. 3732–3747. PMLR, 2020.

Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Geoffrey E. Hinton and Terrence J. Sejnowski. Learning and relearning in boltzmann machines. In David E. Rumelhart, James L. McClelland, and PDP Research Group (eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations*, pp. 282–317. MIT Press, 1986.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Benjamin Hoover, Yuchen Liang, Bao Pham, Rameswar Panda, Hendrik Strobelt, Duen Horng Chau, Mohammed Zaki, and Dmitry Krotov. Energy transformer. *Advances in Neural Information Processing Systems*, 36, 2023.

Benjamin Hoover, Zhaoyang Shi, Krishnakumar Balasubramanian, Dmitry Krotov, and Parikshit Ram. Dense associative memory with epanechnikov energy. *arXiv preprint arXiv:2506.10801*, 2025.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Aapo Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(24):695–709, 2005. URL http://jmlr.org/papers/v6/hyvarinen05a.html.

Aapo Hyvärinen. Some extensions of score matching. *Computational Statistics and Data Analysis*, 51(5):2499–2512, 2007. ISSN 0167-9473. doi: 10.1016/j.csda.2006.09.003. URL https://www.sciencedirect.com/science/article/pii/S0167947306003264.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf. Technical report.

Dmitry Krotov and John J. Hopfield. Dense associative memory for pattern recognition. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/eaae339c4d89fc102edd9dbdb6a28915-Paper.pdf.

Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Justin Matejka and George Fitzmaurice. Same stats, different graphs: Generating datasets with varied appearance and identical statistics through simulated annealing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pp. 1290–1294, 2017. doi: 10.1145/3025453.3025912. URL https://doi.org/10.1145/3025453.3025912.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *arXiv preprint arXiv:1611.02163*, 2016.

Beren Millidge, Tommaso Salvatori, Yuhang Song, Thomas Lukasiewicz, and Rafal Bogacz. Universal hopfield networks: A general framework for single-shot associative memory models. In *International Conference on Machine Learning*, pp. 15561–15583. PMLR, 2022.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018.

Bao Pham, Gabriel Raya, Matteo Negri, Mohammed J Zaki, Luca Ambrogioni, and Dmitry Krotov. Memorization to generalization: Emergence of diffusion models from associative memory. *arXiv preprint arXiv:2505.21777*, 2025.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Bishwajit Saha, Dmitry Krotov, Mohammed J Zaki, and Parikshit Ram. End-to-end differentiable clustering with associative memories. In *International Conference on Machine Learning*. PMLR, 2023.

Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. Step-unrolled denoising autoencoders for text generation. *arXiv preprint arXiv:2112.06749*, 2021.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, 2015.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song and Diederik P Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.

Aaron Van Den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. *Advances in Neural Information Processing Systems*, 30, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural Comput.*, 23(7):1661–1674, July 2011. ISSN 0899-7667. doi: 10.1162/NECO_a_00142. URL https://doi.org/10.1162/NECO_a_00142.

Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, ICML'11, pp. 681–688, 2011.

Lantao Yu, Yang Song, Jiaming Song, and Stefano Ermon. Training deep energy-based models with f-divergence minimization. In *International Conference on Machine Learning*, pp. 10957–10967. PMLR, 2020.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in neural information processing systems*, 32, 2019.

A Additional Details

A.1 Training Setup for Figure 1

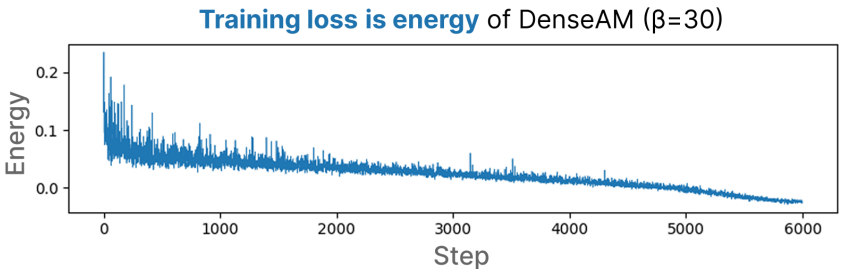

Figure 5: Training loss for the DenseAM run used in Fig. (1). Noisy loss minimization comes from the noise added to the minibatches.

We trained a Gaussian-kernel DenseAM with the energy function (4) on the 2D dinosaur dataset (Matejka & Fitzmaurice, 2017) in Fig. (1), using Eq. (5) as the objective. The dataset has 135 points, and we learned $K = 120$ memory vectors. Memory weights were initialized from a zero-mean Gaussian with standard deviation 1.5. We used a constant inverse temperature $\beta = 30$, the Adam (Kingma, 2014) optimizer with learning rate of 0.007, minibatch size of 64, and 6000 training steps.

Each training step is simple: sample a minibatch of 64 data points, add Gaussian noise to every point in that minibatch, compute the average energy of the noisy minibatch, and update the memory weights with Adam to lower that average energy.

The standard deviation of the noise added to minibatches is linearly decayed over training, from 1.0 at the start to 0.1 at the end of the 6000-th step. Strong perturbations at early steps help activate and train the weights to reduce the number of "dead" weights.

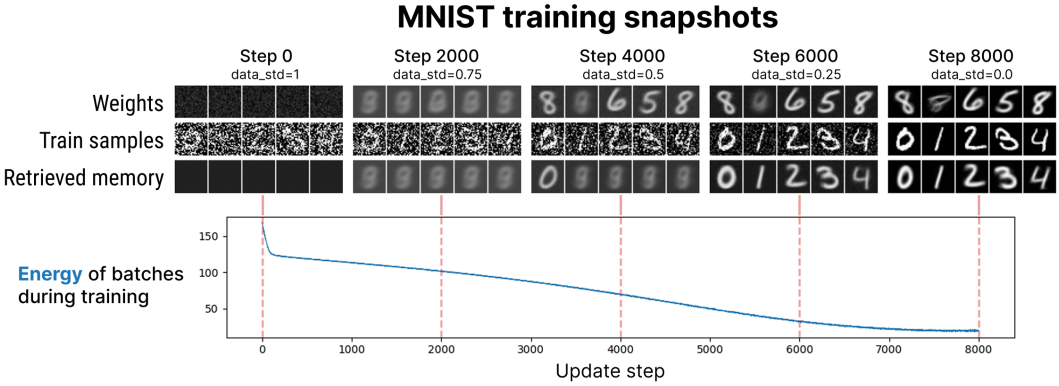

Figure 6: MNIST training snapshots across optimization. (top) Visualizing the effect of training and augmentation on learned weights. Columns show checkpoints at steps 0, 2000, 4000, 6000, and 8000 while the data-noise standard deviation is annealed from 1.0 to 0.0. At each checkpoint, we visualize learned memory weights (top row), augmented noisy training samples from the current batch (middle row), and the retrieved memories after energy-descent dynamics (bottom row). Early memories are diffuse and retrieval is poor; later memories sharpen into class prototypes and retrievals align with denoised digits. The lower panel shows the energy over each minibatch while training, with dashed vertical lines marking the displayed checkpoints.

A.2 Training setup for MNIST

We repeat the experiment of Sec. (A.1) on MNIST, training Gaussian-kernel DenseAM on the same energy-minimization objective (5). Input images were normalized to $[0, 1]$, flattened to $N = 784$, and taken from the first 50,000 training images (with 10,000 test images used only for evaluation). We learned $K = 1000$ memory vectors.

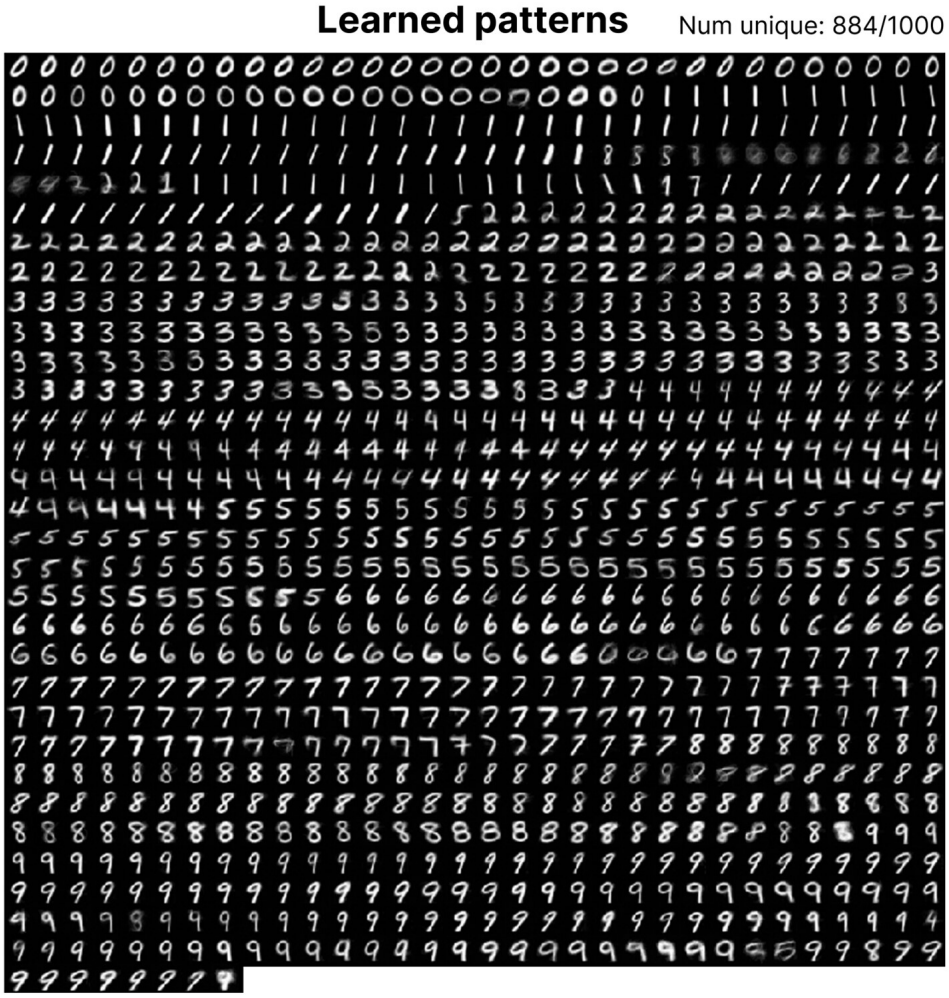

Figure 7: Energy minimization learns diverse and high quality prototype patterns on MNIST. Visualized are all 1000 learned memory patterns trained on MNIST, sorted by their skew and KNN-nearest class label. Each stored pattern serves as an attractor to denoising dynamics.

Memory weights were initialized from a Gaussian with data-dependent parameters:

$$\mu_0 = \mathrm{clip}(\mathrm{mean}(X_{\text{train}}), 0, 1), \quad \sigma_0 = \mathrm{clip}(0.2 \cdot \mathrm{std}(X_{\text{train}}), 0.02, 0.08).$$

We used fixed inverse temperature $\beta = 0.6$, Adam with learning rate 2×10^{-3}, minibatch size 256, and 8000 training steps.

At each step, we sampled a minibatch uniformly (with replacement), added i.i.d. Gaussian noise to each sample, clipped back to $[0, 1]$, computed mean minibatch energy, and updated memory weights with Adam. Like before, we augment each image by adding Gaussian noise, allowing standard deviation to linearly anneal from 1.0 to 0.01 over training. No Langevin noise was added to the memory update rule. The learned memories are shown in Fig. (7), and the training loss curve is shown in Fig. (6).

A.3 Training Setup for CIFAR-10

Using Eq. (7), we trained an autoencoder alongside our DenseAM equipped with the energy function (4) on CIFAR-10 (Krizhevsky, 2009), which has 50,000 training images at 32x32 resolution. The reconstruction loss $\mathcal{L}_{\text{recon}}$ in Eq. (7) is simply $\frac{1}{N}\|g_{\boldsymbol{\phi}}(\mathbf{z}) - \mathbf{x}\|_2^2$, or the mean squared error. Meanwhile, the entire autoencoder is based on the ResNet used in Ho et al. (2020), but without time conditioning.

Residual Block
GroupNorm32()
Swish()
Conv3x3()
GroupNorm32()
Swish()
Conv3x3()
DownConv3x3() [Optional]

Table 2: Residual block architecture. Each 3x3 convolutional layer (no bias) uses padding and stride with a value of 1. There is an optional Conv3x3 layer with a stride of 2 which down-sample the image by half of its resolution.

Hyperparameter	**Value**(s)
Initial Latent Dimension	128
Num. of Residual Blocks	4
Channel Multipliers	[1, 2, 2, 4]
Attention Resolutions	[16, 8]
Attention Heads	4
Memory Dimension	512

Table 3: Hyperparameters for the ResNet used on CIFAR-10 (Krizhevsky, 2009).

Architecture. The encoder consists of four channel-multiplier blocks, where each block upscales the initial embedding dimension by a certain scale and consists of a certain number of residual blocks. Moreover, for each multiplier block, the image resolution is halved, with the exception of the last block. Meanwhile, at certain down-scaled resolutions of the image, a typical attention block (Vaswani et al., 2017) is utilized. At the end of the encoder, the image is flattened and projected to the memory-vector dimension, and RMSNorm (Zhang & Sennrich, 2019) is applied to ensure that the latent embeddings are within a certain scale. Meanwhile, for the decoder, it is the reverse of the encoder, where its first operations undo the flattening, and then subsequent upscale blocks increase the image resolution back to the original by doubling each time. For more details, refer to Tabs. (2) and (3) for the hyperparameters of the ResNet.

Training. For training, we used AdamW (Loshchilov & Hutter, 2017) with the momentum variables $\bar{\beta}_1 = 0.9$ and $\bar{\beta}_2 = 0.95$, where the learning rate was set as 2×10^{-4}. The weight decay value was set as 5×10^{-2}. Additionally, we used linear warmup of the learning rate from 0 to 2×10^{-4} for the beginning 20,000 training steps. Afterwards, we used cosine scheduler to decay the learning rate from the same learning rate back to 0 (Loshchilov & Hutter, 2016). Our minibatch size was set as 64 [2].

Meanwhile, similarly to Ho et al. (2020), we kept track of the exponential-moving-average (EMA) of our trained model and used it during inference. The total number of training steps for the model was 800,000. However, we only utilized the best trained model according to our FID validation. See Fig. (8) for the energy and reconstruction-loss curves.

Our inverse temperature β of the DenseAM was also linearly scheduled from a small value of 1.25×10^{-4} to $\beta = 30$ for the first 50,000 training steps. Moreover, to ensure that our memory bank $\boldsymbol{\Xi}$ converges decently, we randomly replace 10% of our memory vectors every 5,000 steps during those first 50,000 training steps, i.e., we replace memory vectors a total of 10 times. The motivation behind this approach is that our autoencoder learns much faster than our DenseAM, while our memory bank $\boldsymbol{\Xi}$ is still largely stuck in its initial (noisy) initialization. Thus, based on Eq. (4), our latent memories cannot find their nearest data points to learn properly. We also based this motivation on observing the ℓ_2-norms of the gradients of different parameter sets in Fig. (9) from very small training runs of our models.

Sampling. Using our EMA model, we generated samples from latent noise vectors drawn from $\mathcal{N}(0, \sigma^2\, \mathbf{I}_d)$ where $\sigma = 10$. Meanwhile, for Eq. (8), we used a total of 100 steps of energy gradient descent with Langevin noise (Welling & Teh, 2011) added at each step with $\sigma_{\text{langevin}} = 0.005$ and a step size of 0.05. For our DenseAM, we used inverse temperature $\beta = 30$, fixed throughout the descent. At the end of the descent, we used our decoder $g_{\boldsymbol{\phi}}$ to map the latent vectors back to image space. Lastly, for the FID computation (Heusel et al., 2017), we used the CIFAR-10 training dataset as our reference set, as done by related works in Tab. (1).

[2] See https://github.com/Lemon-cmd/Energy-Minimization for the code on latent energy minimization.

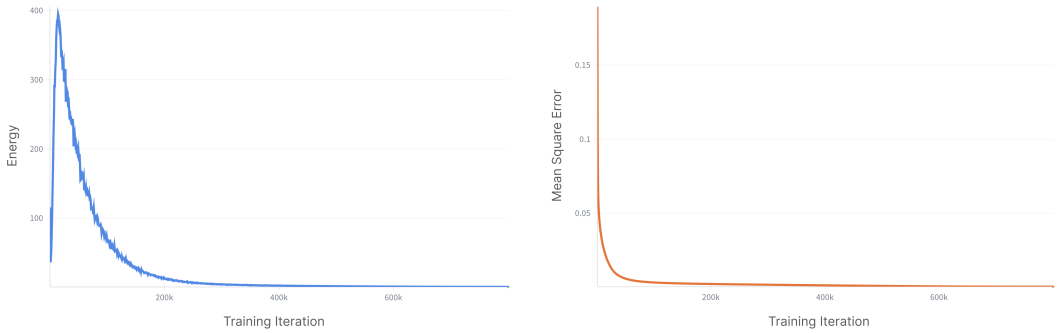

Figure 8: Training loss curve for the model trained on CIFAR-10 (Krizhevsky, 2009) with $K = 4096$ memories. The total loss is the combination of Eq. (4) and $\mathcal{L}_{\text{recon}}$, but they are separated here for illustration. The energy increases initially due to the increasing inverse temperature β for the first 50,000 training steps.

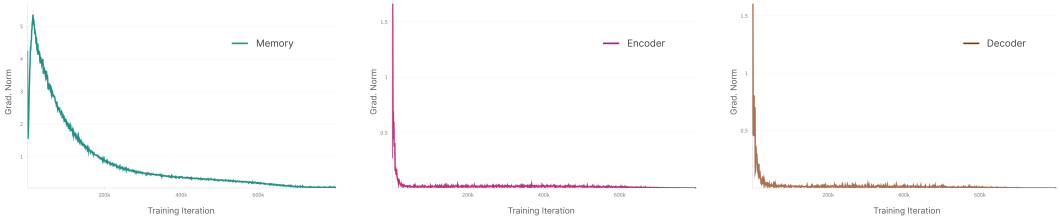

Figure 9: Illustrations of the ℓ_2-norm of the gradients of $\boldsymbol{\Xi}, \boldsymbol{\theta}$ and $\boldsymbol{\phi}$, respectively. Here, the curve of the memory bank's gradient demonstrates that $\boldsymbol{\Xi}$ learns slower than that of the autoencoder's parameters.

B Additional Results

Num. of Memories	FID	Total Param. Count
1536	20.55	$\sim$ 64.6M
3072	14.00	$\sim$ 66.3M
4096	**12.25**	$\sim$ 67.4M

Table 4: Ablation FID results based on varying numbers of memories K for CIFAR-10.

C Usage of LLM Disclaimer

LLMs were utilized in part of the writing of the manuscript, specifically for fine-tuning sentences and syntax correction.

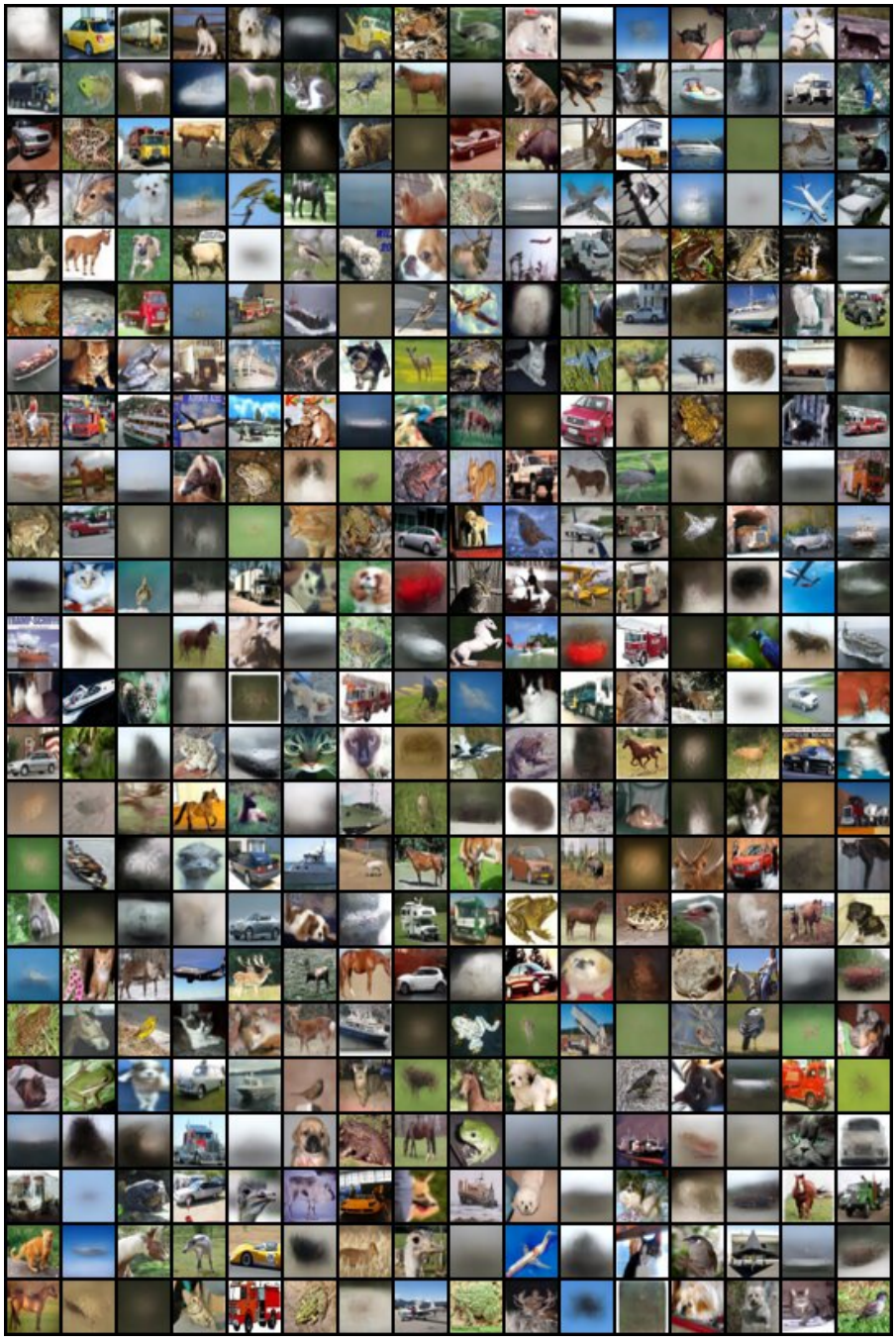

Figure 10: Illustration of 384 decoded latent memories from the learned memory bank Ξ on CIFAR-10. Here, there are prototypical memories alongside training data points which have been stored inside of the model.

Figure 11: Illustration of 384 examples generated from latent noise vectors sampled from Gaussian distribution, by using Eq. (8) alongside Langevin noise at each energy descent step.