Investigations into Compressing Energy-Based Models

Pedro Sousa (pmms2)

Abstract

Energy-based models (EBMs) are powerful deep generative models adept to diverse downstream tasks that map inputs to scalar energy values. Using a customised EBM baseline with ResNet18 as the energy function, quantization, pruning, and distillation are tested to reduce size and increase efficiency. A 1/145th-sized student model created via distillation matches baseline image quality and anomaly detection. Despite requiring further tuning, results suggest distillation can significantly compress EBMs while retaining performance. This initial analysis opens doors for deploying EBMs by overcoming sampling complexity via model compression.

1 Introduction

Energy-based deep learning models (EBMs) were first introduced by [20] as strong contestants to Generative Adversarial Networks (GANs) [6] using *energy functions* to map a sample to its scalar energy value. A less restrictive functional form is possible since *energy functions* can theoretically be parameterised by any nonlinear regression function. EBMs' renewed popularity derives from their natural fit within discriminative frameworks, satisfying researchers' wishes for generative models that benefit downstream problems [8]. [7] initially formalised this potential by introducing Joint Energy Models (JEMs), a hybrid model class designed for both generative and discriminative modelling.

Although advantageous because of its unique flexibility, training EBMs has been documented as an especially difficult process since it requires expensive and unstable sampling techniques to bypass the intractability of computing the likelihood of samples [19]. Therefore, EBMs have been underexplored, particularly in terms of model efficiency and optimisation. Our work aims to bridge this gap by being an introductory study on applying network compression techniques to an EBM. Through empirical investigation, we aimed to reflect on which compression methods yield the best performance on a limited network's footprint.

This study is mainly motivated by EBMs' potential to replace other model classes in real-world deployments. Given their inherent flexibility and dual capacity in generative and discriminative tasks, EBMs appear ideal for resource-constrained environments, potentially obviating the need for separate task-specific models. We recognise the long journey until the mainstream adoption of EBMs, but we hope to contribute with a meaningful investigation into compressing these models.

2 Theoretical Background

2.1 Overview of EBMs

Generative models are designed to estimate the probability distribution over a given dataset, $p(\mathbf{x})$. However, for high-dimensional data, simple neural networks alone struggle to fulfil two key properties of probability distributions: $p(\mathbf{x}) \geq 0$ and $\int_x p(\mathbf{x})\, dx = 1$. EBMs recognise that any function that predicts values larger than zero can be divided by its volume to be turned into a probability distribution.

For a given *energy function* $E_\theta(\mathbf{x}) : \mathbb{R}^D \to \mathbb{R}$, probability theory lets us normalise the energy scores for all possible inputs $\mathbf{x}_{train}$ for $\mathbf{x} \in \mathbb{R}^D$:

$$q_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z(\theta)}, \tag{1}$$

where $Z(\theta)$ is the normalising constant (i.e. partition function):

$$Z(\theta) = \int_x \exp(-E_\theta(\mathbf{x}))\, dx. \tag{2}$$

The exponential function in (1) enforces the assignment of a probability greater than zero to any sample, and the negative sign before $E_\theta(\mathbf{x})$ relates to the energy assignments: samples with low likelihood have high energy and samples with high likelihood have low energy. Since the *energy function* itself is not required to integrate to one, one may choose special-purpose architectures to parameterise it (e.g. CNNs and GNNs). However, for most choices of E_θ, it is impossible to compute or even estimate $Z(\theta)$, which results in the intractability of normalised densities estimation and in additional challenges to compute the maximum likelihood estimation (MLE) of parameters θ.

2.2 Training EBMs

The lack of stability around $Z(\theta)$ invalidates traditional training methods generally used in generative modelling. In order to create a stable training objective, we shift the training focus to compare the likelihood of points and adopt a modified MLE objective that aims to maximise the probability of $\mathbf{x}_{train}$ compared to the model's randomly sampled points:

$$\nabla_\theta \mathcal{L}_{\text{MLE}}(\theta; p) = -\mathbb{E}_{p(\mathbf{x})}[\nabla \log q_\theta(\mathbf{x})] = \mathbb{E}_{p(\mathbf{x})}[\nabla E_\theta(\mathbf{x})] - \mathbb{E}_{q_\theta(\mathbf{x})}[\nabla E_\theta(\mathbf{x})]. \tag{3}$$

With the adjusted MLE, we enable the dual training objective of maximising the energy for sampled points from our model and minimising the energy for data points from the training dataset. A thorough derivation of (3) is provided by [19], but the main idea is that rather than analytically computing the likelihood of an EBM, we leverage the likelihood maximisation with gradient ascent by using the gradient of the log-likelihood. Here, $p(\mathbf{x})$ is the true data distribution, and $q_\theta(\mathbf{x})$ is the distribution defined by the EBM. The first term can be computed directly from the data, but the second one is problematic because it requires expectations over the model distribution, which is, as previously seen, intractable due to the partition function $Z(\theta)$.

We summarise the training goal as maximising the numerator and decreasing the denominator in (1). Figure 1 illustrates the training intuition, where f_θ represents $exp(-E_\theta(\mathbf{x}))$. The "correct answer" point symbolises a data point from $\mathbf{x}_{train}$ and the "wrong answer" a sample from our model, $\mathbf{x}_{sample}$. Since the model is not normalised, simply maximising the un-normalised log-probability $exp(-E_\theta(\mathbf{x}))$ by changing θ does not ensure that $\mathbf{x}_{train}$ increases in likelihood. We must also account for the effect on other "wrong points". Therefore, EBM training implies "pulling up" the probability of data points in the dataset while simultaneously "pushing down" randomly sampled points, hence making $Z(\theta)$ small.

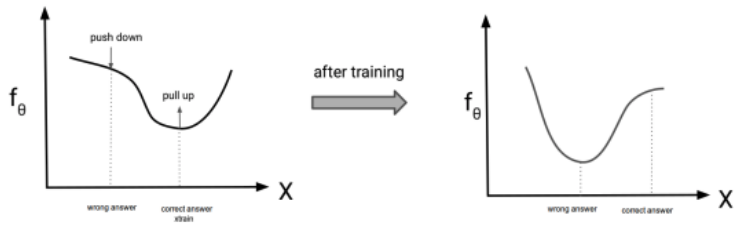

Figure 1: Illustration of EBM training objective. From [5].

2.3 Sampling from EBMs

Unfortunately, an additional constraint of EBMs is the difficulty in drawing samples from $q_\theta(\mathbf{x})$. Since the earliest EBMs [12], the Monte Carlo Markov Chain (MCMC) has been the *de facto* standard sampling technique [18]. Despite little development over the years, recent adaptations of EBMs for high-dimensional data successfully approximated the expectation in (3) using a sampler based on Stochastic Gradient Langevin Dynamics (SGLD) [22], as seen in Algorithm 1 [14]. It starts from a random point and uses the gradients of E_θ to slowly move towards the direction of higher probability. SGLD is incomplete until noise w is added to the sample at each gradient step to better capture the target probability distribution. Theoretically, if we execute for an infinite number of steps, we would be guaranteed to find an exact sample from the modelled distribution. In practice, we use a hyperparameter K to limit the number of steps.

Algorithm 1 Sampling from an EBM using Stochastic Gradient Langevin Dynamics. From [14]

1: Sample $\mathbf{x}^0$ from a Gaussian or uniform distribution;
2: **for** sample step $k = 1$ to K **do**
3: $\quad \mathbf{x}^k \leftarrow \mathbf{x}^{k-1} - \eta \nabla_{\mathbf{x}} E_\theta(\mathbf{x}^{k-1}) + \omega$, where $\omega \sim \mathcal{N}(0, \sigma)$ ▷ Generate sample via Langevin dynamics
4: **end for**
5: $\mathbf{x}_{\text{sample}} \leftarrow \mathbf{x}^K$

2.4 Contrastive Divergence (CD)

CD was introduced by [12] as an alternative to running MCMC until convergence, minimising the added complexity to the training time. It approximates the intractable term of (3), $\mathbb{E}_{q_\theta(\mathbf{x})}[\nabla E_\theta(\mathbf{x})]$, by using samples generated from SGLD. It starts by initialising the MCMC chain with a data sample $\mathbf{x}$ drawn from the true distribution $p(\mathbf{x})$. It follows by running SGLD for K steps to approximate a sample $\mathbf{x}_{sample}$ from $q_\theta(\mathbf{x})$. This will be an approximation from the model distribution but not a sample from the equilibrium distribution because the chain did not run long enough to converge. Once the gradients of the energy function with respect to θ for both $\mathbf{x}_{train}$ and $\mathbf{x}_{sample}$ are compute, the parameters θ are updated according to (3). A popular variant of CD is persistent CD [21], where a persistent state is introduced to the MCMC chain so that it does not have to be restarted when training on a new data point. In practice, we leverage a replay buffer to keep multiple states of the MCMC chain [4]. New chains are initialised by randomly sampling from the buffer.

Despite the efforts to stabilise EBM training, CD-based sampling does not successfully mitigate the risks of divergence when models assign a high probability to training examples from the true distribution $p(\mathbf{x})$, but the sampling algorithm continues to sample noise images. This means the model eventually created multiple local maxima where the noise images fall. The MCMC sampling is compromised when the energy function over which the calculated gradients have "diverged".

3 Methodology

This section details the architectural choices for the EBM and the model compression techniques applied. The EBM implementation and the compression techniques were carried out in PyTorch [15].

3.1 Energy Function

For the purpose of this paper, we mainly concentrated on the generative capabilities of EBMs, specifically training our models on image generation. Our choice of energy function reflected the nature of our task with the baseline model following an adjusted version of the ResNet18 architecture [10]. Simpler Convolutional Neural Networks were used as student models for knowledge distillation.

Activation Function We chose `Swish` as our activation function [16]. Its smoothness appeals to our use case since the gradients computed with respect to the input images should not be sparse. However,

even though it only requires arithmetic operations, `Swish` is more computationally expensive than `ReLu` because it involves computing the sigmoid function.

Removing `BatchNorm` Layers Removing the `BatchNorm` layers was motivated by the architectural choices of [7] where batch-normalisation is not included so that the models' outputs are deterministic functions of the input. However, since then, the authors proved that EBMs can be trained with batch-normalisation and other forms of scholastic regularisation.

Number of Output Dimensions Unlike traditional EBM implementations, we kept the ResNet logits as our final layer, not replacing it with a linear layer with a scalar output for the energy value. Instead, we derived the energy of the input by computing the mean of the logits across the labels' dimensionality. Considering a ResNet that outputs an unnormalised vector $\mathbf{z} \in \mathbb{R}^{10}$, the energy function can be represented as a function of these output dimensions: $E_\theta(\mathbf{x}) = f(\mathbf{z})$ where f aggregates these dimensions into a single scalar energy value. If f is the mean function, then it acts as a form of regularisation:

$$E_\theta(\mathbf{x}) = \frac{1}{10} \sum_{i=1}^{10} z_i. \tag{4}$$

The gradient of $E_\theta(\mathbf{x})$ is the average of the gradients of each dimension, implying that learning involves smoothing the gradients, averaging out noise and fluctuations from individual dimensions:

$$\nabla_\theta E_\theta(\mathbf{x}) = \frac{1}{10} \sum_{i=1}^{10} \nabla_\theta z_i. \tag{5}$$

Averaging the outputs can lead to a smoother and more convex energy landscape, facilitating easier optimisation and potentially leading to better convergence properties. Our experiments showed us that averaging the logits leads to a more balanced and stable learning process.

3.2 Sampler

As previously shown, for the CD's objective to be used to train the EBM, we require a way to generate samples during training. For the sampler, we used the persistent variant of CD alongside SGLD in an effort to minimise the number of iterations inside the MCMC sampling. The sampling buffer stores samples from previous batches to serve as starting points for the MCMC chain in the next batch. We combine the buffer samples with new ones initialised from scratch. 95% from the sampling buffer and 5% newly initialised for each batch. This helps reduce the overall sampling cost by decreasing the number of MCMC steps to converge to usable samples.

Training Algorithm The final version of our training algorithm is in Algorithm 2 [14]. It follows the training logic of persistent CD and an SLGD sampler with an added regularisation loss to enforce close to zero energy values for the real data and slightly lower for the sampled data.

3.3 Model Compression Techniques

3.3.1 Quantization

Quantization compresses models by reducing the precision of parameters and activations from 32-bit floats to lower bit-width integers like 8-bits [3]. It works by mapping the continuous range of floating-point values to a discrete set of values, effectively approximating the original distribution of weights and activations while minimising the information loss. Typically applied post-training or built into quantization-aware training, this allows faster inference and a smaller model footprint. Quantization trades some accuracy for improved efficiency through lower numeric precision and discretisation. When carefully tuned, performance loss can be minimised while gaining runtime speedups.

Algorithm 2 Training an energy-based model for generating images. From [14].

Initialize empty buffer $B \leftarrow \emptyset$
while not converged do **do**
  Sample data from dataset: $\mathbf{x}_i^+ \sim p_{\text{data}}$
  Sample initial fake data: $\mathbf{x}_i^0 \sim B$ with 95% probability, else $U(-1, 1)$
  for sample step $k = 1$ to K **do** ▷ Generate sample via Langevin dynamics
    $\mathbf{x}'_k \leftarrow \mathbf{x}'_{k-1} - \eta \nabla_{\mathbf{x}} E_\theta(\mathbf{x}'_{k-1}) + \omega$, where $\omega \sim \mathcal{N}(0, \sigma)$
  end for
  $\mathbf{x}' \leftarrow \Omega(\mathbf{x}'_K)$ ▷ Ω: Stop gradients operator
  Contrastive divergence: $\mathcal{L}_{\text{CD}} = \frac{1}{N} \sum (E_\theta(\mathbf{x}_i^+) - E_\theta(\mathbf{x}_i^-))$
  Regularization loss: $\mathcal{L}_{\text{RG}} = \frac{1}{N} \sum E_\theta(\mathbf{x}_i^+)^2 + E_\theta(\mathbf{x}_i^-)^2$
  Perform SGD/Adam on $\nabla_\theta(\mathcal{L}_{\text{CD}} + \alpha \mathcal{L}_{\text{RG}})$
  Add samples to buffer: $B \leftarrow B \cup \mathbf{x}^-$
end while

Unfortunately, quantization is not an ideal compressing technique for EBMs, as generative models require backpropagation during sampling. PyTorch's quantization implementation does not support gradients, as it targets inference-only usage. Thus, we focused quantization on the non-generative task of out-of-distribution detection, which solely uses energy function outputs. Without library support for fusing our custom `Swish` activation, we applied static quantization to minimise size and evaluated the impact. Quantizing generative tasks remains difficult since MCMC iterative sampling relies on backpropagating through the model. Overcoming this constraint would enable further compression of EBM sampling.

3.3.2 Pruning

Pruning selectively removes parameters from a neural network's architecture by setting them to zero [9]. Its goal is to reduce model complexity without significantly sacrificing its performance. It can be applied at various granularities, ranging from individual weights (unstructured pruning) to entire filters or channels (structured pruning). Both methods make use of filtering techniques such as l_1 norm, where $\|W\|_1 \leq \alpha$ for $\|W\|_1 = \sum_i \sum_j |w_{ij}|$. Pruned networks become more efficient and can generalize better by reducing overfitting, as the pruning process can be seen as a form of regularization [9]. For our EBM, pruning the energy function parameterised by the customised ResNet18 was done by structured and unstructured pruning applied both globally and to `Conv2d` layers only.

3.4 Knowledge Distillation

Knowledge distillation [11, 1] describes the process of training a smaller, more computationally efficient network - the student - to mimic the behaviour of a larger, pre-trained network - the teacher. The guiding principle is to transfer the rich knowledge encapsulated within the complex teacher network into the simpler student network. Since we deal with energy outputs, we use the Mean Squared Error (MSE) as the distillation loss to help the student:

$$L_{\text{distill}} = \frac{1}{N} \sum_{i=1}^{N} (E_{\text{teacher}}(x_i) - E_{\text{student}}(x_i))^2. \tag{6}$$

We experimented with six student models, from smaller ResNet architectures to even simpler CNN models. By minimising the MSE loss, the student network learns to approximate the teacher's energy function, effectively distilling its knowledge.

4 Experiments

4.1 Metrics

Unlike classifiers, generative models are not evaluated on accuracy. We used three key metrics to assess performance. each providing key insights into the model's capabilities.

Inception Score (IS) It takes a list of generated images and returns a single floating point number measuring their realism. The score reflects the variety and distinctiveness of the images, and it is calculated as $IS = \exp(\mathbb{E}_{x\sim p_g}[D_{KL}(p(y|x)||p(y))])$ where $p(y|x)$ is the conditional class distribution and $p(y)$ is the marginal class distribution. Higher IS values indicate clearer and more diverse images.

Frèchet Inception Score (FID) FID measures the similarity between generated and real images. It is computed as $FID = ||\mu_r - \mu_g||^2 + \text{Tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2})$ where μ and Σ are the mean and covariance of the feature vectors of the real (r) and generated (g) images. Lower FID scores denote better similarity between images.

Out-of-Distribution (OOD) Detection Also known as "anomaly" detection, it helps identify samples wrongly classified by overconfident deep learning models. A robust EBM should effectively identify images that do not conform to the training dataset distribution.

To compute the IS and FID scores, we trained a simple classifier. It's outputs (logits) form the basis of $p(y|x)$ in IS, and the extracted features from both real and generated images are used to compute the mean and covariance in FID. To test OOD detection, we generated transformed images by either adding noise, rotating them, or size reduction.

4.2 Baselines

Due to resource constraints, we evaluated the lower-dimensional MNIST dataset [13] to ease EBM training rather than a more complex high-dimensional colour image dataset. MNIST contains 28x28 grayscale images of handwritten digits. This simpler problem space of 10-digit classes means the energy landscape has reduced complexity. Our baseline EBM model, trained for 60 epochs over 2 days on an NVIDIA A100 GPU, achieved an FID score of 5,275, IS of 1.13, and successful in-distribution detection (OOD of zero).

4.3 Quantization

We applied activation and weight quantization to `int8` tensors to our baseline EBM. As previously mentioned, operator fusing was not applied because we used an unsupported activation function, `Swish`. Relative to the baseline, the post-training quantized model has a compression ratio of 11.9x and 50% faster inference time, as detailed in Table 1.

Table 1: Baseline vs Quantized Stats

Model	Model Size (MB)	Inference Time (s)
Baseline	134.1	0.0263
Static Quantized	11.3	0.0132

Table 2 summarises our results on testing the baseline and the quantized models on OOD detection for digits four, five, and nine. It shows the baseline model is more sensitive to noise, with quantized models less adept at noise detection. Both models struggle with rotated digits, particularly with 5 and 9, likely due to their visual ambiguity when inverted - an upside-down 5 resembling a 2 or an inverted 9 looking like a 6. Lastly, size reductions yield nearly zero scores from both models, indicating only

a marginal ability to recognize these as non-authentic images. The short range of OOD scores for the quantized model may stem from the architecture adjustments for quantization, where tensors must frequently convert between quantized and dequantized states for compatibility with `Swish`, potentially degrading its performance.

Table 2: OOD Detection Results

Model	Noise			Rotation			Reduction		
	4	5	9	4	5	9	4	5	9
Baseline	-0.45	-0.35	-0.21	-0.01	-0.01	0.00	-0.03	-0.04	-0.05
Static Quantization	-0.02	-0.02	-0.02	-0.01	-0.01	-0.01	-0.02	-0.02	-0.02

4.4 Pruning

Unstructured and structured pruning was applied both globally and to `Conv2D` layers only. Pruning ratios varied from 10% to 90%, and the impact on the FID, IS, and OOD scores was assessed. For the locally pruned models, we also recorded the scores for post-training with 7 further epochs.

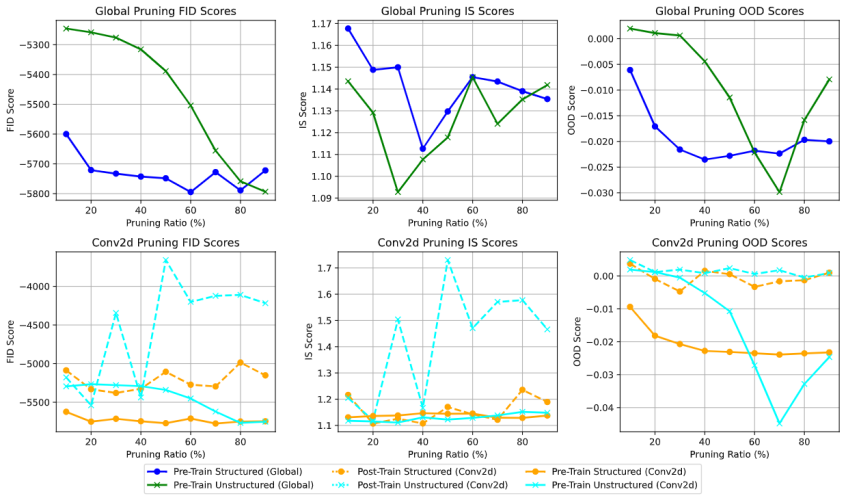

Figure 2: Results of different pruning strategies applied to the baseline model.

From Figure 2, an increased global pruning ratio significantly affects all three scores. The FID score under unstructured pruning declines more sharply yet maintains higher overall values, suggesting better image quality retention than structured pruning. Both methods exhibit volatility for the IS score, with notable drops at 30% and 40% pruning. With OOD scores, ideally, models should return zero since the test images are from the training set; however, unstructured pruning shows less stability but generally better performance over various pruning levels. When only pruning the `Conv2d` layers, the FID and IS scores show a stable pre-training pruning performance slightly superior to its global counterparts, with unstructured pruning maintaining the upper hand. The fine-tuned pruned models show increased performance even at the 80% and 90% pruning levels, maintaining close-to-perfect OOD scores. However, 7 epochs might not be sufficient for more consistent results since EBM training is highly unstable.

4.5 Knowledge Distillation

Distilling knowledge from the baseline teacher model into smaller, simpler student models aimed at verifying the hypothesis that these can leverage a shared representation learnt by the teacher to preserve performance. Due to computational constraints, we trained the student models for 30 epochs,

half the epochs used to train the baseline model. Figure 3 reveals that for medium-sized student models, this strategy is indeed efficient in preserving performance across all metrics. Despite their size, even the smallest student models show a competitive performance to most pruned models from Figure 2, suggesting that knowledge distillation may be a more effective method for model compression than pruning. For direct comparison to the quantized model, Figure 3 also reveals that the student models are either comparable or superior in performance at OOD for transformed images. These results indicate that the knowledge in the energy landscape provides a strong signal for teaching the student. Matching the regression-based energy outputs rather than discrete classes may enable better mimicry. Additionally, the smoothing from averaging across multiple output dimensions and the MSE distillation loss likely aid simpler students in replicating the teacher model. Further analysis is needed to confirm these factors explain the suitability of distillation for compressing EBMs.

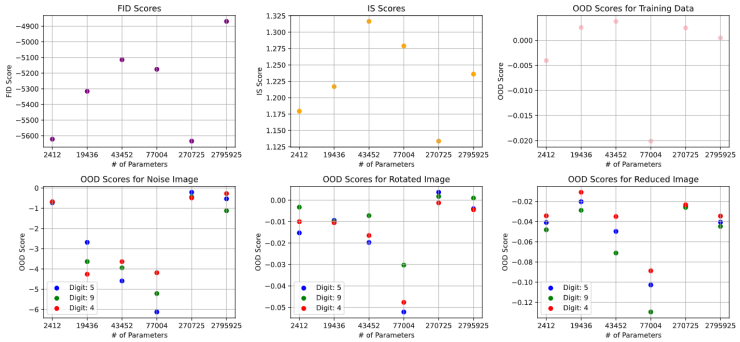

Figure 3: Results of Knowledge Distillation from the baseline model to simpler, smaller students.

5 Conclusion

Figure 4 contains twenty generated images from the baseline model, the top performing student model from Figure 3 (except for OOD on the training data), models with unstructured 60% pruning of `Conv2d` layers, both pre- and post-training. Unlike accuracy for evaluating classifiers, assessing the quality of generated images based on the scores used in this paper is not as straightforward. While useful, the FID and Inception Scores sometimes fail to capture the perceptual quality of generated images hence the need for visual inspection [2]. From the images, we note that the pre-training pruned model has clearer digits, which contradicts the thesis that the scoring dictates the unconditional best image generator. Our experiments concluded that the student model with 77,004 parameters (943 KB) is the most efficient for deployment in resource-constraint environments because of its reduced size, 145x fewer parameters than the baseline, and comparable performance with the larger alternatives.

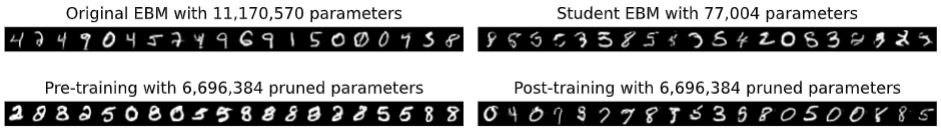

Figure 4: 20 generated images from baseline, pruned, and student models.

As previously mentioned, training EBMs is highly taxing from computational and time standpoints. To overcome challenges with model divergency, we had to reload our models multiple times from stabler and better-performing model checkpoints. In-depth hyperparameter tuning was out of the scope of this paper and further experimentation was constrained by the long training times. We hope to have successfully provided the reader with a first study on compressing EBM architectures. With additional time and resources, future work would expand into exploring joint energy models for concurrent classification and generation tasks [7], assessing their robustness under compression, and employing the "lottery ticket hypothesis" to identify optimally pruned architectures [23, 17].

References

[1] BACILUĂ, C., CARUANA, R., AND NICULESCU-MIZIL, A. Model compression. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining* (2006), Pages 535–541.

[2] BARRATT, S., AND SHARMA, R. A note on the inception score. *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Model* (2018).

[3] CHOUKROUN, Y., KRAVCHIK, E., YANG, F., AND KISILEV, P. Low-bit quantization of neural networks for efficient inference. *IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)* (2019), Pages 3009–3018.

[4] DU, Y., AND MORDATCH, I. Implicit generation and modeling with energy-based models. *33rd Conference on Neural Information Processing Systems* (2019).

[5] ERMON, S. Energy-based models.

[6] GOODFELLOW, I. J., MIRZA, M., XU, B., FARLEY-WARDE, D., POUGET-ABADIE, J., OZAIR, S., COURVILLE, A., AND BENGIO, Y. Generative adversarial nets.

[7] GRATHWOHL, W., DUVENAUD, D., WANG, K., AND SWERSKY, K. Your classifier is secretly an energy based model and you should treat it like one. *ICLR* (2020).

[8] GUO, Q., MA, C., JIANG, Y., YUAN, Z., YU, Y., AND LUO, P. Egc: Image generation and classification via a diffusion energy-based model.

[9] HAN, S., POOL, J., TRAN, J., AND DALLY, W. J. Learning both weights and connections for efficient neural networks.

[10] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. *Computer Vision and Pattern Recognition* (2015).

[11] HINTON, G., VINYALS, O., AND DEAN, J. Distilling the knowledge in a neural network.

[12] HINTON, G. E. Training products of experts by minimizing contrastive divergence. *Neural computation* (2002), Pages 1771–1800.

[13] LECUN, Y., BOTTOU, L., BENGIO, Y., AND HAFFNER, P. Gradient-based learning applied to document recognition. *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Model* (1998), Pages 2278–2324.

[14] LIPPE, P. Deep energy-based generative models.

[15] PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHEIN, N., AND ANTIGA. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* (2019).

[16] RAMACHANDRAN, P., ZOPH, B., AND LE, Q. V. Swish: A self-gated activation function.

[17] RAMANUJAN, V., WORTSMAN, M., KEMBHAVI, A., FARHADI, A., AND RASTEGARI, M. What's hidden in a randomly weighted neural network? *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2020).

[18] ROBERT, C., AND CASELLA, G. A short history of markov chain monte carlo: Subjective recollections from incomplete data. *Statistical Science* (2011), Pages 102–115.

[19] SONG, Y., AND KINGMA, D. P. How to train your energy-based models.

[20] TEH, Y. W., WELLING, M., OSINDERO, S., AND HINTON, G. E. Energy-based models for sparse overcomplete representations. *JMLR* (2003), Pages 1235–1260.

[21] TIELEMAN, T. Training restricted boltzmann machines using approximations to the likelihood gradient. *Proceedings of the 25th International Conference on International Conference on Machine Learning* (2008), Pages 1064–1071.

[22] WELLING, M., AND TEH, Y. W. Bayesian learning via stochastic gradient langevin dynamics. *Proceedings of the 28th International Conference on International Conference on Machine Learning* (2011), Pages 681–688.

[23] YEO, S., JANG, Y., SOHN, J., HAN, D., AND YOO, J. Can we find strong lottery tickets in generative models? *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence* (2023).