

Supervised Classification via Energy-Based Learning on CIFAR10 CIFAR100 and CELEBA

Wahyuningdiah Trisari Harsanti Putri^{1*}, Quintin Kurnia Dikara², Heryudi Ganesha³,
Deshinta Arrova Dewi⁴

^{1,2}Informatics Study Program, Faculty of Engineering Science, Universitas Paramadina, Jakarta, Indonesia

³Devoteam Indonesia, Jakarta, Indonesia

⁴Faculty of Data Science and Information Technology, INTI International University, Malaysia

ABSTRACT

Open Access

Type: Research Paper

Article:

Received: November 05, 2025

Revised: February 16, 2026

Accepted: July 13, 2026

Issued: October 30, 2026

© Putri et al. (2026).



This is an open-access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license

***Correspondence Address:**

wahyuningdiah.trisari@paramadina.ac.id

This research evaluates a Joint Energy Model (JEM) approach to perform supervised classification on the CIFAR10, CIFAR100, and CELEBA datasets. We implement a range of architectures, including custom shallow and deep CNNs and an adapted ResNet-18 modified with 3x3 first-layer convolution to prevent the loss of critical spatial information for small image classification. Following a rigorous experimental protocol involving multiple independent runs, we provide statistical validation via mean and standard deviation for all performance metrics. Our models achieved F1-scores exceeding 80% for both CIFAR datasets and a 90.34% accuracy of CELEBA multi-label classification. Beyond accuracy, we conducted a systematic analysis of the learned energy landscapes. While results confirm a systemic calibration failure in standard discriminative training for CIFAR objects, they reveal the emergence of a robust, globally coherent “face manifold” detector for the CELEBA dataset. These findings quantify the calibration limitations of standard classifiers and demonstrate how post-hoc energy interpretation can reveal the underlying structure of learned visual representations.

Keywords: *Energy Based Model, JEM, machine learning, vision, CIFAR, CELEBA*

1. INTRODUCTION

The field of discriminative and generative artificial intelligence (AI) has experienced rapid progress in the past decades. Areas that have made progress include computer vision [1], [2], [3], [4], [5], [6], natural language processing (NLP) [7], [8], [9], [10], [11], and reinforcement learning [12], [13]. Current AI systems are proficient in narrow domains but struggle with generalization, adaptation to new situations, and common-sense reasoning, abilities that are fundamental to human intelligence. AI vision models can produce beautiful images, but do not understand that the trajectory of a thrown ball will form a parabolic curve due to the influence of gravity. These models generally struggle in physical reasoning, so that during inference they exhibit nonsensical output due to their disconnection to the real world [14], [15], [16]

Beforehand, LeCun offers a completely different approach [17]. LeCun argues that the inability of current AI models in various generalization tasks is because existing models do not have an internal representation of how the world works. The same idea has been put forward by Ha and Schmidhuber in their paper about world models [18]. Furthermore, LeCun proposes an approach using an energy-based model (EBM) as an effort to enable AI models to have a representation of the world, by applying it to a framework consisting of several modules. The framework proposed by LeCun itself is very complete and complex, consisting of several modules such as the configurator module, perception module, world model module, cost module, short-term memory module, and actor module.

In this research, only a small part of this big idea is researched, namely a vision system with EBM. We aimed in solving the need for structural internal representation to improve generalization and calibration in vision tasks. We hope to understand how vision model capture the underlying structure of the world in a way that standard classifiers do not.

EBM is inspired by statistical physics, a model that is designed to capture complex relationships in data by assigning a scalar value called "energy" to each possible configuration of the input data and parameters in the model. Low energy levels are assigned to desired or

possible configurations, while high energy levels are assigned to undesirable configurations. The goal of this energy-based model is to learn an energy function that captures the underlying structure of the data set. For example, in the task of classifying a cat, EBM would use an energy function to model the relationship between the image itself and the labels in each specified category. EBM would assign low energy values to images of cats, and high energy values to other images. The energy function used uses the Boltzmann distribution as follows [19]:

$$P(x, y) = \frac{e^{-E(x, y)}}{Z} \quad (1)$$

$$Z = \sum_{y'} e^{(-E(x, y'))} \quad (2)$$

Z is the global partition function, a summation (or integral) of the un-normalized probability over all possible configurations of x and y' . During training, the model able to learn data energy landscape, therefore EBM can catch the underlying structure of data, just like how human recognize a cat based of their characteristics (features).

We proposed to build a standard discriminative classifier with post-hoc energy interpretation model capable to perform classification task on CIFAR10, CIFAR100, and CELEBA dataset. Unlike fully generative EBM, our model is optimized purely via discriminative cross-entropy.

2. METHODS

We spent a lot of our time in literature review on EBM. Followed by data acquisition and cloud environment setup. The platform of our choice is the Google Cloud Platform (GCP) Southeast Asia region. We chose the server location based on our geographical proximity to ensure low-latency training. Our experiments ran on hardware and libraries as follow:

Table 1. Hardware and Libraries Setup

Aspect	Information
Compute Engine	g2-standard-4 (4 vCPUs, 16 GB Memory)
GPU	1 x NVIDIA L4
Storage	250 GB Boot and Data Disk
Libraries	torch; torchvision; matplotlib; numpy; pillow

2.1 Architecture and Training Setup

As a preliminary baseline, we implement a shallow convolutional neural network trained via standard supervised learning, then we updated the shallow convolutional layer into deep layer. The model was trained using supervised Maximum Likelihood Estimation (MLE) of the conditional distribution. Let $f_{\theta}(x) \in \mathbb{R}^K$ denote the logit vector produced by the network for an input image x over K classes. The model is trained to maximize the conditional log-likelihood $p_{\theta}(y|x)$ using standard Cross-Entropy loss[20]:

$$\mathcal{L}(\theta) = -\sum_{(x,y) \in \mathcal{D}} \log \left(\frac{e^{f_{\theta}(x)[y]}}{\sum_{j=1}^K e^{f_{\theta}(x)[j]}} \right) \quad (3)$$

We chose a Convolutional Neural Network (CNN) architecture for image classification designed as an energy-based model. This architecture is tailored to train EBM, with the following explanation. The input layer is a 3x32x32 image from the CIFAR10 dataset with 3 channels (RGB). We employed two convolutional blocks. For the first block, we increased the 3 channels to 64, with 3x3 kernels, and kept the spatial dimension at 32x32 pixels. Batchnorm stabilized the training process and accelerated convergence, ReLU added nonlinearity, and Maxpool reduced the size to 16x16 pixels. The output of the first block was a feature map with 64x16x16 channels. For the second block, we increased the 64 channels to 128 to increased feature complexity, and then Maxpool reduced the size to 8x8 pixels. The output of the second block was a feature map with 128x8x8 channels. An overview of our architecture is as follows:

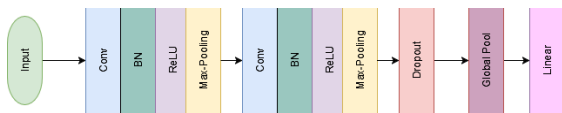


Figure 1. Shallow CNN Architecture

The third block, the channel size was doubled to capture the highest-level features. Regularization was also applied to prevent overfitting by randomly deactivating 30% of the neurons. Feature aggregation performed by reducing the 8x8 feature map to 1x1 (global average). Then, equalization performed by converting the 256x1x1 feature map into a 256-

dimensional vector. The output was 256 features. The energy head for classification was performed by mapping the 256 features to the energy scores assigned to `num_classes`. For EBM, low energy indicates a high probability for that class. Below is our code for the EBM model architecture:

Code Snippet 1. Shallow CNN Architecture

```
# Model Architecture
class EBVisionModel(nn.Module):
    def __init__(self, num_classes=10):
        super().__init__()
        self.backbone = nn.Sequential(
            nn.Conv2d(3, 64,
kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(64),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),

            nn.Conv2d(64, 128,
kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(128),
            nn.ReLU(),
            nn.MaxPool2d(2, 2),

            nn.Conv2d(128, 256,
kernel_size=3, stride=1, padding=1),
            nn.BatchNorm2d(256),
            nn.ReLU(),
            nn.Dropout(0.3),

            nn.AdaptiveAvgPool2d((1, 1)),
            nn.Flatten()
        )
        self.energy_head = nn.Linear(256,
num_classes)

    def forward(self, x):
        return
self.energy_head(self.backbone(x))
```

In traditional classifiers, features imply probabilities, while in EBM, features are energies. A comparison of EBM with standard CNNs is as follows:

Table 2. Comparison between CNN and EBM.

Aspect	Standard CNN	EBM
Output	Logits	Energies
Loss	CrossEntropy(logits, y)	CrossEntropy(-energies, y)
Interpretation	Higher logits = more likely	Lower energy = more likely

This architecture worked due to several reasons, including its progressive complexity, started with a small number of channels 3 and gradually increased to 256. Furthermore, it utilized a spatial hierarchy from 32x32 down to 16x16 and 8x8, ensuring captured patterns at different scales. It also utilized regularization to prevent overfitting.

We updated the architecture to train with the CELEBA dataset. The architecture changed

into 5-layer CNN with Binary Cross-Entropy (BCE) with Logit Loss. For CIFAR100, we switched to ResNet-18 pretrained on ImageNet. Table 3 below list the comparison of our model training, the architecture, the parameters used, and the EBM energy formulation used:

Table 3. Training Comparison of CIFAR10, CIFAR100, and CELEBA

Dimension	CIFAR 100	CIFAR 10	CELEBA
Description	Transfer learning with Post-Hoc JEM interpretation	Custom CNN classifier with energy parameterization	Deep custom CNN to output independent energy values for facial attributes
EBM Energy Formulation	Marginal energy (Eq.4)	Marginal energy where model outputs raw energies (Eq.5)	Joint marginal energy (Eq.8)
Loss Function	Cross-Entropy Loss with Label smoothing (0.1)	Cross-Entropy Loss with Label smoothing (0.1) applied to negative energies	Binary Cross-Entropy (BCE) with Logits Loss
Model Architecture	Adapted ResNet-18 Pretrained on ImageNet	Custom shallow model with 3 CNN layers	Custom deep model with 5 CNN layers
Optimizer	AdamW	Adam	Adam
Optimizer	Default	Default	Custom
Betas	PyTorch AdamW betas	PyTorch Adam betas	GAN-style betas
Learning Rate	1×10^{-3}	1×10^{-3}	2×10^{-4}
Weight Decay	5×10^{-4}	1×10^{-5}	None
Batch Size	128	128	128
Max Epoch	200	50	10
Experimental Seeds	[42, 1337, 2026]	[71, 233, 1597]	[42, 1337, 2026]
Evaluation Metrics	Test accuracy, Macro Precision, Recall, F1-Score, Mean & Standard Deviation across runs	Test accuracy, Macro Precision, Recall, F1-Score, Mean & Standard Deviation across runs	Attribute accuracy, Macro Precision, Recall, F1-Score, Per-attribute Mastery Profile, Mean & Standard Deviation

2.2 Joint Energy Model for Energy Landscape

The last part of our method was to plot the learned energy landscape of our model. To do that, we must first bridge the gap between

our discriminative classifier and EBM. As established in Grathwol et al seminal paper [21], any standard K -class classifier can be re-interpreted as a JEM. We had to define how the model's output whether it was logits or raw energies map to the probability density of the data.

With our CIFAR100 experiment, we used a standard adapted ResNet-18. The model outputs a vector of standard classification logits $f_\theta(x) \in \mathbb{R}^{100}$, where $f_y(x)$ is the logit for class y . In the JEM framework, the joint energy of an image x and a class label y is defined as the negative logit of that class:

$$E(x, y) = -f_y(x)$$

This meant that when the model was highly confident that the image x belonged to class y , the joint energy $E(x, y)$ is highly negative or low energy. To find the overall plausibility of the image x across all 100 classes, we marginalize over the label space y :

$$p(x) = \sum_{y=1}^{100} p(x, y) = \frac{\sum_{y=1}^{100} e^{-E(x, y)}}{Z}$$

$$= \frac{\sum_{y=1}^{100} e^{f_y(x)}}{Z}$$

Since $p(x) = \frac{e^{-E(x)}}{Z}$ we equate the numerators:

$$e^{-E(x)} = \sum_{y=1}^{100} e^{f_y(x)}$$

Taking the negative natural logarithm of both sides yields the final marginal energy function for CIFAR100:

$$E(x) = -\log \sum_{y=1}^{100} e^{f_y(x)}$$

$$= -\text{LogSumExp}_y(f_y(x)) \quad (4)$$

In our CIFAR10 experiment, we used custom CNN where the outputs were explicitly treated as raw energies. In the training loop, we passed `-energies` as the logits to the Cross-Entropy loss as seen in Code Snippet 2 below. Because the negative of our model's raw output energies acted as the logits, we have:

$$f_y(x) = -\text{energies}_y \Rightarrow E(x, y) = \text{energies}_y$$

Here, the model directly outputted the joint energy for each class y . To find the overall marginal energy $E(x)$, we marginalized again over all 10 classes, and came to the same result as equation 4:

$$\begin{aligned} E(x) &= -\log \sum_{y=1}^{10} e^{-\text{energies}_y} \\ &= -\text{LogSumExp}_y(-\text{energies}_y) \end{aligned} \quad (5)$$

For our CELEBA training, which is a multi-label task with 40 independent binary attributes, we used BCE with logits defined as:

$$f_i(x) = -E_i(x) \quad (6)$$

For a single attribute i , the label y_i is binary ($y_i \in \{0,1\}$). the joint probability of the image x and the binary label y_i is:

$$p(x, y_i) = \frac{e^{y_i f_i(x)}}{Z}$$

To find the marginal energy of the image x with respect to this single attribute, we marginalized over the two possible states of y_i (present of absent):

$$\begin{aligned} p(x) &= \sum_{y_i \in \{0,1\}} p(x, y_i) = \frac{e^{0 \cdot f_i(x)} + e^{1 \cdot f_i(x)}}{Z} \\ &= \frac{1 + e^{f_i(x)}}{Z} \end{aligned}$$

Equating this to $p(x) = \frac{e^{-E_i(x)}}{Z}$ gave the free energy for attribute i :

$$E_i(x) = -\log(1 + e^{-f_i(x)})$$

Since our model's logits are defined as equation (6) above, meaning that the $E_i(x)$ is the raw output, this became:

$$E_i(x) = \log(1 + e^{-E_i(x)}) \quad (7)$$

Because the 40 attributes were treated as independent binary variables under the BCE loss, the joint probability of the image x across all attributes is the product of their individual marginal probabilities:

$$p(x) = \prod_{i=1}^{40} p_i(x) = \prod_{i=1}^{40} \frac{e^{-E_i(x)}}{Z_i} = \frac{e^{-\sum_{i=1}^{40} E_i(x)}}{\prod Z_i}$$

This gave us the final Joint Marginal Energy Function for CELEBA, which was the sum of the individual free energies:

$$E_{\text{joint}}(x) = \sum_{i=1}^{40} E_i(x) = -\sum_{i=1}^{40} \log(1 + e^{-E_i(x)}) \quad (8)$$

Crucially, because our models were trained purely discriminately without contrastive phases such as Stochastic Gradient Langevin Dynamics (SGLD) [22] or Score Matching [23], the resulting energy landscape is unconstrained in out-of-distribution directions. This baseline serves to highlight the calibration limitations of standard classifiers and motivates the necessity of explicit generative EBM training for physically consistent result.

3. RESULTS AND DISCUSSION

The training process went on with varying dynamics for each dataset. We name the EBM models based on the success-failure iterations of the experiment. For example, the EBM001_CIFAR10 model represents our base model with the CIFAR10 dataset. If there is a minor change, the iterations are appended with the letters A, B, C, and so on. When there is a major change, the number identifier in the file name is increased.

Furthermore, in this study, we came to understand that in training EBM for classification tasks, the intractable Z partition function can be avoided altogether. For classification task, we did not need to know the probability of a specific image appearing $P(x)$, we only need to observe that given this image x , what is the probability of the label being y' ? This is exactly conditional probability $P(y|x)$ as denoted by equation (5). Therefore, we can derive $P(y|x) = P(x, y)/P(x)$ from equation (1) and we thus substitute the values for EBM classification task:

$$P(y|x) = \frac{P(x, y)}{P(x)} = \frac{\frac{e^{-E(x, y)}}{Z}}{\frac{\sum e^{-E(x, y')}}{Z}} \quad (9)$$

$$P(y|x) = \frac{e^{-E(x, y)}}{\sum e^{-E(x, y')}} \quad (10)$$

Notice here that the intractable global partition function Z appears in both the numerator and the denominator in equation (9), and they cancel out perfectly [24] as shown in equation (10). The cancellation proves that Cross-Entropy Loss is mathematically sound for our model. In the training code, we implemented the equation (10) via the SoftMax function through class `nn.CrossEntropyLoss`. Here is the implementation code (partial):

Code Snippet 2. Negative Energy as Loss

```
...
loss_function =
nn.CrossEntropyLoss(label_smoothing=0.1)
...
Energies = model(images)
Loss = loss_function(-energies, labels)
...
```

This class takes logits as input, and in our code, we used `-energies` as logits. So, by passing negative energies to the `CrossEntropyLoss` class, we implicitly told PyTorch to calculate the conditional probabilities for the 10 classes in the CIFAR10 dataset. We then find the loss by maximizing the log-probability of the correct class.

3.1 Training on CIFAR10, CELEBA, and CIFAR100 Dataset

Our training started with CIFAR10 dataset. It showed desirable accuracy on the third minor iteration, the 001C model. Here is the statistical result of our training on the 50th epoch.

Table 4. Statistical Result of CIFAR10 Training

Aspect	Results for 3 Runs
Seeds	[71, 233, 1597]
Accuracy	86.54% $\pm$ 0.35%
F1-Score	86.47% $\pm$ 0.35% (Macro)
Precision	86.50% $\pm$ 0.31% (Macro)
Recall	86.54% $\pm$ 0.35% (Macro)

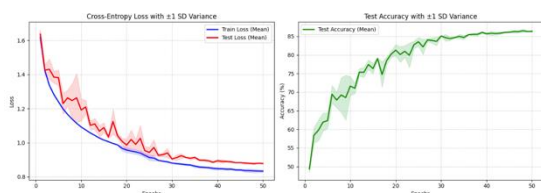


Figure 2. Loss and accuracy over epochs for CIFAR10 dataset.

Our next training was CELEBA dataset with 001C model. While CIFAR10 consists of small, diverse images of everyday objects categorized into 10 classes, CELEBA contains large, high-resolution images of celebrity faces with various attributes. This dataset has 40 binary attributes for each image, which means it faces a multi-label classification problem. We had to employ changes to train CELEBA.

We update the architecture to 5 CNN layers, then resized the image to a consistent square shape of 64x64 pixels since they were originally rectangular. Then, we switched the `CrossEntropyLoss` meant for single-label problems to Binary Cross Entropy with logits for multi-label problem. We also switched the `torch.argmax` accuracy metric with the sigmoid of the logits and then compared against all 40 true labels. Our model's final layer now output `num_attributes(40)` instead of `num_classes`.

The training went as intended. Two out of three training runs for CELEBA our model had early stopped at epoch 9. Here is the statistical result.

Table 5. Statistical Result of CELEBA Training

Aspect	Results for 3 Runs
Seeds	[42, 1337, 2026]
Accuracy	90.34% $\pm$ 0.02%
F1-Score	64.28% $\pm$ 0.53% (Macro)
Precision	73.42% $\pm$ 0.26% (Macro)
Recall	59.47% $\pm$ 0.51% (Macro)

The true measure of our model's performance is the Macro F1-Score. We calculated each of the 40 attributes individually and then take the average. The value of 64.28% meant that our model treated rare attributes like Bald and common attributes like Young with equal weight.

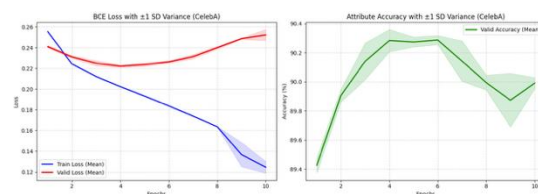


Figure 3. Loss and Accuracy over Epoch for CELEBA Dataset

As for the variance, across our 3 independent runs it showed $\pm 0.02\%$ and $\pm 0.53\%$ which meant it is virtually non-existent to extremely low. These numbers proved that our model performance is due to the stable training pipeline.

The statistical result demonstrated that the model architecture, loss function (BCEWithLogitsLoss), and training setup were well-suited for the problem. Early stopping worked as intended. Our model has a solid baseline, but there is significant room for improvement.

For attribute mastery profile, we had shown below on Figure 4 the result from the first run. We ranked them based on the highest value of F1-Score, Precision, and Recall down to the lowest. The high tier that our model mastered with F1 value over 85% are the ones with global contrast and strong geometry. Attributes like Male, Wearing_Lipstick, and Heavy_Makeup are not confined to a few pixels, they alter the global color distribution and texture of the entire face. The network easily captures these low frequency color and texture shifts.

Eyeglasses and Smiling have highly distinct, sharp, high-contrast geometric shapes as in the dark frames of glasses, the white curve of teeth or lips. Even at low 64x64 resolution, these shapes do not get blurred out. There is also the factor of spurious correlations. The attributes Male and No_Beard are highly correlated in the dataset, as are Wearing_Lipstick and Heavy_Makeup. The model learns to use these joint distributions to boost its confidence, resulting in exceptionally high F1-scores.

The low tier are the ones that our model struggle to master with F1-score below 60%. Attributes like Wearing_Necklace (12.01%) is one such example. A necklace is only depicted as a fine thin line of pixels. At 64x64 resolution, after passing through four downsampling convolutional layers (strides of 2), a necklace is reduced to less than a single pixel of information. It is physically impossible for our network to detect it because the information was discarded during downsampling.

Rank	Attribute Name	F1-Score	Precision	Recall
1	Male	97.47%	96.50%	98.46%
2	No_Beard	96.40%	94.93%	97.93%
3	Mouth_Slightly_Open	92.46%	93.27%	91.65%
4	Smiling	91.66%	94.98%	88.56%
5	Wearing_Lipstick	91.23%	87.61%	95.17%
6	Young	90.71%	88.81%	92.70%
7	Eyeglasses	89.69%	89.43%	89.95%
8	Heavy_Makeup	88.58%	83.99%	93.69%
9	High_Cheekbones	85.73%	89.02%	82.68%
10	Blond_Hair	82.18%	76.58%	88.68%
11	Bangs	81.70%	79.53%	83.98%
12	Attractive	81.07%	78.31%	84.03%
13	Wearing_Hat	79.12%	93.60%	68.51%
14	Gray_Hair	71.22%	77.66%	65.77%
15	Bushy_Eyebrows	70.13%	77.96%	63.72%
16	Black_Hair	69.43%	83.31%	59.51%
17	Goatee	69.12%	75.69%	63.59%
18	Wavy_Hair	68.47%	63.56%	74.19%
19	S_o_Clock_Shadow	67.36%	67.05%	67.68%
20	Brown_Hair	65.00%	68.01%	62.24%
21	Arched_Eyebrows	63.58%	72.44%	56.66%
22	Wearing_Necktie	62.18%	69.37%	56.34%
23	Bald	61.52%	66.57%	57.18%
24	Sideburns	60.26%	80.54%	48.13%
25	Rosy_Cheeks	56.91%	68.79%	48.53%
26	Wearing_Earrings	56.85%	76.28%	45.32%
27	Big_Nose	55.37%	69.90%	45.84%
28	Pale_Skin	55.30%	51.51%	59.70%
29	Receding_Hairline	51.70%	61.27%	44.72%
30	Chubby	45.15%	62.94%	35.20%
31	Double_Chin	45.00%	62.98%	35.08%
32	Pointy_Nose	44.72%	62.99%	34.66%
33	Bags_Under_Eyes	43.27%	66.47%	32.08%
34	Narrow_Eyes	41.00%	55.90%	32.37%
35	Straight_Hair	39.82%	60.74%	29.62%
36	Oval_Face	39.15%	60.78%	28.88%
37	Big_Lips	38.78%	39.47%	38.11%
38	Mustache	32.94%	74.65%	21.14%
39	Blurry	23.76%	71.28%	14.26%
40	Wearing_Necklace	12.01%	56.49%	6.72%

Figure 4. Per-attribute accuracy on CELEBA dataset.

There is also the matter of subjectivity and label noise. Attributes such as Double_Chin or Big_Lips are human-annotated labels that have no clear mathematical decision boundary. When the ground truth labels are inconsistent, then it is expected that our model performed poorly.

From the low tier, we also found that our model is conservative. We derived this conclusion by analyzing the precision versus recall numbers. For example, the attribute Mustache has Precision value of 74.65% and Recall value of 21.14%, the attribute Blurry with similarly high precision of 71.28% and poor Recall of 14.26% and Wearing_Necklace with consecutive value of Precision and Recall at 56.49% and 6.72%. Because these features are rare, our model has learned that predicting “1” or present is highly risky. Therefore, it only predicts Mustache or Blurry when it was certain which led to high Precision. However, our model missed almost 80% of the actual mustaches in the dataset, which led to a poor Recall.

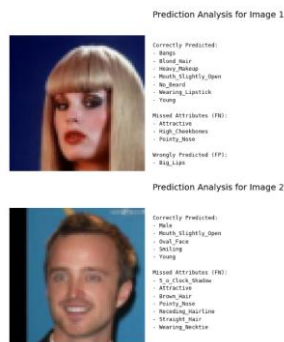


Figure 5. Prediction Sample on CELEBA Dataset

The final training dataset was CIFAR100. Unlike the previous two datasets, this one presented quite an interesting challenge for us to solve. Most of the model training time was spent on this dataset because we repeatedly fell short of the expected accuracy level. In the first iteration, we only achieved an accuracy of 62.53%.

We switched from our baseline EBM001C model and replaced it with the Residual Network (ResNet) architecture, a deep learning architecture that allows modelling residual mapping as the difference between input and output, instead of direct mapping. Residual mapping allows for better backpropagation and training networks consisting of hundreds or even thousands of layers [25]. We also improved data augmentation with AutoAugment. To speed up the convergence process, we also used the OneCycleLR learning rate, which increased the learning rate as a warm-up, reached a peak, and then slowly decreased the learning rate. With the various efforts above, we managed to increase the accuracy to 78.14%.

To achieve higher accuracy, we used a deeper model architecture, ResNet-18. We used mixup and cutmix regularization techniques for data augmentation, forcing the model to learn more complex features by training on images that were combinations of other images. We used a longer training run of 250 epochs and a better scheduler. However, the accuracy decreased to 76%.

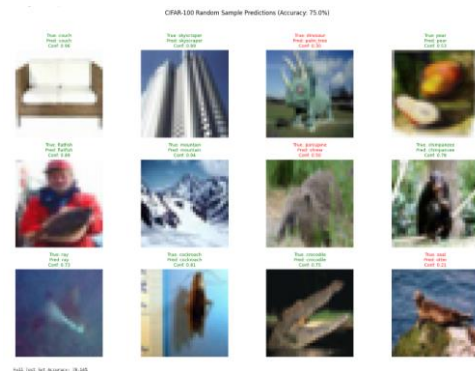


Figure 6. Prediction sample on CIFAR100 dataset.

We tried changing the model architecture, using a deeper ResNet and a Squeeze-Excitation (SE) block, which has been shown to significantly improve CNN performance [26]. We also changed the training strategy to three iterations. The first iteration ran 150 epochs to improve accuracy by 2-3%. The second iteration ran 150 epochs with a cosine learning rate, which balances exploration and exploitation toward model convergence [27]. We used Stochastic Weight Averaging (SWA) to prevent the model from getting stuck in local minima [28] after epoch 100. We tracked the SWA and regular models, saving the best-performing model.

The third iteration used an ensemble model by combining predictions from the trained models, using a technique of averaging logits over negative energy. Knowledge distillation, when the ensemble model is used as a reference for learning a new model, occurs. Essentially, our pipeline consisted of training three baseline models, creating, and evaluating a combination of these three baseline models to create a student model. Despite having carried out quite a long and complicated training technique, the accuracy of the final distilled model worsened to 69.36%.

[illegible]

Figure 7. First iteration model accuracy on CIFAR100 dataset.

[illegible]

Figure 8. *Second iteration model accuracy on CIFAR100 dataset.*

However, as can be seen in the image, the third iteration model also failed to achieve the desired accuracy of 80%. After distillation on all three models, the total accuracy dropped to 69.36%. After deliberation with the team member, we suspected that we might have reached a performance limit for our model, a ResNet built from scratch. Our model, starting with completely random weights, had to learn everything about the visual world such as what constitutes an angle, texture, curvature, all the way to cat fur, all these based on the relatively small CIFAR100 dataset.

Therefore, we decided to transition to a model that has already learned and proven results. We used the pre-trained ResNet-18. This model was trained on 1.2 million images from the ImageNet dataset with 1,000 classes. We will adapt this model for use on the CIFAR100 dataset. The adaptation step we took was to load the pre-trained ResNet-18 model directly through `torchvision.models`, complete with training weights (expert features) from ImageNet. We will remove the last layer and replace it with the output layer for 100 CIFAR100 classes. We fine-tuned this hybrid model by "freezing" the initial layers (which recognize basic shapes and textures) of the model or training them with a very small learning rate, while the final output layer learned to map expert features to the 100 CIFAR100 classes. Unfortunately, this approach was still unsuccessful, the accuracy rate was only 53.51%.

```

4.5e-08
Epoch [78/80] | Train Loss: 2.5313 | Test Loss: 2.2657 | Test Acc: 53.24% | LR:
2.0e-08
Epoch [79/80] | Train Loss: 2.5297 | Test Loss: 2.2657 | Test Acc: 53.51% | LR:
5.0e-09
Accuracy improved: 53.47% -> 53.51%. Saving model to
EBM_03_CIFAR100_TransferLearning/cifar100_transfer_best_model.pth
Epoch [80/80] | Train Loss: 2.5288 | Test Loss: 2.2638 | Test Acc: 53.29% | LR:
0.0e+00

```

Training complete. Final best model test accuracy on CIFAR-100: 53.51%

Figure 9. Naive transfer learning on pre-trained ResNet-18 architecture.

We went back to deliberation and later found out that the problem lies on the pre-trained ResNet-18 architecture itself. This architecture was designed for 224x224 pixel images from ImageNet. The first layer of this architecture is a 7x7 convolution with a stride of 2. This is followed by a pooling layer with a stride of 2. This combination proved very aggressive, down sampling a factor of 4 from a 32x32 pixel CIFAR100 input, meaning it went straight from 32x32 to 8x8. For small images like CIFAR, this was disastrous. This meant 75% of the spatial information was discarded before the main ResNet block even saw it. Our model was trying to classify objects based on blurry 8x8 thumbnails. The architecture's pre-trained weights were missing the input details needed to work effectively.

After understanding our mistakes, we fixed them with a `3x3 model.conv1` layer with a stride of 1. We also removed the initial pooling layer, replacing it with a `nn.Identity` layer that does nothing. The results of these improvements were immediately visible. Our model ultimately achieved an F1-Score of over 80 percent. Here is the statistical result of our training.

Table 6. Statistical Result of CIFAR100 Training

Aspect	Results for 3 Runs
Seeds	[42, 1337, 2026]
Accuracy	80.29% $\pm$ 0.18%
F1-Score	80.28% $\pm$ 0.17% (Macro)
Precision	80.48% $\pm$ 0.15% (Macro)
Recall	80.29% $\pm$ 0.18% (Macro)

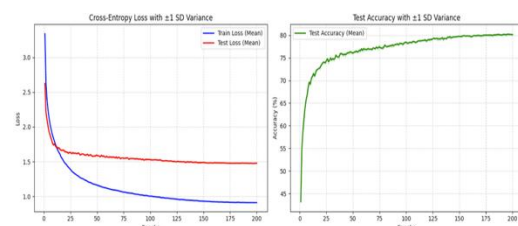


Figure 10. Loss and accuracy over epoch for CIFAR100 dataset.

3.2 Visualizing Energy Landscape

The plot to the energy landscape is shown below on Figure 11 and Figure 12 for our model trained on CIFAR10 and CIFAR100 respectively. What we expected to see is the red dot reside in the lowest basin of the energy landscape, marked by the bright yellow contour.

But as we can see, the image did not reside there. This is because our model was trained as a standard discriminative classifier. It only learned to draw decision boundaries to separate classes. It cared only for the relative difference between logits. Our model assigned lower energy or higher probability to a perturbed image rather than the real image.

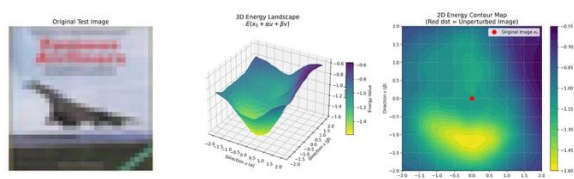


Figure 11. Energy landscape for CIFAR10 Dataset

Our interpretation over the fact is because our model was never trained on negative samples such as fake or noisy images, it has never learned to “push up” the energy of out-of-distribution or perturbed space. If we looked closely, the test image on the left is not a clean, prototypical CIFAR10 plane. It has bright red text at the top and a dark border at the bottom. A standard CIFAR10 plane is typically a clean object silhouetted against a blue sky or sitting on a grey runway.

The same thing happened with our model trained with CIFAR100 dataset. The real image did not reside in the lowest energy area. This is a validation of our experimental setup. The two different datasets with two different architectures (our custom CNN and adapted ResNet-18) proved that this was a systemic, mathematical property of discriminative classifiers, not a random anomaly or a dataset specific fluke

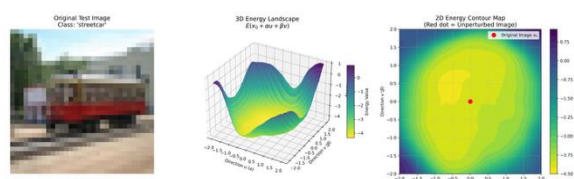


Figure 12. Energy landscape for CIFAR100 Dataset

While the phenomenon is the same, the underlying mechanics are even worse in CIFAR100 due to two factors, the tenfold number of classes increase, and the tightly packed decision boundaries in the high dimensional representation space. Below is the comparison table of both model’s energy landscape:

Table 7. Comparison of CIFAR10 and CIFAR100 Energy Landscape

Metric	CIFAR10	CIFAR100
Unperturbed State $x_0(0,0)$	Reside in a higher energy green region (≈ -1.15)	Reside in a higher energy greenish-yellow region (≈ -3.75)
Global Minimum (Basin)	Located at $(0, -1.0)$ in a deep yellow valley (≈ -1.60)	Located around $(-0.5, -0.5)$ in a deep bright yellow valley (≈ -4.50)
Energy Gap (ΔE)	(≈ 0.45) energy units of uncalibrated overconfidence	(≈ 0.75) energy units of uncalibrated overconfidence

In contrast to the previous energy landscape of our models, the CELEBA trained model was showing a different result. The red dot sat around the lowest energy basin. The single biggest driver of this difference is the transition from SoftMax Cross-Entropy which we used in CIFAR to Binary Cross-Entropy (BCE) used in CELEBA.

SoftMax is a competitive loss function, which pressured the model to extrapolate wildly. If a perturbation made the image noisy but pushed it deeper into the area of decision boundary relative to other classes, the logits exploded, and energy plummeted. This created the uncalibrated yellow basins away from the red dot.

Figure 13 below was the output of our joint mode energy landscape for CELEBA. Unlike the previous two models, this time the red dot sat at the absolute bottom of the lowest basin. Our interpretation of it was that our model has not just learned isolated attributes, rather **it has implicitly constructed a highly robust, globally coherent “face manifold” detector**. While this interpretation is still open to contestation, we conclude that this was due to three factors, (1) the mathematical synergy of summing independent BCE heads; (2) global feature destruction cascade; and (3) the inactive attributes stayed quiet.

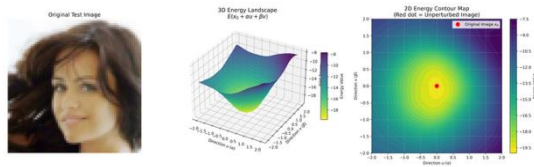


Figure 13. Energy Landscape for CELEBA Dataset

To understand point (1) above, we had to look back at our Joint Energy formula that we implemented, denoted by equation (8). For active attributes (for this example is *Male*, *Smiling*, and *Young*) the model was highly confident therefore the contribution to the joint energy is highly negative. These active attributes acted as anchors that were pulling down the joint energy down. Vice versa, the inactive attributes such as *Bald*, *Mustache*, and *Eyeglasses* to the joint energy is approximately 0. They act as a flat ceiling and do not pull the energy down.

To the second point (2), what we noted was that in the case of CELEBA dataset, noise destroyed geometry, which had cascading effect. *Smiling* is highly specific, localized geometric feature. Defined by the curvature of the lips, visibility of teeth, and the contraction of cheeks. When we perturb the image along the random directions u and v , we were adding high frequency noise to the pixels. This noise blurred the mouth, eyes, hairs, jawlines, and skin texture. It meant that the noise triggered a simultaneous destruction of all active attributes, as these features were destroyed, their individual energies spike toward 0. Because the joint energy constituted the sum of these individual terms, these energy spikes add up. This collective, additive rise in energy creates an incredibly steep, multi-dimensional energy barrier, surrounding the real image.

For the joint landscape to be ideal, one more critical thing had to happen. The inactive attributes had to remain quiet under perturbation is our third point (3). If our model was poorly calibrated, adding random noise to the face might had triggered a false positive for a rare attribute (such as *mustache*). If that happened the *Mustache* energy would have plummeted to high negative value. This would have dragged the joint energy $E_{joint}(x)$ down, creating a false low energy basin away from the red dot. Just like what happened in CIFAR. Because the red dot is in the bottom, it proved our CELEBA model is highly robust against

hallucinating absent features under noise. The inactive attributes stayed at 0 and did not pollute the joint landscape.

CONCLUSION

We were aiming to build a standard discriminative classifier with post-hoc energy interpretation model capable to perform classification task on CIFAR10, CIFAR100, and CELEBA dataset. The model was trained using supervised MLE of the conditional distribution.

We achieved F1-Score over 86% for the CIFAR10 and 100 datasets. We achieved lower F1-Score with CELEBA dataset. We start by using our custom architecture of shallow CNN, updated to deep CNN, and end up with pre-trained ResNet-18 for CIFAR100. We learned that custom model can be good but had a ceiling. Naïve transfer learning can be worse if the architecture is mismatched.

For energy landscape visualization, our preliminary experiments across both CIFAR10 and CIFAR100 confirmed a systemic calibration failure in discriminatively trained classifiers. When re-interpreted as implicit EBM, both networks consistently failed to place the real, unperturbed test images at local energy minima. Instead, random perturbations along orthogonal directions consistently lead to lower energy basins.

But an exciting result was shown by the CELEBA energy landscape. We hope that our interpretation is correct that our model compose multiple independent, non-competitive energy functions (attributes) into a single joint energy function, and the resulting landscape remains stable and calibrated.

We hope to further research this energy function in supporting temporal coherence of a video clip across a certain time span. We also hope to study how multiple energy function works as a composition in physical consistency like gravity or momentum.

DECLARATIONS

Author Contributions: The first author conceptualized the study, conducted the investigation, and wrote the original draft; the second and third authors conducted the GCP preparation and cloud environment adjustment for various experimental setups; and the fourth

author provided data and analysis insights for various challenges that occurred during the research. All authors read and approved the final manuscript.

Funding: This research was funded by the Ministry of Higher Education, Science, and Technology, Republic of Indonesia, under Ministerial Decree Number DIPA-139.04.1.69332/2025 and Contract Number 0998/LL3/AL.04/2025. The funder had no role in the study design, data collection, analysis, interpretation, manuscript preparation, or decision to publish.

Conflict of Interest: The authors declare no conflict of interest.

Generative Artificial Intelligence Use Statement: During the preparation of this manuscript, the authors used gemini-2.5-pro for proofreading and challenging any inconsistencies throughout the paper. The authors reviewed and edited the output and take full responsibility for the content of this publication.

Ethics Approval and Informed Consent: Not applicable. This study used only publicly available benchmark datasets (CIFAR10, CIFAR100, and CelebA) and did not involve the recruitment of human participants or the collection of new personal data.

Data Availability: The CIFAR10 and CIFAR100 datasets are publicly available at <https://www.cs.toronto.edu/~kriz/cifar.html>, and the CelebA dataset is publicly available at <https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html>. The source code is available at <https://github.com/harsantiputri>.

ACKNOWLEDGEMENTS

This research is funded by the Ministry of Higher Education, Science, and Technology, Republic of Indonesia. Ministerial Decree Number DIPA-139.04.1.69332/2025. Contract Number 0998/LL3/AL.04/2025.

REFERENCES

- [1] Goodfellow *et al.*, “Generative adversarial networks,” *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020, doi: 10.1145/3422622.
- [2] M. H. Guo *et al.*, “Attention mechanisms in computer vision: A survey,” *Comput. Vis. Media (Beijing)*, vol. 8, no. 3, pp. 331–368, 2022, doi: 10.1007/s41095-022-0271-y.
- [3] T. Singh and D. K. Vishwakarma, “A deep multimodal network based on bottleneck layer features fusion for action recognition,” *Multimed. Tools Appl.*, vol. 80, no. 24, pp. 33505–33525, 2021, doi: 10.1007/s11042-021-11415-9.
- [4] L. Yuan *et al.*, “Florence: A New Foundation Model for Computer Vision,” 2021.
- [5] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, *A review of convolutional neural networks in computer vision*, vol. 57, no. 4. Springer Netherlands, 2024. doi: 10.1007/s10462-024-10721-6.
- [6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017, doi: 10.1145/3065386.
- [7] I. Ahmed, M. Kajol, U. Hasan, and P. P. Datta, “ChatGPT vs. Bard: A Comparative Study,” 2023.
- [8] A. Borji and M. Mohammadian, “Battle of the Wordsmiths : Comparing ChatGPT, GPT-4, Claude, and Bard,” 2023.
- [9] S. Bubeck *et al.*, “Sparks of Artificial General Intelligence: Early experiments with GPT-4,” 2023.
- [10] N. M.-D. Doris M. and Brennan, “Can ChatGPT and Bard generate aligned assessment items? A reliability analysis against human performance,” *Journal of Applied*

- Learning & Teaching*, vol. 6, no. 1, pp. 75–80, 2023.
- [11] S. Kabir, D. N. Udo-Imeh, B. Kou, and T. Zhang, “Who Answers It Better? An In-Depth Analysis of ChatGPT and Stack Overflow Answers to Software Engineering Questions,” *ArXiv*, 2023.
- [12] T. P. Lillicrap *et al.*, “Continuous control with deep reinforcement learning,” *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*, 2016.
- [13] M. Fang, X. Wang, and N. Z. Gong, “Provably Robust Federated Reinforcement Learning,” 2025, doi: 10.1145/3696410.3714728.
- [14] G. Sreekumar and V. N. Bodetti, “IN PHYRE Discovers: Large Multimodal Models Struggle in Inductive Physical Reasoning,” *arXiv:2506.12263v1*, pp. 1–35, 2025.
- [15] H. Shen *et al.*, “PHY X: Does Your Model Have the ‘ Wits ’ for Physical Reasoning ?,” *arXiv:2505.15929v2*, 2025.
- [16] C. Xue, V. Pinto, C. Gamage, E. Nikonova, P. Zhang, and J. Renz, “Phy-Q as a measure for physical reasoning intelligence,” *Nat. Mach. Intell.*, vol. 5, no. January, pp. 83–93, 2023, doi: 10.1038/s42256-022-00583-4.
- [17] Y. LeCun, “A Path Towards Autonomous Machine Intelligence,” *Iclr*, pp. 1–62, 2022.
- [18] D. Ha and J. Schmidhuber, “Recurrent world models facilitate policy evolution,” *Adv. Neural Inf. Process. Syst.*, vol. 2018-Decem, no. C, pp. 2450–2462, 2018.
- [19] T. Osogami, “Boltzmann machines and energy-based models,” pp. 1–36.
- [20] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. Accessed: Jul. 28, 2026. [Online]. Available: <https://www.deeplearningbook.org/>
- [21] W. Grathwohl, K. C. Wang, J. H. Jacobsen, D. Duvenaud, K. Swersky, and M. Norouzi, “Your Classifier Is Secretly an Energy Based Model and You Should Treat It Like One,” *8th International Conference on Learning Representations, ICLR 2020*, pp. 1–23, 2020.
- [22] M. Welling, D. Bren, and Y. W. Teh, “Bayesian Learning via Stochastic Gradient Langevin Dynamics,” 2011.
- [23] A. Hyvarinen, “Estimation of Non-Normalized Statistical Models by Score Matching,” *Journal of Machine Learning Research*, vol. 6, p. 1, 2005, [Online]. Available: <https://www.jmlr.org/papers/volume6/hyvarinen05a/hyvarinen05a.pdf>
- [24] F. K. Gustafsson, M. Danelljan, R. Timofte, and T. B. Schön, “How to Train Your Energy-Based Model for Regression,” *31st British Machine Vision Conference, BMVC 2020*, 2020, doi: 10.5244/c.34.43.
- [25] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778, doi: 10.1109/CVPR.2016.90.
- [26] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, “Squeeze-and-Excitation Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 8, pp. 2011–2023, 2020, doi: 10.1109/TPAMI.2019.2913372.
- [27] V. Singh *et al.*, “Beyond Cosine Decay: On the effectiveness of Infinite Learning Rate Schedule for Continual Pre-training,” pp. 1–21, 2025.
- [28] H. Guo, J. Jin, and B. Liu, “Stochastic Weight Averaging Revisited,” *Applied Sciences (Switzerland)*, vol. 13, no. 5, 2023, doi: 10.3390/app13052935.