École Polytechnique Fédérale de Lausanne

Design of Energy-based Classifiers for General Classification Tasks

Laboratoire d'Automatique - LA

MASTER SEMESTER PROJECT REPORT

Professor	Giancarlo Ferrari Trecate
Assistants	Clara Galimberti, Muhammad Zakwan
Student	Weitong Zhang
Sciper	322327
Section	Mechanical Engineering
Date	June 13, 2022

Contents

Abstract

Deep neural networks (DNNs) have achieved magnificent success in supervised classification tasks, but they are known to be vulnerable to adversarial attacks. The instability of deep neural networks comes from the problem of vanishing or exploding gradients during weight optimization. Recent studies have demonstrated that DNNs that stem from the discretization of continuous-time Hamiltonian systems ensure non-vanishing gradients by design for an arbitrary network depth. In this project, we follow the previous work on Hamiltonian DNNs (H-DNNs) and conduct classification based on the energy instead of the final state of the discretized systems. The proposed method shows good performance on general classification tasks and achieves high robustness against adversarial attacks.

Keywords: Deep Neural Networks, Supervised Classification, Hamiltonian Systems, Energy-based Classification

1 Introduction

Deep neural networks have shown success in various tasks, such as image classification [1], natural language processing [2], and autonomous driving [3]. However, the robustness of deep neural networks is sometimes not guaranteed, and the vulnerability to adversarial attacks leads to concerns in real-world applications [4].

Recently, studies have shown that by designing deep neural networks from the time discretization of ordinary differential equations (ODEs) [5], the stability properties of ODEs can be leveraged to defend against adversarial attacks. A related problem is the observation of vanishing or exploding gradients during training [6], which suggests using neural networks derived from dynamical systems that produce bounded and non-vanishing state trajectories. An example is to discretize ODEs based on skew-symmetric maps, which has been used to define anti-symmetric deep neural networks [5]. Another example is to recast forward propagation as a Hamiltonian system, which has the following structure:

$$\dot{\mathbf{y}}(t) = \mathbf{J}(t)\frac{\partial H(\mathbf{y}(t), t)}{\partial \mathbf{y}(t)}, \quad \forall t \in [0, T]. \tag{1}$$

Furthermore, a unified framework defining Hamiltonian Deep Neural Networks (H-DNNs) is provided in [7], in which anti-symmetric and Hamiltonian systems are encompassed. A new class of H-DNNs is also proposed with better performance and stability.

In this report, an energy-based H-DNN is proposed for general classification tasks. Instead of conducting classification according to the final state of the dynamical system, the energy of the layer is leveraged for classification. The proposed method is tested on several benchmark classification tasks, including digit classification based on the MNIST dataset.

The remainder of the report is organized as follows. In Section 2, related works about DNNs inspired by ODE discretization and previous works on H-DNNs are introduced. In Section 3, an energy-based classifier is presented, which is followed by several numerical examples and robustness evaluation in Section 4, and 5, respectively. Finally, concluding remarks are provided in Section 6.

2 Related Works

2.1 ODE-inspired DNNs

Popular models such as residual neural networks [8], and recurrent neural network decoders [9] can be obtained by applying a sequence of transformations to a hidden state:

$$\mathbf{y}_{j+1} = \mathbf{y}_j + h\mathbf{f}\left(\mathbf{y}_j, \boldsymbol{\theta}_j\right), \tag{2}$$

where $j = 0, 1, \ldots, N-1$, $h = \frac{T}{N}$ denotes the time step, $f(\cdot)$ is a nonlinear function, and $\boldsymbol{\theta}_j$ is the vector of parameters. The iterative updates between hidden layers can be regarded as forward-Euler discretization of the following nonlinear system:

$$\dot{\mathbf{y}}(t) = \mathbf{f}(\mathbf{y}(t), \boldsymbol{\theta}(t)). \tag{3}$$

2.2 Stability of ODE-inspired DNNs

In order to obtain stable forward propagation, one of the methods is to construct force fields with skew-symmetric Jacobians. Consider the following nonlinear ODE:

$$\dot{\mathbf{y}}(t) = \sigma\left(\mathbf{K}^\top(t)\mathbf{y}(t) + b(t)\right), \tag{4}$$

with a time interval $t = [0, T]$. The ODE is stable if $\mathbf{K}^\top(t)$ changes slow and

$$\max_{i=1,2,\ldots,n} \mathrm{Re}\left(\lambda_i(\mathbf{J}(t))\right) \leq 0, \quad \forall t \in [0, T], \tag{5}$$

where $\mathbf{J}(t)$ is the Jacobian of $\sigma\left(\mathbf{K}^\top(t)\mathbf{y}(t) + b(t)\right)$, and it could be calculated as

$$\mathbf{J}(t) = \mathrm{diag}\left(\sigma'\left(\mathbf{K}(t)^\top\mathbf{y} + b(t)\right)\right)\mathbf{K}(t)^\top. \tag{6}$$

Considering that the activation function is usually non-decreasing, Eq. (5) is satisfied if $\mathbf{K}(t)$ changes slowly and

$$\max_{i=1,2,\ldots,n} \mathrm{Re}\left(\lambda_i(\mathbf{K}(t))\right) \leq 0. \tag{7}$$

To satisfy this condition, the simplest way is to build up a system with skew-symmetric Jacobians [5], [9]. A skew-symmetric matrix is a matrix whose transpose equals its negative, i.e. $\mathbf{J}(t) = -\mathbf{J}^\top(t)$, because all eigenvalues of a skew-symmetric matrix are imaginary:

$$\mathrm{Re}\left(\lambda_i(\mathbf{J}(t))\right) = 0. \tag{8}$$

In order to control the smoothness of $\mathbf{K}(t)$, regularization of the transformation weights is introduced as follows [5]:

$$R(\mathbf{K}) = \frac{1}{2h}\sum \|\mathbf{K}_j - \mathbf{K}_{j-1}\|_F^2, R(b) = \frac{1}{2h}\sum \left(b_j - b_{j-1}\right)^2, \tag{9}$$

where $\|\cdot\|_F$ denotes the Frobenius norm.

2.3 Hamiltonian Neural Networks

Hamiltonian neural networks [7] are derived from a general class of continuous-time Hamiltonian systems to avoid the problem of gradients vanishing/exploding during training. Consider time-varying Hamiltonian systems defined by the ODE

$$\dot{\mathbf{y}}(t) = \mathbf{J}(t)\frac{\partial H(\mathbf{y}(t), t)}{\partial \mathbf{y}(t)}, \tag{10}$$

where $\mathbf{J}(t)$ is skew-symmetric at all times and the continuously differentiable function $H(\cdot)$ is the Hamiltonian function. With $\mathbf{J}$ and $H(\cdot)$ independent of time, the system is generalized as a time-invariant system. Time-invariant Hamiltonian systems are marginally stable when $H(\cdot)$ is a positive definite function, which makes them ideal candidates for deep neural networks. As for time-varying models, the same property remains if $H(\mathbf{y}(t), t)$ changes very slowly over time.

In the following discussion, we focus on the the energy function

$$H(\mathbf{y}(t), t) = [\log(\cosh(\mathbf{K}(t)\mathbf{y}(t) + \mathbf{b}(t)))]^\top \mathbf{1}, \tag{11}$$

where $\mathbf{1} = [1, \ldots, 1]^\top$. Partial derivative of the energy with respect to $\mathbf{y}(t)$ is computed as

$$\frac{\partial H(\mathbf{y}(t), t)}{\partial \mathbf{y}(t)} = \mathbf{K}^\top(t) \tanh(\mathbf{K}(t)\mathbf{y}(t) + \mathbf{b}(t)). \tag{12}$$

Therefore, by with the above energy function (11), the system (10) becomes

$$\dot{\mathbf{y}}(t) = \mathbf{J}(\mathbf{y}, t)\mathbf{K}^\top(t) \tanh(\mathbf{K}(t)\mathbf{y}(t) + \mathbf{b}(t)). \tag{13}$$

Applying forward Euler discretization with time step h and assuming $\mathbf{J}(t)$ constant, we obtain the propagating equation between layers j and $j+1$

$$\mathbf{y}_{j+1} = \mathbf{y}_j + h\mathbf{J}\mathbf{K}_j^\top \tanh\left(\mathbf{K}_j\mathbf{y}_j + \mathbf{b}_j\right), \tag{14}$$

where $\mathbf{J}(\mathbf{y}) = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & 0 \end{bmatrix}$ for H_1-DNN.

3 Energy-based Classifiers

3.1 Classification based on energy

When H-DNNs are applied for classification tasks, the input is propagated through Hamiltonian layers, followed by a binary classification layer to classify according to the final state of the dynamic system. However, as Hamiltonian dynamics is defined by the energy function (11), it is also inspiring to implement classification according to the energy of each state instead of the final state itself. Hamiltonian systems are attractive due to their energy-preservation properties, which can result in more stable deep networks. Accordingly, energy-based networks are expected to have higher robustness, especially against adversarial attacks.

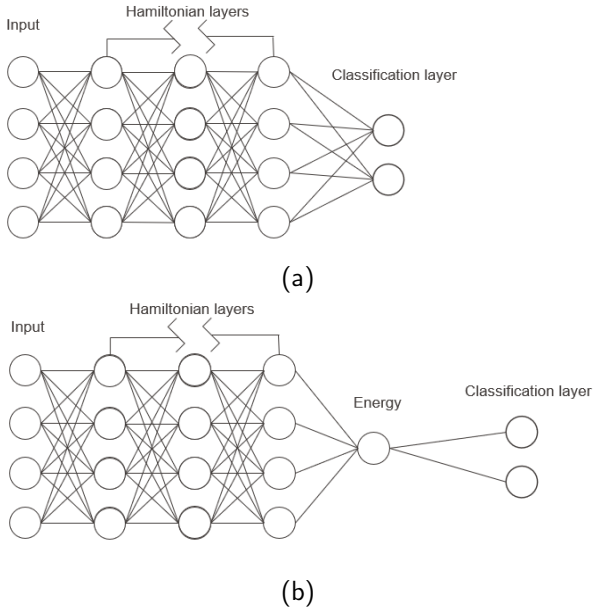


Figure 1: Network architecture of (a) H-DNN classifiers, and (b) energy-based classifiers.

3.2 Training algorithm

In this report, we consider binary or multicategory classification problems where M denotes the number of classes. The networks are trained by solving the following optimization problem

$$\begin{aligned} \min_{\boldsymbol{\theta}} \quad & \tfrac{1}{s}\sum_{k=1}^{s}\mathcal{L}\left(\mathbf{f}_N\left(\mathbf{H}_N^k\right),c^k\right)+\alpha_c R_N\left(\boldsymbol{\theta}_N\right)+\alpha R\left(\mathbf{K}_{0,\ldots,N-1},\mathbf{b}_{0,\ldots,N-1}\right)+\alpha_H R_H\left(\mathbf{H}_{1,\ldots,N}\right), \\ & \text{s.t. } \mathbf{y}_{j+1}^k=\mathbf{y}_j^k+h\mathbf{J}_j\left(\mathbf{y}_j^k\right)\mathbf{K}_j^\top\tanh\left(\mathbf{K}_j\mathbf{y}_j^k+\mathbf{b}_j\right), \quad j=0,1,\ldots,N-1, \end{aligned} \tag{15}$$

where $\mathrm{R_N}(\cdot)$ is the L_2 regularization term of the output layer, $\mathrm{R}(\cdot)$ is the regularization term of layers $0,\ldots,N-1$. The output layer is given by $\mathbf{f}_N\left(\mathbf{H}_N^k\right)=\sigma_c\left(\mathbf{W}\mathbf{H}_N+\mu\right)$, where $\mathbf{H}_N$ is the energy of the final state, $\sigma_c(x)=\frac{1}{1+e^{-x}}$ for binary classification problems and $\sigma_c(x)=\frac{e^x}{\sum_{k=1}^{s}e^{x_k}}$ for multi-class classification problems. The trainable parameters defining the network are $\theta=\{\mathbf{K}_{0,\ldots,N-1},\mathbf{b}_{0,\ldots,N-1},\mathbf{W},\mu\}$. The regularization terms are defined as

$$\begin{aligned} R_K\left(\mathbf{K}_{0,\ldots,N-1}\right)&=\tfrac{h}{2}\textstyle\sum_{j=1}^{N-1}\left\|\mathbf{K}_j-\mathbf{K}_{j-1}\right\|_F^2 \\ R_b\left(\mathbf{b}_{0,\ldots,N-1}\right)&=\tfrac{h}{2}\textstyle\sum_{j=1}^{N-1}\left\|\mathbf{b}_j-\mathbf{b}_{j-1}\right\|^2 \\ R_H\left(\mathbf{H}_{1,\ldots,N}\right)&=\tfrac{h}{2}\textstyle\sum_{j=1}^{N}\left\|\mathbf{H}_j-\mathbf{H}_{j-1}\right\|^2 \end{aligned} \tag{16}$$

The regularization term is to favor weights and energy that vary smoothly between adjacent layers. The coefficients α,α_candα_H are hyperparameters that represent the trade-off between fitting and regularization.

4 Numerical Examples

In order to evaluate the validity and performance of energy-based classifiers, they are first tested on simple binary classification tasks. Then, multi-class classification datasets such as "Peaks" and MNIST are used for evaluation. In this section, only the energy of the final state is used for classification.

4.1 Binary classification examples

4.1.1 Double moons dataset

Double moons dataset is a relatively simple dataset with features in R^2. As is shown in Figure 2, we randomly created 8000 data points, half of which are used for training. As the dataset is simple for classification, we keep the original two features of the input. The optimization problem is solved using the Adam algorithm [10], Sigmoid as the output activation function and binary cross-entropy loss as loss function. With a classifier with eight layers (see Table 1), we are able to obtain 100% test accuracy.

Number of features	2			
Number of layers	2	4	6	8
Test accuracy	88.58%	96.80%	99.45%	100.00%

Table 1: Test accuracy of energy-based classifiers on double moons dataset.

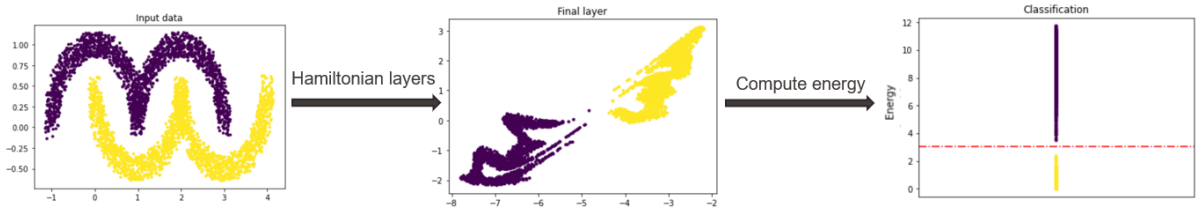
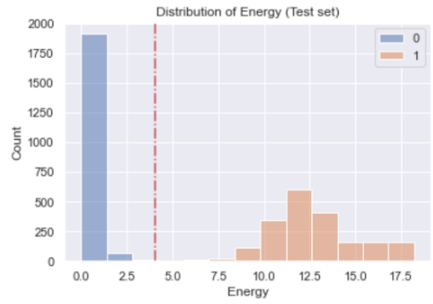
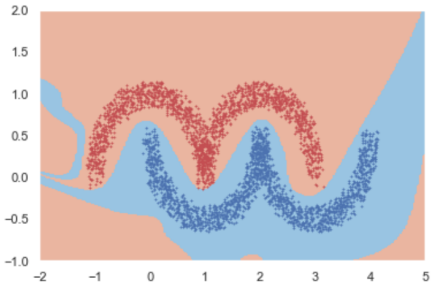


Figure 2: Classification process of energy-based classifiers with 8 layers on double moons.



(a) Distribution of energy



(b) Prediction area of trained classifiers

Figure 3: Results of energy-based classifiers with 8 layers on double moons.

4.1.2 Swiss roll dataset

As for the swiss roll dataset, input features augmentation is applied so as to assure accurate classification. The data is obtained by sampling the vector functions

$$f_1(r,\theta) = r \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}, \quad \text{and} \quad f_2(r,\theta) = (r+0.2) \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix}, \tag{17}$$

where $r \in [0,1]$ and $\theta \in [0, 4\pi]$.

In this case, input feature vectors are augmented as $\begin{bmatrix} (\mathbf{y}_0^k)^\top & 0 & 0 \end{bmatrix}^\top \in R^4$, where $\mathbf{y}_0^k \in R^2$ are original datapoints. The optimization process and model structure are similar to those for "double moons" dataset. With a classifier with 8 layers and 4 features (see Table 2), we are able to obtain 100% test accuracy.

Besides accuracy, what we can observe from Figure 3 and Figure 5 is that for both binary classification tasks, the energy of label 0 is distributed very close to 0, while the energy of label 1 is less centralized with higher values.

Number of features	4			
Number of layers	2	4	8	16
Test accuracy	67.58%	86.88%	98.95%	100.00%

Table 2: Test accuracy of energy-based classifiers on swiss roll dataset.

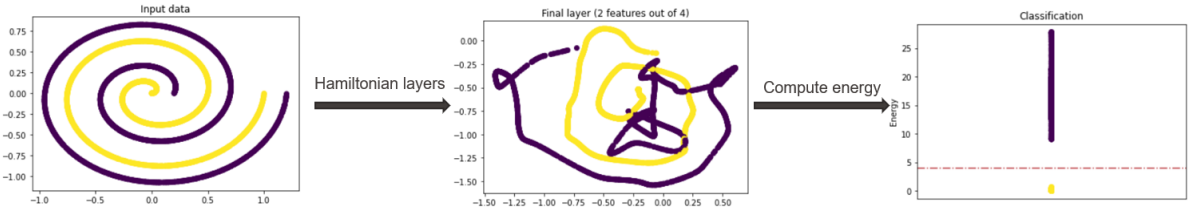
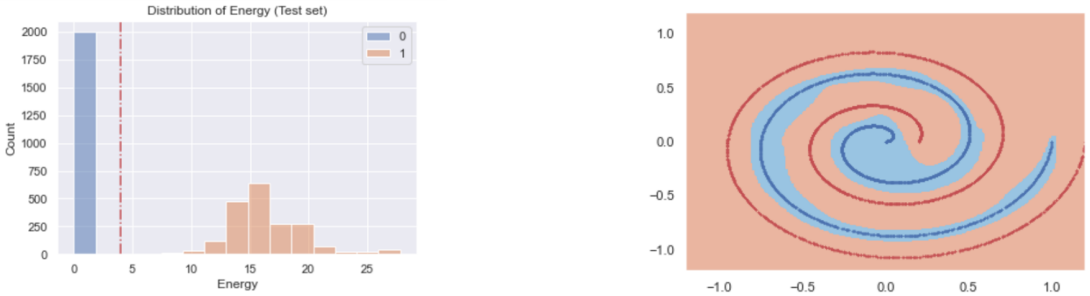


Figure 4: Classification process of energy-based classifiers with 16 layers on swiss roll.



(a) Distribution of energy

(b) Prediction area of trained classifiers

Figure 5: Results of energy-based classifiers with 16 layers on swiss roll.

4.2 Multi-class classification examples

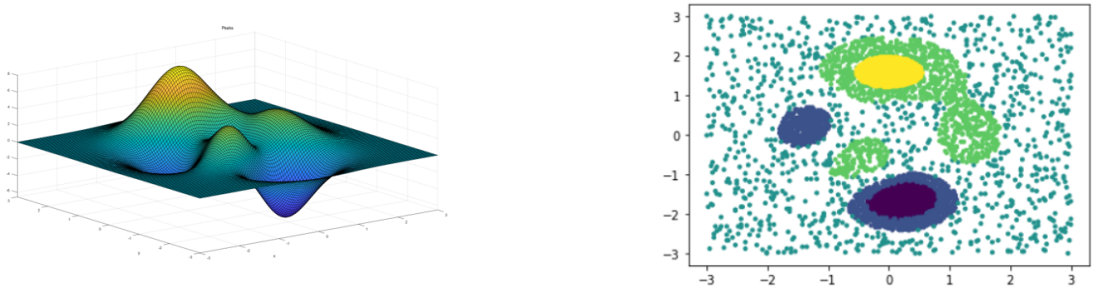
4.2.1 Peaks dataset

Peaks is a new challenging multi-class classification problem proposed by [5]. The dataset is created using the **peaks** function in MATLAB that samples the data with the function

$$\begin{aligned} f(x) = 3\left(1-x_1\right)^2 \exp\left(-\left(x_1^2\right) - \left(x_2+1\right)^2\right) - 10\left(x_1/5 - x_1^3 - x_2^5\right) \\ \exp\left(-x_1^2 - x_2^2\right) - 1/3\exp\left(-\left(x_1+1\right)^2 - x_2^2\right), \end{aligned} \tag{18}$$

where $x \in [-3, 3]^2$. The peaks function creates non-convex and non-linear level sets while the smoothness is maintained. Data points are sampled from a 256×256 grid and divided into five classes according to the function value. Then we randomly pick 1000 sample points in each class, and thus a dataset with 5000 points is created. The data points are illustrated in Figure 6. Half of the dataset is divided as the training set, while the other part is used for testing.

The input data are augmented to 8 features, and energy-based classifiers with different layers are trained for evaluation. As is shown in 3, a neural network with 512 layers achieved 96.16% accuracy, and similar to previous examples, energies with label 0 have extremely low values, while energies in other classes are less centralized with higher values (see Figure 8).



(a) Discretized **peaks** function in MATLAB

(b) Created dataset with 5 classes

Figure 6: The peaks dataset.

Number of features	8			
Number of layers	64	128	256	512
Test accuracy	89.68%	92.24%	94.60%	96.16%

Table 3: Test accuracy of energy-based classifiers on peaks dataset.

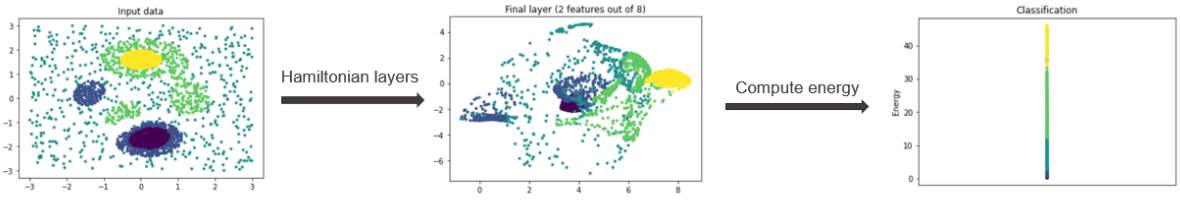


Figure 7: Classification process of energy-based classifiers with 512 layers on peaks.

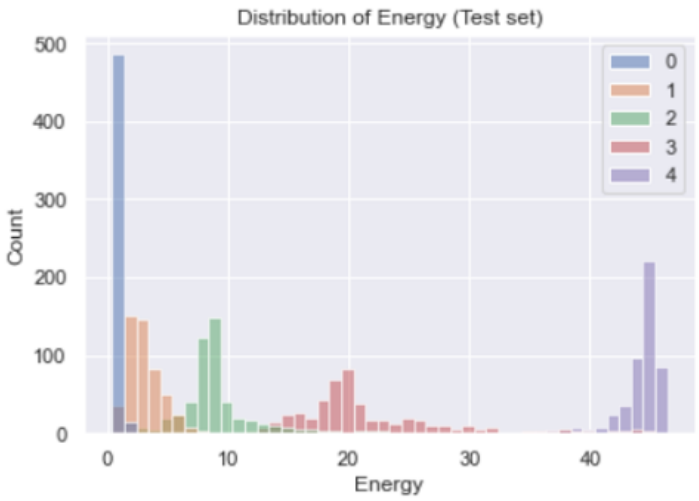


Figure 8: Distribution of energy of classifiers with 512 layers on peaks.

4.2.2 MNIST

MNIST is a standard image classification dataset with 28×28 digital images in the grayscale of hand-written digits from 0 to 9 with corresponding labels. It contains 60,000 images for training and 10,000 images for testing.

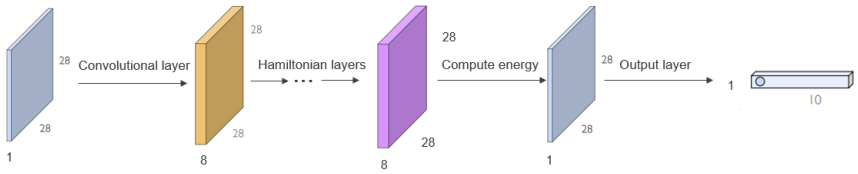
Figure 9 shows the energy-based network architecture for MNIST classification. A convolutional layer is first implemented to augment the input data from 1 to 8 channels, and the energy of each pixel is computed during the propagation through Hamiltonian layers. The output layer uses the energy of final states or all the states for a linear transformation and a softmax activation function to obtain a vector in R^{10} representing the probability of the data belonging to each of the ten classes.

In Table 4, two classifiers in Figure 9 with 2, 4 and 8 layers are compared with H_1-DNN. With more parameters in the output layer, classifiers using the energy of all states perform better than using only the energy of the final state, and with more layers, more energy of all layers can be used for classification. The accuracies between H_1-DNN and all-state-energy-based classifiers are similar with more than 4 layers. Another similar property among the models is that the performance is not improved with a sufficient number of layers, which may result from the fact that the linear output classifier is not complex enough.

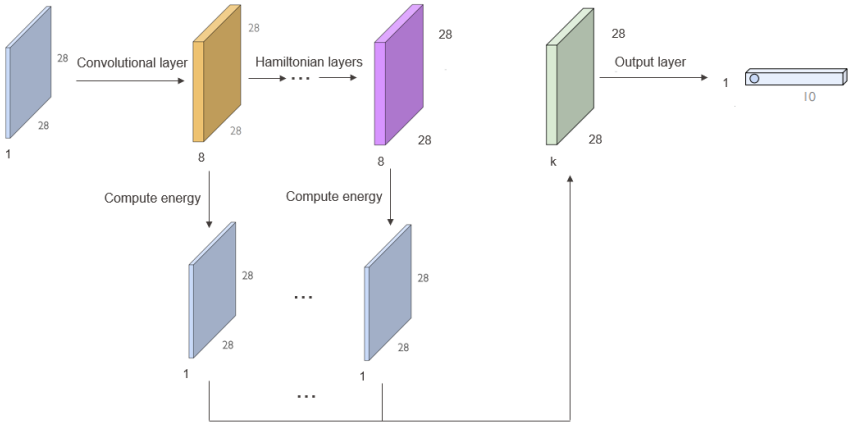
Model / # of layers	$H_1 - DNN$	energy-based (final states)	energy-based (all states)
2	96.11%	92.28%	92.49%
4	96.67%	92.97%	96.94%
8	96.96%	93.30%	97.05%

Table 4: Test accuracy of different methods on MNIST.

In Figure 10, we can observe that the energy tends to maintain a similar shape to the input images. This property will bring us some inspiration when we discuss about robustness evaluation in the following section. From the confusion matrix, while digits such as *0*, *1* and *3* are classified with high accuracy, *2*, *5* and *8* are sometimes misclassified as other digits.

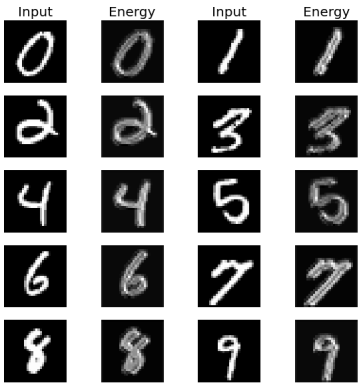


(a) Energy-based classifiers using energy of final states

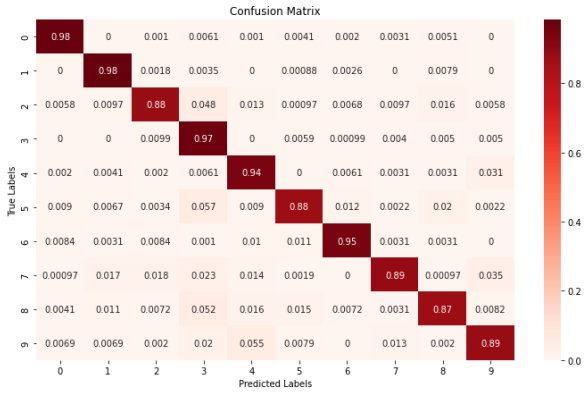


(b) Energy-based classifiers using energy of all states

Figure 9: Comparison of two energy-based methods for MNIST classification.



(a) Energy of the final state comparing to the input



(b) Confusion matrix

Figure 10: Results of final-state-energy-based classifiers with 4 layers.

5 Robustness Testing

The motivation of this part is to explore the robustness of the energy-based classifiers with comparison to H_1-DNN, and Res-Net derived from (2). Methods of testing the robustness are corrupting images with random noises, blurring the images with a Gaussian function, and adversarial attacks. In this part, all the models have 4 hidden layers and 1 output layer for linear classification. As for the optimization algorithm, we use SGD with Adam and cross-entropy loss for 10 epochs. The learning rate is initialized as 0.04 and decayed with $\gamma = 0.8$ in each epoch. For all networks, we set $\alpha = \alpha_c = 8 \times 10^{-3}$, and for energy-based networks, hyperparameter for energy regularization is set as $\alpha_H = 1 \times 10^{-5}$.

5.1 Noises and blur

5.1.1 Gaussian noise

One of the methods of testing the robustness of neural networks is corrupting images with random noises. Gaussian noise is a kind of noise with values that are Gaussian distributed. The probability density function p of a Gaussian random variable is given by

$$p_G(z) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}, \tag{19}$$

where z represents the grey level of noises, μ the grey value, and σ the standard deviation of the value. We change the value of variance σ^2 to control the extent of corruption, and the result is shown in 11. The performance of energy-based classifiers is between Res-Net and H_1-DNN. From Figure 12, we observe that the energy of the final state is similar to that of the input state. Therefore, with the increase in variance noise values, it is more difficult for the energy to be classified.

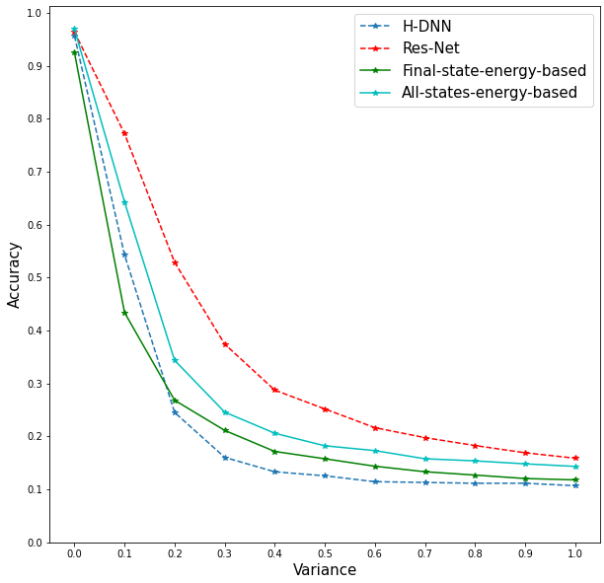


Figure 11: Testing accuracy on MNIST examples corrupted by Gaussian noise.

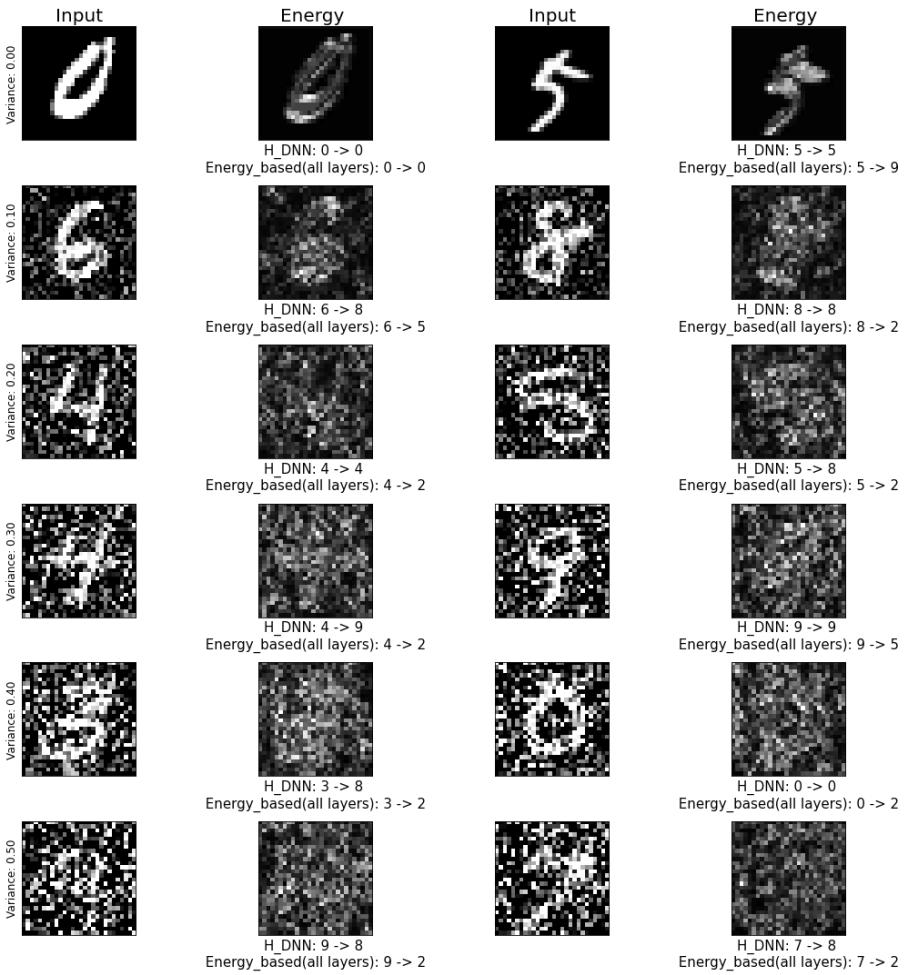


Figure 12: MNIST examples corrupted by Gaussian noise, corresponding energy of the final state, and predictions of two models.

5.1.2 Salt-and-pepper noise

Salt-and-pepper noise, also known as impulse noise, is a form of noise that sparsely occurs as white and black pixels. In this section, equal amounts of salt and pepper noises are added to the images, and we control the extent of corruption by changing the proportion of image pixels replaced with noise.

On salt-and-pepper noise, the performance of energy-based classifiers is even worse than the evaluation on Gaussian noise. An intuitive reason is that this kind of noise is more abrupt and thus leads to random values in energy, as is shown in Figure 14.

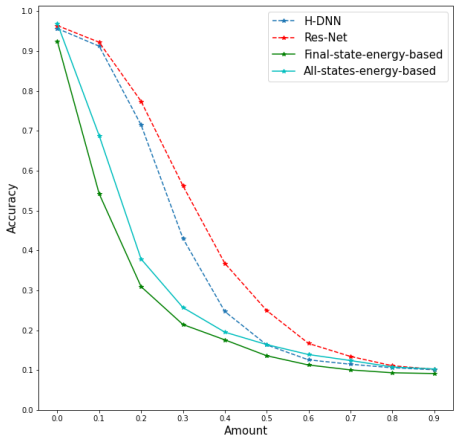


Figure 13: Testing accuracy on MNIST examples corrupted by salt-and-pepper noise.

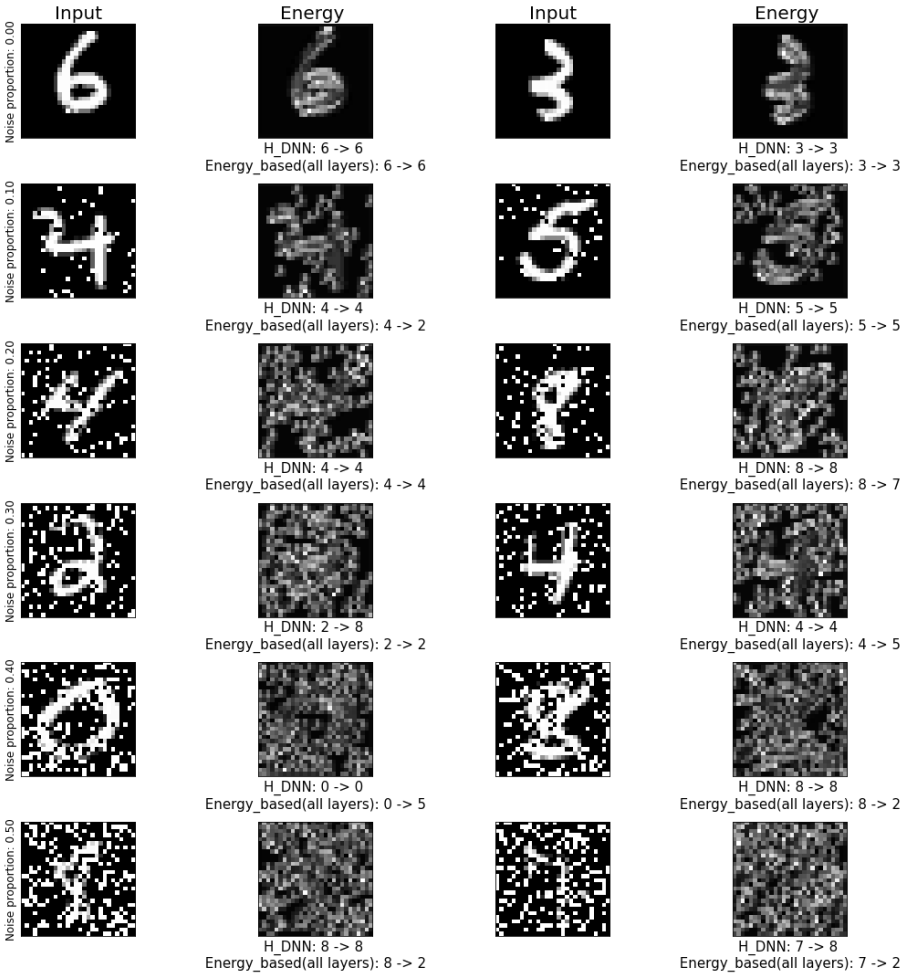


Figure 14: MNIST examples corrupted by salt-and-pepper noise, corresponding energy of the final state, and predictions of 2 models.

5.1.3 Gaussian blur

Gaussian blur, also known as Gaussian smoothing, is a kind of image-blurring filter that uses a Gaussian function for calculating the transformation to apply to each pixel in the image. It can be considered as a nonuniform low-pass filter that preserves low spatial frequency and reduces negligible details in an image. Gaussian blur is typically achieved by convolving an image with a Gaussian kernel in the following form

$$G_{2D}(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{20}$$

where x and y are location indices, and σ is the standard deviation controling the variance around the mean value of the Gaussian distribution. We fix $\sigma = 7$ and control the extent of Gaussian blur by changing the size of the Gaussian kernel.

In this case, the energy-based classifier using only the energy from the final state achieves the best performance among 4 models. An intuitive reason is that both the input images and energy maintain the gray values but they are smoothed by their neighborhoods (as is shown in Figure 16), and they are still recognizable and less likely to lead to misclassification.

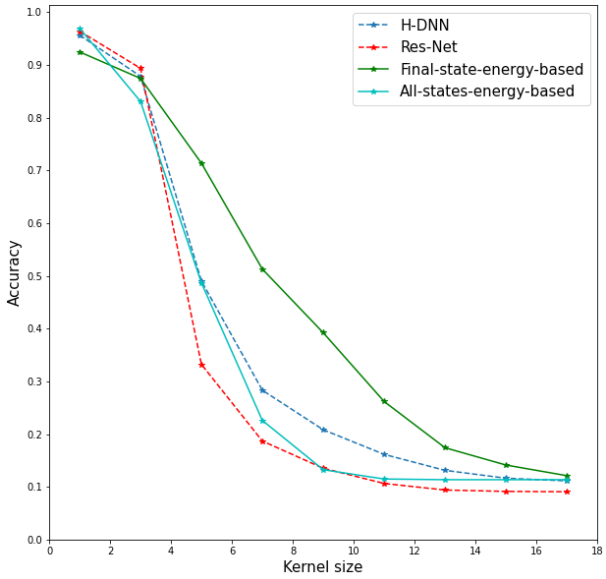


Figure 15: Testing accuracy on MNIST examples corrupted by Gaussion blur.

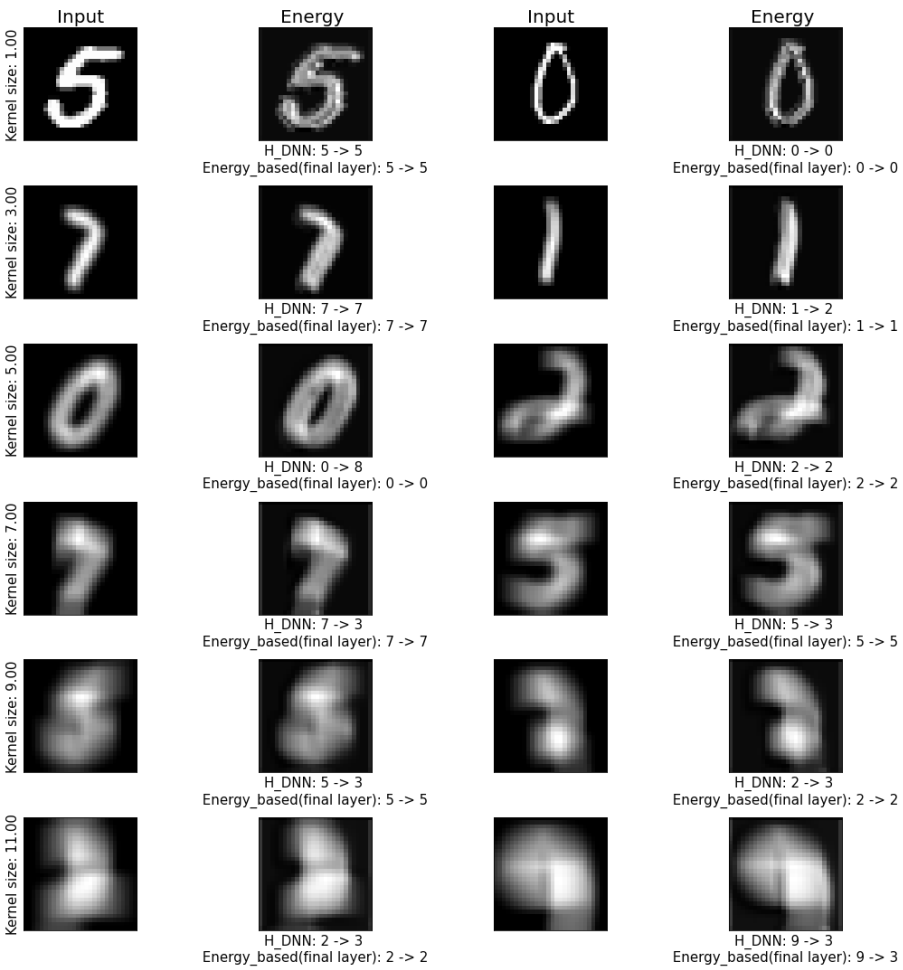


Figure 16: MNIST examples corrupted by Gaussian blur, corresponding energy of the final state, and predictions of two models.

5.2 Adversarial attacks

5.2.1 FGSM

Fast Gradient Sign Attack (FGSM) [11] exploits the gradients of the neural network to build an adversarial image. Instead of minimizing the loss by updating the weights based on back-propagated gradients, the attack adjusts the input data to maximize the loss based on the same propagated gradients. The attacked images can be expressed as

$$\hat{x} = x + \epsilon * \operatorname{sign}\left(\nabla_x J(\theta, x, y)\right), \tag{21}$$

where x and y are the original image and label, $\hat{x}$ is the adversarial image, J denotes the loss function, θ denotes the model weights, and ϵ describes the step of the attack. From Figure 17, with an increasing value of ϵ, the accuracy gradually reduces to 0%, and the all-states-energy-based classifier brings the highest robustness. From Figure 18, we can observe that although input images are attacked based on gradients, they keep the original shape and the energy remain less corrupted, which may explain the robustness of energy-based classifiers.

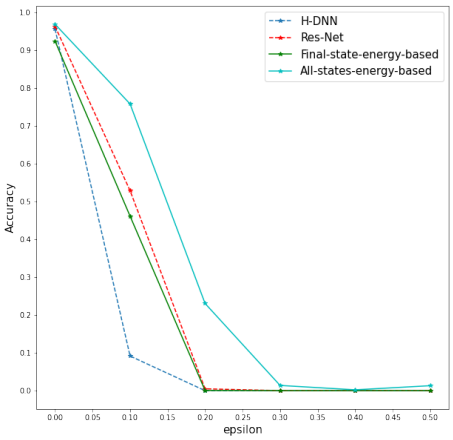


Figure 17: Testing accuracy on MNIST examples attacked by FGSM.

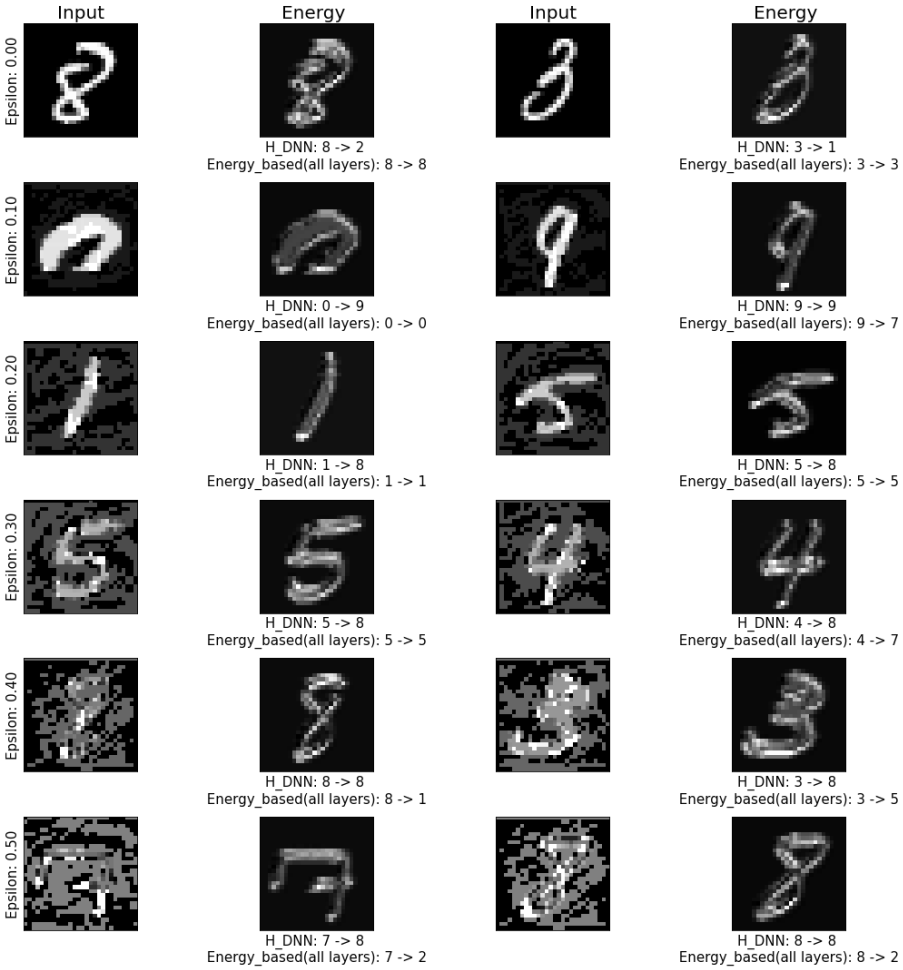


Figure 18: MNIST examples attacked by FGSM, corresponding energy of the final state, and predictions of two models.

5.2.2 PGD

Projected Gradient Descent (PGD) [12] is another popular gradient-based attack method. It attempts to find the perturbation that maximizes the loss of a model on a particular input while keeping the size of the perturbation smaller than a specified amount ϵ. Compared to simple one-step schemes such as FGSM, PGD brings a more powerful adversary by multiple steps:

$$x^{t+1} = \Pi_{x+\mathcal{S}} \left(x^t + \epsilon \operatorname{sgn} \left(\nabla_x L(\theta, x, y) \right) \right). \tag{22}$$

where $\Pi_{x+\mathcal{S}}$ denotes the projection onto a L^∞ ball $\mathcal{S}$.

The result of PGD is similar to that of FGSM, where the all-states-energy-based classifier shows the highest robustness among the four models, and the input images and energy are still recognizable after attack.

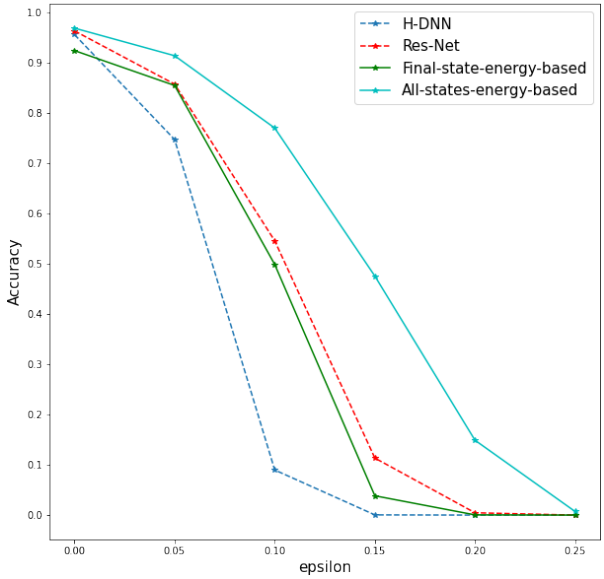


Figure 19: Testing accuracy on MNIST examples attacked by PGD.

Figure 20: MNIST examples attacked by PGD, corresponding energy of the final state, and predictions of two models.

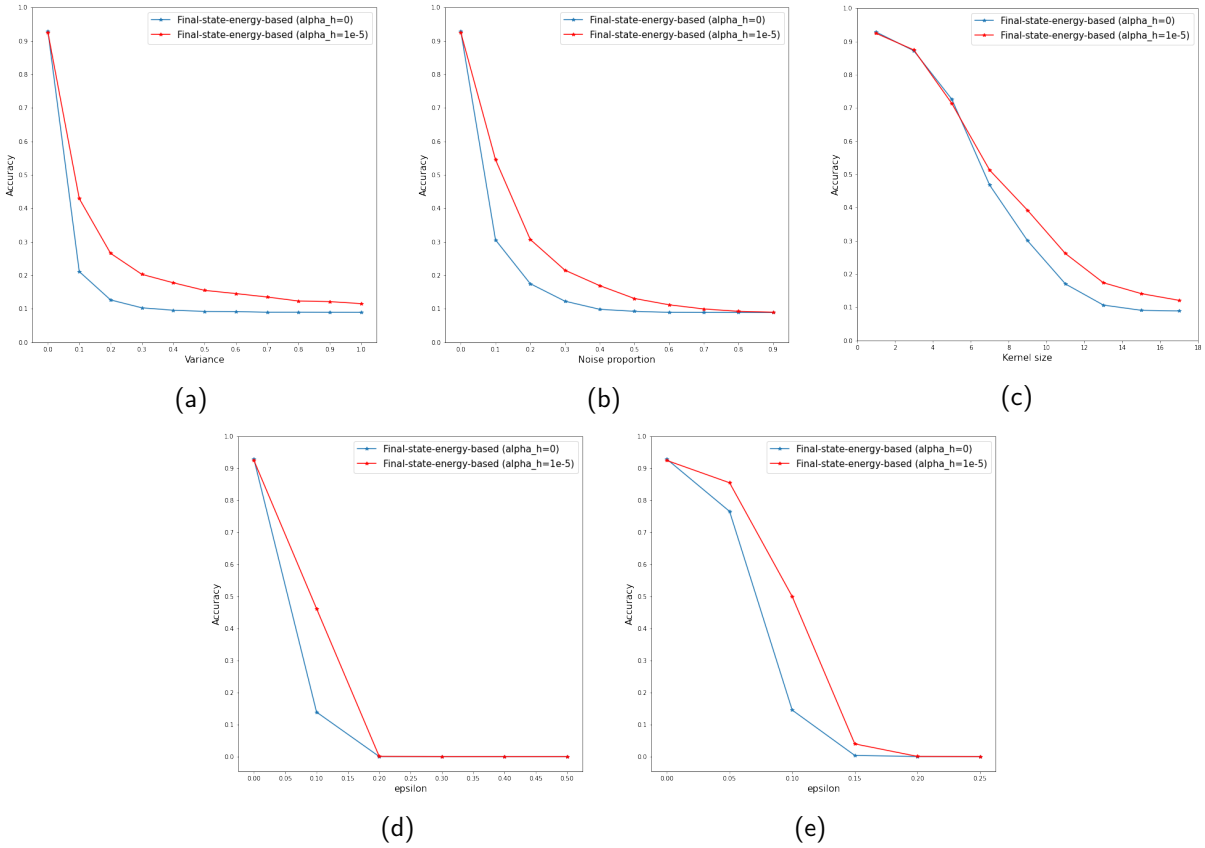
5.3 Robustness testing of energy-based regularization

In Section 3.2, we applied the following regularization terms to the optimization problem:

$$Reg = \alpha_c R_N\left(\boldsymbol{\theta}_N\right) + \alpha R\left(\mathbf{K}_{0,\ldots,N-1}, \mathbf{b}_{0,\ldots,N-1}\right) + \alpha_H R_H\left(\mathbf{H}_{1,\ldots,N}\right), \tag{23}$$

which is to favor weights and energy that vary smoothly between adjacent layers. In [7] it has been shown that the regularization term on weights $\alpha R\left(\mathbf{K}_{0,\ldots,N-1}, \mathbf{b}_{0,\ldots,N-1}\right)$ can provide robustness for H-DNNs. In this section, we show experimentally that the energy-based regularization term $\alpha_H R_H\left(\mathbf{H}_{1,\ldots,N}\right)$ also improves the robustness of the neural networks.

As is shown in Figure 21, various attack methods are applied to 2 models with and without the energy-based regularization term, respectively. From the result, it is obvious that the energy-based regularization term leads to an increase of the model robustness. Therefore, it is useful to limit the difference among the energy in different layers.



(a) (b) (c)

(d) (e)

Figure 21: Test accuracy of 2 models on MNIST examples attacked by: (a) Gaussian noise, (b) salt-and-pepper noise, (c) Gaussian blur, (d) FGSM and (e) PGD.

6 Conclusion and Future Work

Throughout the project, a type of energy-based classifier is designed and implemented for various classification tasks. A regularization term of energy is added to stabilize the network. Then, different methods are applied to test the robustness of the proposed model, which shows high robustness against adversarial attacks. Results are also analyzed by visualization of energy.

However, limitations of the proposed method exist. When trained and tested for different classification tasks, the proposed model does not show higher accuracy than baseline models. Moreover, the result of adversarial attacks is not consistent with that of noises and blur. An intuitive hypothesis of the reason is proposed by energy plots, but it is still difficult to derive a mathematical proof of the robustness of energy-based classifiers. Furthermore, although classification is conducted according to the energy of the states, the gradients are still propagating through states instead of energy during the training, which leads to similar results between H-DNN and energy-based classifiers.

In the future, there are potential methods that may improve performance or avoid limitations. The performance and robustness of the model can be further explored when trained and tested on more complicated datasets such as CIFAR-10. It would also be challenging to provide mathematical or analytical proof of robustness of energy-based classifiers. Finally, other novel models that leverage the stable property of Hamiltonian systems may also be proposed.

References

[1] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[2] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. *arXiv preprint arXiv:1603.06042*, 2016.

[3] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[4] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[5] Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *Inverse problems*, 34(1):014004, 2017.

[6] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[7] Clara Lucía Galimberti, Liang Xu, and Giancarlo Ferrari Trecate. A unified framework for hamiltonian deep neural networks. In *Learning for Dynamics and Control*, pages 275–286. PMLR, 2021.

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

[9] Bo Chang, Minmin Chen, Eldad Haber, and Ed H Chi. Antisymmetricrnn: A dynamical system view on recurrent neural networks. *arXiv preprint arXiv:1902.09689*, 2019.

[10] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[11] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[12] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.