K-NN WITH IDENTITY INITIALIZED RESNET—A DEEP LEARNING MODEL EMPIRICALLY ROBUST AGAINST L-INFINITY PROJECTED GRADIENT DESCENT ATTACK

Yicheng Qian, Boshi An & Bowen Su
Department of Computer Science, Machine Learning Term Project Group
Peking University
Beijing, China

ABSTRACT

We propose a novel deep learning model called K-NN with Identity Initialized ResNet (IdResNet). Our model combines the robustness of k-nearest neighbors with the power of neural networks: it is more robust against Projected Gradient Descent (PGD) attack than previous works for $\epsilon = 8/255$ and $\epsilon = 16/255$. Moreover, it only takes 5 minutes to train. Our code can be retrieved at `https://github.com/totorato/PotentialNet`.

1 PRELIMINARY AND RELATED WORK

1.1 K-NEAREST NEIGHBOURS

K-NN (k-nearest neighbors, Altman (1992)) is a non-parametric supervised learning algorithm for multi-class classification. Given dataset $\mathcal{S}$ and distance measure d, the output of the algorithm on a test data x is obtained by a plurality vote of the top-k nearest neighbors of x in $\mathcal{S}$ under distance measure d.

1.2 IDENTITY INITIALIZATION

Identity initialization is a non-random weight initialization method in which the weight matrices of an MLP consisting of hidden layers (whose input and output shapes are identical) are initialized with identity matrices. Leveraging identity initialization, Kubota et al. (2021) proposed an explainable deep MLP network, consisting dozens of dense layer with each layer identity initialized. In our model, we adopt a somewhat similar network architecture and employ identity initialization.

Identity initialization is critical to our model in the sense that it combines the power of neural networks with the robustness of k-nearest neighbour algorithm. The importance will be explained in the experiment section (section 3).

1.3 SIAMESE NETWORK

Siamese Neural Network is a class of neural network architectures that contain two or more identical subnetworks. They perform well on small datasets and class-imbalanced datasets according to Koch et al. (2015).

Though it is called "Siamese Network", it is in fact a type of traditional neural network with loss function specially designed for. There are two main ways to train Siamese Networks, which corresponds to two types of loss functions for traditional neural networks, namely Triplet loss and Contrastive loss (Lian et al. (2018)).

For the Triplet loss, each time 3 samples (anchor sample x_0, positive sample x_+ and negative sample x_-) are fed into the network. The positive sample and the anchor sample are from the same class, while the negative sample and the anchor sample are from different classes. The Triplet Loss for this triplet is

$$\mathcal{L}_\theta(x_+, x_-, x_0) = \max(||f_\theta(x_+) - f_\theta(x_0)||^2 - ||f_\theta(x_-) - f_\theta(x_0)||^2 + \alpha, 0)$$

where $\alpha > 0$ encourages larger distance between different classes and smaller distance in the same class.

For the Contrastive Loss, each time two samples are fed into the network. The Contrastive loss for the pair $(x_1, y_1), (x_2, y_2)$ is

$$\mathcal{L}_\theta(x_1, x_2) = \frac{1}{2}(1 + y_1 y_2)||f_\theta(x_1) - f_\theta(x_2)||^2 - \frac{1}{2}(1 - y_1 y_2)\max\{0, m - ||f_\theta(x_1) - f_\theta(x_2)||\}^2$$

where $||\cdot||$ denotes the l_2-norm. We'll use Exponential Contrastive loss for our model, which is similar to Contrastive loss.

1.4 Projected Gradient Descent

Projected Gradient Descent (PGD, Madry et al. (2017)) is a simple but efficient attack method for deep learning models. This method performs well on image classification data sets such as MNIST (Yan et al. (1998)).

PGD is an approximation method for solving the adversarial attack problem:

$$\begin{aligned} x^* = \arg\max_{x'} \mathcal{L}(f_\theta(x'), y) \\ s.t.\ ||x' - x||_p \le \epsilon \end{aligned} \tag{1}$$

where (x, y) is the original test data, f_θ is the model, ϵ is the maximal perturbation allowed and $||\cdot||_p$ is the l_p norm.

Algorithm 1 PGD

```
Input: Training data $x$, Iteration $n$, Stepsize $\alpha$, Model $f_\theta$, Loss function $\mathcal{L}$, Maximal perturbation $\epsilon$
Output: An adversarial sample $x'$
  $x' \leftarrow x$
  while $n > 0$ do
      $\Delta \leftarrow \vec{\nabla}_{x'} \mathcal{L}(f_\theta(x'))$
      $x' \leftarrow x' + \alpha\Delta$
      $x' \leftarrow \text{clip}(x', x - \epsilon, x + \epsilon)$
      $n \leftarrow n - 1$
  end while
```

1.5 Laws of Physics

To make our model more interesting, we also incorporate an idea from physics intuition into our model. The idea worked (refer to section 3.4), but we are not sure why.

According to Quantum Mechanics, the evolution of the state of the universe approximately follows Schrödinger equation

$$i\hbar \frac{d}{dt}\psi(t) = \widehat{H}\psi(t)$$

whose solution is

$$\psi(t) = e^{-i\widehat{H}t/\hbar}\psi(0) = \lim_{\Delta t \to 0} \left(1 - \Delta t \frac{i\widehat{H}}{\hbar}\right)^{\frac{t}{\Delta t}} \psi(0)$$

Let $L_{\Delta t} = 1 - \Delta t \frac{i\widehat{H}}{t}$, then the evolution of ψ is equivalent to the iterated application of $L_{\Delta t}$. (The above derivation also applies to classical mechanics. We choose quantum mechanics here because it's easier to formulate).

Note that the operator $L_{\Delta t}$ is very close to the identity operator. In fact, iterated application of operators close to the identity operator is ubiquitous in maths and physics. For example, suppose our universe is described by a continuous-time Markov Chain, and the infinitesimal matrix for this Markov Chain is $\mathbf{A}$. Then the evolution of $\mathbf{P}(t)$, the probability distribution over states, is described by the backward Kolmogorov differential equation

$$\mathbf{P}'(t) = \mathbf{A}\mathbf{P}(t)$$

whose solution is

$$\mathbf{P}(t) = e^{\mathbf{A}t}\mathbf{P}(0) = \lim_{\Delta t \to 0} (1 - \Delta t \mathbf{A})^{\frac{t}{\Delta t}} \mathbf{P}(t)$$

Let $L_{\Delta t} = 1 - \Delta t \mathbf{A}$, then the evolution of $\mathbf{P}$ is equivalent to the iterated application of $L_{\Delta t}$.

Meanwhile, the layer we use in our model, the Identity Initialized Residual layer (or IdRes layer, which will be described below), also resembles an identity operator with a small perturbation. Note that for the Schrödinger example and Markov Chain example, we apply the *same* operator each time. To mimic this, we only need to share weight between *all* intermediate IdRes layers (after which all IdRes layers represent the same operator).

2 METHODS

2.1 EXPONENTIAL CONTRASTIVE LOSS

Suppose a network f_θ takes an image as input and outputs an $N_S d$ dimensional vector. To train a large-margin classifier, we can use the Exponential Contrastive Loss. To be specific, given a batch of training data $\mathcal{B} = \{(x_i, y_i) | i \in [B]\} (y_i \in [K])$, suppose

$$f_\theta(x_i) = (p_i^1, p_i^2, \ldots, p_i^{N_S})(p_i^j \in \mathbb{R}^d)$$

then the loss is defined as

$$\mathcal{L}_\theta(\mathcal{B}) = \frac{1}{B}\sum_{u=1}^{B}\sum_{v=1}^{B} I\left(y_u - y_v, \sum_{j=1}^{N_S} ||p_u^j - p_v^j||\right)\left(e^{-\lambda \sum_{j=1}^{N_S} ||p_u^j - p_v^j||} - e^{-\lambda l_0}\right) + K\mu \sum_{i=1}^{B}\left(\sum_{j=1}^{N_S} ||p_i^j||\right)^2$$

where $||\cdot||$ denotes the l_2-norm, and

$$I(\Delta, l) = \begin{cases} \frac{1}{K}, & \Delta \neq 0 \\ -1, & \Delta = 0 \wedge l \geq l_0 \\ 0, & \Delta = 0 \wedge 0 \leq l < l_0 \end{cases}$$

(we set $I(\Delta, l) = 0$ when $\Delta = 0 \wedge 0 \leq l < l_0$ to avoid producing too large gradient when points of the same class come too close to each other)

In other words, we interpret our network as a function that maps each $x_i (i \in [B])$ into N_S separate d dimensional spaces. We compute Euclidean distance in each space, sum up the distances and compute the Exponential Contrastive loss, which is what the first term stands for. The loss encourages the points of the same class to come together and points of different classes to move away from each other. The second term can be understood as l_2 regularization on the output of our network, which empirically enhances train stability.

The loss can also be understood as the potential energy of the interaction between K different types of charged particles. Suppose there are K types of charged particles, and the potential energy between type s particle at position p_s and type t particle at position p_t is

$$\Phi_{s,t}(p_s, p_t) = I(s - t, d(p_s, p_t))\left(e^{-\lambda d(p_s, p_t)} - e^{-\lambda l_0}\right)$$

Suppose the charge density of type $i (i \in [K])$ particle is $\rho_i : \Omega \to \mathbb{R}$, and suppose there is an extra central force field with potential $\Phi_c(p) = \mu||p||^2$, then the potential energy of the system is

$$\sum_{s=1}^{K}\sum_{t=1}^{K}\int_{\Omega}\int_{\Omega}\rho_s(p_s)\rho_t(p_t)\Phi_{s,t}(p_s,p_t)dp_s dp_t + \mu\int_{\Omega}\rho(p_s)||p_s||^2 dp_s$$

Treat ρ_i as a distribution over $\mathbb{R}^d$ for each $i \in [K]$. Suppose there is a dataset $\mathcal{S} = \{(p_i, y_i)|i \in [B]\}$ containing B/K points sampled from the corresponding distribution for each of the K distributions, then for each $s \in [K]$,

$$\rho_s(p) \approx \frac{K}{B}\sum_{i=1}^{B} 1_{y_i=s}\delta(p_i - p)$$

Thus the potential energy of the system can be estimated by

$$\frac{K^2}{B}\left(\frac{1}{B}\sum_{u=1}^{B}\sum_{v=1}^{B} I\left(y_u - y_v, d(p_u,p_v)\right)\left(e^{-\lambda d(p_u,p_v)} - e^{-\lambda l_0}\right) + K\mu\sum_{i=1}^{B}||p_i||^2\right)$$

2.2 Identity Initialized ResNet

Inspired by ResNet and identity initialization, we propose Identity Initialized Residual (IdRes) layer. An IdRes layer is composed of an identity mapping branch (possibly on part of the input tensor) and a residual branch. The output of the IdRes layer is the element-wise addition of these two branches. The residual branch can be any function approximator as long as its output shape matches the output shape of the identity mapping branch. The residual branch is zero-initialized before training.

Here we give the full description of the IdRes layer:

$$\mathbf{IdRes}_{f_\theta}(x, (i_0, i_1, \ldots, i_{k-1})) = x[:i_0, :i_1, \ldots, :i_{k-1}] + f_\theta(x)$$

where $f_\theta : \mathbb{R}^{x.shape} \to \mathbb{R}^{i_0 \times i_1 \times \cdots \times i_K}$ is the residual branch, which will be zero-initialized before training. Note that if $x.shape = (i_0, i_1, \ldots, i_{k-1})$, then zero-initializing the residual branch is equivalent to identity initializing the IdRes layer.

The Identity Initialized ResNet (IdResNet) we propose consists of three parts: A Dense layer, denoted as $\mathbf{OtS}$ (Observation to State); A "Decoder" consisting of three identical IdRes layers $\mathbf{Dec}_1, \mathbf{Dec}_2, \mathbf{Dec}_3$ with shared parameters; A "Classifier" consisting of a single IdRes layer $\mathbf{Cl}$. Our model can be expressed as the composition of these layers:

$$\mathbf{IdResNet}_\theta = \mathbf{CL} \circ \mathbf{Dec}_3 \circ \mathbf{Dec}_2 \circ \mathbf{Dec}_1 \circ \mathbf{OtS}$$

The input and output shape of these layers are specified as follows,

$$\mathbf{OtS} : \mathbb{R}^{C \times H \times W} \to \mathbb{R}^{W_R + W_P}$$
$$\mathbf{DEC}_i : \mathbb{R}^{W_R + W_P} \to \mathbb{R}^{W_R + W_P} \quad (i = 1, 2, 3)$$
$$\mathbf{CL} : \mathbb{R}^{W_R + W_P} \to \mathbb{R}^{W_P}$$

where $W_P = N_S d$ is the total length of the output vector. Figure 1 gives an overview of our proposed network.

2.3 Training with Noise

We add random noise to the normalized images before feeding it into our neural network.

We implemented a noise layer $\mathbf{Noise}_{\mathcal{D}}$ parameterized by a distribution $\mathcal{D}$, which is defined as

$$\mathbf{Noise}_{\mathcal{D}} : \mathbb{R}^{C \times H \times W} \to \mathbb{R}^{C \times H \times W}$$
$$(\mathbf{Noise}_{\mathcal{D}}(x))_{c,h,w} = x_{c,h,w} + \Delta_{c,h,w}(c \in [C], h \in [H], w \in [W])$$

where $\Delta_{c,h,w} \sim \mathcal{D}$ $(c \in [C], h \in [H], w \in [W])$ are i.i.d random variables. The noise layer is activated at training and disabled at evaluation and PGD attack.

It is possible to incorporate random noise at evaluation and PGD (If so, then the method is called *randomized smoothing*), however, it might make the PGD attack more computationally expensive (because it will become harder to estimate the gradient). We didn't have enough time for this. So, we leave it for future work to replace our *training with noise* with *randomized smoothing*.

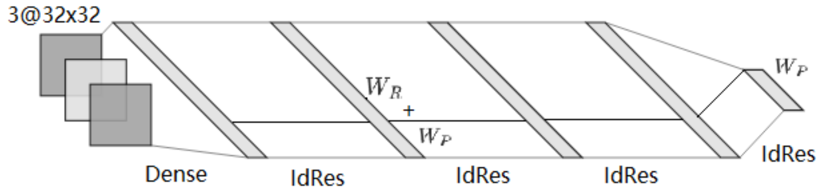


Figure 1: An overview of our proposed network.

2.4 Evaluation using K-nearest Neighbour

When evaluating our model, we feed all training data $x_i^{\text{train}}(i \in [N_{\text{train}}])$ through the model f_θ to obtain the outputs

$$p_i^{1,\text{train}}, p_i^{2,\text{train}}, \ldots, p_i^{N_S,\text{train}} = f_\theta(x_i^{\text{train}})(i \in [N_{\text{train}}])$$

For a test data x_t^{test} $(t \in [N_{\text{test}}])$, we feed it through the model f_θ to obtain the output

$$p_t^{1,\text{test}}, p_t^{2,\text{test}}, \ldots, p_t^{N_S,\text{test}} = f_\theta(x_t^{\text{test}})$$

Then, we compute the total Euclidean distance between $f_\theta(x_t^{\text{test}})$ and $f_\theta(x_i^{\text{train}})$:

$$d_{t,i} = \sum_{j=1}^{N_S} ||p_t^{j,\text{test}} - p_i^{j,\text{train}}||_2$$

where $||\cdot||_2$ stands for l_2-norm.

The predicted label is obtained using k-NN algorithm.

2.5 Projected Gradient Descent

We use a variation of PGD called k-step Fast Gradient Sign Method (kFGSM, Vivek & Babu (2020)) to test the adversarial robustness of our model.

Algorithm 2 kFGSM

```
Input: Training data $x$, Iteration $n$, Stepsize $\alpha$, Model $f_\theta$, Loss function $\mathcal{L}$, Maximal perturbation $\epsilon$
Output: An adversarial sample $x'$
  $x' \leftarrow x$
  while $n > 0$ do
      $\Delta \leftarrow \vec{\nabla}_{x'}\mathcal{L}(f_\theta(x'))$
      $x' \leftarrow x' + \alpha \cdot \text{sgn}(\Delta)$
      $x' \leftarrow \text{clip}(x', x - \epsilon, x + \epsilon)$
      $n \leftarrow n - 1$
  end while
```

This method has the advantage that

(i) Its performance matches that of PGD.

(ii) It works well when the gradient on the input sample is very small or very large.

Since k-NN is not differentiable, we set the loss function to be a function similar to the one we used when training the model (but without the regularization term because it's unnecessary here). To be specific, given training data $\mathcal{D}^{\text{train}} = \{(x_i^{\text{train}}, y_i^{\text{train}}) | i \in [N_{\text{train}}]\}(y_i \in [K])$ and test data $(x_t^{\text{test}}, y_t^{\text{test}})(t \in [N_{\text{test}}])$ suppose

$$\mathbf{IdResNet}_\theta(x_i^{\text{train}}) = (p_i^{1,\text{train}}, p_i^{2,\text{train}}, \ldots, p_i^{N_S,\text{train}})(i \in [N_{\text{train}}], (\forall j \in [N_S], p_i^j \in \mathbb{R}^d))$$

$$\mathbf{IdResNet}_\theta(x_t^{\text{test}}) = (p_t^{1,\text{test}}, p_t^{2,\text{train}}, \ldots, p_t^{N_S,\text{test}})(\forall j \in [N_S], p_t^j \in \mathbb{R}^d)$$

the loss function is defined as

$$\mathcal{L}_\theta(\mathbf{IdResNet}_\theta(x_t^{\text{test}})) = \sum_{i=1}^{N_{\text{train}}} I\left(y_t^{\text{test}} - y_i^{\text{train}}, \sum_{j=1}^{N_S} ||p_t^{j,\text{test}} - p_i^{j,\text{train}}||\right)\left(e^{-\lambda \sum_{j=1}^{N_S} ||p_t^{j,\text{test}} - p_i^{j,\text{train}}||} - e^{-\lambda l_0}\right)$$

3 Experiments

In this section, we conduct several experiments on IdResNet and related networks. We tested our network on the CIFAR-10 dataset (Krizhevsky et al. (2009)), which contains $N_{\text{train}} = 50000$ training data and $N_{\text{test}} = 10000$ testing data. The training dataset is denoted as $\mathcal{D}_{\text{train}}$ and the testing dataset as $\mathcal{D}_{\text{test}}$.

3.1 Experimental Setting

In our experiments, we set network parameter $W_P = 30, W_R = 2048$, training batch size $B = 2048$ and loss function parameter $N_S = 2, d = 15, K\mu = 0.25, l_0 = 7, \lambda = 0.5$. The k for k-NN is set to be 800. We normalize each image (to $\mu = 0, \sigma = 1$) pixilwise before feeding them into IdResNet.

The distribution $\mathcal{D}$ for the noise layer is chosen to be the distribution of $2X - 5$ where $X \sim B(5, 0.5)$ (B stands for the binomial distribution). We chose the Bernoulli distribution because surprisingly, Bernoulli distribution performs better than Uniform distribution and Gaussian distribution. It is interesting to note that the noise is so large that after adding the noise, the image becomes almost impossible for humans to recognize, as shown in Figure 2.

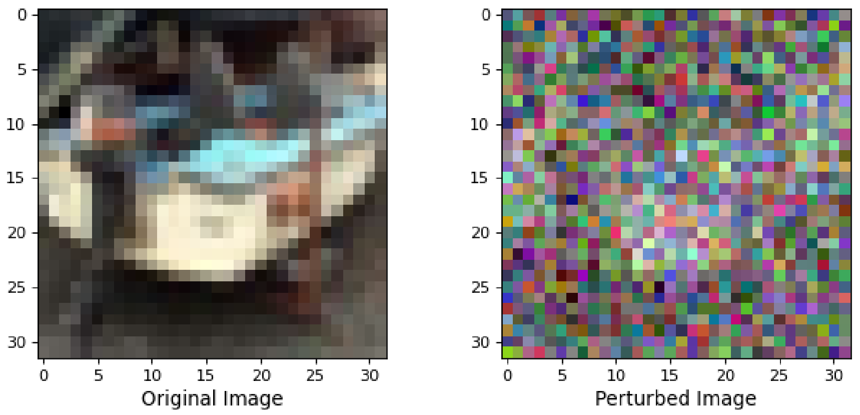


Figure 2: Comparison of original image and image perturbed with noise

We trained our model for 20 epochs and tested it under 30 round l_∞ PGD attack with $\epsilon = 0, 8/255, 16/255$ and stepsize $\epsilon/10$.

3.2 Experimental Results

Our Results are listed in Table 1. The ”Clean” columns shows regular test accuracy. The ”PGD” columns shows test accuracy under 30-round l_∞ PGD attack with corresponding epsilon. All numbers are reported in percentage.

Method	$\epsilon = 8/255$		$\epsilon = 16/255$	
	Clean	PGD	Clean	PGD
IBP (Gowal et al. (2018))	50.99	31.27	31.03	23.34
CROWN-IBP (Zhang et al. (2019))	45.98	34.58	33.94	24.77
CROWN-LBP (Lyu et al. (2021))	48.06	37.95	-	-
l_∞-distance Net (Zhang et al. (2021b))	56.80	37.46	55.05	26.02
l_∞-distance Net (Zhang et al. (2021a))	54.30	41.84	48.50	32.73
K-NN with Identity Initialized ResNet (ours)	53.38	**45.48**	53.38	**35.69**

Table 1: Experimental results and comparison with existing methods.

3.3 Train Efficiency

The training efficiency of our model is significantly higher compared to previous works. For example, l_∞ net took 3.7 hours to train on a NVIDIA RTX 3090 GPU (Zhang et al. (2021b)), while ours only need 5 minutes.

3.4 Ablation

In this section, we conduct ablation study on IdRes layer, parameter sharing and exponential contrastive loss with k-NN. The results are shown in Table 2. We used the same experimental setting as described in section 3.1, except that in test A and B, we use a 5-layer simple MLP and train it with CrossEntropy loss.

From Table 2 we observe that:

- Exponential contrastive loss with k-NN is crucial for a model to be robust under l_∞ PGD attacks.
- Both identity initialization and parameter sharing contributes to the robustness, increasing accuracy by $\sim 7\%$ and $\sim 1.5\%$ respectively. Identity initialization increases both clean accuracy and adversarial robustness. Parameter sharing increases adversarial robustness and decreases clean accuracy.

	identity init resnet	parameter sharing	k-NN and exp loss	Clean	$\epsilon = 8/255$	$\epsilon = 16/255$
A	✗	✗	✗	54.27	7.25	0.28
B	✓	✗	✗	54.73	5.89	0.15
C	✗	✗	✓	53.34	37.38	26.54
D	✗	✓	✓	52.67	39.09	28.69
E	✓	✗	✓	54.24	44.48	34.45
FULL	✓	✓	✓	53.38	**45.48**	**35.69**

Table 2: Ablation studies on CIFAR-10 dataset.

3.5 Justification of K-NN with IdResNet

We conjecture that k-NN with IdResNet combines the robustness of k-NN with the power of neural networks. To illustrate this idea, we random initialized $\mathbf{OtS}$, identity initialized $\mathbf{Dec}_1, \mathbf{Dec}_2, \mathbf{Dec}_3, \mathbf{Cl}$ and test the accuracy of our network on CIFAR-10 *without any training*. Note that we are evaluating with k-NN, which for input data x compares $\mathbf{IdResNet}_\theta(x)$ with $\{(\mathbf{IdResNet}_\theta(x_i), y_i)|(x_i, y_i) \in \mathcal{D}_{\text{train}}\}$. So even though the network is not trained with $\mathcal{D}_{\text{train}}$, it is possible for our model to achieve nontrivial classification accuracy on $\mathcal{D}_{\text{test}}$. The result of this test is interesting:

(i) Test accuracy with clean data is 26.31%

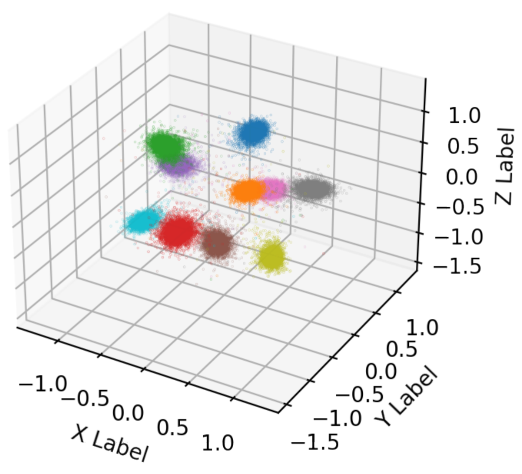
(ii) Test accuracy under 30-round, $\epsilon = 8/255$ l_∞ PGD attack is 9.94%, which is significantly higher than simple MLP trained with $\mathcal{D}_{\text{train}}$.

This test demonstrated that our network is able to leverage the robustness of k-NN algorithm. For comparison, we random initialized a 5-layer MLP (width of hidden layers $= 2078$) and test it on $\mathcal{D}_{\text{test}}$. The result is as expected:

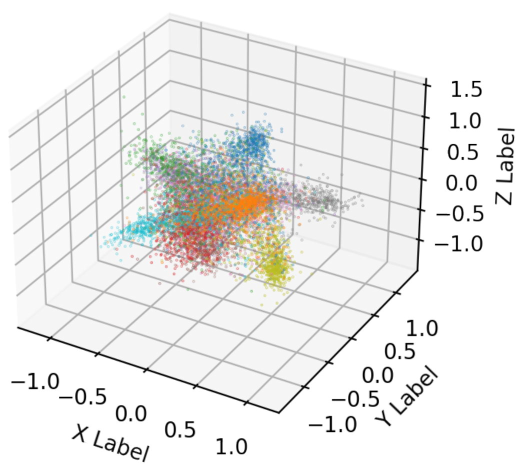
(i) Test accuracy with clean data is 9.81%

(ii) Test accuracy under 30-round, $\epsilon = 8/255$ l_∞ PGD attack is 0.0%

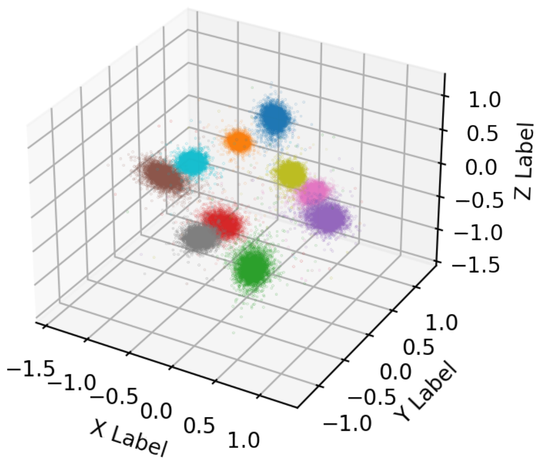
3.6 Clustering Effect of Exponential Contractive Loss



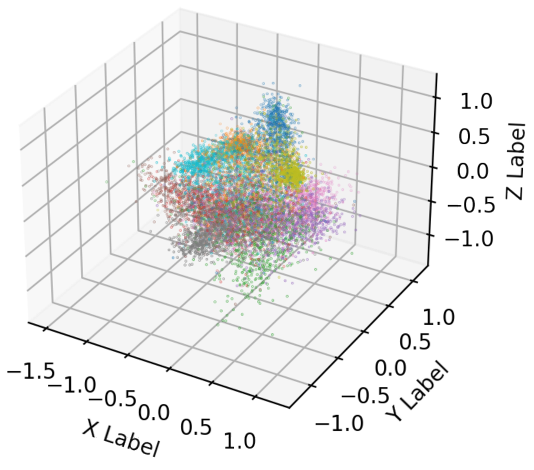
(a)



(b)



(c)



(d)

Figure 3: (a) $\mathcal{P}^{1,\text{train}}$; (b) $\mathcal{P}^{1,\text{test}}$;(c) $\mathcal{P}^{2,\text{train}}$; (d) $\mathcal{P}^{2,\text{test}}$;

To illustrate the clustering effect of exponential contrastive loss, we set $N_S = 4$ and $d = 3$ and train $\mathbf{IdResNet}_\theta$ on $\mathcal{D}_{\text{train}}$ without adding noise to images. Then, we calculate

$$p_i^{1,\text{train}}, p_i^{2,\text{train}}, p_i^{3,\text{train}}, p_i^{4,\text{train}} = \mathbf{IdResNet}_\theta(x_i^{\text{train}}) \text{ for } (x_i^{\text{train}}, y_i^{\text{train}}) \in \mathcal{D}_{\text{train}}$$
$$p_i^{1,\text{test}}, p_i^{2,\text{test}}, p_i^{3,\text{test}}, p_i^{4,\text{test}} = \mathbf{IdResNet}_\theta(x_i^{\text{test}}) \text{ for } (x_i^{\text{test}}, y_i^{\text{test}}) \in \mathcal{D}_{\text{test}}$$

where $p_i^{j,\text{train}}, p_i^{j,\text{test}} \in \mathbb{R}^3$ for all i, j. Finally, we plot the point sets

$$\mathcal{P}^{1,\text{train}} = \{p_i^{1,\text{train}} | i \in [50000]\}, \mathcal{P}^{1,\text{test}} = \{p_i^{1,\text{test}} | i \in [10000]\}$$
$$\mathcal{P}^{2,\text{train}} = \{(p_i^{2,\text{train}} | i \in [50000]\}, \mathcal{P}^{2,\text{test}} = \{p_i^{2,\text{test}} | i \in [10000]\}$$

the plots are shown in Figure 3.

4 Future Work

4.1 Parameter Tuning

We haven't carefully adjusted the hyper-parameters and design choices in this project. We believe that theses hyper-parameters still needs to be tuned:

(i) Loss function parameters: N_S, d, μ
(ii) Number of layers in our model
(iii) Width of hidden layers: $W_R + W_P$
(iv) The parameter k of k-NN

and these design choices still needs to be adjusted:

(i) The distance metric in k-NN
(ii) The distribution of noise: $\mathcal{D}$

4.2 Extend to CNN

The residual branch in our IdRes layer can be any function approximators. One promising choice is CNN. Since CNN layers can be incorporated into residual layers (He et al. (2015)), and there exists euclidean distance metric for images (Wang et al. (2005)), our model can be extended to include CNN.

However, in our tentative experiments, we found that IdRes with CNN suffers from a problem related to receptive field size. Since the output of intermediate IdRes layers has output shape identical to input shape, the residual CNN layers can't perform pooling. Thus, the receptive field of output neurons are significantly smaller than that of regular CNNs. We observe that when we set convolution size $= 3 \times 3$, IdRes with CNN underperforms IdRes with dense layers (on clean data), but when we set convolution size $= 5 \times 5$, IdRes with CNN outperforms IdRes with dense layers (on clean data), but still underperforms regular CNN by a large margin.

We haven't tested the adversarial robustness of IdRes with CNN. We leave it for future work to figure out how to incorporate CNN into our model to boost test accuracy and adversarial robustness.

4.3 Adversarial Attack for Our Model

When performing Projected Gradient Descent, we set the loss function to be identical to the one we use when we train the model. However, performing PGD on this loss function might not be the most effective way to attack the K-NN classifier we use. It is possible that by PGD on some well-designed loss functions, our model will exhibit weaker adversarial robustness. It is also possible that attack methods other than PGD can do better at attacking our model.

In our experiments, we haven't found any other loss functions for PGD or other adversarial attack methods which perform better than the one we used. We leave it for future work to either find an effective attack method for our model, or to prove that our model is indeed robust.

Note that though there have already been some works on attacking Siamese Networks, such as Wu et al. (2019), their attack objectives are very different from ours, hence it is likely that our model is immune to those attacks.

4.4 Randomized Smoothing and Adversarial Training

Previous works have shown certified robustness through adding random noise to the input of the network (Cohen et al. (2019)). As mentioned in section 2, our *training with noise* can be replaced with *randomized smoothing*, which is likely to increase the adversarial robustness of our model.

It is also possible to replace *training with noise* with *adversarial training*. However, this will significantly increase training time.

References

Naomi S Altman. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

Jeremy M Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing, 2019. URL `https://arxiv.org/abs/1902.02918`.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Relja Arandjelovic, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL `https://arxiv.org/abs/1512.03385`.

Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, pp. 0. Lille, 2015.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Shohei Kubota, Hideaki Hayashi, Tomohiro Hayase, and Seiichi Uchida. Layer-wise interpretation of deep neural networks using identity initialization. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3945–3949. IEEE, 2021.

Zheng Lian, Ya Li, Jianhua Tao, and Jian Huang. Speech emotion recognition via contrastive loss under siamese networks. In *Proceedings of the Joint Workshop of the 4th Workshop on Affective Social Multimedia Computing and first Multi-Modal Affective Computing of Large-Scale Multimedia Data*. ACM, oct 2018. doi: 10.1145/3267935.3267946. URL `https://doi.org/10.1145%2F3267935.3267946`.

Zhaoyang Lyu, Minghao Guo, Tong Wu, Guodong Xu, Kehuan Zhang, and Dahua Lin. Towards evaluating and training verifiably robust neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4308–4317, 2021.

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.

BS Vivek and R Venkatesh Babu. Single-step adversarial training with dropout scheduling. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 947–956. IEEE, 2020.

Liwei Wang, Yan Zhang, and Jufu Feng. On the euclidean distance of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1334–1339, 2005. doi: 10.1109/TPAMI.2005.165.

Xugang Wu, Xiaoping Wang, Xu Zhou, and Songlei Jian. Sta: Adversarial attacks on siamese trackers, 2019. URL `https://arxiv.org/abs/1909.03413`.

L Yan, C Corinna, and CJ Burges. The mnist dataset of handwritten digits, 1998.

Bohang Zhang, Tianle Cai, Zhou Lu, Di He, and Liwei Wang. Towards certifying l-infinity robustness using neural networks with l-inf-dist neurons. In *International Conference on Machine Learning*, pp. 12368–12379. PMLR, 2021a.

Bohang Zhang, Du Jiang, Di He, and Liwei Wang. Boosting the certified robustness of l-infinity distance nets. *arXiv preprint arXiv:2110.06850*, 2021b.

Huan Zhang, Hongge Chen, Chaowei Xiao, Sven Gowal, Robert Stanforth, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. *arXiv preprint arXiv:1906.06316*, 2019.