Parameterized Rate-Distortion Stochastic Encoder - Supplementary Material

Quan Hoang [1] **Trung Le** [1] **Dinh Phung** [1]

We provide additional details to supplement the main paper. We focus on some key proofs for self-completeness, details on mini-batch weighted sampling scheme, our proposed posterior matching objective, pseudo-codes, experimental parameters and additional experimental results for reproducibility.

1. Theoretical framework

1.1. Proof of Lemma 1.

Lemma 1. *Let* $p(\mathbf{x},\mathbf{z}) = p(\mathbf{x})\,p(\mathbf{z}\mid\mathbf{x})$ *be a given joint distribution. The distribution* $q^\star(\mathbf{z})$ *that minimizes the Kullback-Leibler (KL) divergence* $KL[p(\mathbf{x},\mathbf{z}) \parallel p(\mathbf{x})\,q(\mathbf{z})]$ *is the marginal distribution* $p(\mathbf{z})$ *corresponding to* $p(\mathbf{z}\mid\mathbf{x})$*:*

$$\underset{q(\mathbf{z})}{argmin}\; KL[p(\mathbf{x},\mathbf{z}) \parallel p(\mathbf{x})\,q(\mathbf{z})] = p(\mathbf{z})$$

where

$$p(\mathbf{z}) = \sum_{\mathbf{x}} p(\mathbf{x})\,p(\mathbf{z}\mid\mathbf{x})$$

Proof. Consider

$$\begin{aligned}
&KL[p(\mathbf{x},\mathbf{z}) \parallel p(\mathbf{x})\,q(\mathbf{z})] - KL[p(\mathbf{x},\mathbf{z}) \parallel p(\mathbf{x})\,p(\mathbf{z})] \\
&= \sum_{\mathbf{x},\mathbf{z}} p(\mathbf{x},\mathbf{z}) \log \frac{p(\mathbf{x},\mathbf{z})}{p(\mathbf{x})\,q(\mathbf{z})} - \sum_{\mathbf{x},\mathbf{z}} p(\mathbf{x},\mathbf{z}) \log \frac{p(\mathbf{x},\mathbf{z})}{p(\mathbf{x})\,p(\mathbf{z})} \\
&= \sum_{\mathbf{x},\mathbf{z}} p(\mathbf{x})\,p(\mathbf{z}\mid\mathbf{x}) \log \frac{p(\mathbf{x})\,p(\mathbf{z})}{p(\mathbf{x})\,q(\mathbf{z})} \\
&= \sum_{\mathbf{z}} p(\mathbf{z}) \log \frac{p(\mathbf{z})}{q(\mathbf{z})} \\
&= KL[p(\mathbf{z}) \parallel q(\mathbf{z})] \geq 0 \qquad (1)
\end{aligned}$$

The inequality in 1 results from the non-negativity of the KL divergence. □

[1]Department of DSAI, Faculty of Information Technology, Monash University, Australia. Correspondence to: Quan Hoang <qhoang.ai@gmail.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

1.2. Proof of Equation 4.

Lemma 2. *For a fixed distribution* $q(\mathbf{z})$*, the optimal distribution* $p^\star(\mathbf{z}\mid\mathbf{x})$ *that minimizes the functional* $\mathcal{F}[p(\mathbf{z}\mid\mathbf{x}), q(\mathbf{z})]$*:*

$$\begin{aligned}
\mathcal{F}[p(\mathbf{z}\mid\mathbf{x}), q(\mathbf{z})] = &\sum_{\mathbf{x},\mathbf{z}} p(\mathbf{x})\,p(\mathbf{z}\mid\mathbf{x}) \log \frac{p(\mathbf{x})\,p(\mathbf{z}\mid\mathbf{x})}{p(\mathbf{x})\,q(\mathbf{z})} \\
&+ \beta \sum_{\mathbf{x},\mathbf{z}} p(\mathbf{x})\,p(\mathbf{z}\mid\mathbf{x})\,d(\mathbf{x},\mathbf{z}) \qquad (2)
\end{aligned}$$

is

$$p^\star(\mathbf{z}\mid\mathbf{x}) = \frac{q(\mathbf{z}) \exp(-\beta d(\mathbf{x},\mathbf{z}))}{\sum_{\mathbf{z}} q(\mathbf{z}) \exp(-\beta d(\mathbf{x},\mathbf{z}))} \qquad (3)$$

Proof. Using the method of Lagrange multipliers, we set up the functional:

$$\begin{aligned}
\mathcal{J}(p(\mathbf{z}\mid\mathbf{x})) = &\sum_{\mathbf{x},\mathbf{z}} p(\mathbf{x})\,p(\mathbf{z}\mid\mathbf{x}) \log \frac{p(\mathbf{z}\mid\mathbf{x})}{q(\mathbf{z})} \\
&+ \beta \sum_{\mathbf{x},\mathbf{z}} p(\mathbf{x})\,p(\mathbf{z}\mid\mathbf{x})\,d(\mathbf{x},\mathbf{z}) \\
&+ \sum_{\mathbf{x}} \nu(\mathbf{x}) \left[\sum_{\mathbf{z}} p(\mathbf{z}\mid\mathbf{x}) - 1\right]
\end{aligned}$$

where the $\nu(\mathbf{x})$ is the Lagrange multiplier to constrain $p(\mathbf{z}\mid\mathbf{x})$ to be a conditional probability distribution. Taking the functional derivative of $\mathcal{J}(p(\mathbf{z}\mid\mathbf{x}))$ with respect to $p(\mathbf{z}\mid\mathbf{x})$, we have:

$$\begin{aligned}
\frac{\delta\mathcal{J}}{\delta p(\mathbf{z}\mid\mathbf{x})} = &\; p(\mathbf{x}) \log \frac{p(\mathbf{z}\mid\mathbf{x})}{q(\mathbf{z})} + p(\mathbf{x}) + \\
&+ \beta p(\mathbf{x})\,d(\mathbf{x},\mathbf{z}) + \nu(\mathbf{x})
\end{aligned}$$

Setting the functional derivative to 0, we have:

$$p(\mathbf{x}) \left[\log \frac{p^\star(\mathbf{z}\mid\mathbf{x})}{q(\mathbf{z})} + 1 + \beta d(\mathbf{x},\mathbf{z}) + \frac{\nu(\mathbf{x})}{p(\mathbf{x})}\right] = 0$$

or

$$p^\star(\mathbf{z}\mid\mathbf{x}) = q(\mathbf{z}) \exp\left[-\beta d(\mathbf{x},\mathbf{z}) - 1 - \frac{\nu(\mathbf{x})}{p(\mathbf{x})}\right]$$

Since $\sum_{\mathbf{z}} p^\star(\mathbf{z} \mid \mathbf{x}) = 1$ and note that $-1 - \nu(\mathbf{x})/p(\mathbf{x})$ ís constant w.r.t. $\mathbf{z}$, one must have:

$$p^\star(\mathbf{z} \mid \mathbf{x}) = \frac{q(\mathbf{z}) \exp(-\beta d(\mathbf{x}, \mathbf{z}))}{\sum_{\mathbf{z}} q(\mathbf{z}) \exp(-\beta d(\mathbf{x}, \mathbf{z}))}$$

□

1.3. Mini-batch Weighted Sampling (MWS)

In this part, we present Mini-batch Weighted Sampling (MWS) introduced in (Chen et al., 2018). Given the empirical dataset $(\mathbf{x}_1, ..., \mathbf{x}_N)$, we identify each data point with a unique integer index and define $p(\mathbf{z} \mid n) = p(\mathbf{z} \mid \mathbf{x}_n)$, $p(\mathbf{z}, n) = p(\mathbf{z} \mid n) p(n) = p(\mathbf{z} \mid n) \frac{1}{N}$ and $p(\mathbf{z}) = \sum_{n=1}^{N} p(\mathbf{z} \mid n) p(n)$. Let $\mathcal{B}_M = \{n_1, ..., n_M\}$ be a mini-batch of M indices sampled i.i.d from the discrete uniform distribution $\mathcal{U}\{1, N\}$. The probability of a sampled mini-batch $\mathcal{B}_M$ is $p(\mathcal{B}_M) = (1/N)^M$. Let $r(\mathcal{B}_M \mid n)$ denote the probability of a sampled mini-batch that consists of a fixed element n and other elements sampled i.i.d from $\mathcal{U}\{1, N\}$. Then, $r(\mathcal{B}_M \mid n) = (1/N)^{M-1}$. As the objective is to maximize $\mathbb{E}_{p(\mathbf{z})} \log p(\mathbf{z})$, we can approximate its lower bound:

$$\begin{aligned}
&\mathbb{E}_{p(\mathbf{z})} [\log p(\mathbf{z})] \\
=&\mathbb{E}_{p(\mathbf{z},n)} \left[\log \mathbb{E}_{n' \sim p(n)} [p(\mathbf{z} \mid n')]\right] \\
=&\mathbb{E}_{p(\mathbf{z},n)} \left[\log \mathbb{E}_{p(\mathcal{B}_M)} \left[\frac{1}{M} \sum_{m=1}^{M} p(\mathbf{z} \mid n_m)\right]\right] \\
\geq&\mathbb{E}_{p(\mathbf{z},n)} \left[\log \mathbb{E}_{r(\mathcal{B}_M|n)} \left[\frac{p(\mathcal{B}_M)}{r(\mathcal{B}_M \mid n)} \frac{1}{M} \sum_{m=1}^{M} p(\mathbf{z} \mid n_m)\right]\right] \quad (4) \\
=&\mathbb{E}_{p(\mathbf{z},n)} \left[\log \mathbb{E}_{r(\mathcal{B}_M|n)} \left[\frac{1}{NM} \sum_{m=1}^{M} p(\mathbf{z} \mid n_m)\right]\right] \quad (5)
\end{aligned}$$

The inequality 4 is because the support of r is a subset of that of p. During training, Chen et al. (2018) approximates $\mathbb{E}_{p(\mathbf{z})} \log p(\mathbf{z})$ from a mini-batch of samples $\{n_1, ..., n_M\}$:

$$\mathbb{E}_{p(\mathbf{z})} \log p(\mathbf{z}) \approx \frac{1}{M} \sum_{i=1}^{M} \left[\log \sum_{j=1}^{M} p(\mathbf{z}_i \mid n_j) - \log(NM)\right] \quad (6)$$

where $\mathbf{z}_i$ is sampled from $p(\mathbf{z} \mid n_i)$. Empirically, we find that the model is easier to train when dropping $p(\mathbf{z}_i \mid n_i)$ from the summation in Eq. 6. Thus, we use the following estimator:

$$\mathbb{E}_{p(\mathbf{z})} \log p(\mathbf{z}) \approx \frac{1}{M} \sum_{i=1}^{M} \left[\log \sum_{j \neq i} p(\mathbf{z}_i \mid n_j) - \log(NM)\right] \quad (7)$$

1.4. Posterior matching objective

Given two diagonal Gaussian distributions $\mathcal{N}(\mu, \text{diag}(\sigma^2))$ and $\mathcal{N}(\mu', \text{diag}(\sigma'^2))$ on the d-dimensional space $\mathbb{R}^d$, we employ the following discrepancy measure:

$$\sum_{j=1}^{d} \alpha_j |\mu_j - \mu'_j| + \sum_{j=1}^{d} \frac{1}{d} |\log \sigma_j - \log \sigma'_j| \quad (8)$$

where

$$\alpha_j = \max\left(\frac{1}{d}, \frac{|\mu_j - \mu'_j|}{\sum_{k=1}^{d} |\mu_k - \mu'_k|}\right)$$

The discrepancy in Eq. 8 consists of two terms penalizing the mismatches between the means and variances of the two distributions. For the means, the weight α_j adds more penalty to the dimensions where the mismatches between means of the two distributions are larger than average.

1.5. Input sampling for posterior matching

To sample the set of points $\mathbf{x}' \overset{\text{iid}}{\sim} \mathcal{S}(\mathbf{x})$ that should be in the same partitioning, we consider the following procedure. First, a perturbation magnitude s is sampled from the uniform distribution $\mathcal{U}(0, m)$ where m is a predefined maximum value. Then, a noise $\mathbf{u}$ is sampled from the standard Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$. Finally, $\mathbf{x}'$ is sampled as $\mathbf{x} + s\mathbf{u}$. Sampling the perturbation magnitude s ensures the diversity of the distance between $\mathbf{x}'$ and $\mathbf{x}$. The maximum value m is set to $12/255$ for Cifar-10 and $16/255$ for ImageNet.

1.6. Algorithms

The pseudo-code of learning PARADISE for the supervised and unsupervised settings are described in Alg. 1 and Alg. 2, respectively. To compute the mini-batch approximation of rate, the differential entropy of the Gaussian posteriors $\mathcal{N}(\mu_i, \text{diag}(\sigma_i^2))$ is computed using the following formula:

$$\mathbb{H}(\mathcal{N}(\mu_i, \text{diag}(\sigma_i^2))) = \sum_{j=1}^{D} \left(\frac{\log 2\pi}{2} + \frac{1}{2} + \log \sigma_{ij}\right)$$

where D is the dimensionality of μ_i and σ_i. Let $\mathbf{z}_i$ be sampled from the Gaussian posteriors $\mathcal{N}(\mu_i, \text{diag}(\sigma_i^2))$, $\text{MWS}\left(\{(\mu_i, \sigma_i)\}_{i=1}^{M}\right)$ is approximated following Eq. (7) as:

$$\frac{1}{M} \sum_{i=1}^{M} \log \sum_{j \neq i} \mathcal{N}(\mathbf{z}_i \mid \hat{\mu}_j, \text{diag}(\hat{\sigma}_j^2))$$

where $\hat{\mu}_j = \text{StopGradient}(\mu_j)$, $\hat{\sigma}_j = \text{StopGradient}(\sigma_j)$ and $\mathcal{N}(\mathbf{z}_i \mid \hat{\mu}_j, \text{diag}(\hat{\sigma}_j^2))$ is the probability density:

$$\mathcal{N}(\mathbf{z}_i \mid \hat{\mu}_j, \text{diag}(\hat{\sigma}_j^2)) = \frac{\exp\left[-\frac{1}{2} \sum_{t=1}^{D} \left(\frac{z_{it} - \hat{\mu}_{jt}}{\hat{\sigma}_{jt}}\right)^2\right]}{\prod_{t=1}^{D} (\sqrt{2\pi} \hat{\sigma}_{jt})}$$

For the supervised setting, feeding forward the sampled representation to the softmax classifier is computationally cheap, so we set $K = 12$ in our experiments.

2. Experiment Details

We use TensorFlow version 1.12 to implement our experiments and will release the code after publication. For all experiments, we parameterize $p(\mathbf{z} \mid \mathbf{x})$ as a diagonal Gaussian distribution $\mathcal{N}\left(\mu_\theta(\mathbf{x}), \text{diag}\left(\sigma_\theta^2(\mathbf{x})\right)\right)$ with $\mu_\theta(\mathbf{x})$ and $\sigma_\theta(\mathbf{x})$ being outputs of a neural network with parameter θ. Two separate linear layers are applied on the top layer of the neural network to compute $\mu_\theta(\mathbf{x})$ and $\sigma_\theta(\mathbf{x})$. Each element of $\sigma_\theta(\mathbf{x})$ is bound to $(0, 1)$ by applying the sigmoid function to the outputs of the linear operation. For supervised and robust learning experiments, we modify a standard CNN architecture by adding two separate linear layers on the top layer for the computation of $\mu_\theta(\mathbf{x})$ and $\sigma_\theta(\mathbf{x})$. In addition, the classifier is a softmax classifier with a linear layer applied to $\mathbf{z}$ sampled from $\mathcal{N}\left(\mu_\theta(\mathbf{x}), \text{diag}\left(\sigma_\theta^2(\mathbf{x})\right)\right)$ to compute the logits to the softmax function. At inference time, $\mu_\theta(\mathbf{x})$ is fed to the softmax classifier instead of sampling $\mathbf{z}$ from $\mathcal{N}\left(\mu_\theta(\mathbf{x}), \text{diag}\left(\sigma_\theta^2(\mathbf{x})\right)\right)$, which we find leading to similar results.

MNIST For the experiment on MNIST (LeCun et al., 1998), the encoder is a neural network with two fully-connected layers of 1,000 hidden units using the ReLU activation function. It takes the flatten input of 784 dimensions and outputs two 2-dimensional vectors $\mu_\theta(\mathbf{x})$ and $\sigma_\theta(\mathbf{x})$. The decoder has a similar architecture but in reverse order. It takes as inputs the 2-dimensional vector $\mathbf{z}$ sampled from $\mathcal{N}\left(\mu_\theta(\mathbf{x}), \text{diag}\left(\sigma_\theta^2(\mathbf{x})\right)\right)$, and outputs the reconstruction vector of 784-dimensions. We use Adam (Kingma & Ba, 2014) optimizer with the learning rate of 0.0001, the first order momentum of 0.9 and the second-order momentum of 0.999. The batch size is set to 128. The number of iterations is set to 50,000.

CelebA For the experiment on CelebA (Liu et al., 2015), the architectures of the encoder and decoder follow those in (Chen et al., 2018) except that the number of dimensions of $\mathbf{z}$ is 64. We find that adding constraint to the code space makes it easier to generate images from random noise. Therefore, we bound the value of $\mu_\theta(\mathbf{x})$ to the range $[-1, 1]$ by applying the $\tanh$ function after the linear operation, and bound $\sigma_\theta(\mathbf{x})$ to the range $[\exp(-1), \exp(1)]$ by applying the $\tanh$ function after the linear operation to compute $\log(\sigma_\theta(\mathbf{x}))$. The mean absolute value is used instead of mean square error as the distortion measure to get better image reconstruction. We use Adam (Kingma & Ba, 2014) optimizer with the learning rate of 0.0001, the first order momentum of 0.9 and the second-order momentum of 0.999. The rate parameter α, the batch size and the number of iterations is set to 0.0001, 128 and 100,000 respectively.

Cifar-10 and ImageNet For supervised learning, we conduct experiment on the Cifar-10 data set using the pre-activation ResNet (He et al., 2016b) with different number of layers, including 20, 32, 56 and 110 (He et al., 2016a). The number of dimensions for $\mathbf{z}$ is 64. For robust learning experiment, we apply use the base wide ResNet WRN-28-10 architecture (Zagoruyko & Komodakis, 2016) for Cifar-10 and the pre-activation ResNet with 34 layers for ImageNet (He et al., 2016a). The number of dimensions of for $\mathbf{z}$ is 128 for WRN-28-10 on Cifar-10 and 512 for ResNet-34 on ImageNet. Data pre-processing, hyper-parameters and learning schedule strictly follow those for standard model (Zagoruyko & Komodakis, 2016; He et al., 2016a). For ImageNet, however, we reduce learning rate by 10 times at epoch 90 in addition to the reduction at epoch 30 and 60 as in (He et al., 2016b). To select the rate parameter α, an heuristic is employed where starting with a value such as 0.001, we train a model for a few epochs to observe the rate on the train and validation sets. We keep dividing α by 10 until there is no sign of overfitting on the rate. α is set to 0.0001 for Cifar-10 and 0.00001 for ImageNet.

For robust learning experiments, we train PARADISE-PM and DVIB-PM on Cifar-10 with parameter search over $\alpha \in \{1e-4, 1e-5\}$ and $\gamma \in \{10, 25, 50\}$. The best γ is 25 for both models while the best α is $1e-4$ for PARADISE-PM and $1e-5$ for DVIB-PM. On ImageNet, α is set to $1e-5$ for PARADISE-PM and γ is searched over $\{50, 100, 200\}$. The best γ is 100.

Adversarial attacks We evaluate adversarial robustness using strong PGD-based untargeted attack, random targeted attack and multi-targeted attack (Madry et al., 2017; Gowal et al., 2019). We employ the loss function and optimization settings for PGD attacks from (Qin et al., 2019). The loss functions are summarized in Table 1. For attacks on Cifar-10, we use Adam optimizer (Kingma & Ba, 2014) with the update on the adversarial perturbation of the form $\delta \leftarrow Proj_\epsilon(\delta + \eta Adam(\nabla_\delta l(\mathbf{x} + \delta, y)))$ where $Proj_\epsilon(\delta) = \underset{\xi:\|\xi\|_\infty \leq \epsilon}{\text{argmin}} \|\delta - \xi\|_\infty$. For multi-targeted attack, the learning rate η is set to 0.1 and the number of steps is 200. For untargeted and random-targeted attacks, we use the learning rate schedule of $\eta = 0.1$ for the first 100 steps, then 0.01 for the next 50 steps and 0.001 for the last 50 steps. For untargeted attack on each single image, we use 20 different random initializations for perturbations to craft 20 attacks and consider the attacks successful if any of them can cause the model to predict incorrectly. For ImageNet, we craft attacks on 2,000 random validation images, including 2 images for each class, and use only one random initialization for the perturbation due to time constraint.

Algorithm 1 Alternative training of using stochastic gradient descent.

for number of training iterations
* Sample a minibatch of M data-label pairs $\{(\mathbf{x}_i, y_i)\}_{i=1}^{M}$
* Feed forward the minibatch to the encoder to compute the parameters (μ_i, σ_i)
* $\epsilon_i^k \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I})$ for $i \in [1, M], k \in [1, K]$
* $\mathbf{z}_i^k := \mu_i + \epsilon_i^k \sigma_i$
* Feed forward $\mathbf{z}_i^k$ to the softmax classifer to get the conditional distribution $p_\phi\left(y \mid \mathbf{z}_i^k\right)$
* rate $:= -\frac{1}{M}\sum_{i=1}^{M} \mathbb{H}\left(\mathcal{N}\left(\mu_i, \text{diag}\left(\sigma_i^2\right)\right)\right) - \text{MWS}\left(\{(\mu_i, \sigma_i)\}_{i=1}^{M}\right)$
* distortion $:= -\frac{1}{MK}\sum_{i=1}^{M}\sum_{k=1}^{K} \log p_\phi\left(y_i \mid \mathbf{z}_i^k\right)$
* $\mathcal{L} :=$ distortion $+ \alpha \times$ rate
* Update θ and ϕ by descending along the gradients $\nabla_\theta \mathcal{L}$ and $\nabla_\phi \mathcal{L}$.
endfor

Algorithm 2 Alternative training of using stochastic gradient descent.

for number of training iterations
* Sample a minibatch of M data points $(\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M)$
* Feed forward the minibatch to the encoder to compute the parameters (μ_i, σ_i)
* $\epsilon_i \overset{\text{iid}}{\sim} \mathcal{N}(\mathbf{0}, \mathbf{I})$ for $i \in [1, M]$
* $\mathbf{z}_i := \mu_i + \epsilon_i \sigma_i$
* Feed forward $\mathbf{z}_i$ to the decoder to get the reconstruction $\hat{\mathbf{x}}_i$
* rate $:= -\frac{1}{M}\sum_{i=1}^{M} \mathbb{H}\left(\mathcal{N}\left(\mu_i, \text{diag}\left(\sigma_i^2\right)\right)\right) - \text{MWS}\left(\{(\mu_i, \sigma_i)\}_{i=1}^{M}\right)$
* distortion $:= -\frac{1}{M}\sum_{i=1}^{M} \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2$
* $\mathcal{L} :=$ distortion $+ \alpha \times$ rate
* Update θ and ϕ by descending along the gradients $\nabla_\theta \mathcal{L}$ and $\nabla_\phi \mathcal{L}$.
endfor

Table 1: The loss functions for different attacks we use for evaluation of adversarial robustness. $f_c(\mathbf{x})$ represents the logit corresponding to the class $c \in [1, C]$. t and r denote the ground-truth and a random class, respectively. s denotes the class with highest logit value excluding the logit corresponding to the correct class t. For random-targeted attack, r is chosen randomly (and is different from t) at the beginning of the optimization. For multi-targeted attack, we maximize $f_i(\mathbf{x}+\delta) - f_t(\mathbf{x}+\delta)$ for all $i \in [1, C]$ and consider the attack successful if any of the attacks on each target class i causes the classifier to predict a class different from t. The evaluation metric for random-targeted attack is attack success rate - the percentage of times an attacker successfully causes the model to predict a target class r. Lower attack success rate is better.

Attack Name	**Loss Function**	**Evaluation Metric**
Random-Targeted	$\max_{\delta \in B(\mathbf{x},\epsilon)} f_r(\mathbf{x}+\delta) - f_t(\mathbf{x}+\delta)$	Attack Success Rate
Untargeted	$\max_{\delta \in B(\mathbf{x},\epsilon)} f_s(\mathbf{x}+\delta) - f_t(\mathbf{x}+\delta)$	Adversarial Accuracy
Multi-Targeted	$\max_{\delta \in B(\mathbf{x},\epsilon)} \max_{i \in [1,C]} f_i(\mathbf{x}+\delta) - f_t(\mathbf{x}+\delta)$	Adversarial Accuracy

Table 2: Test accuracy (in %) on Cifar-10. To compute test accuracy, we take the average results of models trained using 10 different random seeds.

	RN-20	RN-32	RN-56	RN-110
Standard	91.20	92.16	92.81	93.22
PARADISE	**91.56**	**92.51**	**92.95**	**93.48**

Table 3: Test accuracy (in %) on ImageNet.

	Top-1	Top-5
Standard	72.54	90.73
PARADISE	71.73	90.27

3. Additional experiment results

3.1. CelebA

To generate random images using PARADISE, we first sample random noise $\mathbf{z}$ from a multivariate Gaussian distribution fit to the empirical aggregate posterior of unseen samples. The sampled noise $\mathbf{z}$ is fed to the decoder to generate a face image. Then, we perform a series of reconstructions. The idea is to explore what each neighborhood of $\mathbf{z}$ represents. Fig. 1 shows that this series of reconstructions result in images of similar-looking faces with transformation such as smile, face orientation and shape, eyeglasses, gender, beard and hair style. Interestingly, we also observe linearization of these semantic transformations (Fig. 2). For instance, adding the difference between the $\mathbf{z}$-vector of a smiling face and that of a neutral face to the $\mathbf{z}$-vector of another person's neutral face generates the smiling face of that person.

3.2. Supervised experiments

The result for Cifar-10 is reported in Tab. 2. For all settings, we train models using 10 random seeds and take the average test accuracy, except for ImageNet due to limited resources. PARADISE improves test accuracy about 0.3% across architectures. On ImageNet, however, top-1 and top-5 accuracy drops about 0.8% and 0.5%, respectively (Tab. 3). We make a mild conclusion that the rate $\mathcal{R}(\theta)$ term in the objective function of PARADISE acts as a regularizer that can be useful in certain settings but did not help for ImageNet where the base ResNet-34 model shows little sign of overfitting.

3.3. Robust Learning

Gradient obfuscation in Deep Variational Information Bottleneck (DVIB) (Alemi et al., 2016). Alemi et al. (2016) claims that Deep Variational Information Bottleneck can improve adversarial robustness. However, the DVIB was evaluated only on white-box attacks. For a well-behaved model, the accuracy on white-box attacks should be lower than that of black-box attacks because white-box attackers has access to more information. However, Athalye et al. (2018) pointed out that many of previously proposed defense methods relied on gradient obfuscation, which makes white-box attacks weaker but does not really improve adversarial robustness. To test DVIB, we train the DVIB model on Cifar-10 using the wide ResNet WRN-28-10 architecture (Zagoruyko & Komodakis, 2016) and consider two simple black-box attacks, FGSM (Goodfellow et al., 2014) and R+FGSM (Tramèr et al., 2017), crafted using a naturally trained model on the Cifar-10 dataset, and take the min accuracy as an upper-bound of black-box accuracy. We trained DVIB using the the rate parameter $\alpha \in \{1e-2, 1e-3, 1e-4\}$. The model got numerical error at $\alpha = 1e-2$. Tab. 4 shows that DVIB achieves significantly higher accuracy on white-box FGSM attacks than on the black-box attacks. The issue is more serious with the higher value of α. The black-box accuracy is just slightly better than the standard model.

Table 4: Adversarial accuracy (in %) of DVIB with different values of the rate parameter α on Cifar-10. $\alpha = 0$ corresponds to the standard model.

$\boldsymbol{\alpha}$	Natural	FGSM	Black-box
$1e-3$	95.71	55.45	36.64
$1e-4$	95.72	43.50	35.18
0	95.77	32.29	30.07

Angles between gradients of different models. Fig. 3 plots the histogram of angles between the gradient of the standard model and that of PARADISE-PM w.r.t. the same input image. The angles are computed for are computed for 2,000 random images from the ImageNet validation set, consisting 2 images for each class. The visualization demonstrates that the gradient direction of the two models are orthogonal to each other, which is consistent with the observation in Liu et al. (2016).

Loss surface visualization. We visualize of the loss surface and decision boundary of PARADISE-PM and the standard model when moving an input image along the signed gradient (adversarial) direction and another random Rademacher vector orthogonal to the signed gradient in Fig. 4 for some ImageNet validation images and in Fig. 5 for some Cifar-10 test images. Compared to the standard model on ImageNet, PARADISE-PM has smoother loss surface that is virtually constant along random direction, and has a simpler decision boundary. On Cifar-10, PARADISE-PM's loss surface is almost locally constant around the input data, which explains its adversarial robustness.

Targeted attacks. Fig. 6 compares successful targeted attacks with perturbation size of $\epsilon = 16/255$ on PARADISE-PM and those on the standard model. Successful attacks on PARADISE-PM significantly change the original images to look like the target, while the perturbations by successful

Figure 1: CelebA image generation. Images in column 1 are generated from random noise. Images in columns 2 to 10 are successive reconstructions of the previous column. The series of reconstruction help visualize what each neighborhood of $\mathbf{z}$ represents. In each rows, one can observe similar-looking faces with transformations such as smile, face orientation and shape, eyeglasses, gender and hairstyle.

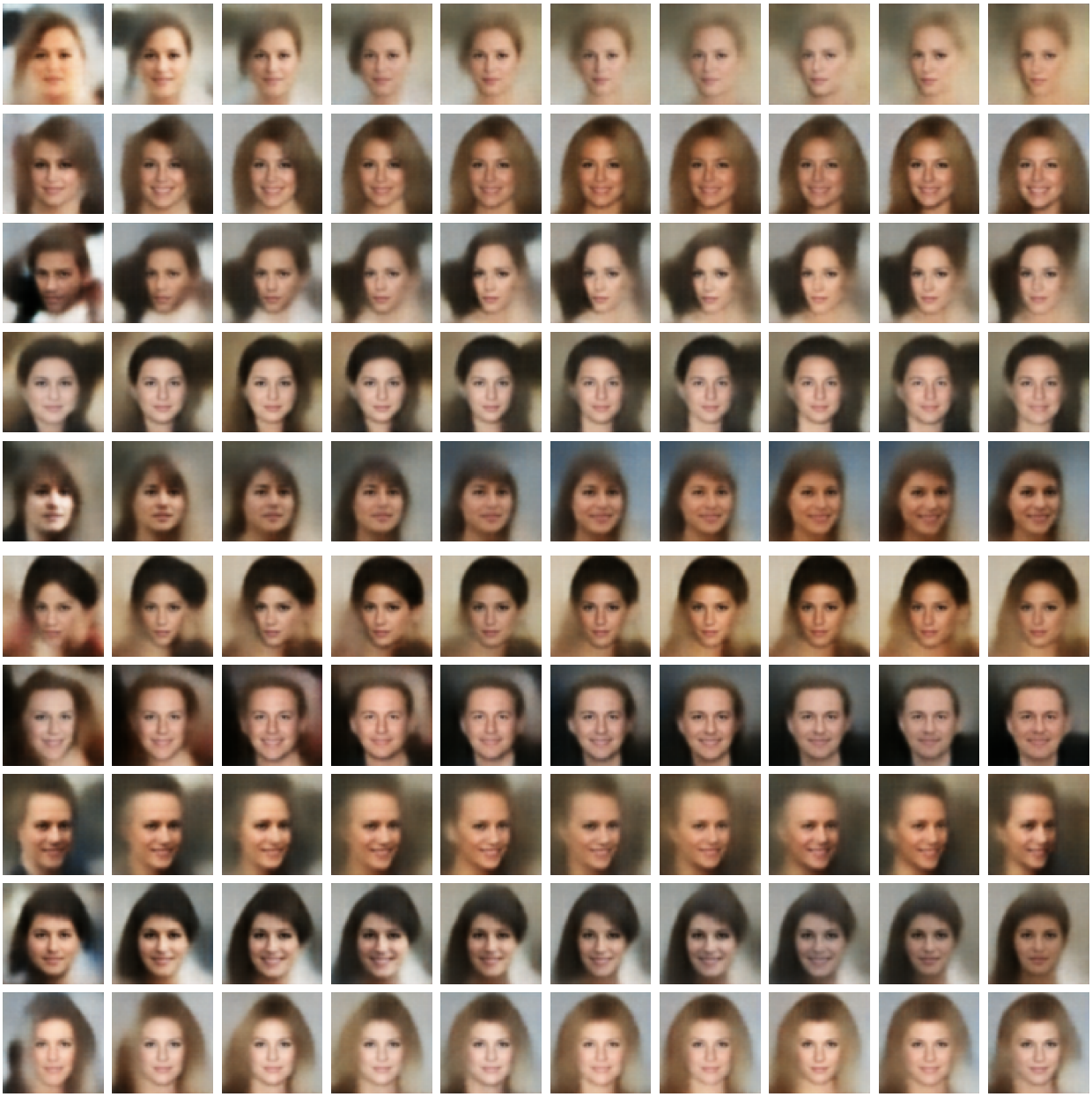


Figure 2: Examples of semantic transformation represented by linear operation in the latent space. In each of the sub-figure, the difference between the $\mathbf{z}$-vector corresponding to the images on the right and left in the first row is added to the left images of the other rows to generate the right images. The captions describe the semantic transformation observed.

(a) Smile. (b) Gender, Beard. (c) Hair, round face. (d) Sunglasses.

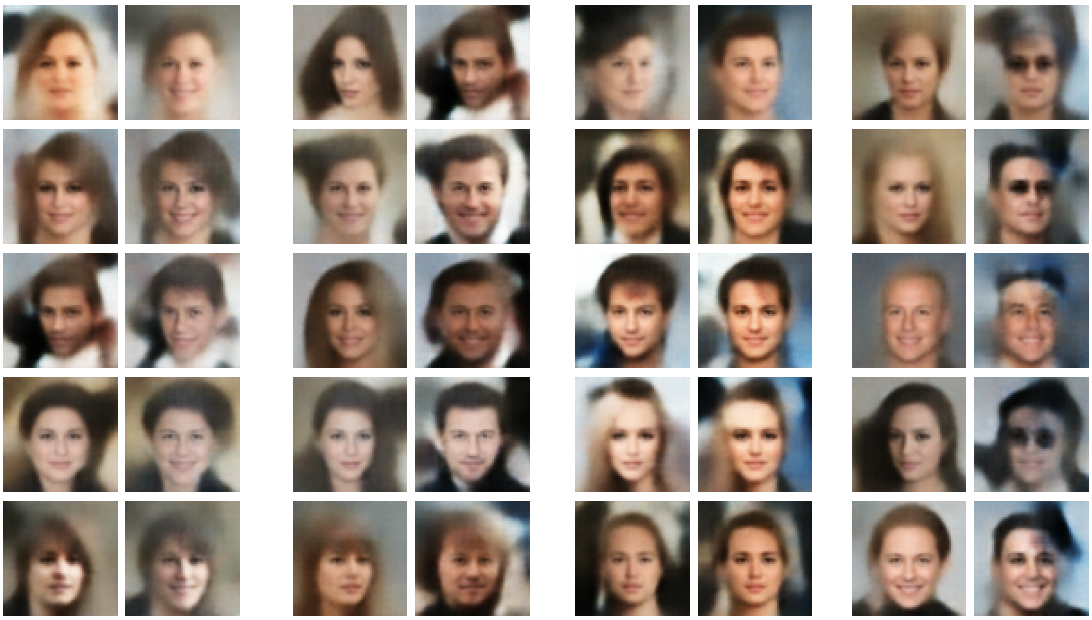
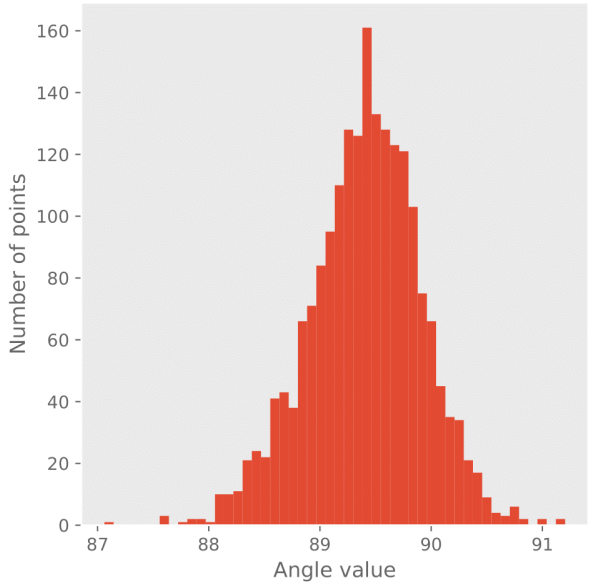


Figure 3: Histogram of angles between the gradient of the standard model and that of PARADISE-PM w.r.t. the same input image. The angles are computed for 2,000 random images from the ImageNet validation set, consisting 2 images for each class. Most of the angles are around 90 degrees.



attacks on the standard model are imperceptible to human.

Figure 4: Comparison of the loss surface and decision boundary of PARADISE-PM and the standard model for ImageNet validation images. In each sub-figure, the top images plots generated by moving the input along the signed gradient (adversarial) direction and another random Rademacher vector orthogonal to the signed gradient; the bottom images plot the corresponding decision boundary; green represents the ground-truth class. PARADISE-PM has smoother loss surface that is virtually constant along random direction, and has a simpler decision boundary.

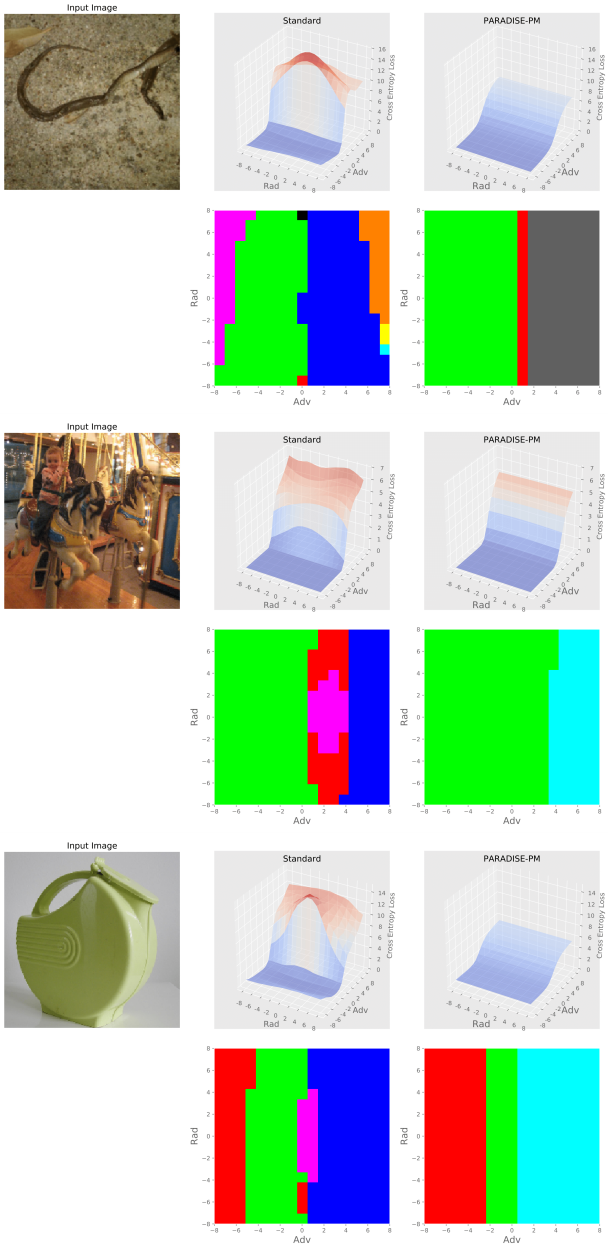


Figure 5: Comparison of the loss surface and decision boundary of PARADISE-PM and the standard model for Cifar-10 test images. In each sub-figure, the top images plots generated by moving the input along the signed gradient (adversarial) direction and another random Rademacher vector orthogonal to the signed gradient; the bottom images plot the corresponding decision boundary; green represents the ground-truth class. PARADISE-PM's loss surface is almost locally constant around the input data.

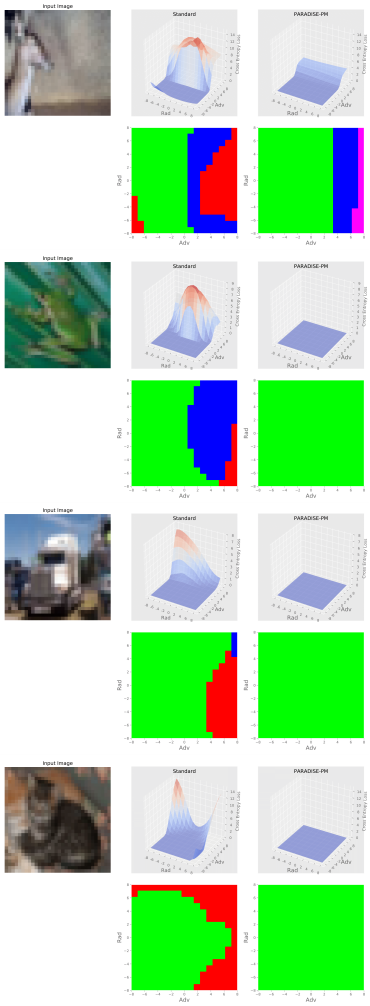


Figure 6: Comparison of successful targeted attacks ($\epsilon = 16/255$). Left: the original input; middle: the attack on the standard model; right: the attack on PARADISE-PM. Successful attacks on PARADISE-PM makes add perturbation that looks like the target class to the input images.

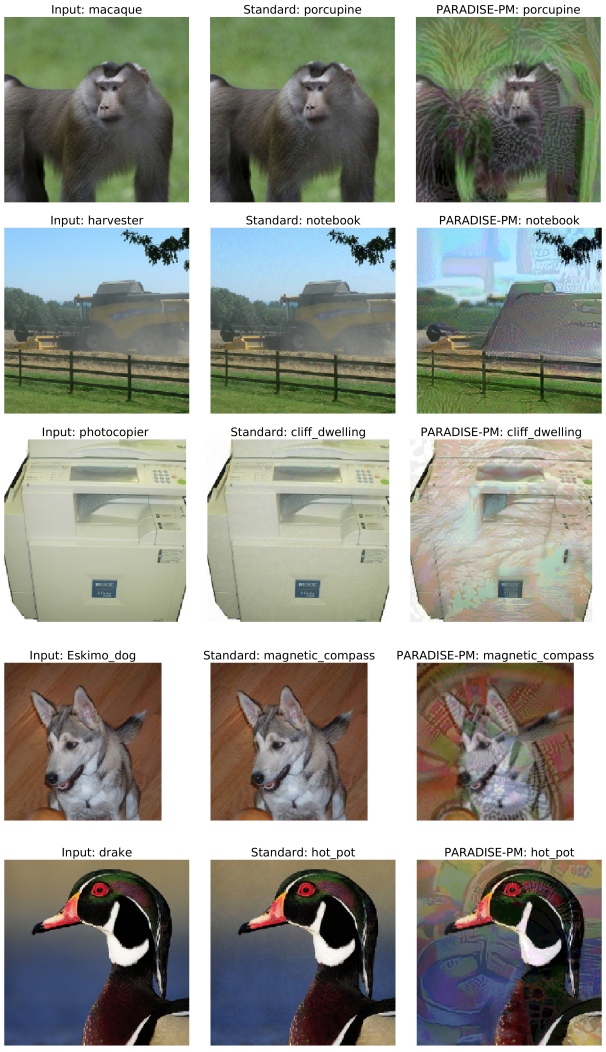
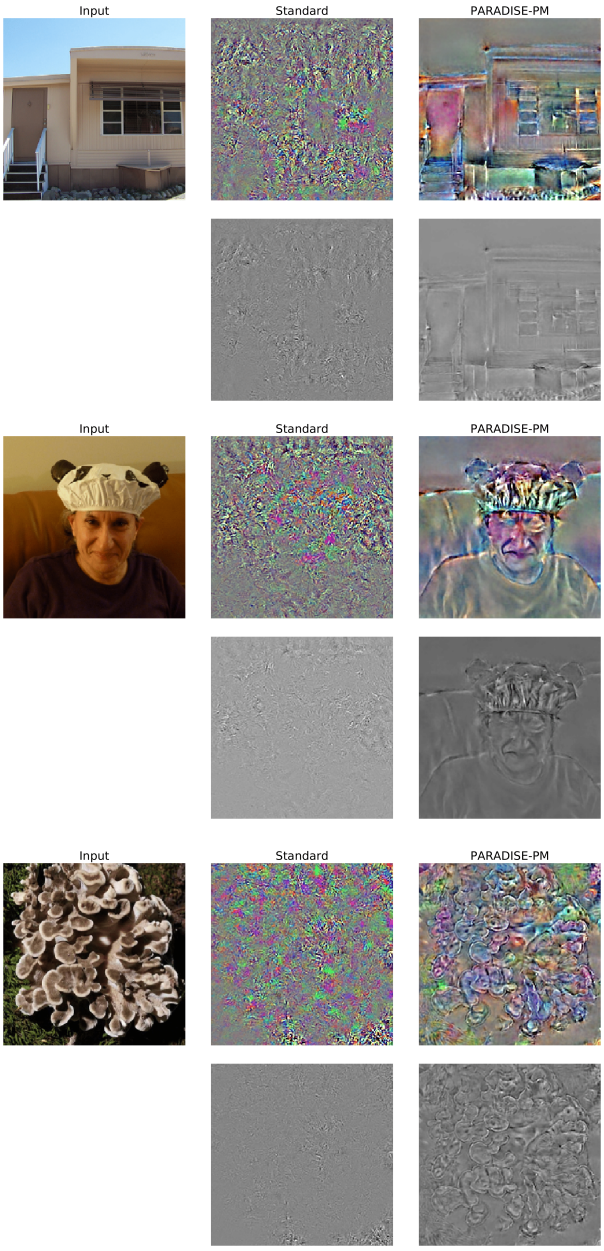


Figure 7: Visualizations of the loss gradient w.r.t. input pixels. The gradients are clipped within ± 3 standard deviations of their mean and rescale to the range $[0, 1]$. In each sub-figure, the top plots visualize gradients in RGB mode; in the bottom plots, gradients are summed across the RGB channels and visualized in gray scale.



References

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *arXiv preprint arXiv:1802.00420*, 2018.

Chen, T. Q., Li, X., Grosse, R. B., and Duvenaud, D. K. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems*, pp. 2610–2620, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Gowal, S., Uesato, J., Qin, C., Huang, P.-S., Mann, T., and Kohli, P. An alternative surrogate loss for pgd-based adversarial testing. *arXiv preprint arXiv:1910.09338*, 2019.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016a.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Liu, Y., Chen, X., Liu, C., and Song, D. Delving into transferable adversarial examples and black-box attacks. *arXiv preprint arXiv:1611.02770*, 2016.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3730–3738, 2015.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Qin, C., Martens, J., Gowal, S., Krishnan, D., Dvijotham, K., Fawzi, A., De, S., Stanforth, R., and Kohli, P. Adversarial robustness through local linearization. In *Advances in Neural Information Processing Systems*, pp. 13824–13833, 2019.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204*, 2017.

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.