On the Benefits of Defining Vicinal Distributions in Latent Space [1]

Puneet Mangla , Vedant Singh, Shreyas Havaldar & Vineeth N Balasubramanian

Department of Computer Science
Indian Institute of Technology, Hyderabad, India

[1]Acknowledging DST, Govt of India (IMPRINT program) for funding; IIT-Hyderabad and JICA for provision of GPU servers .

Table of contents

Empirical Risk Minimization

Empirical Risk Minimization (ERM) : minimize the average error over the training dataset

Empirical Risk Minimization

Empirical Risk Minimization (ERM) : minimize the average error over the training dataset

$$p_{actual}(x, y) \approx p_{\delta}(x, y) = \frac{1}{N} \cdot \sum_{i=1}^{N} \delta(x = x_i, y = y_i)$$

Empirical Risk Minimization

Empirical Risk Minimization (ERM) : minimize the average error over the training dataset

$$p_{actual}(x, y) \approx p_\delta(x, y) = \frac{1}{N} \cdot \sum_{i=1}^{N} \delta(x = x_i, y = y_i)$$

$$w^* = arg \min_w \int \mathcal{L}(F_w(x), y) \cdot dp_\delta(x, y) = arg \min_w \frac{1}{N} \cdot \sum_{i=1}^{N} \mathcal{L}(F_w(x_i), y_i)$$

Empirical Risk Minimization

Empirical Risk Minimization (ERM) : minimize the average error over the training dataset

$$p_{actual}(x, y) \approx p_{\delta}(x, y) = \frac{1}{N} \cdot \sum_{i=1}^{N} \delta(x = x_i, y = y_i)$$

$$w^* = arg \min_{w} \int \mathcal{L}(F_w(x), y) \cdot dp_{\delta}(x, y) = arg \min_{w} \frac{1}{N} \cdot \sum_{i=1}^{N} \mathcal{L}(F_w(x_i), y_i)$$

Drawback: Overparametrized NNs suffer from memorization → leads to undesirable behavior of network outside the training distribution.

Vicinal Risk Minimization

→ Popularly known as data augmentation.

Vicinal Risk Minimization

→ Popularly known as data augmentation.
→ define a vicinity or neighbourhood around each training example (eg. in terms of brightness, contrast, imperceptible noise, etc.)

$$p_v(x, y) = \frac{1}{N} \cdot \sum_{i=1}^{N} v(x, y | x_i, y_i)$$

, where v is the *vicinal distribution* that calculates the probability of a data point (x, y) in the vicinity of other samples (x_i, y_i).

Vicinal Risk Minimization

Expected Vicinal Risk is given by

$$w^* = arg \min_w \int \mathcal{L}(F_w(x), y) \cdot dp_v(x, y) = \frac{1}{N} \cdot \sum_{i=1}^{N} g(F_w, \mathcal{L}, x_i, y_i)$$

where $g(F_w, \mathcal{L}, x_i, y_i) = \int \mathcal{L}(F_w(x), y) \cdot dv(x, y|x_i, y_i)$.

Vicinal Risk Minimization

Expected Vicinal Risk is given by

$$w^* = arg \min_w \int \mathcal{L}(F_w(x), y) \cdot dp_v(x, y) = \frac{1}{N} \cdot \sum_{i=1}^{N} g(F_w, \mathcal{L}, x_i, y_i)$$

where $g(F_w, \mathcal{L}, x_i, y_i) = \int \mathcal{L}(F_w(x), y) \cdot dv(x, y|x_i, y_i)$.
$\rightarrow$ *Gaussian* and *Mixup* are few popular examples of vicinal distributions.

Mixup

Mixup is a popular technique to train models for better generalisation

[2] Pang et al., Mixup inference: Better exploiting mixup to defend adversarial attacks, ICLR 2020

[3] Hendrycks et al., Augmix: A simple method to improve robustness and uncertainty under data shift, ICLR 2020

[4] Lamb et al., Interpolated adversarial training : Achieving robust neural networks without sacrificing too much accuracy. AISec'19

[5] Thulasidasan et al., On mixup training: Improved calibration and predictive uncertainty for deep neural networks. NeurIPS 2019

Mixup

Mixup is a popular technique to train models for better generalisation

- Pang et al., 2020 [2]; Hendrycks et al., 2020 [3]; Lamb et al., 2019 [4] - Mixup to improve the robustness of models.

[2] Pang et al., Mixup inference: Better exploiting mixup to defend adversarial attacks, ICLR 2020

[3] Hendrycks et al., Augmix: A simple method to improve robustness and uncertainty under data shift, ICLR 2020

[4] Lamb et al., Interpolated adversarial training : Achieving robust neural networks without sacrificing too much accuracy. AISec'19

[5] Thulasidasan et al., On mixup training: Improved calibration and predictive uncertainty for deep neural networks. NeurIPS 2019

Mixup

Mixup is a popular technique to train models for better generalisation

- Pang et al., 2020 [2]; Hendrycks et al., 2020 [3]; Lamb et al., 2019 [4] - Mixup to improve the robustness of models.
- Thulasidasan et al., 2019 [5] - Mixup-trained networks are significantly better calibrated.

[2] Pang et al., Mixup inference: Better exploiting mixup to defend adversarial attacks, ICLR 2020

[3] Hendrycks et al., Augmix: A simple method to improve robustness and uncertainty under data shift, ICLR 2020

[4] Lamb et al., Interpolated adversarial training : Achieving robust neural networks without sacrificing too much accuracy. AISec'19

[5] Thulasidasan et al., On mixup training: Improved calibration and predictive uncertainty for deep neural networks. NeurIPS 2019

Contributions

- Propose a new vicinal distribution called *VarMixup (Variational Mixup)* to sample better Mixup images.

Contributions

- Propose a new vicinal distribution called *VarMixup (Variational Mixup)* to sample better Mixup images.
- Experiments shows that VarMixup boosts the robustness to out-of-distribution shifts as well calibration.

Contributions

- Propose a new vicinal distribution called *VarMixup (Variational Mixup)* to sample better Mixup images.
- Experiments shows that VarMixup boosts the robustness to out-of-distribution shifts as well calibration.
- Additional analysis show that VarMixup significantly decreases the local linearity error of the neural network.

Motivation

- Generative models like VAEs - capture the latent space from which a distribution is generated provides us an unfolded manifold

Motivation

- Generative models like VAEs - capture the latent space from which a distribution is generated provides us an unfolded manifold
- linearity in between training examples is more readily observed.

Motivation

- Generative models like VAEs - capture the latent space from which a distribution is generated provides us an unfolded manifold

- linearity in between training examples is more readily observed.

- Leverage such latent to capture the induced global linearity in between examples, and define Mixup.

Details

- We opt for an MMD-VAE [6] because of its advantage over vanilla KL based VAE [7]:

$$\mathcal{L}_{MMD-VAE} = \gamma \cdot MMD(q_\phi(z) \| p(z)) + \mathbb{E}_{x \sim p_{actual}} \mathbb{E}_{z \sim q_\phi(z|x)} [\log(p_\theta(x|z))]$$

[6] Zhao et al., Infovae: Information maximizing variational autoencoder, 2017
[7] Kingma et al., Auto-encoding variational bayes, 2013

Details

- We opt for an MMD-VAE [6] because of its advantage over vanilla KL based VAE [7]:

$$\mathcal{L}_{MMD-VAE} = \gamma \cdot MMD(q_\phi(z) \| p(z)) + \mathbb{E}_{x \sim p_{actual}} \mathbb{E}_{z \sim q_\phi(z|x)} [\log(p_\theta(x|z))]$$

- Mixup vicinal distribution in the latent space of the trained VAE as $v_{VarMixup}(z, y | x_i, y_i) =$

$$\frac{1}{n} \cdot \sum_{j=1}^{N} \mathbb{E}_\lambda [\delta(z = \lambda \cdot \mathbb{E}_z[q_\phi(z|x_i)] + (1-\lambda) \cdot \mathbb{E}_z[q_\phi(z|x_j)], y = \lambda \cdot y_i + (1-\lambda) \cdot y_j)]$$

[6] Zhao et al., Infovae: Information maximizing variational autoencoder, 2017

[7] Kingma et al., Auto-encoding variational bayes, 2013

Details

- We opt for an MMD-VAE [6] because of its advantage over vanilla KL based VAE [7]:

$$\mathcal{L}_{MMD-VAE} = \gamma \cdot MMD(q_\phi(z) \| p(z)) + \mathbb{E}_{x \sim p_{actual}} \mathbb{E}_{z \sim q_\phi(z|x)}[\log(p_\theta(x|z))]$$

- Mixup vicinal distribution in the latent space of the trained VAE as $v_{VarMixup}(z, y|x_i, y_i) =$

$$\frac{1}{n} \cdot \sum_{j=1}^{N} \mathbb{E}_\lambda[\delta(z = \lambda \cdot \mathbb{E}_z[q_\phi(z|x_i)] + (1-\lambda) \cdot \mathbb{E}_z[q_\phi(z|x_j)], y = \lambda \cdot y_i + (1-\lambda) \cdot y_j)]$$

- Equivalent to constructing VarMixup samples as:

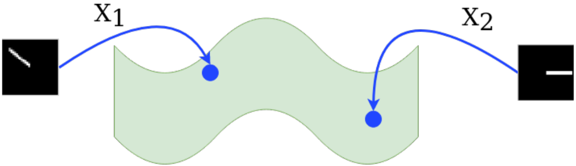
$$x' = \mathbb{E}_x \left[p_\theta \left(x | \lambda \cdot \mathbb{E}_z[q_\phi(z|x_i)] + (1 - \lambda) \cdot \mathbb{E}_z[q_\phi(z|x_j)]\right)\right]$$

$$y' = \lambda \cdot y_i + (1 - \lambda) \cdot y_j$$

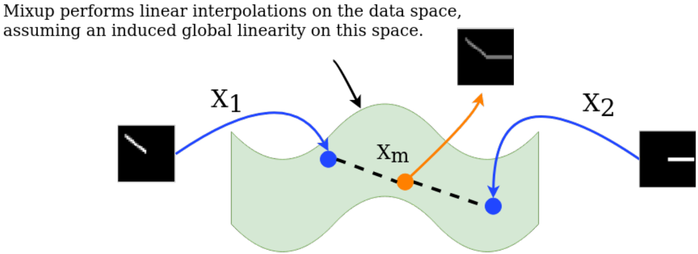
[6] Zhao et al., Infovae: Information maximizing variational autoencoder, 2017

[7] Kingma et al., Auto-encoding variational bayes, 2013

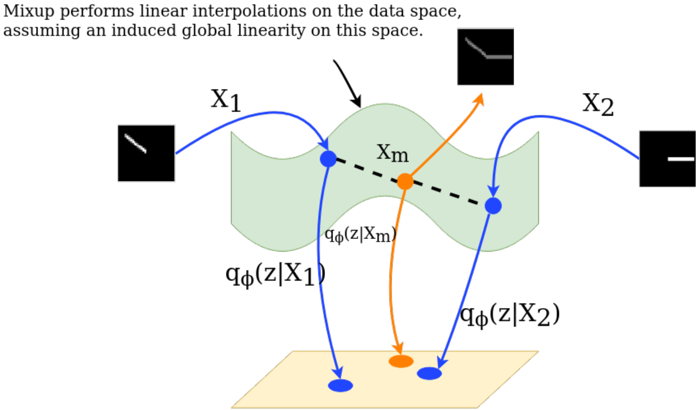
Illustration



Illustration

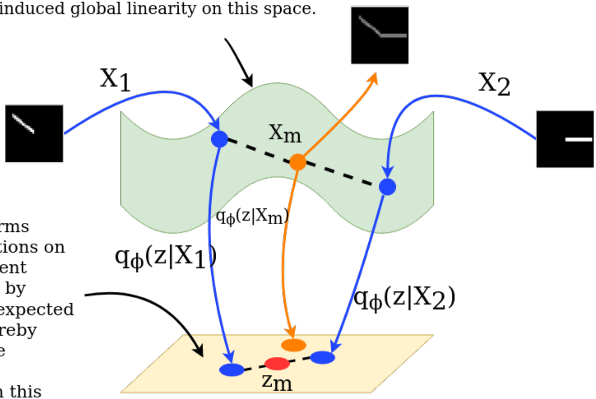


Illustration



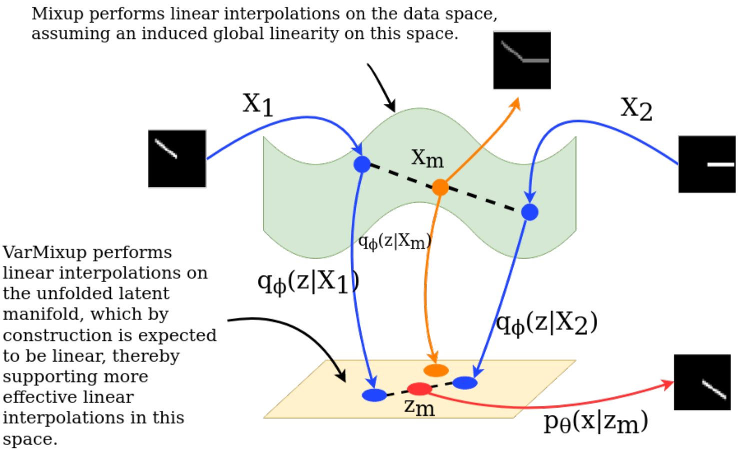
Illustration

Mixup performs linear interpolations on the data space, assuming an induced global linearity on this space.



VarMixup performs linear interpolations on the unfolded latent manifold, which by construction is expected to be linear, thereby supporting more effective linear interpolations in this space.

Illustration



Out-of-Distribution Generalization

- Geirhos et al. (2018) [8]:
  training against distortions can often fail to generalize to unseen distortions

Out-of-Distribution Generalization

- Geirhos et al. (2018) [8]:
  training against distortions can often fail to generalize to unseen distortions
- Hendrycks et al. (2019) propose benchmarks for measuring generalization to unseen corruptions

Out-of-Distribution Generalization

- Geirhos et al. (2018) [8]: training against distortions can often fail to generalize to unseen distortions
- Hendrycks et al. (2019) propose benchmarks for measuring generalization to unseen corruptions

Figure 2: Example ImageNet-C corruptions. These corruptions are encountered only at test time and not

Out-of-Distribution Generalization

- Robustness to common input corruptions on CIFAR-10-C, CIFAR-100-C and Tiny-Imagenet-C
- *adv*-VarMixup : Variant of VarMixup where we use adversarial robust VAE.

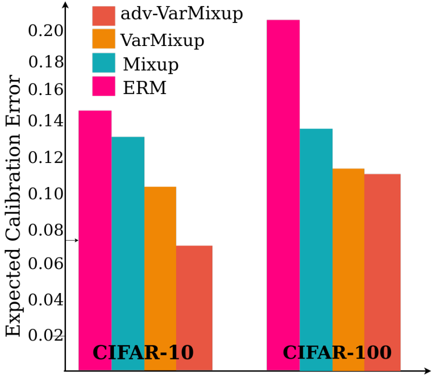
Out-of-Distribution Generalization

- Robustness to common input corruptions on CIFAR-10-C, CIFAR-100-C and Tiny-Imagenet-C
- *adv*-VarMixup : Variant of VarMixup where we use adversarial robust VAE.

Method	CIFAR-10-C	CIFAR-100-C	Tiny-Imagenet-C
AT (Madry et al., 2018)	73.12 ± 0.31 (85.58 ± 0.14)	45.09 ± 0.31 (60.28 ± 0.13)	15.74 ± 0.36 (22.33 ± 0.16)
TRADES (Zhang et al., 2019)	75.46 ± 0.21 (88.11 ± 0.43)	45.98 ± 0.41 (63.3 ± 0.32)	16.20 ± 0.23 (26.12 ± 0.38)
IAT (Lamb et al., 2019)	81.05 ± 0.42 (89.7 ± 0.33)	50.71 ± 0.25 (62.7 ± 0.21)	18.69 ± 0.45 (18.08 ± 0.34)
ERM	69.29 ± 0.21 (94.5 ± 0.14)	47.3 ± 0.32 (64.5 ± 0.10)	17.34 ± 0.27 (49.96 ± 0.12)
Mixup	74.74 ± 0.34 (95.5 ± 0.35)	52.13 ± 0.43 (76.8 ± 0.41)	21.55 ± 0.37 (53.83 ± 0.17)
Mixup-R	74.27 ± 0.22 (89.88 ± 0.11)	43.54 ± 0.15 (62.24 ± 0.21)	21.34 ± 0.32 (53.5 ± 0.28)
Manifold-Mixup	72.54 ± 0.14 (95.2 ± 0.18)	41.42 ± 0.23 (75.3 ± 0.48)	-
VarMixup	**82.57 ± 0.42** (93.91 ± 0.45)	<u>52.57 ± 0.39</u> (73.2 ± 0.44)	<u>24.87 ± 0.32</u> (50.98 ± 0.11)
adv-VarMixup	<u>82.12 ± 0.46</u> (92.19 ± 0.32)	**54.0 ± 0.41** (72.13 ± 0.34)	**25.36 ± 0.21** (50.58 ± 0.23)

Calibration

- measures how good softmax scores are as indicators of the actual likelihood of a correct prediction.
- We measure the *Expected Calibration Error (ECE, lower the better)* [9]



[9] Guo et al., On calibration of modern neural networks , 2017

Local Linearity of Loss Surfaces

- Qin et al., 2020 [10] showed that the local linearity of loss landscapes of NNs correlates robustness.

[10] Qin et al., Adversarial robustness through local linearization, NeurIPS 2019

Local Linearity of Loss Surfaces

- Qin et al., 2020 [10] showed that the local linearity of loss landscapes of NNs correlates robustness.
- The more the loss landscapes are linear, the more the robustness.

[10] Qin et al., Adversarial robustness through local linearization, NeurIPS 2019

Local Linearity of Loss Surfaces

- Qin et al., 2020 [10] showed that the local linearity of loss landscapes of NNs correlates robustness.
- The more the loss landscapes are linear, the more the robustness.
- Local linearity at a data-point x within a neighbourhood $B(\epsilon)$ as $\gamma(\epsilon, x, y) =$

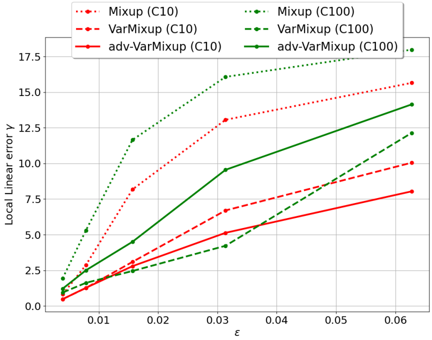
$$\max_{\delta \in B(\epsilon)} |\mathcal{L}(F_w(x+\delta), y) - \mathcal{L}(F_w(x), y) - \delta^T \nabla_x \mathcal{L}(F_w(x), y)|$$

[10] Qin et al., Adversarial robustness through local linearization, NeurIPS 2019

Local Linearity of Loss Surfaces

- Qin et al., 2020 [10] showed that the local linearity of loss landscapes of NNs correlates robustness.
- The more the loss landscapes are linear, the more the robustness.
- Local linearity at a data-point x within a neighbourhood $B(\epsilon)$ as $\gamma(\epsilon, x, y) =$

$$\max_{\delta \in B(\epsilon)} |\mathcal{L}(F_w(x+\delta), y) - \mathcal{L}(F_w(x), y) - \delta^T \nabla_x \mathcal{L}(F_w(x), y)|$$



[10] Qin et al., Adversarial robustness through local linearization, NeurIPS 2019

Conclusion

- Proposed VarMixup, which performs linear interpolation on an unfolded latent manifold where linearity in between training examples is likely to be preserved.

Conclusion

- Proposed VarMixup, which performs linear interpolation on an unfolded latent manifold where linearity in between training examples is likely to be preserved.
- VarMixup trained models are more robust to corruptions and are better calibrated.

Conclusion

- Proposed VarMixup, which performs linear interpolation on an unfolded latent manifold where linearity in between training examples is likely to be preserved.

- VarMixup trained models are more robust to corruptions and are better calibrated.

- Highlights the efficacy of defining vicinal distributions by using neighbors on unfolded latent manifold rather than data manifold.