Understanding KL-PGD: A Hybrid Approach to Adversarial Training

Kenneth*

August 8, 2025

1 Introduction

Adversarial robustness research focuses on training models that can resist small but malicious perturbations to input data. Among the most widely used defenses is **adversarial training**, where the model is trained on adversarially perturbed inputs.

A classic algorithm for generating these perturbations is **Projected Gradient Descent** (PGD) [1], which iteratively updates the input in the direction of the gradient to maximize the loss, while keeping the perturbation within a specified norm bound.

KL-PGD is a variant of PGD-based adversarial training that incorporates the **Kullback–Leibler (KL) divergence** into the loss function. The main idea is to not only push the model to classify perturbed inputs correctly, but also to align the probability distribution over classes between clean and perturbed inputs.

2 Key Components

2.1 Projected Gradient Descent (PGD)

PGD generates adversarial examples by:

1. Starting from a clean input x.
2. Iteratively applying gradient ascent on the loss $L(\theta, x, y)$ with respect to the input x.
3. Projecting the perturbed input back into an ℓ_p-ball of radius ϵ around x to enforce the perturbation bound.

Mathematically:

$$x^{t+1} = \Pi_{\mathcal{B}(x,\epsilon)} \left(x^t + \alpha \cdot \text{sign} \left(\nabla_x L(\theta, x^t, y) \right) \right)$$

where $\Pi_{\mathcal{B}(x,\epsilon)}$ is the projection operator, α is the step size, and ϵ is the perturbation limit.

*This article is for educational purposes. If KL-PGD refers to a specific internal or unpublished method, the description here is illustrative and based on general principles of KL divergence and Projected Gradient Descent (PGD).

2.2 Kullback–Leibler Divergence

KL divergence measures how one probability distribution P diverges from another Q:

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

In classification, P and Q can represent the model's predicted probability distributions for two different inputs.

2.3 Combining KL and PGD

In KL-PGD adversarial training, the loss function often includes:

- A **classification loss** (e.g., cross-entropy) on clean inputs.
- A **KL divergence term** between the model's predictions on clean inputs and its predictions on adversarial inputs.

A typical combined objective might be:

$$\mathcal{L}(\theta) = \mathbb{E}_{(x,y)} \left[L_{\mathrm{CE}}(f_\theta(x), y) + \lambda \cdot D_{\mathrm{KL}}\big(f_\theta(x) \parallel f_\theta(x_{\mathrm{adv}})\big) \right]$$

where x_{adv} is generated via PGD, and λ controls the weight of the KL regularization.

3 Intuition

The KL term encourages the model to produce similar output distributions for clean and adversarial versions of the same input. This has two benefits:

1. **Smoothness:** The model's predictions change more gradually around data points.
2. **Stability:** Reduces sensitivity to small perturbations, improving robustness.

4 Practical Notes

- The choice of λ is critical: too small, and the KL term has little effect; too large, and it may hurt clean accuracy.
- KL-PGD can be more computationally expensive than standard PGD training because it requires computing predictions for both clean and adversarial inputs.
- Variants may use symmetric KL divergence or Jensen–Shannon divergence instead.

5 Conclusion

KL-PGD represents a hybrid strategy that combines the strong perturbation generation of PGD with the distribution-matching power of KL divergence. While it may not be an official standardized method in the literature, the principles behind it are well-grounded in adversarial training research.

Disclaimer

If KL-PGD refers to a specific benchmarked method (e.g., in PRBench), please refer to its official documentation for exact definitions and hyperparameters.

References

[1] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.