Automatic Data Augmentation via Invariance Constrained Learning

DISCLAIMER: Summarized by AI

Problem they are trying to solve / Purpose of method

Goal:
Automatically adapt data augmentation strategies during training by leveraging invariance properties of data, without relying on fixed, handcrafted augmentation policies or expensive search-based methods.

Challenges with previous methods:

- Fixed augmentation policies can introduce bias if transformations are misaligned with data distribution.
- Learning good augmentation policies typically requires:
  - Large search spaces
  - Computationally intensive techniques (e.g., reinforcement learning, gradient-based optimization)
  - Assumptions about differentiability of transformations
- Embedding invariance in model architectures is complex and computationally costly.

Motivation:

Rather than statically defining how and when to augment, the authors propose to **treat augmentation as an invariance constraint in learning**, and use optimization to dynamically adjust the augmentation process.

How does it differ from other methods?

Key Differences:

- **Formulation**: Casts data augmentation as a **constrained optimization problem**, where the constraint enforces invariance (i.e., stability of model predictions under transformations).
- **No predefined augmentation policy**: The transformation distribution is discovered during training by solving a dual optimization problem.
- **Handles non-differentiable transformations**: Uses **Monte Carlo Markov Chain (MCMC) sampling**, not gradients, to find augmentations — avoids a major limitation of previous gradient-based methods.
- **Dynamic control**: The method automatically adjusts:
  - Whether to augment
  - How much to augment
  - Which transformations to apply — **adapts based on data and model behavior**

Advantages over prior work:

- Does not require expensive policy search (e.g., AutoAugment).
- No need for gradient computation with respect to transformations.
- Learns sample-specific, task-aligned augmentation distributions.

How the method works

Simple Overview:

1. Define a set of transformations (e.g., rotations, translations).
2. Require the model to be **invariant** to these transformations.
3. Formulate this as a **constraint** on the learning problem — the model's loss should remain stable under transformations.
4. Solve the learning problem using a **primal-dual algorithm**, adapting both model parameters and augmentation distribution.
5. Use **MCMC sampling** to sample transformations in a differentiability-agnostic manner.

Detailed Steps:

- **Invariance loss**: Define a loss that captures how much the model's prediction changes under a transformation.
- **Constraint**: Require the average transformed loss to stay within a certain threshold (ϵ).
- **Optimization**:
  - Dual formulation introduces a Lagrange multiplier (γ) to weight the invariance constraint.
  - Primal-dual updates alternate between optimizing model weights and adjusting γ.
- **Sampling**:
  - Instead of computing gradients w.r.t. transformations, sample them using **Metropolis-Hastings MCMC**, based on how much loss each transformation induces.
  - Allows use of non-differentiable transforms.
- **Adaptivity**:
  - The augmentation distribution evolves throughout training.
  - γ controls how much augmentation is applied — it shrinks to zero if invariance is satisfied naturally.