Improving Domain Generalization with Interpolation Robustness

DISCLAIMER: Summarized by AI

Problem they are trying to solve / Purpose of method

The goal is to improve **domain generalization (DG)**—the ability of models to generalize to unseen domains during training. Most existing DG methods rely heavily on **domain labels** and focus on aligning distributions across domains, which can be unreliable or unavailable.

Problems:

- Models often overfit to the training domains and fail on unseen domains.
- Domain alignment methods may collapse representations and remove semantic information.
- Most DG methods need **domain labels**, which may not always be provided.

Purpose of the method:

- Introduce a method that improves generalization without needing domain labels.
- Focus on **robustness to feature interpolation**, encouraging models to generalize by learning from interpolated examples between training samples.

How does it differ from other methods?

Unique aspects:

- Does **not require domain labels**, unlike many prior DG approaches.
- Introduces **interpolation robustness training (IRT)**—penalizing sensitivity to interpolated features from different samples.
- Works by constructing interpolations in **feature space** rather than input space, which avoids unnatural interpolations and better preserves semantic structure.

Compared to others:

- Domain alignment methods (e.g., CORAL, MMD) try to minimize distribution shifts but can degrade performance when alignment harms semantics.
- Data augmentation methods like Mixup operate in input space, which might introduce artifacts or semantically implausible examples.
- This method uses interpolated features and directly regularizes model behavior on them.

How the method works

Overview:

- The core idea is to **train the model to be robust to linear interpolations in feature space**.
- They create feature pairs from different samples, interpolate them, and enforce that model predictions remain smooth over these interpolations.

Detailed steps:

1. Pass two inputs through the encoder to get features: $f(x_i), f(x_j)$.
2. Create an interpolated feature:

$$f_{\text{interp}} = \lambda f(x_i) + (1 - \lambda) f(x_j)), where(\lambda \sim \text{Beta}(\alpha, \alpha)$$

   .
3. Predict on the interpolated feature.
4. Use the original labels y_i, y_j to construct a mixed label: $y_{\text{interp}} = \lambda y_i + (1 - \lambda) y_j$.
5. Compute a loss (e.g., KL divergence) between the predicted output and y_{interp}.
6. Add this interpolation robustness loss to the normal classification loss.

This encourages the model to behave smoothly between known data points, thereby increasing its generalization ability to unseen domains.