Angry Birds

Romanian National AI Olympiad 2025 - Problem Editorial

AI Olympiad Committee

June 23, 2025

Abstract

This editorial presents a comprehensive analysis of the "Angry Birds" problem, a challenging bias mitigation task in image classification. The problem requires participants to build an unbiased classifier that can distinguish between Earth and Water clan birds despite spurious correlations in the training data. We explore the key challenges of distribution shift, shortcut learning, and worst-case evaluation, presenting a systematic approach to overcome these obstacles through bias identification and balanced training.

Contents

1 Problem Overview

1.1 Context and Motivation

The "Angry Birds: Peace Treaty" problem presents a unique narrative-driven approach to one of the most pressing challenges in modern machine learning: building robust classifiers that generalize well despite biased training data. Set in a world where Earth and Water clan birds have been at war for centuries, the problem casts the participant as Mahatma GAIdhi, an AI peacemaker who must correctly identify birds from both clans to facilitate peace negotiations.

This problem is fundamentally about **bias mitigation** and **robust classification** under distribution shift. The challenge simulates real-world scenarios where training data contains spurious correlations that can lead to catastrophic failures when the model encounters different distributions at test time.

1.2 Technical Specification

The problem involves binary classification with the following constraints:

- **Classes**: Earth birds (class 0) vs Water birds (class 1)
- **Architecture constraint**: Must use ResNet50 pretrained on ImageNet
- **Training bias**: Earth bird images contain artificial red squares as propaganda
- **Background bias**: Earth birds typically appear with land backgrounds, Water birds with water backgrounds
- **Test distribution**: Balanced across all four combinations (Earth/Water $\times$ Land/Water background)
- **Evaluation**: Worst-case accuracy across the four scenarios

1.3 Key Challenges

1.3.1 Spurious Correlations

The training data contains two main sources of bias:

1. **Artificial markers**: Red squares added to Earth bird images (absent at test time)
2. **Background correlation**: Strong correlation between bird type and background

1.3.2 Distribution Shift

The test distribution is deliberately balanced to expose models that rely on spurious features:

- **LL**: Earth birds with Land background (easy, abundant in training)
- **LW**: Earth birds with Water background (hard, rare in training)
- **WL**: Water birds with Land background (hard, rare in training)
- **WW**: Water birds with Water background (easy, abundant in training)

1.3.3 Worst-Case Evaluation

The final score is determined by the minimum accuracy across all four scenarios, making the problem particularly challenging as the model must perform well on the hardest cases.

2 Solution Approach

2.1 High-Level Strategy

The solution follows a three-stage approach:

1. **Bias Identification**: Systematically identify which training samples belong to the "easy" vs "hard" groups
2. **Data Balancing**: Create a balanced training procedure that gives equal weight to all four combinations
3. **Robust Training**: Train a classifier that performs well across all scenarios

2.2 Stage 1: Preprocessing and Feature Extraction

2.2.1 Bias Equalization Through Preprocessing

To handle the red square bias, the solution employs a clever preprocessing strategy:

Algorithm 1 Red Square Bias Equalization

```
Require: Training images, Test images
Ensure: Preprocessed images with equalized red square bias
1: for each image in Test set OR Water class do
2:     Generate random square size $s \in [10, 20]$
3:     Choose random edge $e \in \{top, left, right, bottom\}$
4:     Generate random position $p$ along edge $e$
5:     Set RED channel $= 255$ in square region
6: end for
```

Listing 1: Preprocessing Implementation

```python
if split == 'test' or y[i] == 1:  # Test images or Water birds
    square_size = np.random.randint(10, 21)
    position = np.random.randint(0, 224 - square_size)
    placement = np.random.randint(0, 4)

    if placement == 0:  # Top edge
        im[:square_size, position:position + square_size, 0] = 255
    elif placement == 1:  # Left edge
        im[position:position + square_size, :square_size, 0] = 255
    elif placement == 2:  # Right edge
        im[position:position + square_size, -square_size:, 0] = 255
    else:  # Bottom edge
        im[-square_size:, position:position + square_size, 0] = 255
```

This preprocessing ensures that the red square feature is no longer discriminative between classes.

2.2.2 Feature Extraction

The solution uses ResNet50 pretrained on ImageNet as a fixed feature extractor:

Listing 2: Feature Extraction Setup

```python
weights = models.ResNet50_Weights.IMAGENET1K_V1
resnet = models.resnet50(weights=weights).eval()
```

```python
resnet.fc = torch.nn.Identity()  # Remove final classification layer
preprocess = weights.transforms()

# Extract features for all images
with torch.no_grad():
    features = resnet(images)  # Shape: [batch_size, 2048]
```

2.3 Stage 2: Bias Identification

The most critical part of the solution is identifying which training samples belong to the "biased" groups (LL and WW) versus the "unbiased" groups (LW and WL).

2.3.1 Iterative Bias Detection

The solution uses an iterative approach to identify biased samples:

Algorithm 2 Iterative Bias Detection

```
Require: Training features, Training labels
Ensure: Binary array indicating biased samples
1: Initialize $selection \leftarrow$ all samples
2: for $iteration = 1$ to 7 do
3:     Train linear classifier on $selection$ for 1 epoch
4:     Evaluate classifier on all training samples
5:     $selection \leftarrow$ correctly classified samples
6:     Print size of $selection$
7: end for
8: $is\_biased \leftarrow$ final $selection$
9: return $is\_biased$
```

The intuition behind this approach is:

- Initially, the model learns to classify using the easiest features (background correlation)
- Correctly classified samples after minimal training are likely from the "easy" groups (LL, WW)
- By iteratively training only on these samples, we amplify the bias
- The final selection represents samples that can be classified purely based on spurious correlations

Listing 3: Iterative Bias Detection Implementation

```python
selection = None
for iteration in range(7):
    train_ds = Dataset('train', selection=selection)
    print(f'Training on: {len(train_ds)} samples')

    # Train for one epoch
    model.train()
    for images, labels in train_dl:
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

    # Find correctly classified samples
    model.eval()
    is_biased = []
    for images, labels in train_dl:
        outputs = model(images)
        is_biased.append(
            (torch.argmax(outputs, axis=-1) == labels).int().detach().
                cpu()
        )
    is_biased = torch.cat(is_biased)
    selection = is_biased.bool().numpy()
```

2.4 Stage 3: Balanced Training

2.4.1 Four-Group Classification

Once biased samples are identified, we can categorize the training data into four groups:

$$\text{Group 0 (LW)} : \text{Earth birds} \cap \text{Unbiased samples} \tag{1}$$
$$\text{Group 1 (LL)} : \text{Earth birds} \cap \text{Biased samples} \tag{2}$$
$$\text{Group 2 (WL)} : \text{Water birds} \cap \text{Unbiased samples} \tag{3}$$
$$\text{Group 3 (WW)} : \text{Water birds} \cap \text{Biased samples} \tag{4}$$

2.4.2 Balanced Sampling Strategy

The solution implements a custom dataset that randomly samples from each group with equal probability:

Algorithm 3 Balanced Group Sampling

```
Require: Four groups of samples: $G_0, G_1, G_2, G_3$
Ensure: Balanced training batch
1: for each training step do
2:     Randomly select group $g \in \{0, 1, 2, 3\}$ with equal probability
3:     Randomly sample instance from $G_g$
4:     Return sampled instance with its true label
5: end for
```

Listing 4: Balanced Dataset Implementation

```python
class BalancedDataset(torch.utils.data.Dataset):
    def __init__(self, split='train', a=None):
        # Load data and create group divisions
        self.groups = dict()
        self.groups[0] = ((self.y == 0).int() * (self.a == 0).int()).
            bool()  # LW
        self.groups[1] = ((self.y == 0).int() * (self.a == 1).int()).
            bool()  # LL
        self.groups[2] = ((self.y == 1).int() * (self.a == 0).int()).
            bool()  # WL
        self.groups[3] = ((self.y == 1).int() * (self.a == 1).int()).
            bool()  # WW

        # Separate embeddings and labels by group
        for i in range(4):
            self.y_group[i] = self.y[self.groups[i]]
            self.embeddings_group[i] = self.embeddings[self.groups[i]]

    def __getitem__(self, sample_n):
        # Randomly select group
        group_n = torch.randint(0, 4, (1,))[0].item()
        # Randomly select sample from group
        sample_n = torch.randint(0, len(self.y_group[group_n]), (1,))
            [0].item()
        return self.embeddings_group[group_n][sample_n], self.y_group[
            group_n][sample_n]
```

2.5 Final Training

The final classifier is trained using standard techniques but with the balanced sampling strategy:

Listing 5: Final Training Loop

```python
model = torch.nn.Linear(2048, 2)
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)
criterion = torch.nn.CrossEntropyLoss()

train_dl = torch.utils.data.DataLoader(
    BalancedDataset(a=is_biased),
    batch_size=256,
    shuffle=True,
    num_workers=10
)

for epoch_n in range(3):
    model.train()
    for images, labels in train_dl:
        model.zero_grad()
        outputs = model(images)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
```

3 Alternative Approaches

3.1 Domain Adaptation Methods

Alternative approaches could include:

3.1.1 Adversarial Training

Train a classifier while simultaneously training an adversarial network to predict the bias attribute:

$$\mathcal{L}_{total} = \mathcal{L}_{task} - \lambda \mathcal{L}_{adversarial} \tag{5}$$

where $\mathcal{L}_{adversarial}$ tries to predict whether a sample comes from a biased group.

3.1.2 Environment-Invariant Learning

Methods like IRM (Invariant Risk Minimization) could be applied:

$$\min_{\Phi,w} \sum_{e\in\mathcal{E}} R^e(\Phi, w) + \lambda \sum_{e\in\mathcal{E}} \|w \odot \nabla_{w|w=1.0} R^e(\Phi, w)\|^2 \tag{6}$$

3.2 Data Augmentation Strategies

3.2.1 Background Mixing

Artificially create samples with mixed backgrounds to reduce correlation:

Listing 6: Background Mixing

```python
def mix_backgrounds(earth_img, water_img, alpha=0.3):
    # Extract foreground objects using segmentation
    earth_fg = extract_foreground(earth_img)
    water_bg = extract_background(water_img)

    # Composite foreground on different background
    mixed = alpha * earth_fg + (1 - alpha) * water_bg
    return mixed
```

3.2.2 Counterfactual Data Generation

Generate synthetic samples for underrepresented groups using generative models.

4 Theoretical Analysis

4.1 Why the Approach Works

The success of this approach can be understood through several theoretical lenses:

4.1.1 Empirical Risk Minimization vs. Worst-Case Risk

Standard ERM minimizes:

$$\hat{R}_{ERM}(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i), y_i) \tag{7}$$

But our evaluation focuses on:

$$R_{worst}(f) = \max_{g\in\{LL,LW,WL,WW\}} R_g(f) \tag{8}$$

The balanced training approximates:

$$\hat{R}_{balanced}(f) = \frac{1}{4} \sum_{g\in\{LL,LW,WL,WW\}} R_g(f) \tag{9}$$

4.1.2 Bias-Variance Trade-off in Group Performance

By identifying and balancing groups, we trade off some performance on easy groups (LL, WW) for better performance on hard groups (LW, WL), ultimately improving worst-case performance.

4.2 Limitations and Failure Cases

4.2.1 Bias Detection Accuracy

The iterative approach assumes that:

1. Models initially learn spurious correlations
2. Correctly classified samples after minimal training are predominantly from biased groups

This may fail if:

- The spurious correlation is weak
- The model learns robust features quickly
- There's significant label noise

4.2.2 Group Identification Errors

Misclassifying samples into wrong groups can hurt performance, especially if:

- Hard samples are misidentified as easy (reduces training on hard cases)
- Easy samples are misidentified as hard (dilutes hard case training)

5 Implementation Details

5.1 Hyperparameter Choices

Key hyperparameters and their justification:

Parameter	**Value**	**Justification**
Bias detection iterations	7	Sufficient for convergence
Learning rate (bias detection)	0.01	Fast learning of spurious features
Learning rate (final training)	0.001	Stable training with Adam
Final training epochs	3	Prevent overfitting
Batch size	256	Balance efficiency and stability

Table 1: Key hyperparameters used in the solution

5.2 Computational Complexity

- **Preprocessing**: $O(N)$ where N is number of images
- **Feature extraction**: $O(N)$ single forward pass through ResNet50
- **Bias detection**: $O(7 \times N)$ for iterative training
- **Final training**: $O(E \times B)$ where E is epochs, B is batches

Total complexity is dominated by the ResNet50 forward passes, making the approach computationally efficient.

6 Experimental Insights

6.1 Ablation Studies

Key components and their impact:

6.1.1 Preprocessing Impact

Removing the red square equalization preprocessing would likely result in:

- Models relying heavily on red square presence/absence
- Poor generalization to test images without red squares
- Catastrophic failure on Earth bird classification

6.1.2 Balanced Training Impact

Training without group balancing would result in:

- High accuracy on LL and WW groups
- Poor accuracy on LW and WL groups
- Low worst-case performance

6.1.3 Bias Detection Sensitivity

The number of iterations in bias detection affects:

- Too few: Insufficient bias amplification
- Too many: Risk of overfitting to specific samples
- 7 iterations: Empirically good balance

7 Extensions and Future Work

7.1 Automated Hyperparameter Tuning

The approach could be enhanced with:

- Cross-validation for iteration count selection
- Learning rate scheduling
- Dynamic group balancing based on performance gaps

7.2 Multi-Attribute Bias Handling

Extending to handle multiple bias sources simultaneously:

- Multiple spurious correlations
- Hierarchical bias structures
- Continuous bias attributes

8 Conclusion

The ”Angry Birds” problem demonstrates several important principles in robust machine learning:

1. **Bias identification is crucial**: Systematic identification of biased samples enables targeted mitigation
2. **Balanced training improves worst-case performance**: Equal treatment of all groups improves robustness
3. **Simple solutions can be effective**: Linear classifiers with good features often outperform complex end-to-end models
4. **Preprocessing matters**: Addressing obvious biases through preprocessing prevents shortcut learning

The solution provides a practical framework for handling distribution shift in classification tasks, with broad applicability beyond the specific contest problem. The approach is particularly valuable in scenarios where:

- Training data contains known spurious correlations
- Test distribution differs from training distribution
- Worst-case performance is critical (e.g., safety-critical applications)
- Limited computational resources prevent complex domain adaptation methods

This problem showcases how contest problems can effectively teach fundamental concepts in machine learning robustness while providing engaging, narrative-driven challenges that make complex topics accessible to learners.