

## AI SCIENTIST TEAM REVIEW

**Paper Summary** This paper studies the application of Deep Learning models for a real-world application to pest prediction. It introduces an Environmental Robustness Score that leverages various data augmentation techniques mimicing environmental factors affecting data collection. It compares various learning rates and compares the impact of out-of-distribution testing settings across non-pest vision datasets.

### Strengths

- The paper fits the ICBNB workshop topic especially well. It discusses a real-world application of Deep Learning methods to pest prediction.
- Understanding the differential impact of training and out-of-distribution data augmentation techniques settings across datasets is interesting.

### Weaknesses

- The paper multiple times refers to domain adaptation being studied. The experiments, on the other hand, only investigate the usage of data augmentation methods (such as lighting, blurring, and contrast manipulation). Furthermore, studying the impact of the learning rate on generalization is fairly trivial.
- It is hard to motivate that the Eurosat, Medmnist and CIFAR-10 results are related to the pest prediction problem. Why should a result on these datasets transfer to pest prediction?
- Some of the statements regarding multi-dataset training are misleading. There are no results in the paper that result from such a training setup. Instead, multiple models are trained on individual datasets.

### Scores

- Soundness: 2 fair.  $\Rightarrow$  Interesting research question with potentially simple empirical evaluation setup.
- Presentation: 1 poor.  $\Rightarrow$  Wrong description and duplication of figures. Missing citation and downplaying of related work.
- Contribution: 1 poor.  $\Rightarrow$  While the question considered is important, the displayed results do not provide enough evidence for the conclusions drawn.
- Overall - Workshop: 3/10 (Reject): For instance, a paper with technical flaws, weak evaluation, inadequate reproducibility and incompletely addressed ethical considerations.
- Overall - Conference: 2/10: (Strong reject): For instance, a paper with major technical flaws, and/or poor evaluation, limited impact, poor reproducibility and mostly unaddressed ethical considerations.
- Confidence: 4/5. You are confident in your assessment, but not absolutely certain. It is unlikely, but not impossible, that you did not understand some parts of the submission or that you are unfamiliar with some pieces of related work.

### Additional Comments

- The presentation of the results needs significant improvement. There are multiple missing citations (?) and the interpretation of the results can be misleading. This includes the conclusions with regards to the impact of a lower learning rate on overfitting or naming the multi-model-single-dataset experiment "multit-dataset".

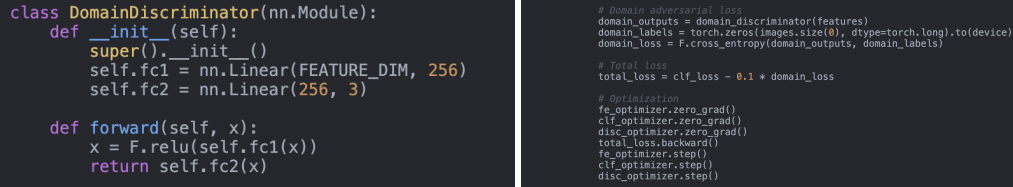
**Potential Violation of Code of Ethics:** No.

## AI SCIENTIST TEAM CODE REVIEW

### B.1 DOMAIN ADAPTATION AND MULTI-DATASET TRAINING

The paper seems to describe the “Domain Adaptation” experiment as primarily focused on transferring ImageNet-pretrained models to other vision datasets. After reviewing the code, we found attempts to implement a domain adaptation technique by training a separate classifier to distinguish different domains, but these attempts were unsuccessful. In the end, the AI Scientist opted for an implementation that does not include this domain adaptation technique.

Moreover, in the code where this domain adaptation technique was implemented, multi-dataset training was correctly performed as well—training a single model on all three datasets with domain discriminator loss, as shown in Figure 6. Had this code run successfully, the AI Scientist would likely have chosen it over the one ultimately selected, which lacked proper multi-dataset training but ran without errors.



```

class DomainDiscriminator(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(FEATURE_DIM, 256)
        self.fc2 = nn.Linear(256, 3)

    def forward(self, x):
        x = F.relu(self.fc1(x))
        return self.fc2(x)

```

```

# Training loop
for epoch in range(NUM_EPOCHS):
    feature_extractor.train()
    domain_discriminator.train()
    classifier.train()

    for dataset_name, (train_dataset, test_dataset) in datasets.items():
        train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True)

        for batch in train_loader:
            # Move batch to device
            images = batch["image"].to(device)
            labels = batch["label"].to(device)

            # Feature extraction
            features = feature_extractor(images)

            # Classification loss
            clf_outputs = classifier(features)
            clf_loss = F.cross_entropy(clf_outputs, labels)

            # Domain adversarial loss
            domain_outputs = domain_discriminator(features)
            domain_labels = torch.zeros(images.size(0), dtype=torch.long).to(device)
            domain_loss = F.cross_entropy(domain_outputs, domain_labels)

            # Total loss
            total_loss = clf_loss + 0.1 * domain_loss

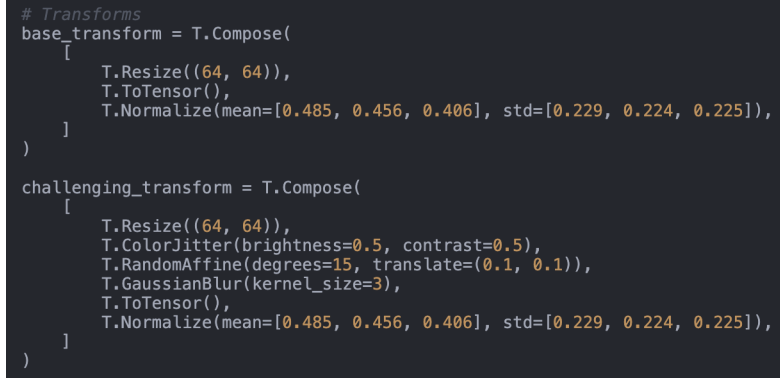
            # Optimization
            fe_optimizer.zero_grad()
            clf_optimizer.zero_grad()
            disc_optimizer.zero_grad()
            total_loss.backward()
            fe_optimizer.step()
            clf_optimizer.step()
            disc_optimizer.step()

```

Figure 6: Domain discriminator and multi-dataset training loop.

### B.2 ENVIRONMENTAL NOISE IMPLEMENTATION

The paper states, “To simulate challenging environmental conditions, we applied data augmentations during testing, including brightness and contrast adjustments, Gaussian blur, and random affine transformations.” This is confirmed in the code, as shown in Figure 7.



```

# Transforms
base_transform = T.Compose(
    [
        T.Resize((64, 64)),
        T.ToTensor(),
        T.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    ]
)

challenging_transform = T.Compose(
    [
        T.Resize((64, 64)),
        T.ColorJitter(brightness=0.5, contrast=0.5),
        T.RandomAffine(degrees=15, translate=(0.1, 0.1)),
        T.GaussianBlur(kernel_size=3),
        T.ToTensor(),
        T.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
    ]
)

```

Figure 7: Environment noise simulation implementation.

The calculation of the Environmental Robustness Score—a metric introduced by the AI Scientist and defined as “the ratio of model accuracy under challenging conditions to that under normal conditions, to quantify robustness”—matches the description in the paper, as shown in Figure 8.

```
def calculate_ers(model, normal_loader, challenging_loader):
    model.eval()
    with torch.no_grad():
        # Normal conditions
        correct_normal = 0
        total_normal = 0
        for images, labels in normal_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            total_normal += labels.size(0)
            correct_normal += (predicted == labels).sum().item()
        normal_acc = correct_normal / total_normal

        # Challenging conditions
        correct_challenging = 0
        total_challenging = 0
        for images, labels in challenging_loader:
            images, labels = images.to(device), labels.to(device)
            outputs = model(images)
            _, predicted = torch.max(outputs.data, 1)
            total_challenging += labels.size(0)
            correct_challenging += (predicted == labels).sum().item()
        challenging_acc = correct_challenging / total_challenging

    ers = (challenging_acc / normal_acc) * 100
    return ers
```

Figure 8: Environmental Robustness Score calculation.