AI SCIENTIST TEAM REVIEW

Paper Summary This paper studies the impact of label noise on model calibration using different noise models. More specifically, the paper contrasts symmetric (unstructured label perturbations) and asymmetric (structured label perturbations) noise. The empirical experiments consider standard small-scale vision datasets (i.e. MNIST, Fashion-MNIST and CIFAR-10) and demonstrate that asymmetric noise leads to higher expected calibration error.

Strengths

- The research question is of real-world importance and shines light on the impact of noisy labels beyond their effect on prediction accuracy.
- The study design is simple and focuses on a single key factor, i.e. the impact of different noise models (asymmetric noise increasing ECE more than symmetric noise). The considered datasets are appropriate for a workshop submission.
- The impact of the different noise models on the downstream model calibration is robust and consistent across the considered datasets.

Weaknesses

- There are multiple instances where the written interpretation of results are not substantially supported by the empirical results presented. E.g. the paragraph interpreting figure 3 refers to ECE measures, which are not displayed in the figures.
- The paper states that it compares different calibration methods, but the paper does not provide any results. The same holds for the mentioned reliability diagrams.
- Furthermore, the supplementary material includes duplicate figures, a missing citation for SVHN and a corresponding missing figure.

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 biggest flaw of this paper is the mentioning of results which are not substantiated by results. This includes the assessment of various methods tailored to uncertainty calibration as well as the usage of reliability diagrams. The paper could be substantially improved if these results were added and the selection of displayed figure results was better curated.
- The readability of figure 2 should be improved by splitting the 6 plots across 2 rows. Furthermore, the related work section appears to dismiss efforts by the scientific community to relate calibration and noisy data.

Potential Violation of Code of Ethics: No.

AI SCIENTIST TEAM CODE REVIEW

A.5 TEMPERATURE SCALING

In our review of the paper, we noted that it lacked experiments involving temperature scaling. Upon inspecting the generated code, we found that the AI Scientist had implemented temperature scaling as can be seen in Figure 7 but never actually used it.

During the paper writing stage, the AI Scientist had access to a set of generated experiment code and its initial plans before generating the code. As a result, it is likely that the paper was influenced by these plans and code, which included temperature scaling, but the AI Scientist failed to realize that the experiments using temperature scaling were never actually conducted.

```python
class TemperatureScaling(nn.Module):
    def __init__(self):
        super(TemperatureScaling, self).__init__()
        self.temperature = nn.Parameter(torch.ones(1))

    def forward(self, logits):
        return logits / self.temperature
```

Figure 7: Temperature scaling implementation.

A.6 DATASET CLASS

We found that the initial implementation of the dataset class lacked an option for symmetric/asymmetric noise distribution, even though it was part of the initial plan. The AI Scientist recognized this mistake and later implemented the correct version, as shown in Figure 8.

In the main paper, the AI Scientist wrote: "Assymetric Noise: Labels are flipped to specific incorrect classes based on a predefined confusion matrix, simulating more realistic mislabeling." The asymmetric noise implementation in the generated code always maps class i to class (i+1) % NUM CLASSES. While this is a valid approach, it is worth noting that there are other ways to implement asymmetric noise.

```python
# Data preparation with noise injection
class NoisyDataset(Dataset):
    def __init__(self, dataset, noise_rate=0.2):
        self.dataset = dataset
        self.noise_rate = noise_rate
        self.noisy_labels = self._inject_noise()

    def _inject_noise(self):
        labels = np.array([y for _, y in self.dataset])
        mask = np.random.rand(len(labels)) < self.noise_rate
        noisy_labels = labels.copy()
        noisy_labels[mask] = np.random.randint(0, NUM_CLASSES, mask.sum())
        return torch.LongTensor(noisy_labels)

    def __getitem__(self, index):
        image, _ = self.dataset[index]
        return image, self.noisy_labels[index]

    def __len__(self):
        return len(self.dataset)
```

```python
class NoisyDataset(Dataset):
    def __init__(self, dataset, noise_type="symmetric", noise_rate=0.2):
        self.dataset = dataset
        self.noise_rate = noise_rate
        self.noise_type = noise_type
        self.noisy_labels = self._inject_noise()

    def _inject_noise(self):
        labels = np.array([y for _, y in self.dataset])
        if self.noise_type == "symmetric":
            mask = np.random.rand(len(labels)) < self.noise_rate
            noisy_labels = labels.copy()
            noisy_labels[mask] = np.random.randint(0, NUM_CLASSES, mask.sum())
        else:  # asymmetric noise
            noisy_labels = labels.copy()
            for i in range(NUM_CLASSES):
                mask = (labels == i) & (np.random.rand(len(labels)) < self.noise_rate)
                noisy_labels[mask] = (i + 1) % NUM_CLASSES
        return torch.LongTensor(noisy_labels)

    def __getitem__(self, index):
        image, _ = self.dataset[index]
        return image, self.noisy_labels[index]

    def __len__(self):
        return len(self.dataset)
```

Figure 8: Noisy dataset class implementation.

A.7 EVALUATION FUNCTION

The evaluation function used to compute the Expected Calibration Error is shown in Figure 9. We manually created test cases and used the MulticlassCalibrationError function with norm='l1' from torchmetrics as the ground truth. Since the MulticlassCalibrationError function expects probability inputs, we omitted the softmax operation in the first line to align with the implementation details. After this adjustment, we confirmed that both functions produce the same results, apart from minor numerical differences.

```python
def compute_ece(logits, labels, n_bins=15):
    softmaxes = softmax(logits, dim=1)
    confidences, predictions = torch.max(softmaxes, 1)
    accuracies = predictions.eq(labels)

    bin_boundaries = torch.linspace(0, 1, n_bins + 1)
    bin_lowers = bin_boundaries[:-1]
    bin_uppers = bin_boundaries[1:]

    ece = torch.zeros(1, device=logits.device)
    for bin_lower, bin_upper in zip(bin_lowers, bin_uppers):
        in_bin = confidences.gt(bin_lower.item()) * confidences.le(bin_upper.item())
        prop_in_bin = in_bin.float().mean()
        if prop_in_bin.item() > 0:
            accuracy_in_bin = accuracies[in_bin].float().mean()
            avg_confidence_in_bin = confidences[in_bin].mean()
            ece += torch.abs(avg_confidence_in_bin - accuracy_in_bin) * prop_in_bin
    return ece.item()
```

Figure 9: The implementation for Expected Calibration Error.