TrackTempo: Loss Function Strategy Summary

1. Current Setup

The current model uses:

```
criterion = nn.BCEWithLogitsLoss()
```

This treats each horse in a race independently and predicts the probability of that horse winning. It does not enforce that only one horse wins per race, and can lead to incorrect confidence scaling across a race field.

2. Why This is Problematic for Racing

- In a real race, only one horse wins.
- BCE does not model the mutual exclusivity of outcomes.
- You can end up with multiple horses with high win probabilities.
- This becomes more extreme as epochs increase (softmax skew).

3. Recommended Upgrade: CrossEntropyLoss

Use: `criterion = nn.CrossEntropyLoss()`

Treat the race as a multi-class classification problem:
- Output shape: [B, N] where N is number of horses per race
- Target: index of the winning horse
- Softmax normalization ensures one clear winner is selected

Pros:
- Better aligned with racing reality
- Simple to implement
- Easily supports ranking visualizations

4. Alternative Strategy: Ranking Loss

Use: MarginRankingLoss or Listwise ranking loss

These optimize the order of horses, not just the winner:

- Output: scores for each horse
- Target: derived from true finishing positions
- Compares relative strength between pairs or lists

Pros:

- Encourages full race ranking fidelity
- Great for exotic bets (place, forecasts, etc.)
- Better for leaderboard scoring

Cons:

- More complex to implement
- Harder to evaluate with simple metrics

5. Can You Combine Them?

Yes - in advanced systems you may combine both:

```
loss = alpha * CrossEntropyLoss + (1 - alpha) * RankingLoss
```

This blends:

- Winner-picking accuracy
- Ordered finish predictions

It is used in information retrieval and elite recommendation systems, but may be overkill early on.

6. Recommended Structure

Build a modular loss factory:

- Use argparse or config.yaml to select loss:
  --loss cross_entropy
  --loss ranking
  --loss bce

Example:

```
    if config.loss_type == "ranking":
        criterion = CustomRankingLoss()
    elif config.loss_type == "cross_entropy":
        criterion = nn.CrossEntropyLoss()
```

Log:

- Epoch loss
- Accuracy, Top-k, and ranking correlation (Kendall, Spearman)

This gives flexibility to experiment with loss strategies cleanly and compare performance on validation/test races.