My research focuses on **understanding and improving optimization algorithms in modern machine learning and deep learning**. Optimization involves learning network parameters to minimize the loss function, which measures the discrepancy between the model's predictions and target. This is often achieved via algorithms like (stochastic) gradient descent, where model parameters are updated iteratively based on a small batch of data. This process is crucial for the success of language models, image recognition and generation, and robotics.

Historically, theoretical insights from numerical optimization have guided practical algorithm design and provided performance guarantees. However, modern algorithms like Adam evolved from intuitions and heuristics, and can often perform better in modern architureres such as Transformers. Common theoretical assumptions fail to capture the characteristics of deep learning tasks, making it hard to explain the advantages of these optimization methods. Consequently, guaranteeing performance and convergence speed is challenging, and these methods can be inefficient and unstable. Practitioners extensively tune hyper-parameters throughout repetitive training which requires unsustainable compute resources, and the resulting algorithm often fails to transfer to other tasks directly.

Understanding the specific properties of modern learning tasks and algorithms can offer insights into practically verifiable assumptions in theoretical analysis, allowing better guarantees, and guiding algorithmic designs to reduce training costs and improve performance. To this end, here are several long-term goals and concrete examples:

- **Examine the role of theoretical assumptions in practice**

Adam is rotation non-invariant. Common assumptions like convexity and smoothness assume rotation invariance and cannot explain Adam's advantage over SGD. What are some rotation non-invariant assumptions that capture Adam's adaptivity? Can we verify them in practice?

- **Understand large model training dynamics**

Prior works attribute SGD's success in deep learning to implicit bias from mini-batch noise, which is thought to help find better local minima in non-convex tasks. Can this phenomenon be verified and explained? Additionally, can we understand how the optimization algorithms, other with their hyperparameters and heuristics like batch size, warmup, dropouts, and normalization techniques, influence training dynamics?

- **Improve the efficiency of existing algorithms**

Compared to SGD-based optimization, Adam optimizer results in more than double the memory overhead, posing a significant challenge in training large models. Are all information in the algorithms necessary during all stages of training? Can we identify and optimize the key components of these algorithms to improve efficiency while maintaining performance?

- **Develop task-aware optimizers**

Optimization tools for reinforcement learning (RL) can differ significantly from those for supervised learning. RL is more challenging because model predictions can influence future data, and reward signals are often delayed or sparse. However, noise and randomness in RL can encourage exploration. How can we identify useful theoretical assumptions that capture these characteristics and improve algorithms tailored to each task?