# CS 744: PYTORCH

Shivaram Venkataraman

Fall 2022

# ADMINISTRIVIA

Assignment 2 out! Due Oct 12th 11 AM!

Paper review deadline – 12:59pm on Tue/Thu

Bid on topics, submit group (1 sentences) – Oct 11

Title confirmed – Oct 15

Project Proposal (2 pages) – Oct 25

- Introduction
- Related Work
- Timeline (with eval plan)

# WRITING AN INTRODUCTION

1-2 paras: what is the problem you are solving

why is it important (need citations)

1-2 paras: How other people solve and why they fall short

1-2 paras: How do you plan on solving it and why your approach is better

1 para: Anticipated results or what experiments you will use

# RELATED WORK, EVAL PLAN

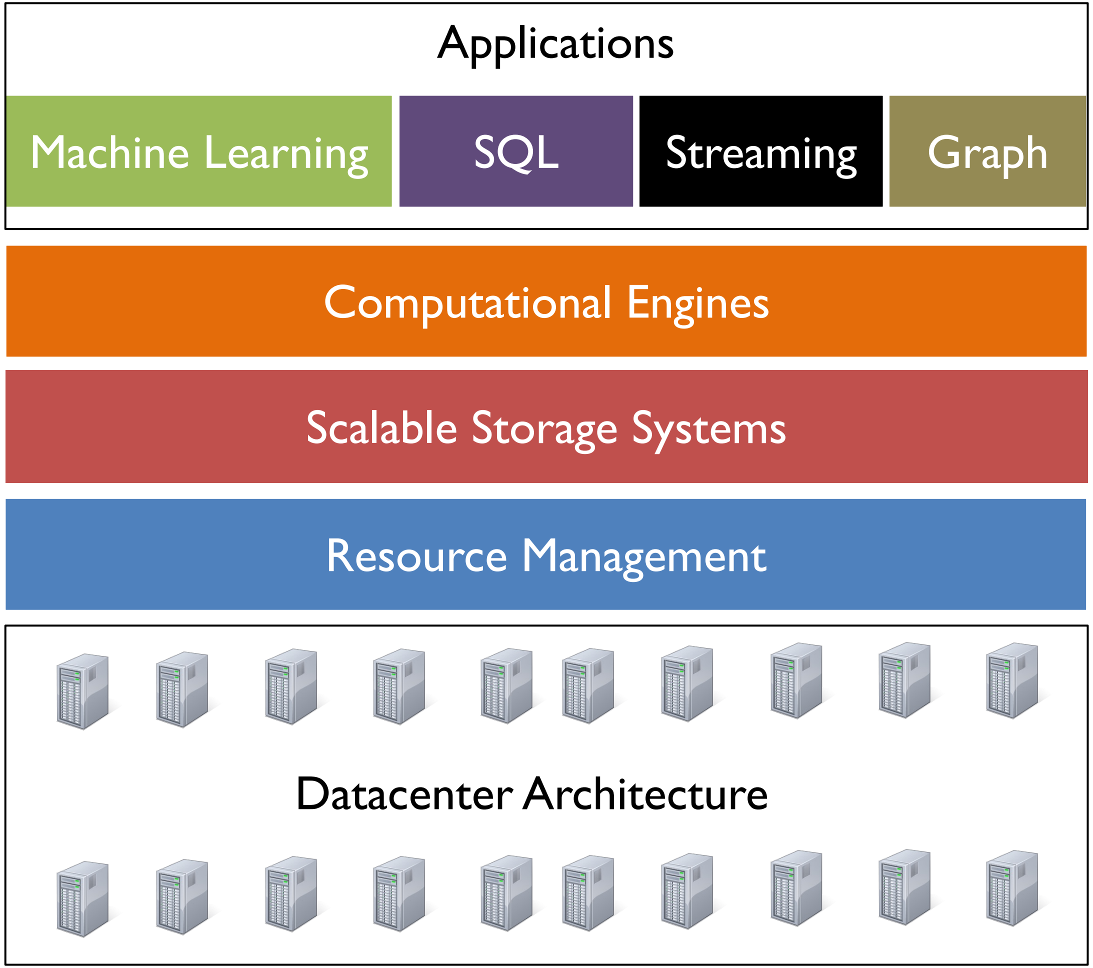
Group related work into 2 or 3 buckets (1-2 para per bucket)

Explain what the papers / projects do

Why are they different / insufficient

Eval Plan

- Describe what datasets, hardware you will use
- Available: Cloudlab, Google Cloud (~$150), Jetson TX2 etc.


Applications
Machine Learning
SQL
Streaming
Graph
Computational Engines
Scalable Storage Systems
Resource Management
Datacenter Architecture

# EMPIRICAL RISK MINIMIZATION

$$\min_{w \in \mathbb{R}^d} \quad \sum_{i=1}^{N} f(w, z_i) + P(w)$$

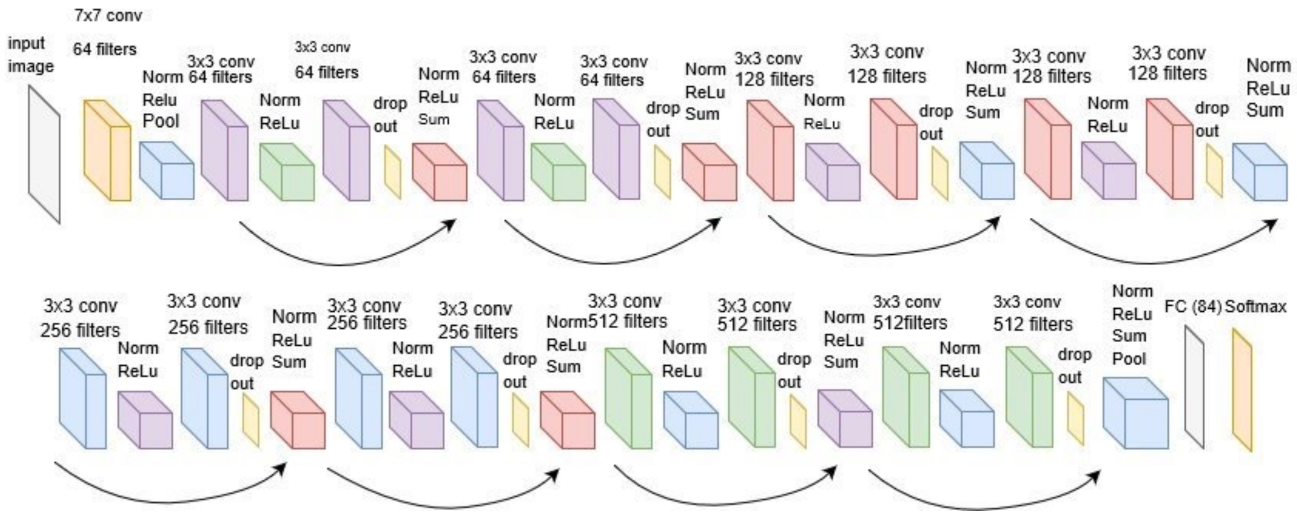
Regularization

Function

Model

Data (Examples)

# DEEP LEARNING




## ResNet18

Convolution
ReLU
MaxPool
Fully Connected
SoftMax

# STOCHASTIC GRADIENT DESCENT

$$w^{(k+1)} = w^{(k)} - \alpha_k \nabla f(w^{(k)})$$

Initialize w

For many iterations:

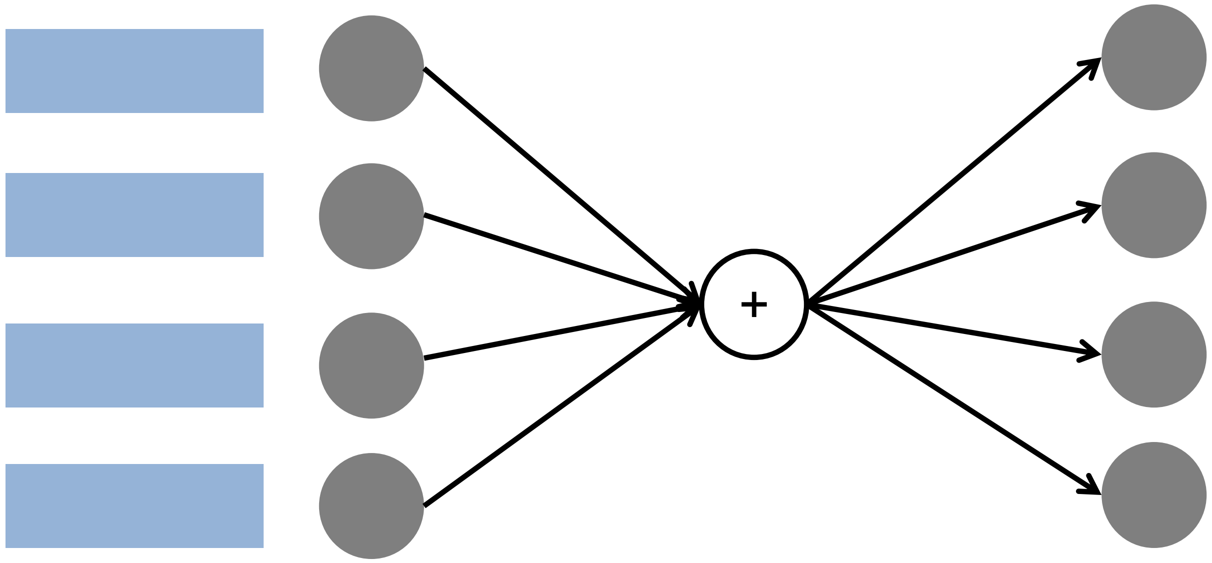
    Loss = Forward pass

    Gradient = backward

    Update model

End

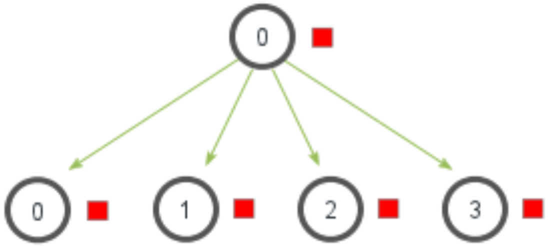
# DATA PARALLEL MODEL TRAINING



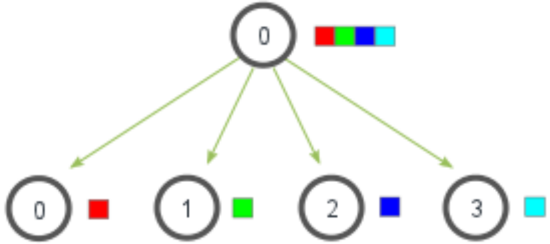
# COLLECTIVE COMMUNICATION

## Broadcast, Scatter

MPI_Bcast

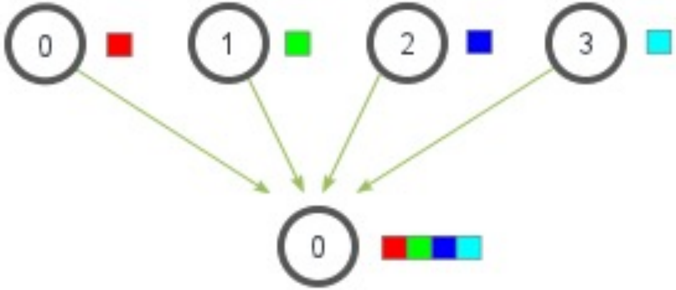



MPI_Scatter

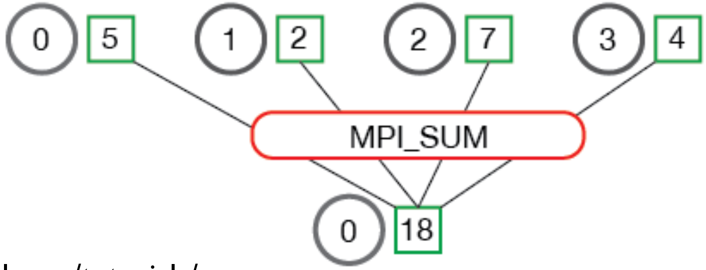



## Gather, Reduce

MPI_Gather

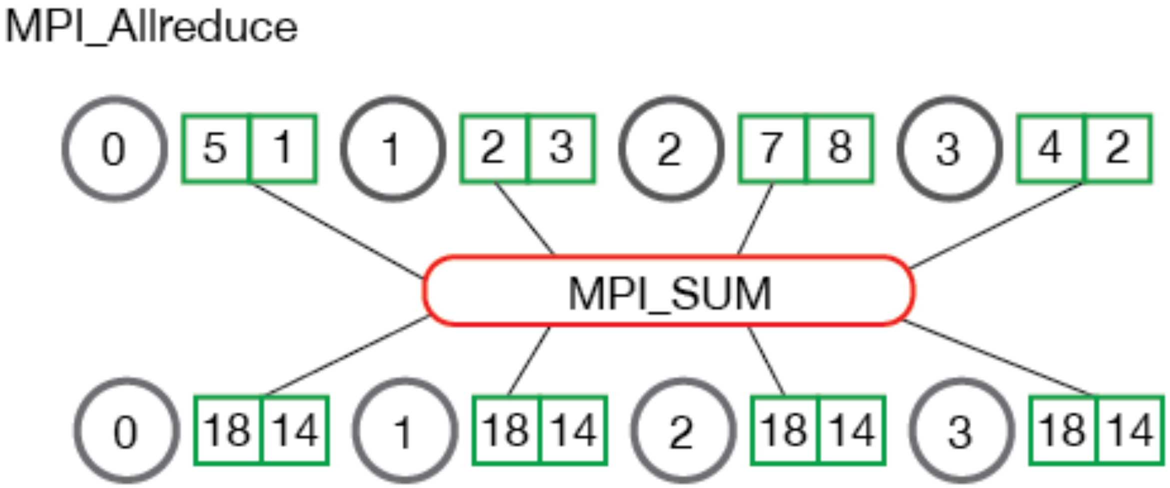



MPI_Reduce




From https://mpitutorial.com/tutorials/

# ALL REDUCE USING A RING



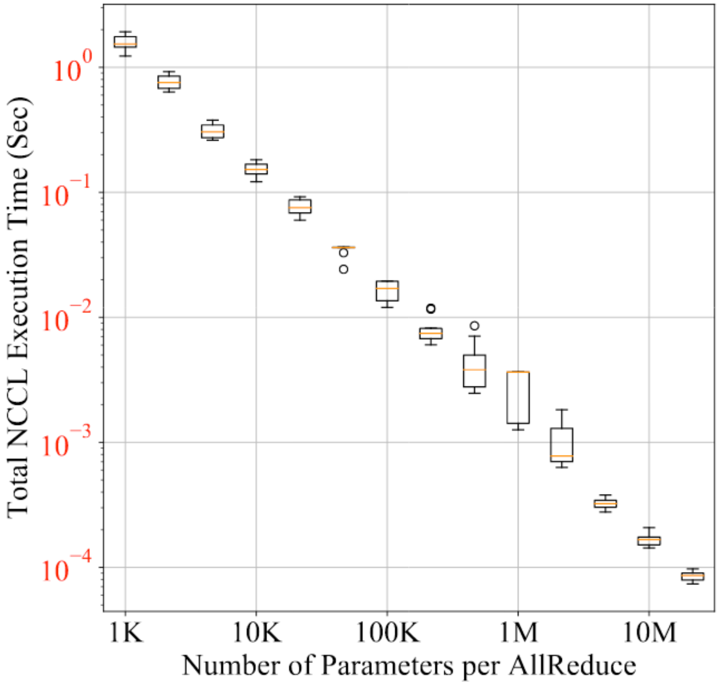

From https://mpitutorial.com/tutorials/

# DISTRIBUTED DATA PARALLEL API

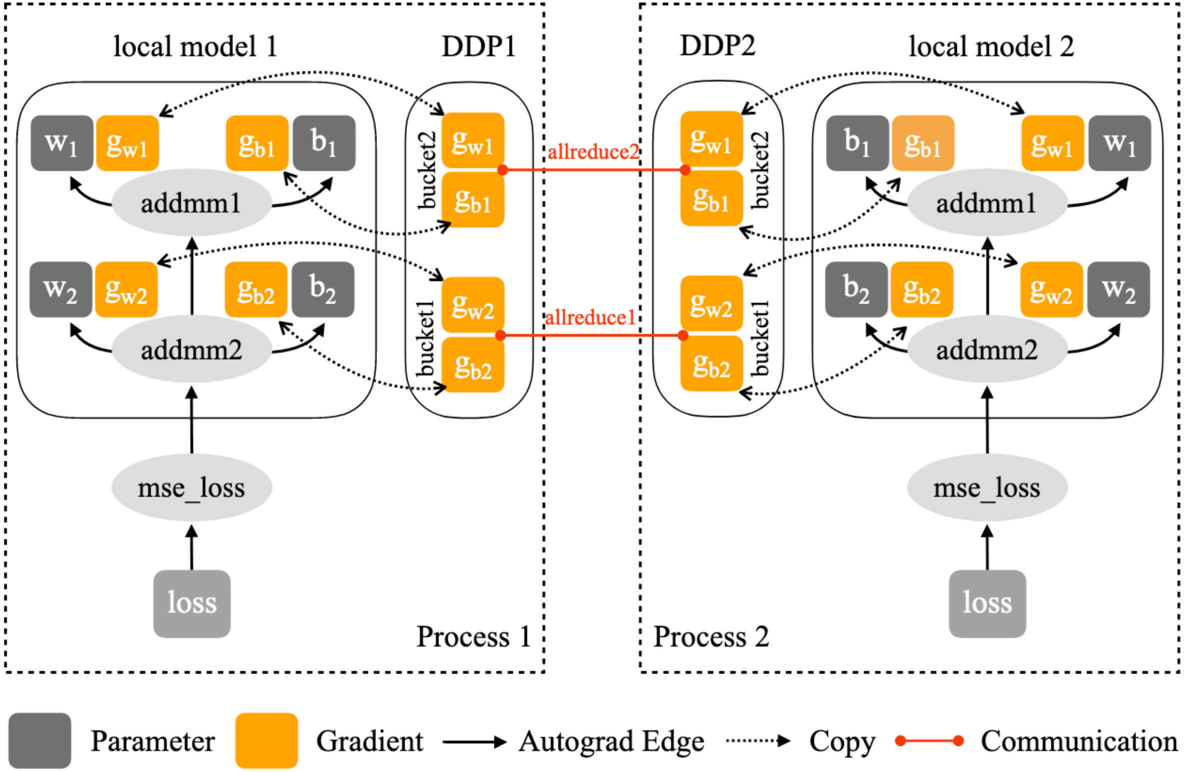
```
# setup model and optimizer
net = nn.Linear(10, 10)
net = par.DistributedDataParallel(net)
opt = optim.SGD(net.parameters(), lr=0.01)

# run forward pass
inp = torch.randn(20, 10)
exp = torch.randn(20, 10)
out = net(inp)

# run backward pass
nn.MSELoss()(out, exp).backward()

# update parameters
opt.step()
```

# GRADIENT BUCKETING

Why do we need gradient bucketing?



# GRADIENT BUCKETING + ALL REDUCE



# GRADIENT ACCUMULATION

```
ddp = DistributedDataParallel(net)
with ddp.no_sync():
  for inp, exp in zip(inputs, expected_outputs):
    # no synchronization, accumulate grads
    loss_fn(ddp(inp), exp).backward()
# synchronize grads
loss_fn(ddp(another_inp), another_exp).backward()
opt.step()
```

# IMPLEMENTATION

Bucket_cap_mb

Parameter-to-bucket mapping

Round-robin ProcessGroups

# SUMMARY

Pytorch: Framework for deep learning

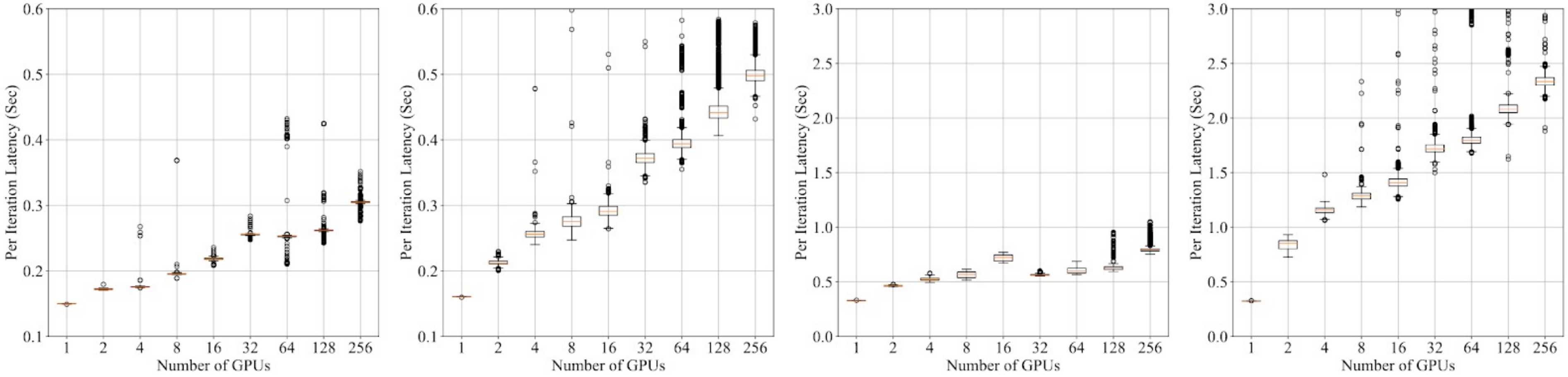
DistributedDataParallel API

Gradient bucketing, AllReduce

Overlap computation and communication

# DISCUSSION

https://forms.gle/jivzEEo5oz8tugYH9




(a) ResNet50 on NCCL (b) ResNet50 on Gloo (c) BERT on NCCL (d) BERT on Gloo

**Figure 9: Scalability**

# What could be some challenges in implementing similar optimizations for AllReduce in Apache Spark?

# NEXT STEPS

Next class: PipeDream

Assignment 2 is out!

Project Proposal – Check Piazza!

# BREAKDOWN

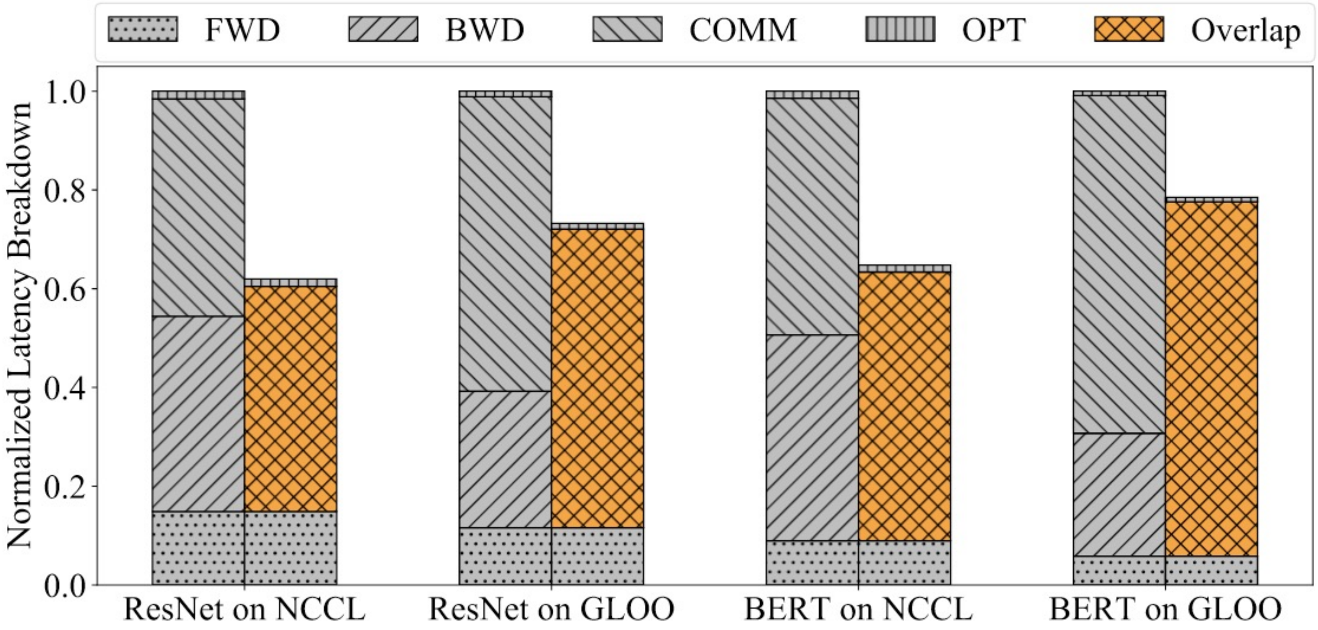



**Figure 6: Per Iteration Latency Breakdown**