1 NLP reproduce ability

1.1 Lora

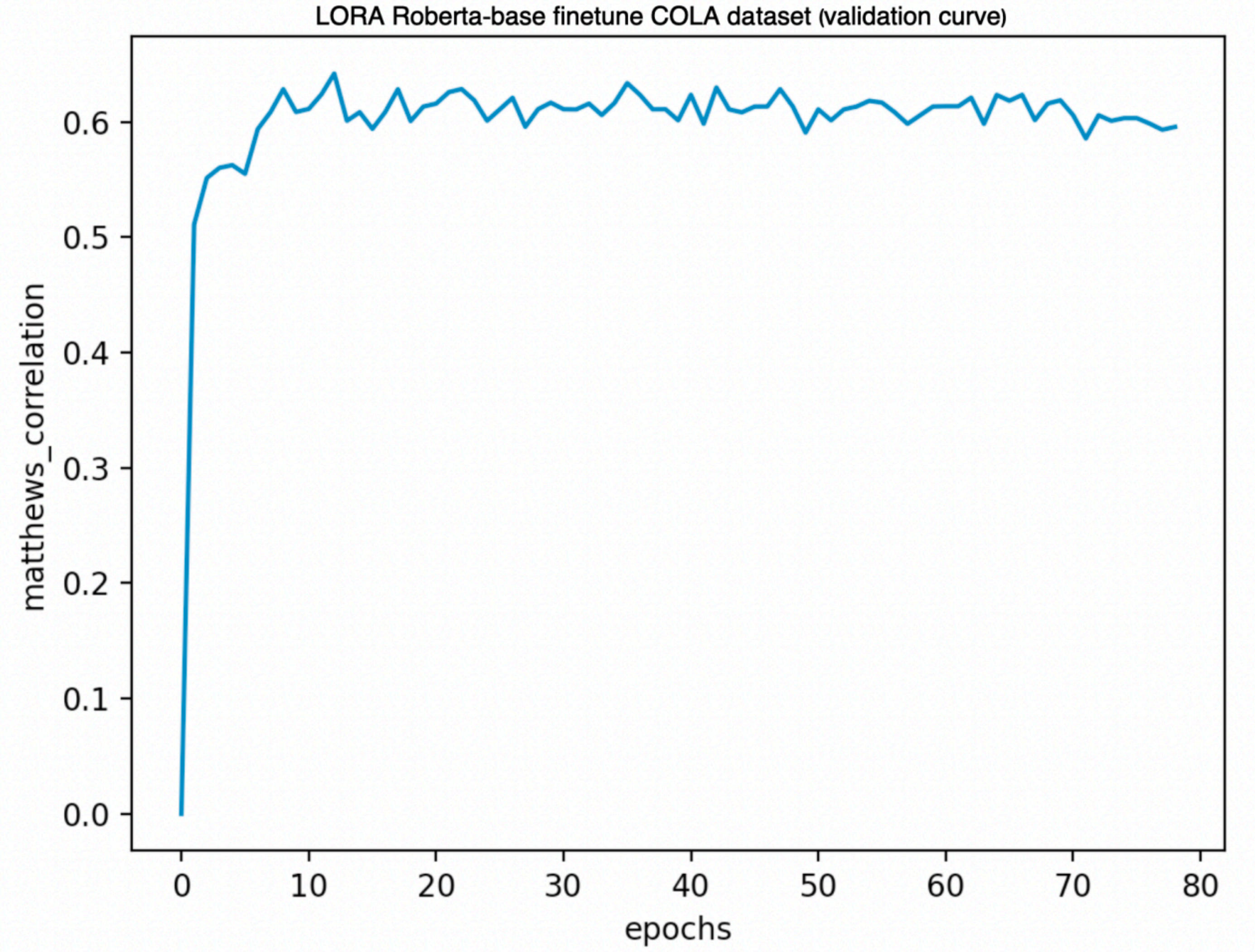
Here is a reproduction of LoRA

https://github.com/microsoft/LoRA

The best accuracy gets at 13 epochs. However, it has a set of about 80epochs. At the 13th epoch, the best accuracy occurs.

Then it slowly decays.

There is a graph about LORA implemented Roberta-base finetune COLA dataset, the y-axis is validation matthews correlation, and the x-axis is epoch number.



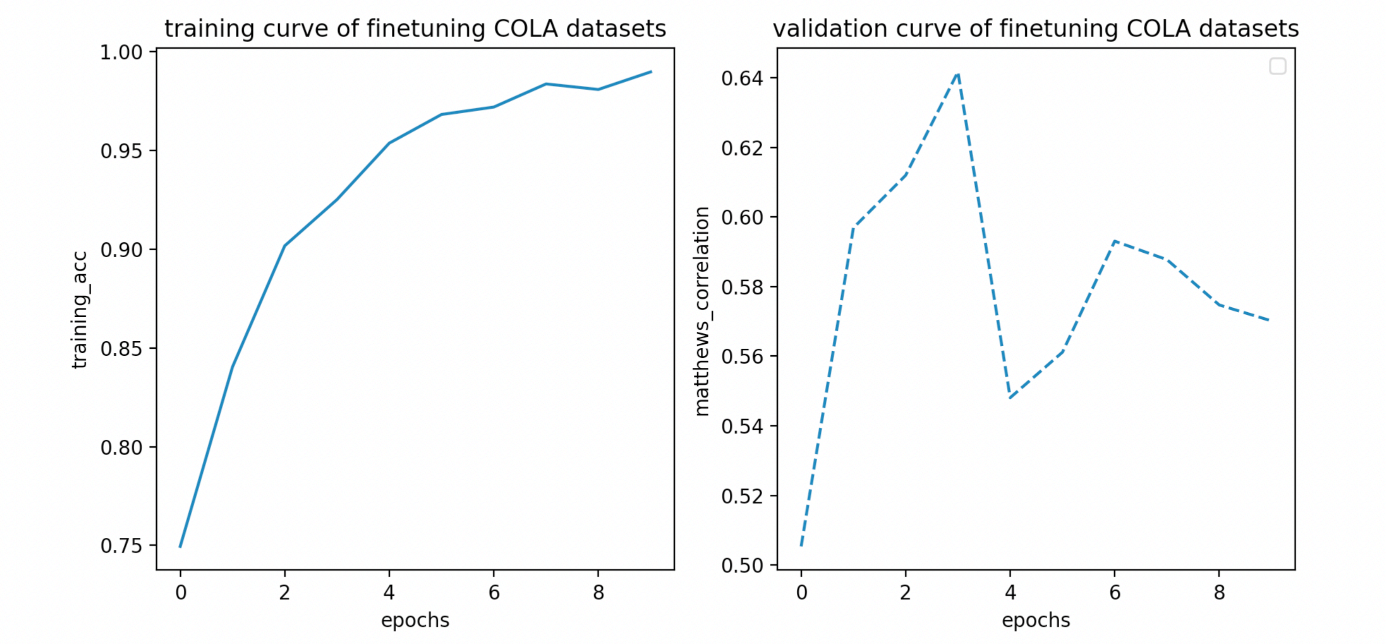
1.2 Fairseq

Also, fairest provides a set of parameters.

https://github.com/facebookresearch/fairseq/blob/main/examples/roberta/config/finetuning/cola.yaml

Backend	Roberta-base
Dataset	COLA
Batch size	16
Lr	1e-5
Weight decay	0.1
Optimizer	Adam(eps = 1e-6)
Epochs	10
Method	mixed precision

It only trains10 epochs. But the same problems occurs.

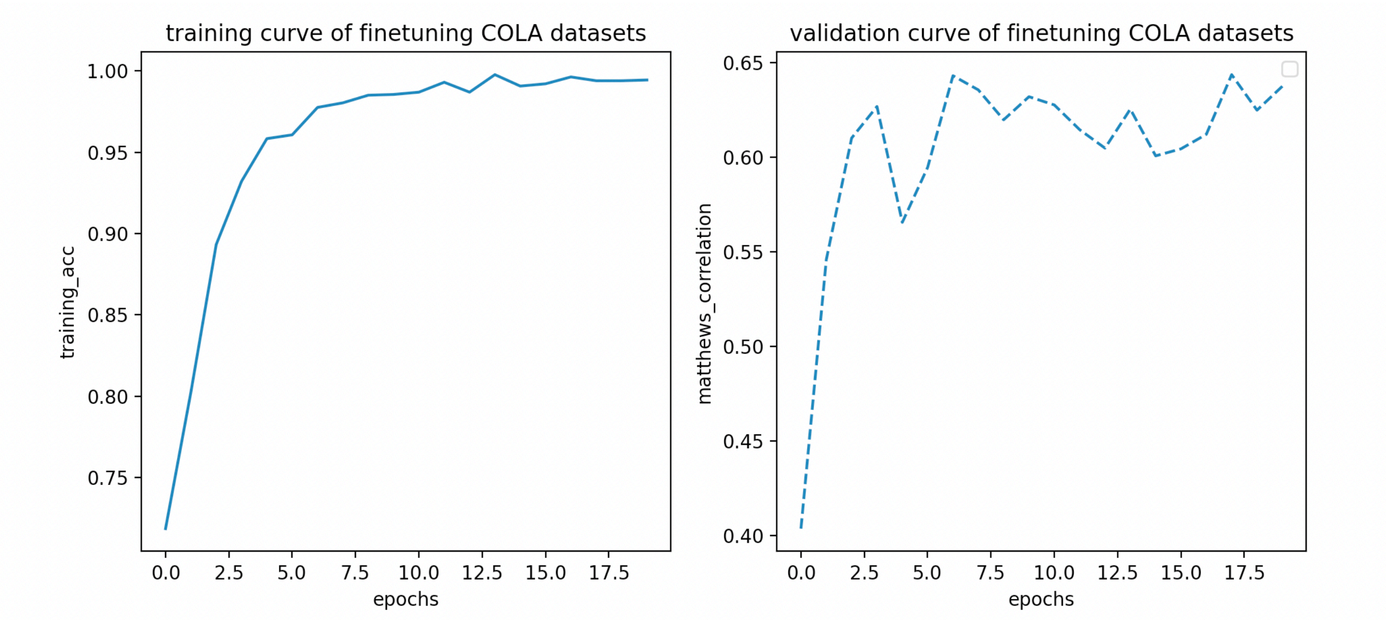


1.3 Gpipe settings

And my settings generate this curve.

Settings

Backend	Roberta-base
Dataset	COLA
Batch size	32
Lr	2e-5
Weight decay	1e-4
Optimizer	Adam(eps = 1e-6)
Epochs	20
Method	mixed precision



2 Pipeline training on two machines

2.1 Settings

Here I use 40CPUs to simulate the client and one GTX1080 to simulate the server

Backend	MobileNetV2
Dataset	CIFAR10
Batch size	64
Image size	[3,224,224]
Lr	0.005
Weight decay	0.0
Optimizer	SGD with momentum
Momentum	0.1
Partition	Client: Conv+bn，Classifier Server: Relu + features[1:]
Chunk	1,4,8

2.2 Results

Here are the definitions of the bandwidth.

$$Bandwidth_{avg} = data(send\ or\ recv)/calculation\_time$$
$$Bandwidth_{peak,client} = max(Bandwidth_{send})$$
$$Bandwidth_{peak,server} = max(Bandwidth_{recv})$$

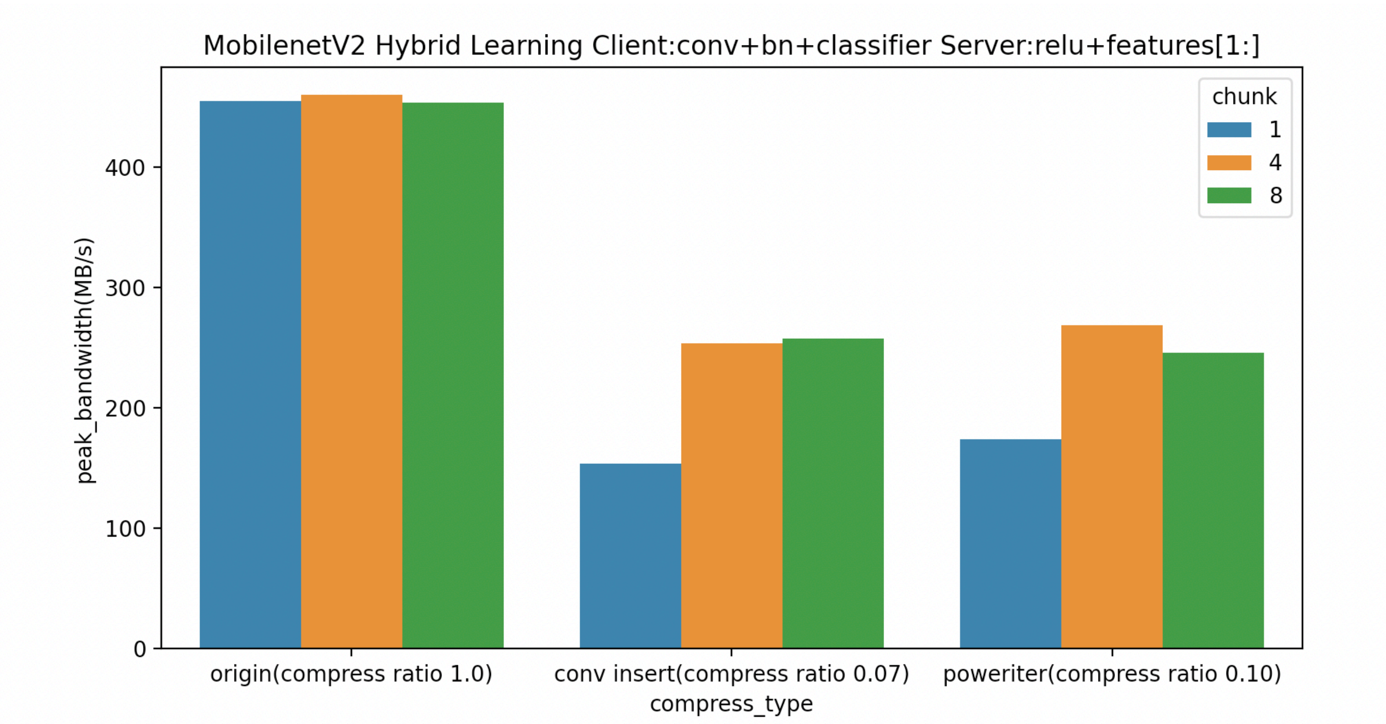
And there are two parameters for power iteration.

As we all know. power iteration is a way of PCA which has similar to SVD_lowrank. But SVD is a high-cost algorithm. Power iteration cost less since it uses QR decomposition instead of SVD decomposition. But QR decomposition could cost a lot when the rank is bigger.

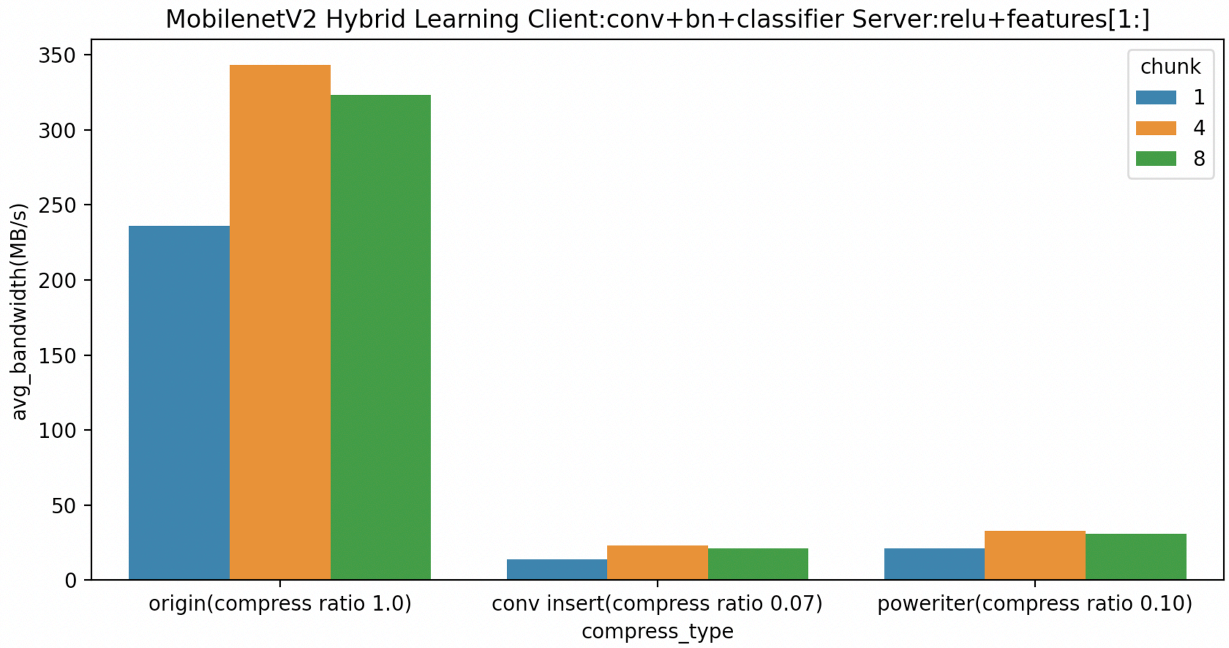
For CV tasks. Activation memory has a size of `[B,C,H,W]`. I unsqueeze the last two ranks to `[B,C,H*W]`. And then I use power iteration to spread it to `[B,C,rank]` and `[B,rank,H*W]`. As you can see, power iteration(3,7), means two ranks of the two sizes of activation memory.

Compress Method	Compress Rate	Acc	Bandwidth(Avg)	Bandwidth(Peak,Client)	Bandwidth(Peak,Server)	Computation Time	Total Time per batch	Chunk
None	1.0	95.86	236.07MB/s	455MB/s	454MB/s	0.48s	2.47s	1
None	1.0	96.04	343.37MB/s	460MB/s	457MB/s	0.33s	2.15s	4
None	1.0	95.94	323.75MB/s	454MB/s	458MB/s	0.35s	2.07s	8
Conv Insert	0.097	96.01	18.10MB/s	147MB/s	149MB/s	0.55s	0.38s	1
Conv Insert	0.097	96.01	29.27MB/s	251MB/s	253MB/s	0.34s	0.38s	4
Conv Insert	0.097	96.01	27.65MB/s	252MB/s	253MB/s	0.36s	0.38s	8
Conv Insert	0.070	95.89	13.92MB/s	154MB/s	148MB/s	0.55s	0.62s	1
Conv Insert	0.070	95.89	22.52MB/s	254MB/s	255MB/s	0.34s	0.37s	4
Conv Insert	0.070	95.87	21.27MB/s	258MB/s	261MB/s	0.36s	0.37s	8
Poweriter(3,7)	0.101	95.54	20.50MB/s	174MB/s	177MB/s	0.56s	0.66s	1
Poweriter(3,7)	0.101	95.54	32.80MB/s	269MB/s	249MB/s	0.35s	0.43s	4
Poweriter(3,7)	0.101	95.54	31.03MB/s	246MB/s	248MB/s	0.37s	0.43s	4

Peak bandwidth with different chunks and different compress algorithms.



Average bandwidth with different chunks and different compress algorithms.


MobilenetV2 Hybrid Learning Client:conv+bn+classifier Server:relu+features[1:]
avg_bandwidth(MB/s)
0
50
100
150
200
250
300
350
chunk
1
4
8
origin(compress ratio 1.0)
conv insert(compress ratio 0.07)
poweriter(compress ratio 0.10)
compress_type