Fine-tune the transformer model with LoRA

1. Task Analysis

Internet Movie Database (IMDb) is a Movies Reviews dataset for binary sentiment classification.

User Reviews

An excellent family movie... gives a lot to think on... There's absolutely nothing wrong in this film. Everything is just perfect. The script is great - it's so... real... such things could happen in everyone's life. And don't forget about acting - it's just awesome! Just look at Frankie and You'll know what I thought about... This picture is a real can't-miss!!!

Positive : 1

This is the worst film I have ever seen.I was watching this film with some friends and after 40 minutes we had enough. The plot was bad and there wasn't a single likeable character.I could get more entertainment watching static. I gave this movie a 1 only because the scale didn't go into negative numbers. Avoid this movie at all costs.

Negative : 0

Dataset overview

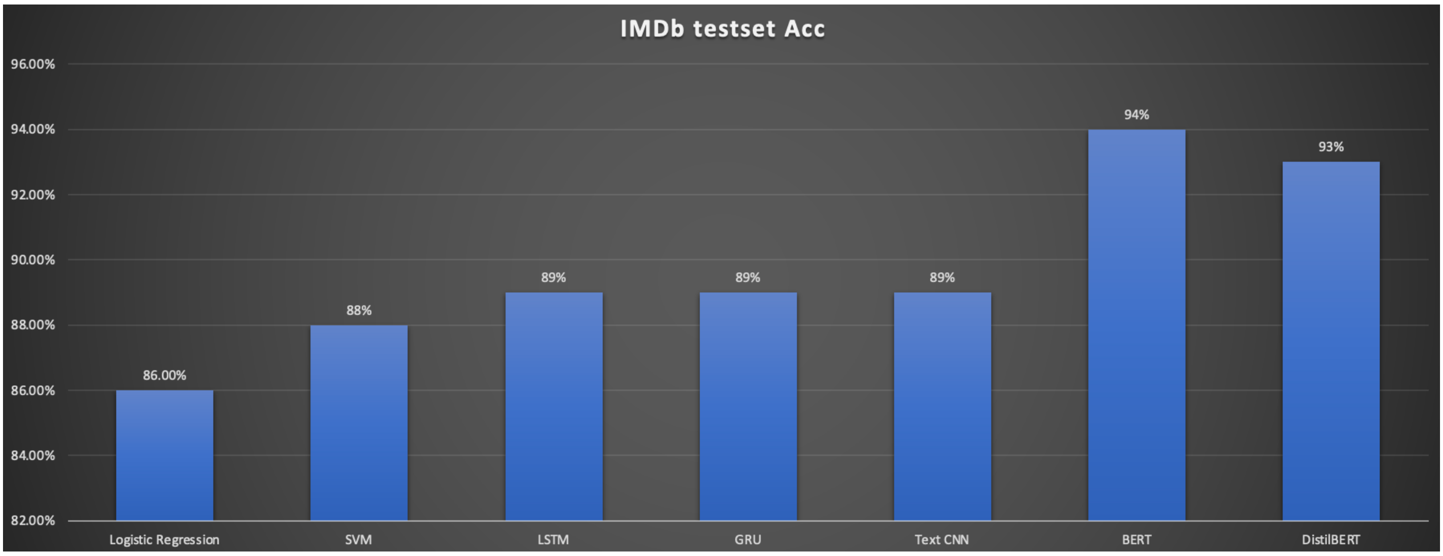
```
DatasetDict({
    train: Dataset({
        features: ['text', 'label'],
        num_rows: 25000
    })
    test: Dataset({
        features: ['text', 'label'],
        num_rows: 25000
    })
    unsupervised: Dataset({
        features: ['text', 'label'],
        num_rows: 50000
    })
})
```

	text	label
0	I rented I AM CURIOUS-YELLOW from my video sto...	0
1	"I Am Curious: Yellow" is a risible and preten...	0
2	If only to avoid making this type of film in t...	0
3	This film was probably inspired by Godard's Ma...	0
4	Oh, brother...after hearing about this ridicul...	0
...	...	...
24995	A hit at the time but now better categorised a...	1
24996	I love this movie like no other. Another time ...	1
24997	This film and it's sequel Barry Mckenzie holds...	1
24998	'The Adventures Of Barry McKenzie' started lif...	1
24999	The story centers around Barry McKenzie who mu...	1

25000 rows × 2 columns

	text_length
count	25000.00000
mean	1325.06964
std	1003.13367
min	52.00000
25%	702.00000
50%	979.00000
75%	1614.00000
max	13704.00000

2. How to classify?



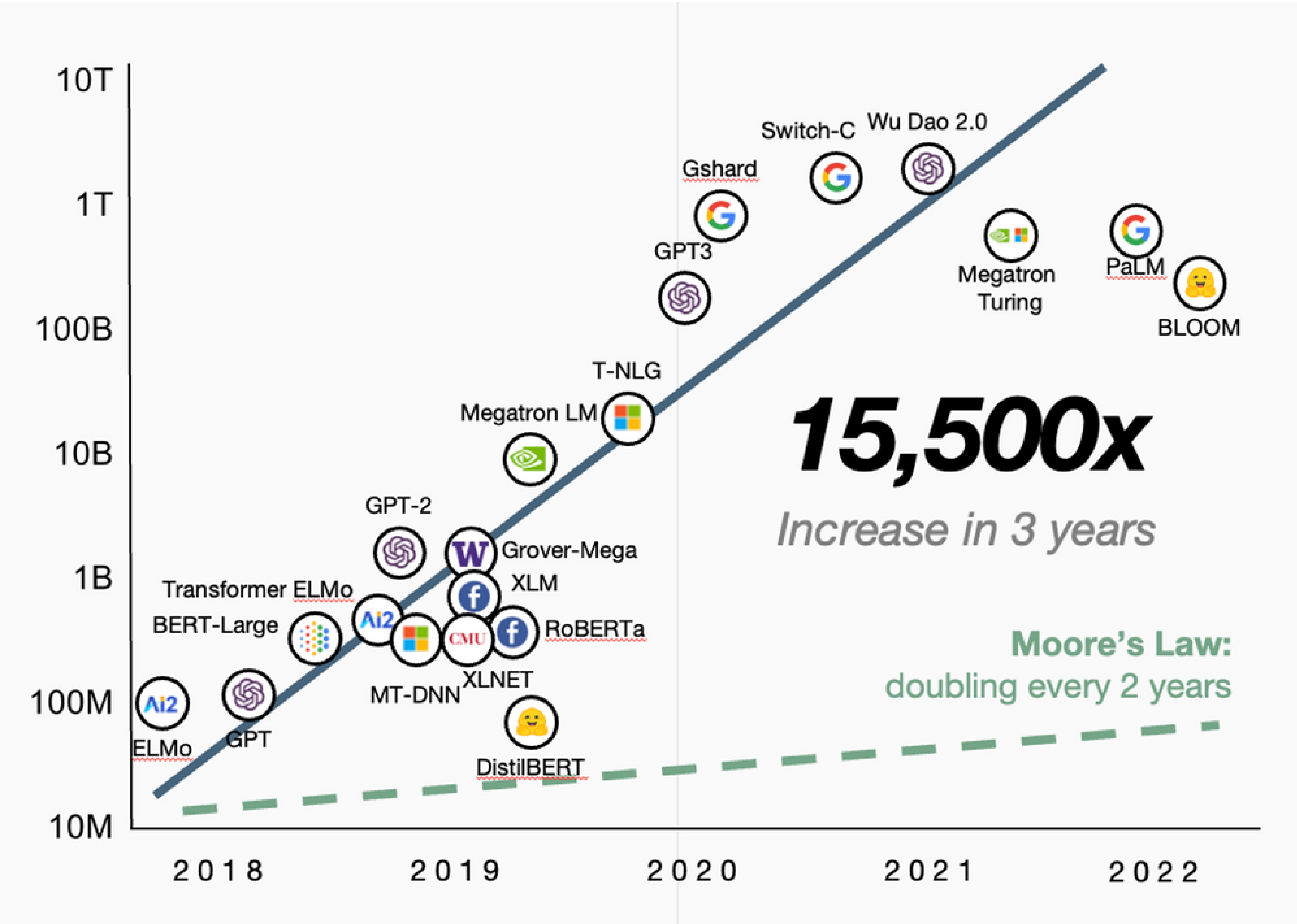
Lightweight Models

- Relatively lower accuracy
- Need more text preprocessing:
  - LowerCasing Text
  - Remove HTML Tags
  - Remove URLs
  - Remove Punctuations
  - Handling ChatWords, like: ASAP, AFAIK, B4, IC, THX, JK, ZZZ, CSL
  - Spelling Correction

- Handling StopWords
- Handling Emojies
- Stemming
- Lemmatization

Large Language Models

- BERT-related model NLU has higher accuracy
- But have a large number of parameters and a huge amount of computation
- Difficult to train

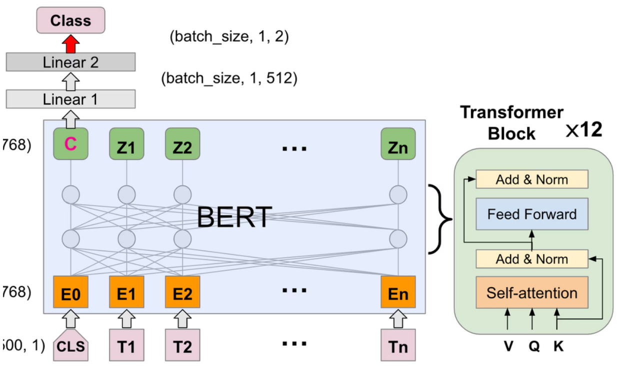


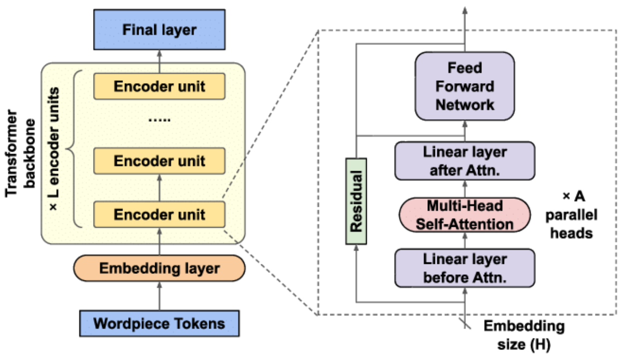
How to finetune LLMs more efficiently?

- Low-Rank Adaption (LoRA)

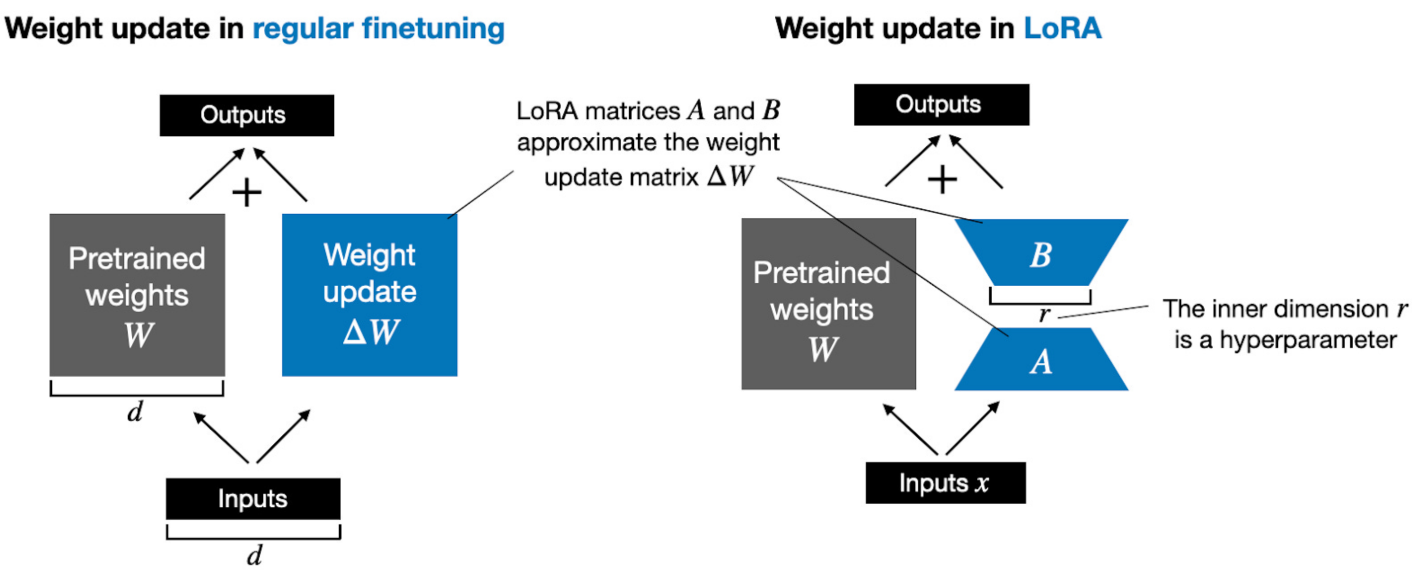
3. Understanding LoRA

- Parameter-Efficient Fine-Tuning (PEFT)
- Instead of adjusting all the parameters of a deep neural network, LoRA focuses on updating only a small set of low-rank matrices.



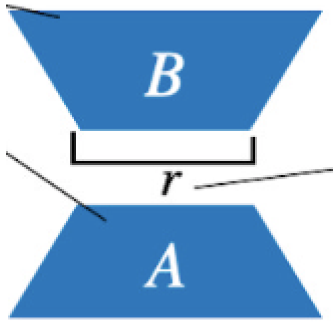



```
BertModel(
  (embeddings): BertEmbeddings(
    (word_embeddings): Embedding(28996, 768, padding_idx=0)
    (position_embeddings): Embedding(512, 768)
    (token_type_embeddings): Embedding(2, 768)
    (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
    (dropout): Dropout(p=0.1, inplace=False)
  )
  (encoder): BertEncoder(
    (layer): ModuleList(
      (0-11): 12 x BertLayer(
        (attention): BertAttention(
          (self): BertSdpaSelfAttention(
            (query): Linear(in_features=768, out_features=768, bias=True)
            (key): Linear(in_features=768, out_features=768, bias=True)
            (value): Linear(in_features=768, out_features=768, bias=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
          (output): BertSelfOutput(
            (dense): Linear(in_features=768, out_features=768, bias=True)
            (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
            (dropout): Dropout(p=0.1, inplace=False)
          )
        )
        (intermediate): BertIntermediate(
          (dense): Linear(in_features=768, out_features=3072, bias=True)
          (intermediate_act_fn): GELUActivation()
        )
        (output): BertOutput(
          (dense): Linear(in_features=3072, out_features=768, bias=True)
          (LayerNorm): LayerNorm((768,), eps=1e-12, elementwise_affine=True)
          (dropout): Dropout(p=0.1, inplace=False)
        )
      )
    )
  )
  (pooler): BertPooler(
    (dense): Linear(in_features=768, out_features=768, bias=True)
    (activation): Tanh()
  )
)
```



- In regular finetuning, we compute the weight updates of a weight matrix ***W*** as ***ΔW***
- In LoRA, we approximate ***ΔW*** through the matrix **multiplication** of two smaller matrices A, B
- ***W*(updated)** = ***W*** + ***ΔW***
- Just like PCA or SVD, consider this as decomposing ***ΔW*** into ***A*** and ***B***

- Suppose the weight matrix ***W***
- It has a size of 5,000x10,000
- 50M parameters in total



- We choose a rank ***r=8***
- Metrix ***A*** size of 5,000x8, Metrix ***B*** size of 8x10,000
- 40,000 + 80,000 = 120,000 parameters in total

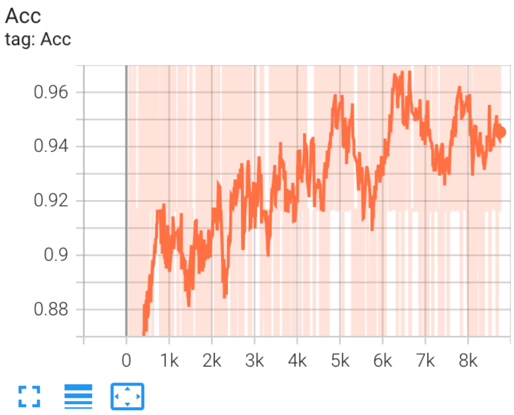
Which is ***400 times smaller*** than the 50M parameters.

4. Coding LoRA from Scratch

- **Best accuracy of LoRA**

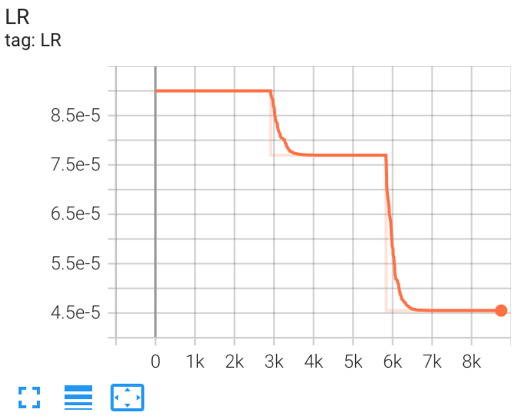
```
--------------------------------------------------
lora_r: 8
lora_alpha: 1
lora_query: True
lora_key: False
lora_value: True
lora_projection: False
lora_mlp: True
lora_head: False

Val acc:   92.72%
Test acc:  93.46%
--------------------------------------------------
```

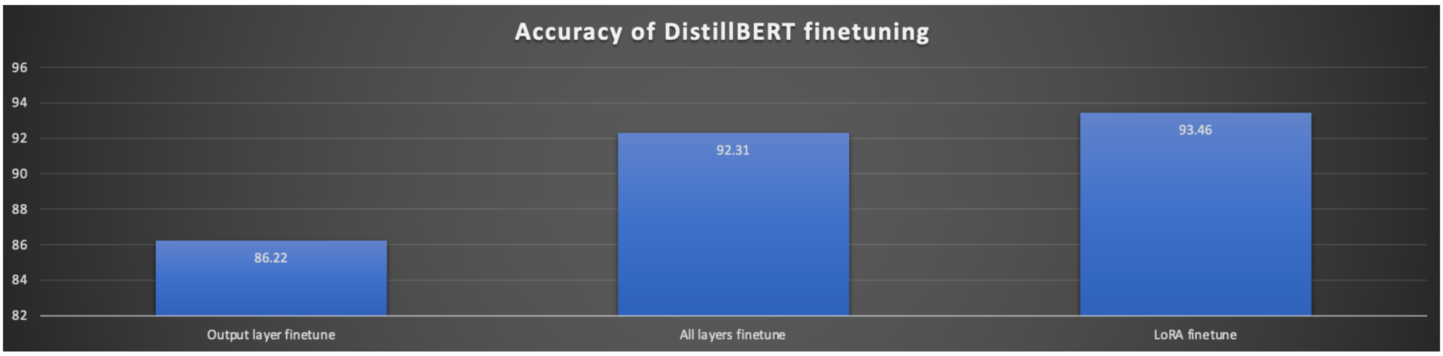



```
**** Total number of parameters:  67471106
**** Number of trainable parameters: 516096 (0.76%)
```

LR

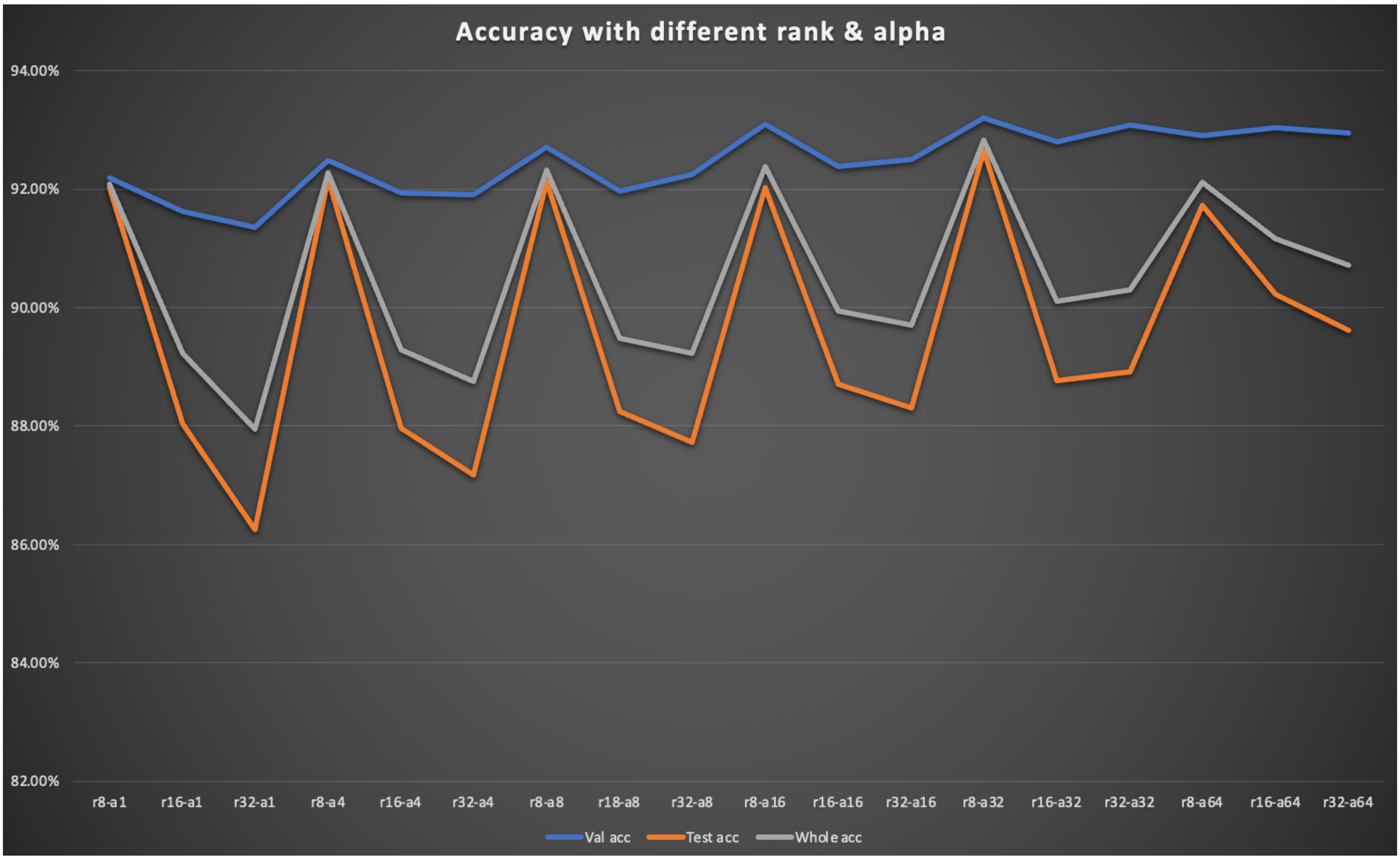


- **Accuracy comparison**

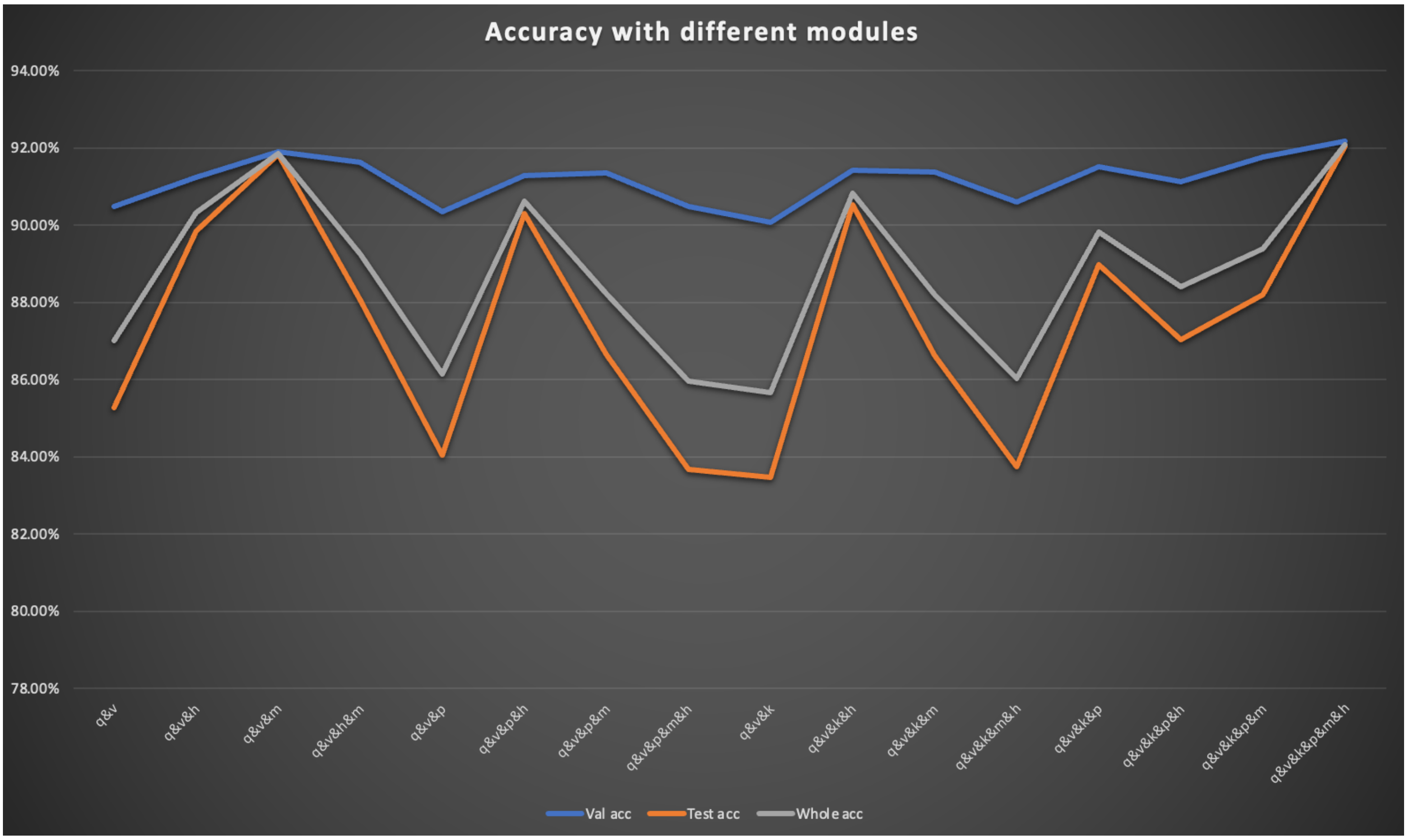


5. Summary

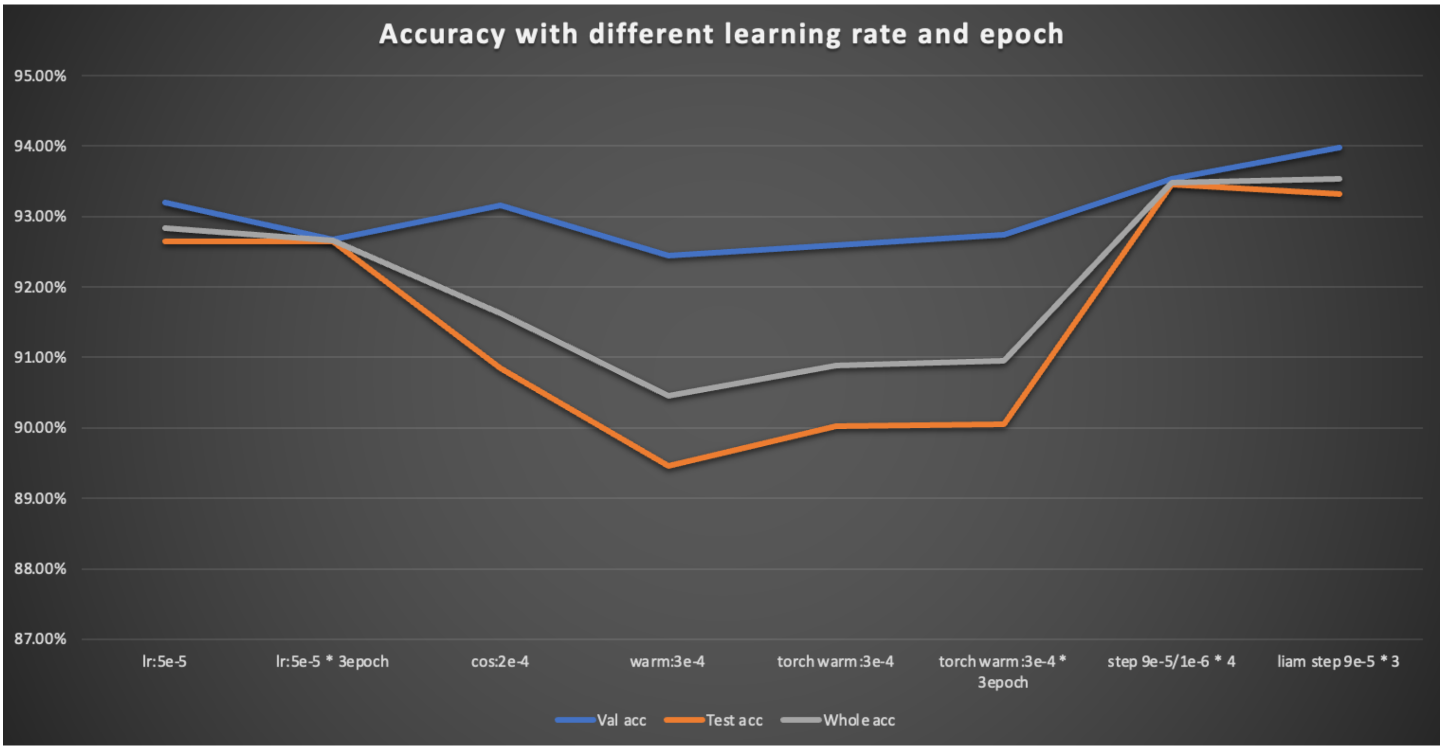
- **LoRA is effective**
- **Choosing rank & alpha**



- **Choosing modules**



- **Choosing learning rate scheduler**



Accuracy with different learning rate and epoch
95.00%
94.00%
93.00%
92.00%
91.00%
90.00%
89.00%
88.00%
87.00%
lr:5e-5
lr:5e-5 * 3epoch
cos:2e-4
warm:3e-4
torch warm:3e-4
torch warm:3e-4 * 3epoch
step 9e-5/1e-6 * 4
liam step 9e-5 * 3
Val acc
Test acc
Whole acc