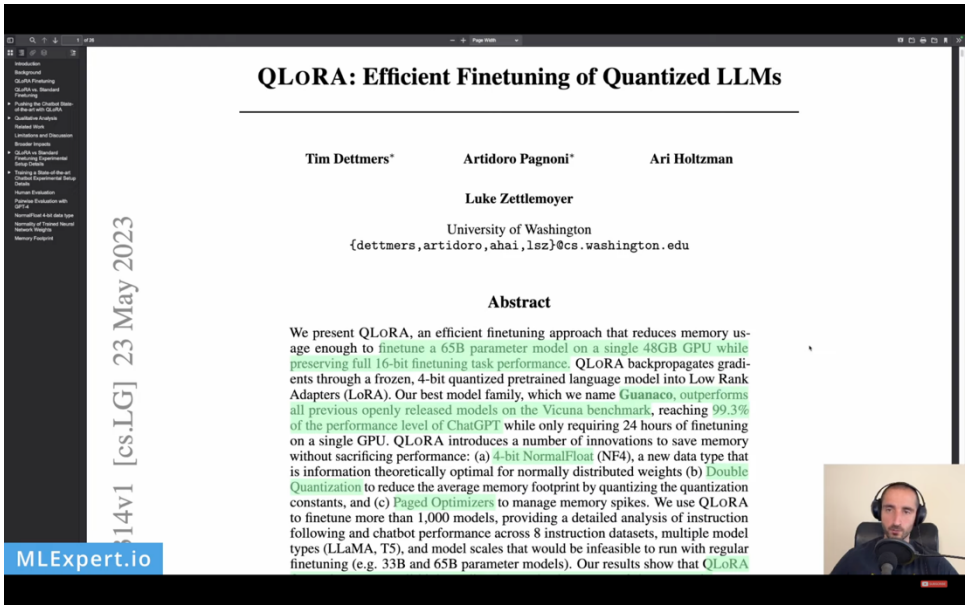
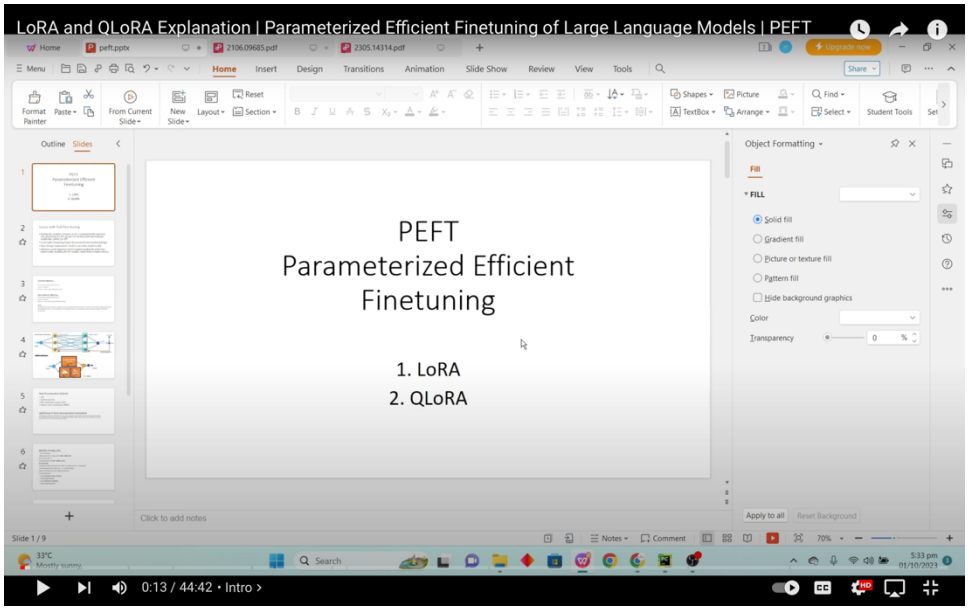
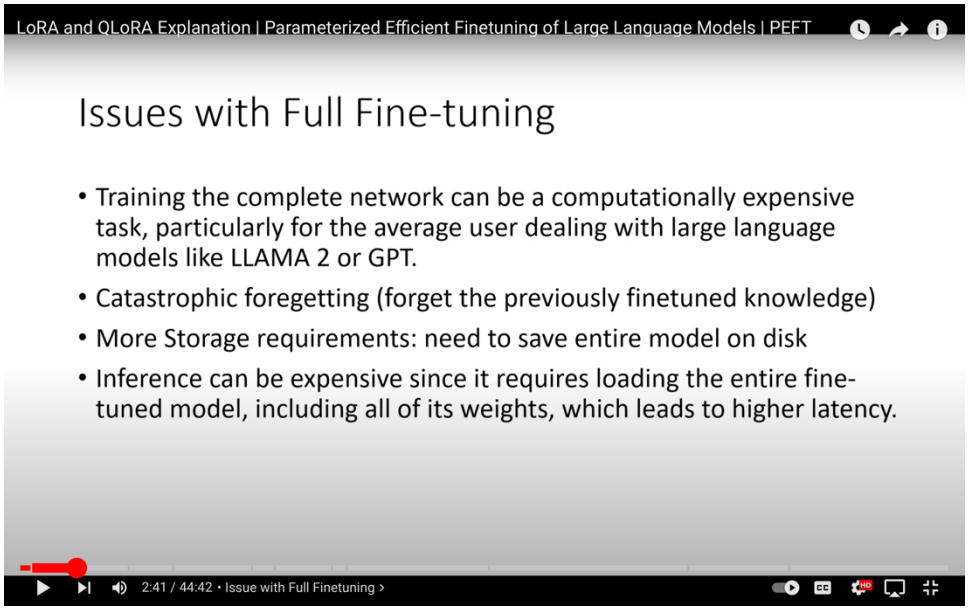
LoRA and QLoRA notes:

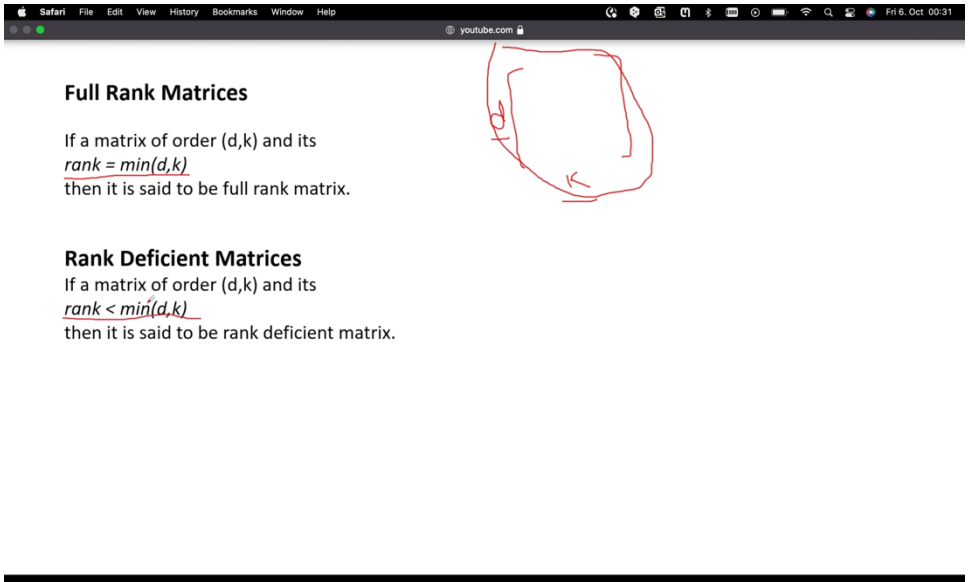


[LoRA and QLoRA Explanation | Parameterized Efficient Finetuning of Large Language Models | PEFT]

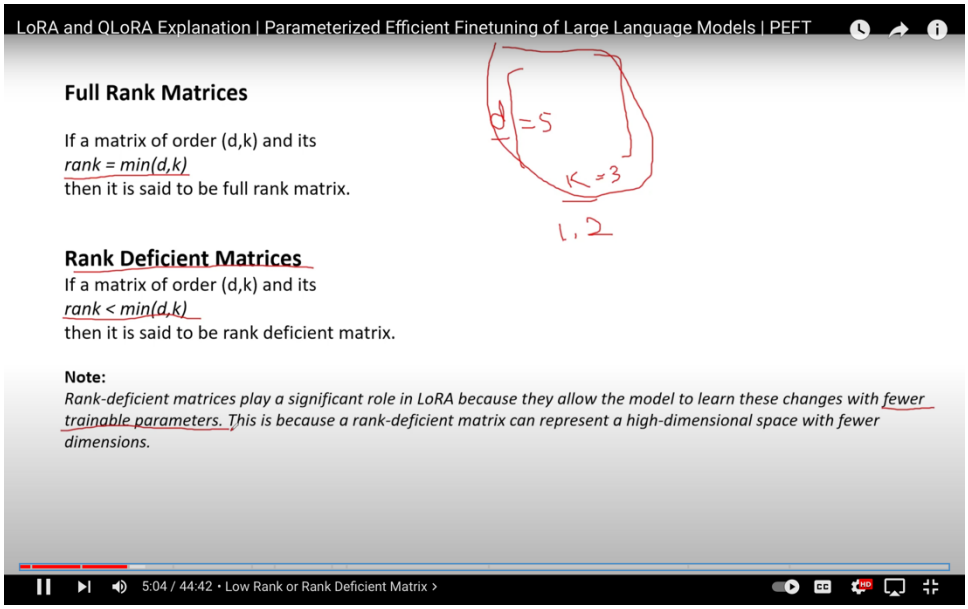

SUBSCRIBE
PARAMETERIZED EFFICIENT FINETUNING OF LLMS
LoRA & QLoRA
Best Explanation
Pretrained Weights

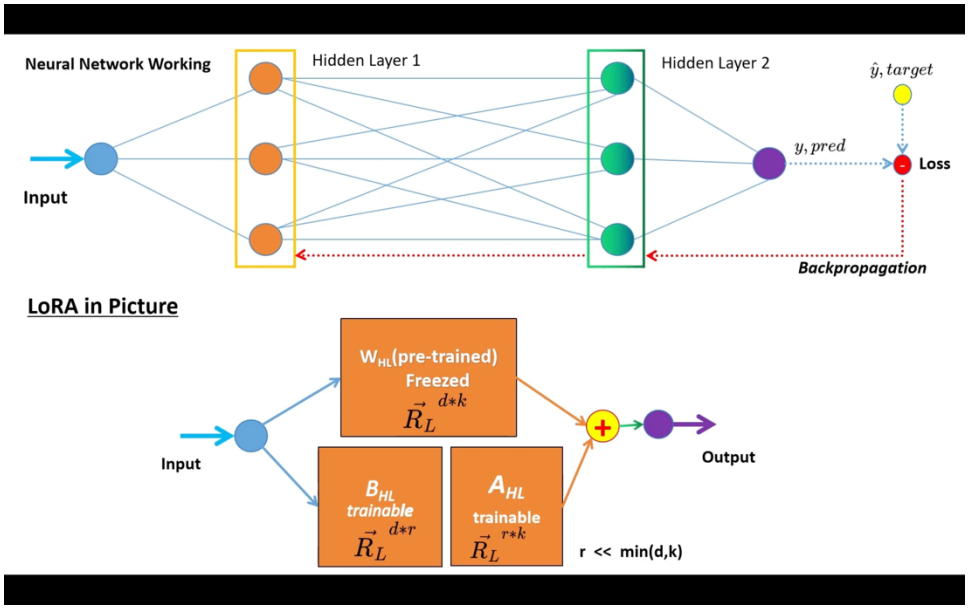

LoRA and QLoRA Explanation | Parameterized Efficient Finetuning of Large Language Models | PEFT
PEFT
Parameterized Efficient Finetuning
1. LoRA
2. QLoRA
0:13 / 44:42 • Intro ›

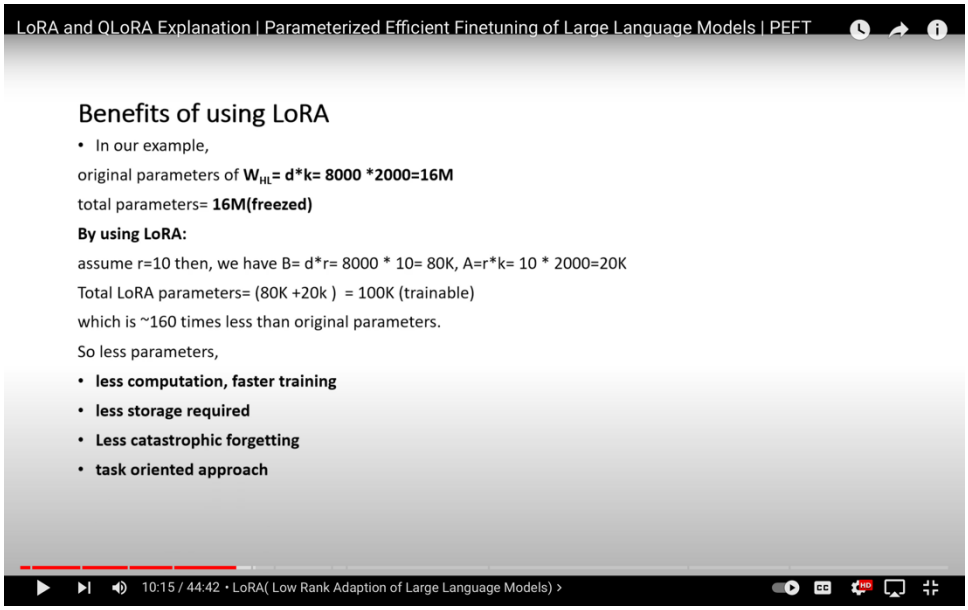

LoRA and QLoRA Explanation | Parameterized Efficient Finetuning of Large Language Models | PEFT
Issues with Full Fine-tuning
• Training the complete network can be a computationally expensive task, particularly for the average user dealing with large language models like LLAMA 2 or GPT.
• Catastrophic foregetting (forget the previously finetuned knowledge)
• More Storage requirements: need to save entire model on disk
• Inference can be expensive since it requires loading the entire fine-tuned model, including all of its weights, which leads to higher latency.
2:41 / 44:42 • Issue with Full Finetuning ›

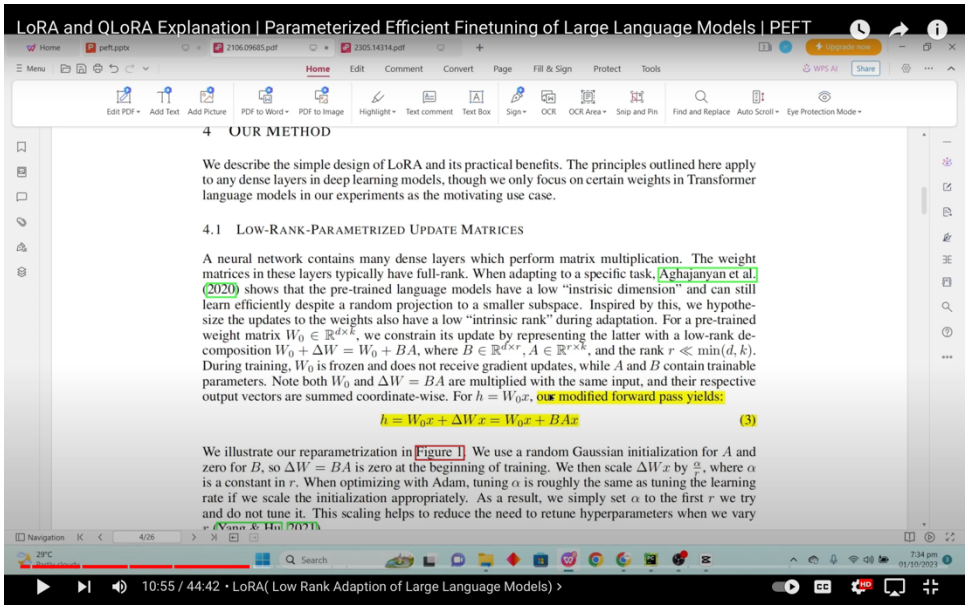


Rank of matrix is total number of independent rows and columns




Neural Network Working
Hidden Layer 1
Hidden Layer 2
$\hat{y}, target$
$y, pred$
Loss
Input
Backpropagation
LoRA in Picture
W_{HL}(pre-trained)
Freezed
$\vec{R}_L^{\,d*k}$
Input
Output
B_{HL}
trainable
$\vec{R}_L^{\,d*r}$
A_{HL}
trainable
$\vec{R}_L^{\,r*k}$
r << min(d,k)


LoRA and QLoRA Explanation | Parameterized Efficient Finetuning of Large Language Models | PEFT
Benefits of using LoRA
• In our example,
original parameters of W_{HL}= d*k= 8000 *2000=16M
total parameters= 16M(freezed)
By using LoRA:
assume r=10 then, we have B= d*r= 8000 * 10= 80K, A=r*k= 10 * 2000=20K
Total LoRA parameters= (80K +20k) = 100K (trainable)
which is ~160 times less than original parameters.
So less parameters,
• less computation, faster training
• less storage required
• Less catastrophic forgetting
• task oriented approach
10:15 / 44:42 • LoRA(Low Rank Adaption of Large Language Models) ›

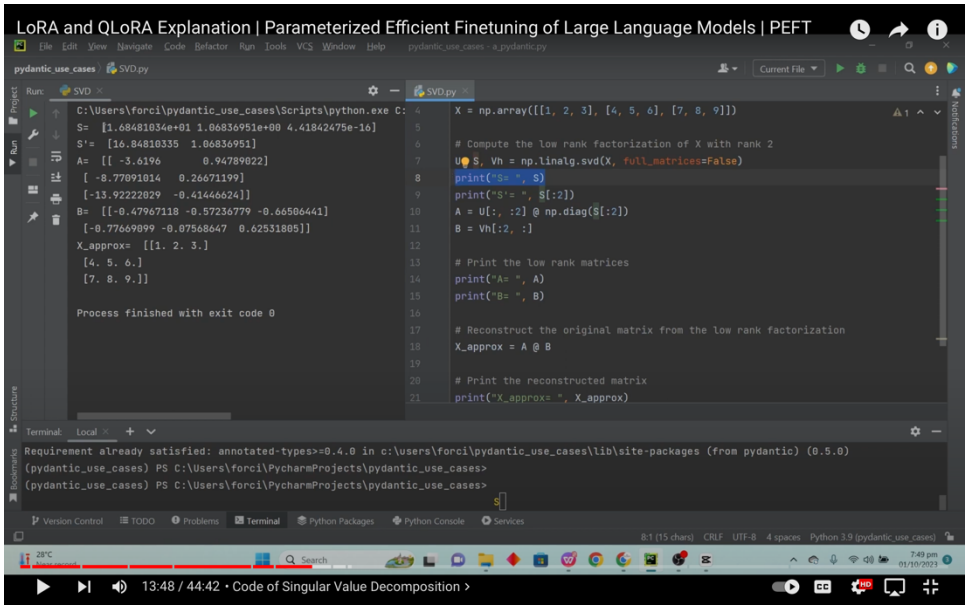


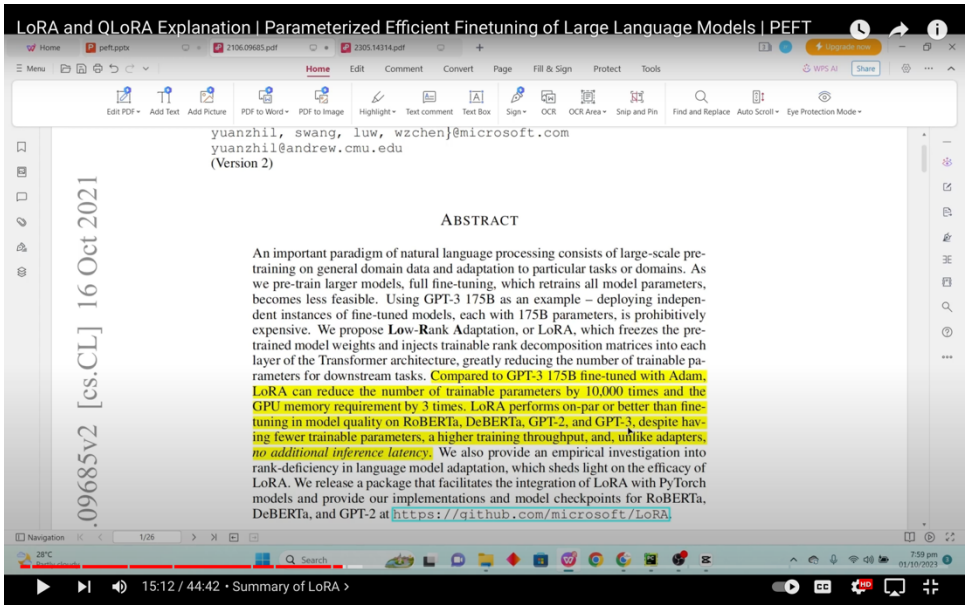
Rank Decomposition Methods

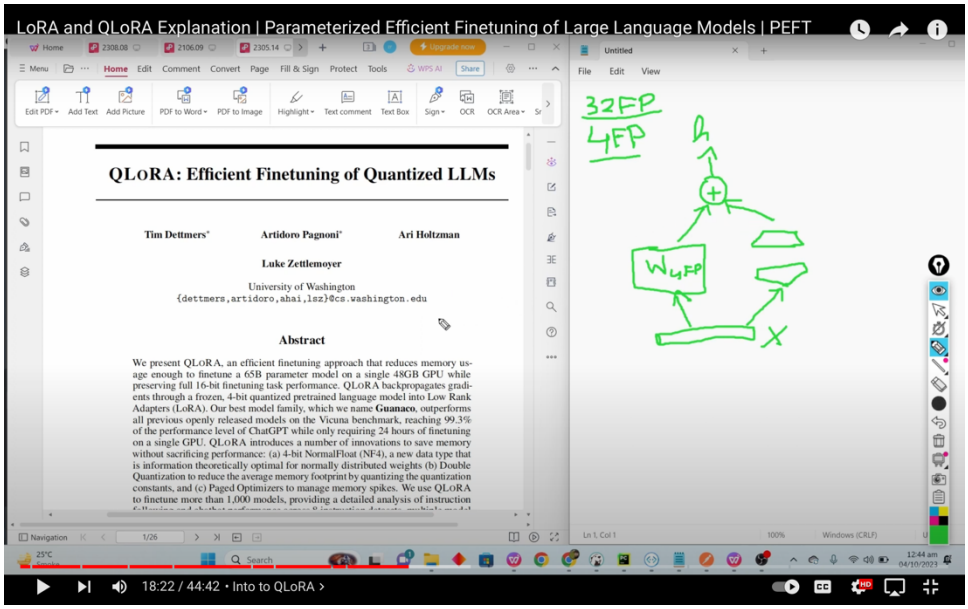
- SVD
- Randomized SVD
- Alternating least squares (ALS)
- Nuclear norm minimization (NNM)

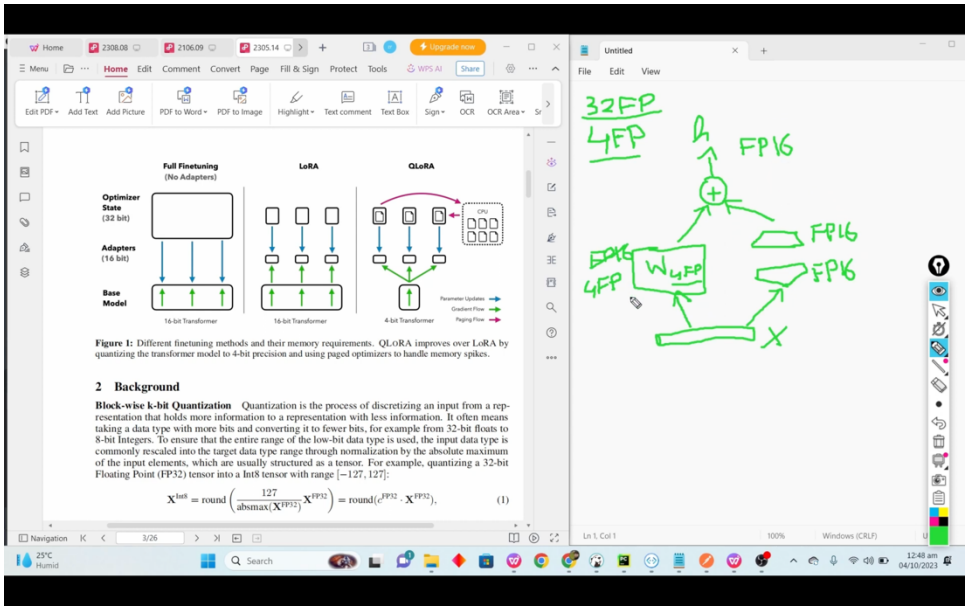
Significance of Rank Decomposition/Factorization

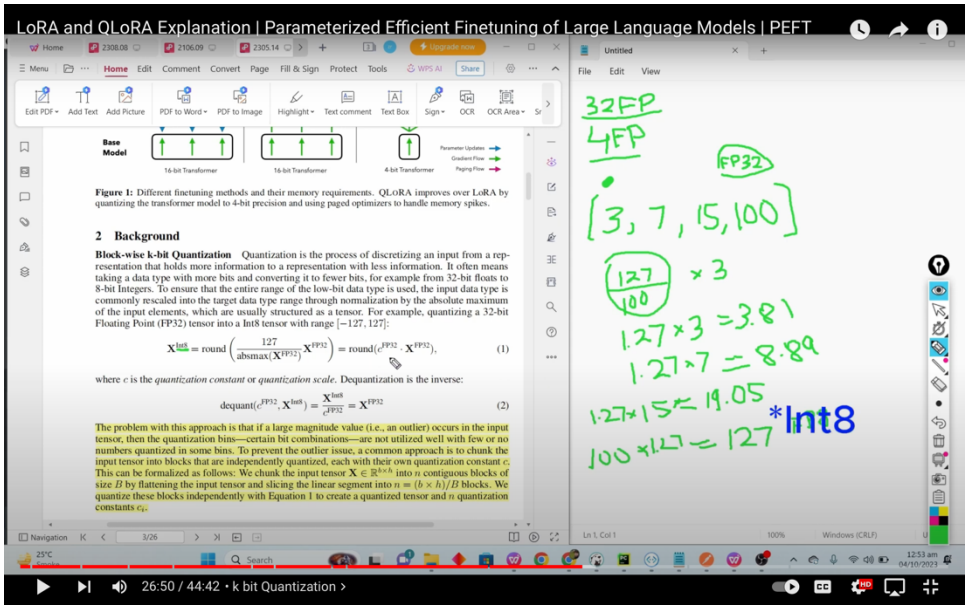
we obtain a resultant matrices B and A that repeats with high accuracy the characteristics of the initial W matrix, but has a smaller dimension, because it does not take into account redundant and unnecessary information.


LoRA and QLoRA Explanation | Parameterized Efficient Finetuning of Large Language Models | PEFT
C:\Users\forci\pydantic_use_cases\Scripts\python.exe C:
S= [1.68481034e+01 1.06836951e+00 4.41842475e-16]
S'= [16.84810335 1.06836951]
A= [[-3.6196 0.94789022]
[-8.77091014 0.26671199]
[-13.92222029 -0.41446624]]
B= [[-0.47967118 -0.57236779 -0.66506441]
[-0.77669099 -0.07568647 0.62531805]]
X_approx= [[1. 2. 3.]
[4. 5. 6.]
[7. 8. 9.]]
Process finished with exit code 0
X = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
# Compute the low rank factorization of X with rank 2
U, S, Vh = np.linalg.svd(X, full_matrices=False)
print("S= ", S)
print("S'= ", S[:2])
A = U[:, :2] @ np.diag(S[:2])
B = Vh[:2, :]
# Print the low rank matrices
print("A= ", A)
print("B= ", B)
# Reconstruct the original matrix from the low rank factorization
X_approx = A @ B
# Print the reconstructed matrix
print("X_approx= ", X_approx)
Requirement already satisfied: annotated-types>=0.4.0 in c:\users\forci\pydantic_use_cases\lib\site-packages (from pydantic) (0.5.0)
(pydantic_use_cases) PS C:\Users\forci\PycharmProjects\pydantic_use_cases>
(pydantic_use_cases) PS C:\Users\forci\PycharmProjects\pydantic_use_cases>
13:48 / 44:42 • Code of Singular Value Decomposition ›

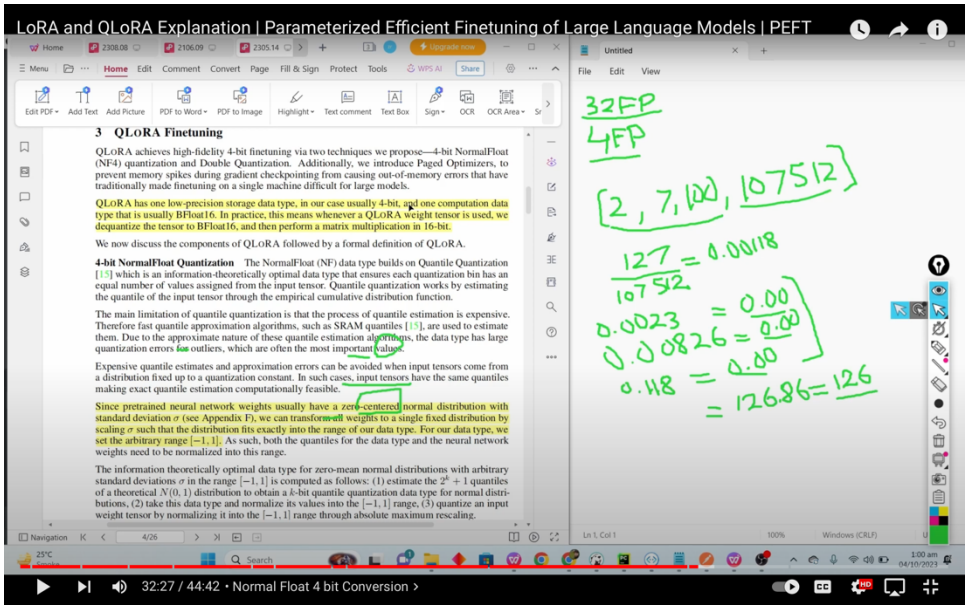

LoRA and QLoRA Explanation | Parameterized Efficient Finetuning of Large Language Models | PEFT
yuanzhil, swang, luw, wzchen}@microsoft.com
yuanzhil@andrew.cmu.edu
(Version 2)
ABSTRACT
An important paradigm of natural language processing consists of large-scale pre-training on general domain data and adaptation to particular tasks or domains. As we pre-train larger models, full fine-tuning, which retrains all model parameters, becomes less feasible. Using GPT-3 175B as an example – deploying independent instances of fine-tuned models, each with 175B parameters, is prohibitively expensive. We propose Low-Rank Adaptation, or LoRA, which freezes the pre-trained model weights and injects trainable rank decomposition matrices into each layer of the Transformer architecture, greatly reducing the number of trainable parameters for downstream tasks. Compared to GPT-3 175B fine-tuned with Adam, LoRA can reduce the number of trainable parameters by 10,000 times and the GPU memory requirement by 3 times. LoRA performs on-par or better than fine-tuning in model quality on RoBERTa, DeBERTa, GPT-2, and GPT-3, despite having fewer trainable parameters, a higher training throughput, and, unlike adapters, no additional inference latency. We also provide an empirical investigation into rank-deficiency in language model adaptation, which sheds light on the efficacy of LoRA. We release a package that facilitates the integration of LoRA with PyTorch models and provide our implementations and model checkpoints for RoBERTa, DeBERTa, and GPT-2 at https://github.com/microsoft/LoRA.
09685v2 [cs.CL] 16 Oct 2021
15:12 / 44:42 • Summary of LoRA ›

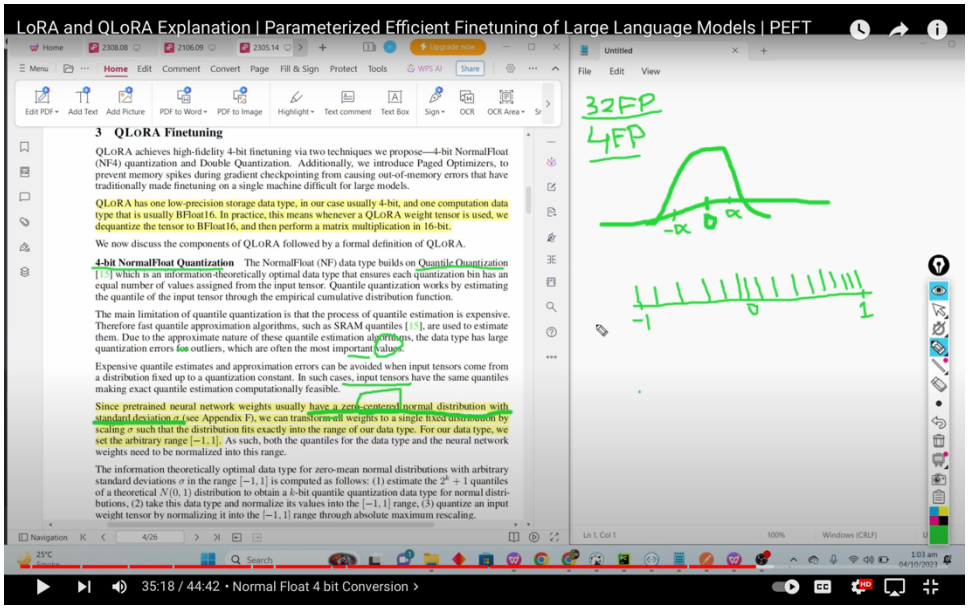

LoRA and QLoRA Explanation | Parameterized Efficient Finetuning of Large Language Models | PEFT
QLoRA: Efficient Finetuning of Quantized LLMs
Tim Dettmers*
Artidoro Pagnoni*
Ari Holtzman
Luke Zettlemoyer
University of Washington
{dettmers,artidoro,ahai,lsz}@cs.washington.edu
Abstract
We present QLoRA, an efficient finetuning approach that reduces memory usage enough to finetune a 65B parameter model on a single 48GB GPU while preserving full 16-bit finetuning task performance. QLoRA backpropagates gradients through a frozen, 4-bit quantized pretrained language model into Low Rank Adapters (LoRA). Our best model family, which we name Guanaco, outperforms all previous openly released models on the Vicuna benchmark, reaching 99.3% of the performance level of ChatGPT while only requiring 24 hours of finetuning on a single GPU. QLoRA introduces a number of innovations to save memory without sacrificing performance: (a) 4-bit NormalFloat (NF4), a new data type that is information theoretically optimal for normally distributed weights (b) Double Quantization to reduce the average memory footprint by quantizing the quantization constants, and (c) Paged Optimizers to manage memory spikes. We use QLoRA to finetune more than 1,000 models, providing a detailed analysis of instruction
32FP
4FP
h
W4FP
X
18:22 / 44:42 • Into to QLoRA ›

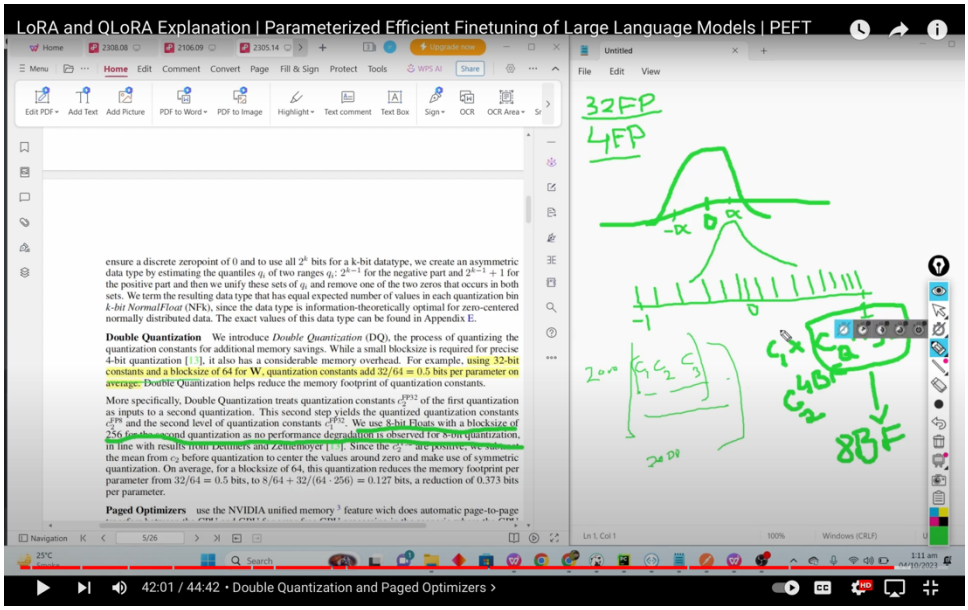

Full Finetuning
(No Adapters)
LoRA
QLoRA
Optimizer State (32 bit)
Adapters (16 bit)
Base Model
CPU
16-bit Transformer
16-bit Transformer
4-bit Transformer
Parameter Updates
Gradient Flow
Paging Flow
Figure 1: Different finetuning methods and their memory requirements. QLoRA improves over LoRA by quantizing the transformer model to 4-bit precision and using paged optimizers to handle memory spikes.
2 Background
Block-wise k-bit Quantization Quantization is the process of discretizing an input from a representation that holds more information to a representation with less information. It often means taking a data type with more bits and converting it to fewer bits, for example from 32-bit floats to 8-bit Integers. To ensure that the entire range of the low-bit data type is used, the input data type is commonly rescaled into the target data type range through normalization by the absolute maximum of the input elements, which are usually structured as a tensor. For example, quantizing a 32-bit Floating Point (FP32) tensor into a Int8 tensor with range [−127, 127]:
$$\mathbf{X}^{\text{Int8}} = \text{round}\left(\frac{127}{\text{absmax}(\mathbf{X}^{\text{FP32}})}\mathbf{X}^{\text{FP32}}\right) = \text{round}(c^{\text{FP32}} \cdot \mathbf{X}^{\text{FP32}}), \quad (1)$$
32FP
4FP
h
FP16
FP16
FP16
FP16
4FP
W4FP
X

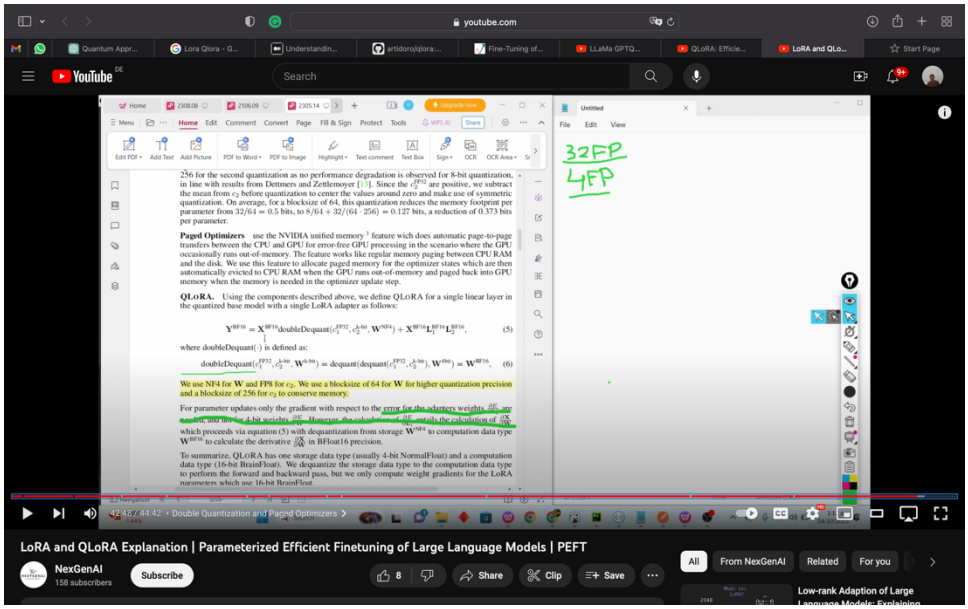


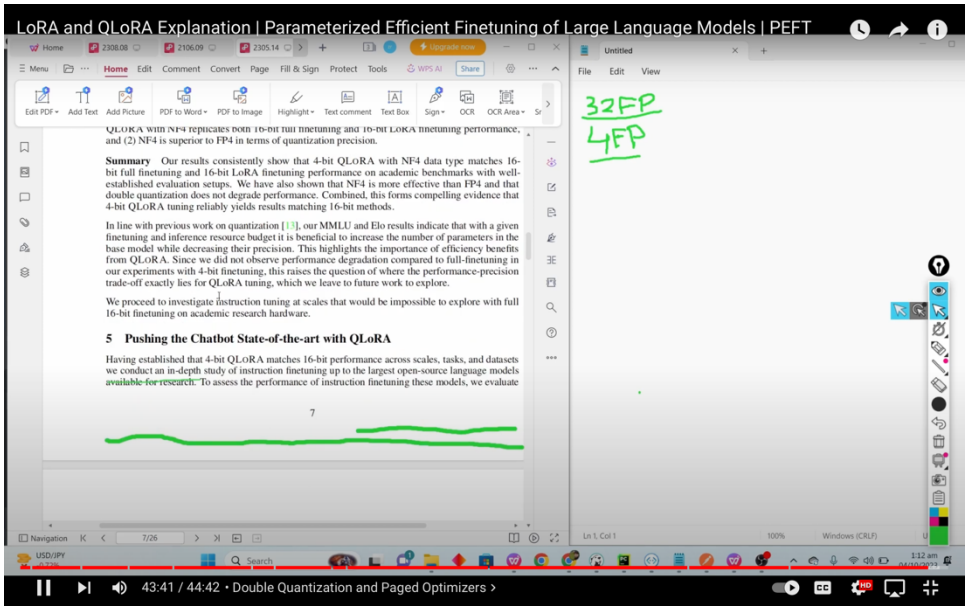
Outlier problem tackled by normal 4 bit precison, and breaking in chunks above.




LoRA and QLoRA Explanation | Parameterized Efficient Finetuning of Large Language Models | PEFT
3 QLoRA Finetuning
4-bit NormalFloat Quantization
32FP
4FP
35:18 / 44:42 • Normal Float 4 bit Conversion ›


LoRA and QLoRA Explanation | Parameterized Efficient Finetuning of Large Language Models | PEFT
Double Quantization
Paged Optimizers
32FP
4FP
8BF
42:01 / 44:42 • Double Quantization and Paged Optimizers ›


youtube.com
YouTube
Search
32FP
4FP
42:48 / 44:42 • Double Quantization and Paged Optimizers
LoRA and QLoRA Explanation | Parameterized Efficient Finetuning of Large Language Models | PEFT
NexGenAI
158 subscribers
Subscribe
8
Share
Clip
Save
All
From NexGenAI
Related
For you
Low-rank Adaption of Large


LoRA and QLoRA Explanation | Parameterized Efficient Finetuning of Large Language Models | PEFT
32FP
4FP
Summary Our results consistently show that 4-bit QLoRA with NF4 data type matches 16-bit full finetuning and 16-bit LoRA finetuning performance on academic benchmarks with well-established evaluation setups. We have also shown that NF4 is more effective than FP4 and that double quantization does not degrade performance. Combined, this forms compelling evidence that 4-bit QLoRA tuning reliably yields results matching 16-bit methods.
In line with previous work on quantization [13], our MMLU and Elo results indicate that with a given finetuning and inference resource budget it is beneficial to increase the number of parameters in the base model while decreasing their precision. This highlights the importance of efficiency benefits from QLoRA. Since we did not observe performance degradation compared to full-finetuning in our experiments with 4-bit finetuning, this raises the question of where the performance-precision trade-off exactly lies for QLoRA tuning, which we leave to future work to explore.
We proceed to investigate instruction tuning at scales that would be impossible to explore with full 16-bit finetuning on academic research hardware.
5 Pushing the Chatbot State-of-the-art with QLoRA
Having established that 4-bit QLoRA matches 16-bit performance across scales, tasks, and datasets we conduct an in-depth study of instruction finetuning up to the largest open-source language models available for research. To assess the performance of instruction finetuning these models, we evaluate
7
43:41 / 44:42 • Double Quantization and Paged Optimizers