Low Rank Approximation on Deep Learning Models

Shiwei Chen

A report submitted as part of the requirements for
the honors program at the
City College of San Francisco,
San Francisco, CA

June 2024

Supervisor Dr. Jamey Bass

Abstract

Low Rank Approximation is a technique which lossy compresses Singular Value Decomposition matrices into lower rank while maintaining the "energy" within the matrix, given by the monotonically decreasing sequence of singular values δ_i [6]. In a process called **Lo**w **R**ank **A**daptation, we freeze the pretrained model weights of deep learning models and inject trainable Low Rank decomposition matrices onto their weights to quickly adapt them to specific tasks [2]. We first adapt a generalized language model (bloom-3) to perform reading comprehension tasks, and subsequently adapt a generic stable diffusion model to produce colorful and pastel-colored backgrounds.

Contents

Chapter 1

Introduction

This chapter will cover the concepts necessary to understand Low Rank Adaptation and its applications to Deep Learning models at a high level. This report assumes the reader is familiar with the concepts taught in an introductory Linear Algebra class. We begin with an explanation and proof of Singular Value Decomposition, and how Low Rank Approximation relates to SVDs. We then build upon this concept by introducing Low Rank Adaptation and applying it to existing models to increase training speed and effectiveness.

1.1 Background

1.1.1 Singular Value Decomposition

SVD combines important concepts of the subject into one significant theorem yielding many applications. It states that any matrix, **regardless of symmetry, dimension, or rank**, can be unconditionally decomposed into three matrices [7].

Definition: Let m and n be arbitrary. Given $A \in \mathbb{C}^{m \times n}$, not necessarily of full rank, a *singular value decomposition* **(SVD)** of A is a factorization:

$$A = U\Sigma V^*$$

where

$$\begin{aligned} &U \in \mathbb{C}^{m\times m} \text{ is unitary,} \\ &V \in \mathbb{C}^{n\times n} \text{ is unitary,} \\ &\Sigma \in \mathbb{R}^{m\times n} \text{ is diagonal.} \end{aligned} \tag{1.1}$$

In addition, it is assumed that the diagonal entries σ_j of Σ are nonnegative and in nonincreasing order; that is, $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_p \geq 0$, where $p = \min(m, n)$ [5].

Existence Theorem for SVDs: Every matrix $A \in \mathbb{C}^{m\times n}$ has a singular value decomposition.

Proof. The following proof follows Gilbert Strang's Linear Algebra lectures at MIT in 2005 [8][9][10]. Consider a matrix A that is $m \times n$ with rank r. This implies the matrix multiplication $A^T A$ is symmetric. In addition, it is positive semi-definite, and we will prove this property: Consider any matrix B such that $B = A^T A$. This implies

$$x^T Bx = x^T A^T Ax = (Ax)^T Ax = (Ax)^2 \geq 0$$

Since $A^T A$ is symmetric with PSD, it is therefore diagonalizable by an orthonormal matrix, with all its eigenvalues $\lambda_i \geq 0$. If we order such eigenvalues such that

$$\sigma_1^2 \geq ... \geq \sigma_r^2 \geq 0,$$

where $r = \text{rank}(A^T A) = \text{rank}(A)$. Let $v_1, ..., v_r$ be an orthonormal set of corresponding eigenvectors for the eigenvalues σ_i. Now, define a new vector u_i such that $u_i = Av_i/\sigma_i$. We want to show that u_i is a unit eigenvector of AA^T

Proof. Note that

$$AA^T u_i = AA^T (Av_i/\sigma_i) = AA^T Av_i \frac{1}{\sigma_1} = A(\sigma_i)^2 v_i \frac{1}{\sigma_i} = (\sigma_i)^2 u_i$$

To show that $u - i$ is a unit eigenvector, we note that

$$u_i^T u_i = (\frac{Av_i}{\sigma_i})^T \frac{Av_i}{\sigma_i} = (\frac{v_i^T A^T}{\sigma_i}) \frac{Av_i}{\sigma_i} = \frac{v_i^T A^T Av_i}{(\sigma_i)^2} = \frac{v_i^T (\sigma_i)^2 v_i}{(\sigma_i)^2} = v_i^T v_i = 1$$

$\square$

Putting everything together, if V_r is the $n \times n$ matrix whose ith column is $v_i, 0 \leq i \leq r$ and Σ_r is the $r \times r$ diagonal matrix whose ith entry is σ_i and U_r is the $m \times r$ matrix whose ith column is $u_i = \frac{1}{\sigma_i} AV_i$, we have

$$U_r = AV_r\sigma_r^{-1} \Rightarrow U_r\Sigma_r = AV_r.$$

If we multiply both sides of the equation by V_r^T, we have by the orthonormality of V_r^T :

$$U_r\Sigma_rV_r^t = AI_r = A.$$

□

An alternate proof for SVD existence, written by Trefethen and Bau is as follows: [5].

Proof. First, we isolate the direction of the largest action of A, and then proceed by induction on the dimension of A. Set $\sigma_1 = ||A||_2$. By a compactness argument, there must be vectors $v_1 \in \mathbb{C}^m$ and $u_1 \in \mathbb{C}^m$ with $||v_1||_2 = ||u_1||_2 = 1$ and $Av_1 = \sigma_1 u_1$. Consider any extensions of v_1 to an orthonormal basis $\{v_j\}$ of $\mathbb{C}^n$ and of u_1 to an orthonomoral basis $\{u_j\}$ of $\mathbb{C}^m$, and let U_1 and V_1 denote the unitary matrices with columns u_j and v_j respectively. Then we have

$$U_1^*AV_1 = S = \begin{bmatrix} \sigma_1 & w^* \\ 0 & B \end{bmatrix},$$

where 0 is a column vector of dimension $m-1, w^*$ is a row vector of dimension $n-1$ and B has dimensions $(m-1)\times(n-1)$. Furthermore,

$$\left|\left|\begin{bmatrix} \sigma_1 & w^* \\ 0 & B \end{bmatrix}\begin{bmatrix} \sigma_1 \\ w \end{bmatrix}\right|\right|_2 \geq \sigma_1^2 + w^*w = (\sigma_1^2 + w^*w)^{1/2}\left|\left|\begin{bmatrix} \sigma_1 \\ w \end{bmatrix}\right|\right|_2$$

This implies $||S||_2 \geq (\sigma_1^2 + w^*w)^{1/2}$. Since U_1 and V_1 are unitary, we know that $||S||_2 = ||A||_2 = \sigma_1$, so this implies $w = 0$. If $n = 1$ or $m = 1$, we are done. Otherwise, the submatrix B describes the action of A on the subspace orthogonal to v_1. By the induction hypothesis, B has an SVD $B = U_2\Sigma_2V_2^*$. Now it is easy to verify that

$$A = U_1\begin{bmatrix} 1 & 0 \\ 0 & U_2 \end{bmatrix}\begin{bmatrix} \sigma_1 & 0 \\ 0 & \Sigma_2 \end{bmatrix}\begin{bmatrix} 1 & 0 \\ 0 & V_2 \end{bmatrix}^* V_1^*$$

is an SVD of A, completing the proof of existence.

For the uniqueness claim, we argue that if the semiaxis lengths of a hyperellipse are distinct, then the semiaxes themselves are determined by the geometry, up to signs. Algebraically, we argue as follows: First, note that σ_1 is uniquely determined by the condition that it is equal to $||A||_2$ as follows from the definition of an SVD. Suppose that in addition to v_1, there is another linearly independent vectorw w with $||w||_2 = 1$ and $||Aw||_2 = \sigma_1$. Define a unit vector v_2 orthogonal to v_1 as a linear combination of v_1 and w,

$$v_2 = \frac{w - (v_1^* w)v_1}{||w - (v_1^* w)v_1||_2}.$$

Since $||A||_2 = \sigma_1, ||Av_2||_2 \leq \sigma_1$ this must be an equality, for otherwise $w = v_1 c + v_2 s$ for some constants c and s with $|c|^2 + |s|^2 = 1$ we would have $||Aw||_2 < \sigma_1$. This vector v_2 is a second right singular vector of A corresponding to the singular value σ_1; it will lead to the appearance of a vector y (equal to the last $n-1$ components of $V_1^* v_2$) with $||y||_2 = 1$ and $||By||_2 = \sigma_1$. We conclude that if the singular vector v_1 is not unique, then the corresponding singular value σ_1 is not simple. □

1.1.2 Low Rank Approximation

Memory constraints must often be considered when storing and working with large matrices. If we want to reduce the amount of space required to store a matrix, one way we can do so is to approximate a given matrix A with a rank-k matrix, for some $k \in \mathbb{N}$. Such a matrix is called a **low-rank approximation.**

Low-Rank SVDs: We want to find a method to remove the least important aspects of a matrix, while keeping the most important ones. Given that any matrix can be factored into an SVD, and that the diagonal terms of Σ, σ_i are monotonically decreasing, SVDs seem to be a good candidate to perform low-rank approximation.

Low-Rank Approximations from the SVD

A natural idea is to keep only the first k terms given by Σ [6]. That is, for an SVD factorization **A** and a target rank k the proposed rank-k approximation is:

$$\hat{A} = \sum_{i=1}^{k} \sigma_i \cdot u_i v_i^T.$$

The K-rank Approximation Process [6]

1: Compute the SVD of A, $A = U\Sigma V^T$
2: Keep only the top k right singular vectors. Set V_k^T equal to the first k rows of V^T.
3: Keep only the top k left singular vectors. Set U_k equal to the first k columns of U.
4: Keep only the top k singular values. Set Σ_k equal to the first k rows and columns of S corresponding to the k largest singular values of **A.**

5: The rank-k approximation is then

$$A_k = U_k \Sigma_k V_k^T.$$

Choosing k

How many σ's do we keep? What rank should our low-rank approximation be? Generally, for any mattrix, it is a good idea to keep 80-90 percent of the "energy", with energy in this case being the summation $\sum_{i=1}^{n} \sigma_i^2$ [11]. Formulated mathematically,

$$\sum_{i=1}^{k} \sigma_i^2 \geq \beta \sum_{i=1}^{n} \sigma_i^2, \beta \in [0.8, 0.9]$$

1.1.3 Low Rank Adaptation

Many applications in deep learning rely on adapting one large-scale, pre-trained model to **multiple** downstream applications. Adaptation is typically one via fine-tuning, which updates all the parameters of the pre-trained model. The downside of fine-tuning is that the new model contains as many parameters as in the original model, and as models become increasingly large, it becomes increasingly unfeasible to train multiple large models to perform domain-specific tasks [2].

Low Rank Adaptation (LoRA) is a procedure which freezes the pre-trained model weights of a Machine Learning model and injects trainable **low-rank** approximations of SVDs into it. Following up from previous research, the authors of LoRA, Hu et al. (2021) propose that the change in weights during model adaptation have **low intrinsic rank** [2]. That is, the first k terms in Σ hold far more importance than those in the middle or at the end, allowing for significant reductions in matrix rank without drops in performance. These low-rank matrices are then trained to perform specific tasks, and in-turn, can adapt generalist models to specialized tasks. As we do not alter the base weights, different low-rank matrices can be quickly switched in or out, allowing us to use one large pre-trained model for multiple domain tasks.

LoRA Process [2]

Given a pre-trained weight matrix, $W_0 \in \mathbb{R}^{d \times k}$, we constrain its update by representing its adapted parameters ΔW as

$$W_0 + \Delta W = W_0 + BA$$

where $B \in \mathbb{R}^{d \times r}, A \in \mathbb{R}^{r \times k}$ and where the rank r is far below $\min(d, k)$

In principle, we can apply LoRA to any weight matrix in a neural network to reduce the number of trainable parameters. In the transformer architecture, there are four weight matrices in the self-attention module (W_q, W_k, W_v, W_o) and two in the MLP module, which we can perform Low Rank Adaptation on.

Benefits

The most significant benefit comes from a reduction in storage and memory usage. In the original Low Rank Adaptation paper by Hu et al. (2021) we see that for a large Transformer trained using the Adam optimizer, VRAM usage can be reduced by up to 2/3 if r is significantly smaller than the dimensions of the weight matrices due to not needing to store the optimizer states for the frozen parameters. Furthermore, VRAM consumption decreased by a factor of 10,000 (from 350GB to 35MB) when adapting GPT-3 with $r = 4$. Finally, a speedup on individual GPUs of 25 percent was noted due to not needing to compute the gradient for the vast majority of parameters. (32.5 tokens/V100 GPU vs 43.1 tokens/V100 GPU).

These benefits allow us to train with significantly fewer GPUs and avoid I/O bottlenecks to adapt a pre-trained model. Such a decrease in hardware requirements also means that, for the first time, members of the general public are able to easily fine-tune large-pretrained models for specific tasks. With as little as 5 images, amateurs can create LoRAs for pre-train Stable Diffusion models to draw in a specific style, or setting.

1.2 About this Report

This is a project of *Shiwei Chen*, submitted to the Mathematics Department at the City College of San Francisco as a part of the Mathematics Honors program.

1.3 Chapter List

Chapter 2 Design: Software dependencies, and general structure of a basic LoRA on a generic Language Model.

Chapter 3 Implementation: Implementation of design chapter, featuring training results and significance.

Chapter 4 Stable Diffusion LoRA: Low Rank Adaptation on Stable Diffusion models, adapting a pre-trained model to draw colorful scenery and images.

Chapter 5 Conclusion

Chapter 2

Design

To demonstrate Low-Rank Adaptation, we set our goal to adapt a language model to succeed in reading comprehension tasks by answering questions about an article that we provide the model.

In adapting the our pre-trained language model to answer prompts, we will use Loralib, a software library developed by Microsoft to implement the process described in Edward et al. (2021). We will also use the PEFT (Parameter Efficient Fine Tuning) and Transformers library from Huggingface. Lastly, we will be implementing our LoRA with PyTorch.

2.1 Model

We will be using the Bloom Language Model as our untrained base model. Bloom's architecture is modified from GPT2, and uses 3 billion parameters. Due to the Bloom LM's open source status and its training data being comprised of multiple languages, it will be suitable for us to adapt for this project.

2.2 Datasets

We will be using the Stanford Question Answering Dataset (SQuAD) reading comprehension dataset, which consists of questions posed by contributors on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage. Find more about the squad_v2 dataset by clicking here.

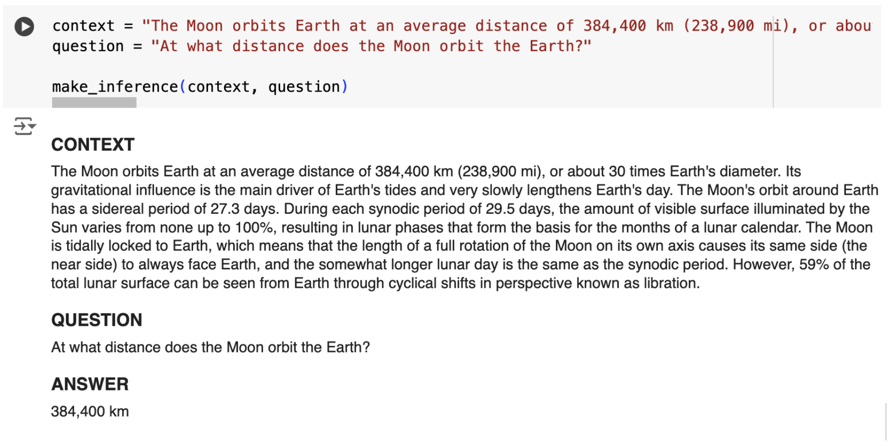
Chapter 3

Implementation

Implementation of the LoRA model was developed on Google Colab, and you may find the source code by clicking here.

The Low Rank Adaptation for bloom-3b was implemented with rank $r = 8$, leading to a trainable parameter count of 2.4 million, as opposed to the 3 billion frozen pre-trained weights parameters. This significant reduction in parameters reflects on the training time of the Low Rank Adaptation, which parsed through the 100 entries on the squad dataset in 102 seconds using the Google Colab free-tier GPU, a Nvidia Tesla K80.

Training loss generally observed slight a shift downwards, averaging 2.53627 by the end of training, but beginning at around 3.0.



```
context = "The Moon orbits Earth at an average distance of 384,400 km (238,900 mi), or abou
question = "At what distance does the Moon orbit the Earth?"

make_inference(context, question)
```

CONTEXT

The Moon orbits Earth at an average distance of 384,400 km (238,900 mi), or about 30 times Earth's diameter. Its gravitational influence is the main driver of Earth's tides and very slowly lengthens Earth's day. The Moon's orbit around Earth has a sidereal period of 27.3 days. During each synodic period of 29.5 days, the amount of visible surface illuminated by the Sun varies from none up to 100%, resulting in lunar phases that form the basis for the months of a lunar calendar. The Moon is tidally locked to Earth, which means that the length of a full rotation of the Moon on its own axis causes its same side (the near side) to always face Earth, and the somewhat longer lunar day is the same as the synodic period. However, 59% of the total lunar surface can be seen from Earth through cyclical shifts in perspective known as libration.

QUESTION

At what distance does the Moon orbit the Earth?

ANSWER

384,400 km

Figure 3.1: Output of adapted Bloom-3 model, given a context and question as inputs

Chapter 4

Stable Diffusion LoRA

4.1 Introduction

Stable Diffusion is a generative deep learning model used for creating high-quality images through iterative refinement of noise. It leverages a diffusion process where noise is gradually added to an image and then reversed using neural networks to generate coherent images from random noise. Stable Diffusion has significant applications in art generation, image inpainting, and super-resolution, allowing for the creation of detailed and visually appealing images from minimal text input.

Low-Rank Adaptation (LoRA) can be applied to a pre-trained Stable Diffusion model to adapt it to specific art styles, characters, or settings. LoRA's ease of use and low barriers to entry have contributed to the flourishing of generating art with Stable Diffusion as a hobby.

Figure 4.1: 'An oil painting of a latent space.' drawn by a Latent Diffusion Model [3]

4.2 Architecture

4.2.1 U-net

The U-net is a convolutional neural network originally developed for biomedical image segmentation. The model consists of a contracting path and then an expansive path, giving it it's u-shaped architecture [13]. The contracting path consists of repeated **convolutions**, each followed by a ReLU activation function and a max pooling operation. As an image goes through the U-net, it is continuously compressed, losing spatial information while feature information is increased.

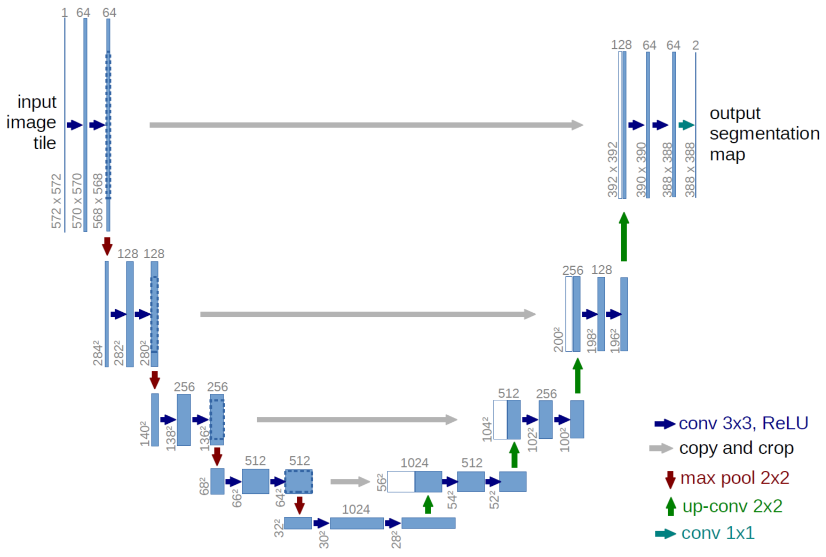


Figure 4.2: Diagram of U-net with Resnet as a backbone [13]

4.2.2 Variational Auto-Encoder

A **Latent Space** is a lower-dimensional representation where each point corresponds to a potential data point.

A **variational autoencoder** is a generative model used for learning latent representations of data in an unsupervised manner. The encoder takes an input data point (such as an image) and maps it to a probability distribution in a latent space. Instead of directly outputting a fixed latent representation, the encoder outputs the parameters (mean and variance) of a probability distribution that represents the latent space.

Stable Diffusion consists of 3 parts: A **variational autoencoder (VAE)** a **U-net**, and an optional text encoder [3]. The VAE encoder compresses the image from pixel space to a smaller dimensional latent space, capturing a more fundamental meaning of the image. **Gaussian Noise** is then continuously applied to the compressed latent representation. Then, the result is passed into the U-Net to be denoised.

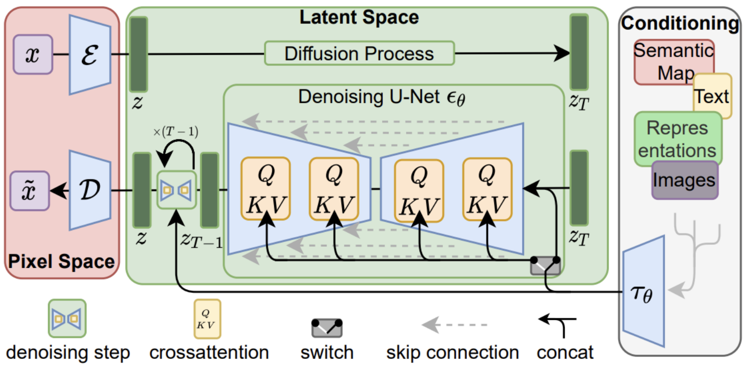


Figure 4.3: Diagram of latent diffusion architecture used by Stable Diffusion [3]

Importantly, the denoising step can be flexibly conditioned on a string of text, and image, or another modality. The encoded conditioning data is exposed to denoising U-Nets via a **cross-attention** mechanism, where it affects the denoising of an image, leading to image generation.

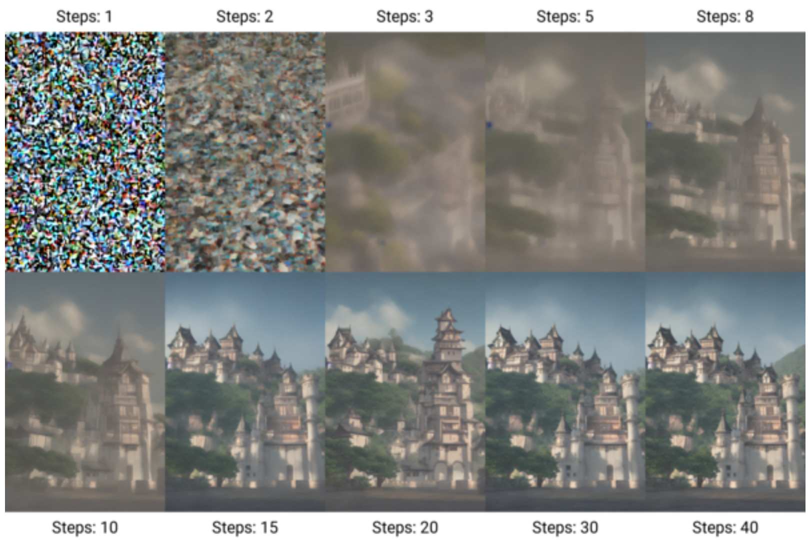


Figure 4.4: Denoising process used by Stable Diffusion [12]

4.2.3 LoRA on Stable Diffusion

Low Rank Adaptation can replace any trainable parameters. Most commonly, LoRA is used on the weights, biases, and cross-attention mechanism of the model. Simo Ryu's

LoRA Stable Diffusion library implements these features in a popular repository with over 6k stars.

4.3 Implementation

The dataset consists of 171 handpicked images selected based on their color, saturation, and scenery. Corresponding tags for dataset were generated by the Khoya trainer model and manually pruned. Interested readers may click here to access this training dataset.

4.4 Results and Evaluation

The LoRA was given one hour of training on a u-net learning rate of $5e-4$ and text-encoder learning rate of $1e-4$ with ten repeats on ten epochs. All pictures were produced at 1080 x 720 resolution. We present a selection of works produced by the "colorful scenery LoRA" below.

Chapter 5

Conclusion

Since it's original application to Large Language Models in 2021, Low-Rank Adaptation has found use in many other subdomains of Deep Learning. Research building upon the original LoRA paper continues. QLoRA, a new approach presented by Dettmers et al (2023), introduces a number of innovations to save memory without sacrificing performance [1]. Another algorithm, Low-Rank Hadamard Product (LoHa) approximates a large weight matrix with more low-rank matrices and combines them with the **Hadamard Product** to create even more parameter-efficient models.

As the field progresses, it is expected that further refinements and novel applications of low-rank adaptation will continue to emerge, driving forward the capabilities of modern models and expanding their applicability across various domains. The ongoing exploration and adaptation of these techniques will likely remain a fertile ground for future research, offering promising avenues for both theoretical advancements and practical implementations.

5.1 References

[1] Dettmers, Tim et al. "QLoRA: Efficient Finetuning of Quantized LLMs." ArXiv abs/2305.14314 (2023): n. pag.

[2] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen: "LoRA: Low-Rank Adaptation of Large Language Models", 2021; http://arxiv.org/abs/2106.09685arXiv:2106.09685.

[3] Rombach, Robin et al. "High-Resolution Image Synthesis with Latent Diffusion Models." 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2021): 10674-10685.

[4] Simo Ryu. Low-rank adaptation for fast text-to-image diffusion fine-tuning.https://github.com/cloneofsimo/lora

[5] Trefethen, Lloyd N., and David Bau III. Numerical Linear Algebra. Society for Industrial and Applied Mathematics, 1997.

[6] Valiant, Gregory, and Tim Roughgarden. "The Singular Value Decomposition (SVD) and Low-Rank Approximations", 30 Apr. 2024, web.stanford.edu/class/cs168/l/l9.pdf.

[7] Visual Kernel, "SVD Visualized, Singular Value Decomposition Explained — SEE Matrix , Chapter 3." Youtube, 23 Apr. 2022, www.youtube.com/watch?v=vSczTbgc8Rc&t=606s. Accessed 20 May 2024.

[8] Strang, Gilbert. "Lec 29 — MIT 18.06 Linear Algebra, Spring 2005." YouTube, YouTube, 7 May 2009, www.youtube.com/watch?v=Nx0lRBaXoz4.

[9] Gundersen, Gregory. "Proof of the Singular Value Decomposition." Proof of the Singular Value Decomposition, 20 Dec. 2018, gregorygundersen.com/blog/2018/12/20/svd-proof/.

[10] Matthew Leingang (https://math.stackexchange.com/users/2785/matthew-leingang), Strang's proof of SVD and intuition behind matrices U and V, URL (version: 2019-07-12): https://math.stackexchange.com/q/2276493

[11] Leskovec, et al. "Lecture 49 - SVD Gives the Best Low Rank Approximation (Advanced) — Stanford." YouTube, YouTube, 13 Apr. 2016, www.youtube.com/watch?v=c7e-D2tmRE0.

[12] "File:X-Y plot of algorithmically-generated AI art of European-style castle in Japan demonstrating DDIM diffusion steps.png." Wikimedia Commons. 7 Aug 2023, 15:54

UTC. 22 May 2024, 17:13 https://commons.wikimedia.org/w/index.php?title=File:X-Y_plot_of_algorithmically-generated_AI_art_of_European-style_castle_in_Japan_demonstrating_DDIM_diffusion_steps.png&oldid=790927243.

[13] Ronneberger, Olaf et al. "U-Net: Convolutional Networks for Biomedical Image Segmentation." ArXiv abs/1505.04597 (2015): n. pag.