LoRA - Group Project for Deep Learning for Natural Language Processing - 2024WS

Lucas Gugler
e12019849@student.tuwien.ac.at

Yikai Yang
e1427424@student.tuwien.ac.at

Alexander Gube
e12023988@student.tuwien.ac.at

1 LoRA

LoRA (Low-Rank Adaptation) is a technique designed to efficiently fine-tune large language models (LLMs) by introducing a small, trainable, low-rank matrix into their architecture. Instead of updating all the model's parameters, which is computationally expensive and memory-intensive, LoRA modifies a subset of the parameters using these low-rank matrices. This significantly reduces the number of trainable parameters and lowers the computational cost, making it more feasible to fine-tune large models on specific tasks or domains without requiring massive computational resources.

The key idea behind LoRA is to freeze the pre-trained model weights and insert trainable rank-decomposition matrices into certain layers of the model, typically at places like the attention mechanisms. These matrices capture task-specific knowledge during fine-tuning, while the original model's general knowledge remains intact. Once trained, LoRA modules can be combined with the base model at inference time, effectively enabling task-specific performance without the need for full model retraining or duplication. This approach is particularly useful for scenarios requiring fine-tuning on multiple tasks, as it allows for lightweight, modular adaptations.Hu et al. [2021]

2 Project Plan

The aim of the project is to reproduce selected parts of the results presented in the original LoRA paper Hu et al. [2021]. We focused on reproducing the RoBERTa Base and GPT-2 experiments in particular. Furthermore, we devised some extensions to the experiments that dealt with quantization, image classification, and rank/task relationships.

3 RoBERTa Base

3.1 Experiment Setup

In the original paper multiple sizes of the RoBERTa model: `base` (125M parameters) and `large` (355M parameters) were selected to test the LoRA implementation. Due to time constraints and with each run of the training setup taking multiple hours, we narrowed down our test runs to the base model.

3.2 Training Setup

For the training setup, we initially aimed to replicate the exact setup which were used in the original LoRA paper which can be found as follows:

Method	Dataset	MNLI	SST-2	MRPC	CoLA	QNLI	QQP	RTE	STS-B
	Optimizer	AdamW							
	Warmup Ratio	0.06							
	LR Schedule	Linear							
RoBERTa base LoRA	Batch Size	16	16	16	32	32	16	32	16
	# Epochs	30	60	30	80	25	25	80	40
	Learning Rate	5E-04	5E-04	4E-04	4E-04	4E-04	5E-04	5E-04	4E-04
	LoRA Config.	$r_q = r_v = 8$							
	LoRA α	8							
	Max Seq. Len.	512							

Figure 1: Hyperparameters chosen for the RoBERTa base model in the LoRA paper

However, we discovered that each run would take approximately 8-10 hours to finish, hence we decided to scale down the hyperparameters such that each run would finish in approx. 2 hours.

3.3 Datasets

As in the original paper, the RoBERTa model is evaluated by using eight different datasets for fine-tuning, which we will list in the following with a short description:

3.3.1 MNLI

The MNLI (Multi Genre Language Inference) dataset contains 433k entries of sentence pairs (premise and hypothesis sentences) of various genres which describe potential entailments. Beside the test-set data, each pair of those sentences are labeled with -1 (contradition = no entailment), 0 (neutral), or 1 (entailment). `https://cims.nyu.edu/~sbowman/multinli/`

3.3.2 SST-2

The SST2 (Stanford Sentiment Treebank) dataset is a sentiment analysis dataset containing 215k phrases derived from movie reviews. `https://paperswithcode.com/dataset/sst-2` Each phrase was annotated by a human with either 0 (negative) or 1 (positive), while neutral phrases were discarded from the dataset.

3.3.3 MRPC

The MRPC (Microsoft Research Paraphrase Corpus) contains 5.8k pairs of sentences derived from news articles. `https://paperswithcode.com/dataset/mrpc` The sentence pair is labeled with either 1, if both sentences are paraphrases, or a 0 otherwise.

3.3.4 CoLA

The CoLA dataset (Corpus of Linguistic Acceptability) is a dataset with 10.6k sentences. Each sentence is labeled with either 1, if the sentence is grammatically correct or a 0 otherwise. `https://nyu-mll.github.io/CoLA/`

3.3.5 QNLI

The QNLI (Question answering Natural Language Inference) dataset contains 20k question, answer pairs which are labeled with either 0, if the question is anwered correctly, or 1 if not. `https://paperswithcode.com/dataset/qnli`

Since the training dataset (5k entries) is not labeled (-1), we decided to dismiss this subset of the dataset.

3.3.6 QQP

The QQP (Quora question pairs) dataset comprises of over 400k pairs of two different questions from the platform "Quora". The label 1 indicates whether two questions are paraphrases of other, while 0 equals non related questions. `https://paperswithcode.com/dataset/quora-question-pairs`

3.3.7 RTE

The RTE (Recognizing Textual Entailment) is a sentence pair dataset derived mainly from excerpts from Wikipedia or news sites. If there is an entailment between the two sentences of the sentence pair, the label is 0, and 1 if there is no entailment. `https://paperswithcode.com/dataset/rte`

3.3.8 STS-B

The STS-B (Semantic Textual Similarity-Benchmark) dataset is composed of sentence pairs which were collected from various data sources, such as news headlines, forums, captions of image or video data, and other natural language datasets. Each sentence pair was rated by humans with a score of between 1 (low similarity) to 5 (high similarity). `https://paperswithcode.com/dataset/sts-benchmark`

3.4 Data Processing

Since some of the datasets contain multiple subsets of training, validation, and test sets, it was necessary to consolidate those by merging those together. In addition, some datasets (MNLI, SST2, QNLI, QQP, RTE) contain unlabeled data in their test sets. Since, it is not possible to validate on unlabeled data using the metrics, as used in the original paper, we decided to create a new dataset split (training/evaluation/test) by merging all data entries which are labeled and re-splitting the newly created dataset in a ratio of 80% Training data, 10% Evaluation data, 10% Test data.

3.5 Implementation

We are utilizing high level interfaces from the HuggingFace library to train our models, specifically we make use of the `transformers` library for loading the models, `datasets` library to load the various datasets. Training was done using the `Trainer` component. For data processing we explored the data structure manually, and wrote our own preprocessing methods.

3.5.1 Model Configuration

We replicated the hyper-parameter setups for each dataset run according to the same hyperparameters in the LoRa paper. In cases where a single run would have taken a high amount of time (>5 hrs), we decided to reduce the number of epochs. This was the case for the MNLI dataset, where we reduced the number of epochs from 30 to 8, which yielded a total training runtime of 5hrs instead of over 20hrs on an A40 server instance.

Furthermore, we only conducted a single run for each dataset, while in the original paper, the end result consists of the average of five different runs. In addition, we follow the procedure for training the MRPC, RTE, and STS-B datasets, where we set the initial LoRA weights to the best MNLI checkpoint before starting training.

3.5.2 Evaluation metrics

We use the same evaluation metrics used by the study authors of the LoRa paper. In case of the MNLI, SST-2, MRPC, QNLI, QQP, and RTE datasets, we measure the `accuracy` on the test set. Since the STS-B dataset contains continuous numeric values it is evaluated by using the Pearson's correlation. For the CoLA dataset, the Matthew's correlation is used. All metrics are loaded using the `evaluate` package from the HuggingFace platform.

3.5.3 Results:

In the following table, we compare our results with the reference implementation:

Table 1: Comparison of the reproduction and reference results. Superscripted results with [1] mean that LoRA weights were initialized to the best MNLI checkpoint before training, while results with [2] designate results with random initialization. **Bolded** results denote results which significantly differ from reference results.

Dataset	Reference Implementation	Our Implementation	Hyperparameter adjustments
MNLI	87.5	84.47	✓# Epochs: reduced from 30 to 8
SST-2	95.1	94.69	-
MRPC	89.7	**73.45**[1] / 86.55[2]	-
CoLA	63.4	60.25	-
QNLI	93.3	**62.47**	-
QQP	90.8	89.68	✓# Epochs: reduced from 25 to 12
RTE	86.6	**54.87**[1] / 74.73[2]	-
STS-B	91.5	**70.82**[1] / 90.82[2]	-
Avg.	87.2	73.84[1] / 80.46[2]	

3.5.4 Discussion

We were able to replicate the results of the LoRA paper to a limited degree with 4 out of 8 datasets, precisely in the cases of `MNLI`, `SST-2`, `CoLa`, and `QQP` datasets. We observe that in cases where we had initialized the LoRA weights to the best MNLI-checkpoint, namely MRPC, RTE, and STS-S our scores were consistently significantly worse with an average of -22.89% and maximum deficit of -31.73% in the case of RTE.

This may seem strange since our result of MNLI (84.47%) is inline with the reference implementation (87.5%). However, it should be noted that we reduced the number of epochs when training MNLI, which could have had an impact on the robustness of the model. To find out whether the initialization of the LoRA weights to the MNLI checkpoint is in fact the main contributor to the low performance, we decided to conduct an additional run without initializing the weights and setting the weights to it's default initialization implementation.

We discovered that by not initializing the LoRA weights to our best MNLI checkpoint,and using the default method of using a gaussian random field initialization, the scores improved significantly and were more comparable with the results of the study. Our MRPC score (86.55%) improved to a similar level compared to the score presented by the study (89.7%). We were able to observe the same improvements with the RTE dataset, where a score of (74.73%) was achieved compared to the study result of (86.6%). We therefore conclude that the initialization to the MNLI checkpoint, as it was done originally in the LoRA paper, was not beneficial in our case. With regard to the reasons to why the scores were diminished, it either could be that by loading the MNLI LoRA weights the model was missing robustness since it was initially tailored to the different MNLI dataset. Another explanation could be that the model was converging against a local optimum during the whole training cycle due to the initialization to the MNLI weights. Indications for that could be seen in the training loss function which remained consistently stable through the entire training cycle.

In addition to those results, we were not able to reproduce the score with the QNLI dataset (Ours: 62.47%, Reference: 93.3%). It remains an exception since we are not able to explain why the score deviates so much from the original result, since we can exclude issues with data pre-processing, and initialization errors since there were none necessary following the proceedings of the paper. We conducted an additional run with a different seed which brought no improvement with a score of 62.44%.

4 GPT-2

In this section, we present our reproduction of the LoRA experiments using GPT-2 on natural language generation tasks. Specifically, we focus on the E2E NLG Challenge dataset, implementing the low-rank adaptation approach as described in the original LoRA paper Hu et al. [2021].

4.1 Experimental Setup

Our implementation utilizes the GPT-2-medium model as the base architecture, which contains 345M parameters. We employ the Hugging Face Transformers library for model implementation and the PEFT (Parameter-Efficient Fine-tuning) library for LoRA integration. The experiments are conducted using PyTorch as the underlying framework.

4.1.1 Dataset

The primary dataset used in our experiments is the E2E NLG Challenge dataset Novikova et al. [2017], which is designed for training natural language generation systems in the restaurant domain. The dataset consists of meaning representations (MR) paired with corresponding natural language references. As such, it is commonly used for data-to-text evaluation. The dataset consists of ca. 42,000 training and 4, 600 test and validation examples each. Each MR contains various restaurant attributes (such as name, food type, price range, etc.) and multiple possible natural language realizations. For example, an MR might look like:

`name[The Eagle], food[English], price[moderate], customer rating[3/5]`

with a corresponding reference text:

`The Eagle is a moderately priced English restaurant with a 3/5 customer rating.`

We also prepared infrastructure for the WebNLG and DART datasets, though our primary experiments focused on E2E NLG.

4.2 Implementation Details

4.2.1 Model Configuration

The LoRA configuration was implemented with the following hyperparameters:

- Rank (r): 4
- LoRA alpha: 32
- Target modules: Query and key attention matrices ("c_attn")
- LoRA dropout: 0.1
- Weight initialization: Gaussian

Here's the key implementation of the LoRA configuration:

```
lora_config = LoraConfig(
r=4,
lora_alpha=32,
target_modules=["c_attn"],
lora_dropout=params['dropout_prob'],
init_lora_weights="gaussian",
bias="none"
)
model = get_peft_model(model, lora_config)
```

4.2.2 Data Processing

We implemented specialized preprocessing functions for the E2E dataset. The data processing pipeline includes:

Grouping test data by meaning representations to handle multiple references Formatting input sequences by concatenating meaning representations and references with a separator Tokenization with special token handling

A critical aspect of our implementation was the proper handling of multiple references in the E2E dataset. Unlike simpler text generation tasks, the E2E dataset contains multiple valid reference texts for each meaning representation. This characteristic required special handling during evaluation.

One of the more challenging aspects of our implementation was setting up proper batching with the Hugging Face DataCollator. The complexity arose from the need to handle both the input sequences and their corresponding references while maintaining the grouped structure of our data. The DataCollator properly pads the sequences within each batch, but we needed to handle the reference texts separately to avoid inappropriate padding of the reference lists. By processing the instance with a DataCollator we avoid unnecessary padding of texts, which would waste memory.

4.3 Training Process

The training process was executed over 5 epochs with the following specifications:

- Batch size: 8
- Learning rate: 2e-4
- Weight decay: 0.01
- Warmup steps: 500
- Label smoothing: 0.1

We implemented a linear learning rate scheduler with warmup and used the AdamW optimizer. The training loop includes gradient accumulation and proper handling of pad tokens:

```
optimizer = torch.optim.AdamW(model.parameters(),
lr=2e-4,
weight_decay=params['weight_decay'])
scheduler = get_linear_schedule_with_warmup(
optimizer,
num_warmup_steps=500,
num_training_steps=num_training_steps
)
```

4.4 Evaluation

We implemented a comprehensive evaluation suite using multiple metrics to assess the quality of generated text:

- BLEU: Measures n-gram precision
- METEOR: Accounts for stemming and synonyms
- ROUGE-L: Captures longest common subsequences
- NIST: Modified BLEU that weights n-gram informativeness
- CIDEr: Consensus-based evaluation

Our generation strategy for evaluation employs beam search with the following parameters:

- Beam width: 10
- Length penalty: 0.9
- Repetition penalty: 1.0
- No-repeat n-gram size: 4
- Maximum new tokens: 64

In our reproduction study, we faced an interesting challenge regarding metric implementation. The original LoRA paper utilized evaluation scripts that relied on Perl and Java dependencies, which were not suitable for our Jupyter notebook environment. Instead, we implemented Python-based versions of these metrics using the Hugging Face `evaluate` library and `nlgmetricverse`

```
bleu = evaluate.load('bleu')
rouge = evaluate.load('rouge')
nist = evaluate.load('nist_mt')
cider = NLGMetricverse(metrics=load_metric("cider"))
meteor = NLGMetricverse(metrics=load_metric("meteor"))
```

Table 2: Comparison of evaluation metrics between reference and our implementation

Metric	Reference Implementation	Our Implementation
BLEU	0.6603	0.6619
METEOR	0.8139	0.8183
ROUGE-L	0.6750	0.6530
NIST	7.2166	7.1026
CIDEr	2.1189	2.1522

To validate our metric implementations, we conducted a comparative analysis by running both our Python-based metrics and the original evaluation scripts from the LoRA paper on the same outputs. The results showed comparable values, with only minor differences attributable to implementation details. We also used the original LoRA code available at Github and retrained the gpt-2 model to generate outputs which we also ran through the two evaluation suites. They yielded similar metrics as our implementation. These validation steps were crucial to ensure the reliability of our reproduction study.

Table 2 shows the comparison of evaluation metrics between both implementations run on our evaluation suite.

4.5 Output Processing

A notable aspect of our implementation is the careful handling of generated outputs. We implemented a format_outputs function that ensures consistent text normalization. This processing ensures fair comparison across different references and generated outputs by standardizing spacing and case.

4.6 Implementation Challenges and Solutions

During the implementation, we encountered and addressed several technical challenges:

Token Handling: Special attention was required for proper padding token configuration:

```
tokenizer.padding_side = 'left'
tokenizer.add_special_tokens({'pad_token': '[PAD]'})
model.config.pad_token_id = tokenizer.pad_token_id
```

Memory Management: Efficient batch processing and gradient accumulation were implemented to handle memory constraints.
Parameters: We tried reproducing the same hyperparameters used for training and evaluation. It was not sufficient to reuse the parameters mentioned in the paper since these produced subpar results. We had to explicitly check the reference implementation on GitHub which used quite a few parameters that were not mentioned in the paper, especially in the evaluation part. This cost us quite a bit of time.

4.7 Rank

This section refers to the reproduction of Table 18 in the original LoRA paper Hu et al. [2021]. Thereby, the above evaluation setup is utilized with varying values for the rank parameter. The following section provides a detailed explanation to the rank in the context of LoRA, which is then followed by our metrics found. As such, this section answers the second research question of the referenced paper, which asks for the optimal rank.

4.7.1 Rank in LoRA

When pre-trained language models are adapted to a specific task, their "intrinsic dimension" is low Aghajanyan et al. [2020]. Based on this, the LoRA paper hypothesizes that weight updates to specialize on downstream tasks, i.e. ΔW, also come with a low "intrinsic" rank. This inspires LoRA training, where the weight matrices of the transformer modules are frozen and the actual adaption is dependent on two low-rank matrices A and B, which model the weight update of low "intrinsic

Table 3: Rank-dependent evaluation metrics (reference implementation)

Rank r	**BLEU**	**NIST**	**METEOR**	**ROUGE**	**CIDEr**
1	0.687	8.7215	0.4565	0.7052	2.4329
2	0.691	8.7413	0.4590	0.7052	2.4639
4	0.738	8.8439	0.4689	0.7186	2.5349
8	0.695	8.7457	0.4636	0.7196	2.5196
16	0.696	8.7483	0.4629	0.7177	2.4985
32	0.693	8.7736	0.4642	0.7105	2.5255
64	0.692	8.7174	0.4651	0.7180	2.5070
128	0.687	8.6718	0.4628	0.7127	2.5030
256	0.689	8.6982	0.4629	0.7128	2.5012
512	0.687	8.6857	0.4637	0.7128	2.5025
1024	0.693	8.7495	0.4659	0.7149	2.5090

Table 4: Rank-dependent evaluation (our implementation)

Rank r	**BLEU**	**NIST**	**METEOR**	**ROUGE**	**CIDEr**	**% train Params**
1	0.644	6.900	0.8016	0.6547	2.0525	0.0277
2	0.653	7.086	0.8011	0.6763	2.2700	0.0554
4	0.633	6.879	0.7970	0.6706	2.2281	0.1107
8	0.642	7.054	0.8020	0.6735	2.2715	0.2211
16	0.662	7.214	0.8153	0.6840	2.3526	0.4413
32	0.668	7.315	0.8172	0.6853	2.3717	0.8788
64	0.649	7.049	0.8010	0.6748	2.2799	1.7422
128	0.663	7.171	0.8169	0.6857	2.3591	3.4248
256	0.657	7.140	0.8105	0.6837	2.3159	6.6228
512	0.660	7.186	0.8133	0.6769	2.3547	12.4228
1024	0.670	7.269	0.8190	0.6855	2.3223	22.1001

dimension". Specifically, for a matrix $W_0 \in \mathbb{R}^{d \times k}$ we have $B \in \mathbb{R}^{d \times r}$ and $A \in \mathbb{R}^{r \times k}$. This results in the following modification of each low-rank adapted layer W_0:

$$W_0 + \Delta W_x = W_0 x + BA$$

4.7.2 Evaluation & Discussion

This section presents our reproduction results of Table 18 in the LoRA paper. Thereby, we use the exact same evaluation setup described above except for the rank which is adapted accordingly. Table 3 shows the rank evaluation results reported by the reference implementation, and table 4 shows our results. Table 5 illustrates the percentual difference captured. Except for the meteor score, we can reproduce the reported scores within a margin of approx. 10%.

The evaluation results help answer the question asking for the perfect rank. However, as stated in Aghajanyan et al. [2020], each task presumably results in a different low "intrinsic dimension", which depends on the task and/or domain of application. Therefore, we can conclude that there is no global best rank, instead, it is task-specific. However, as can be seen from the results, a minimal rank of 1 already results in improvements.

4.7.3 Relationship between Rank and Model Size

As stated in the LoRA paper, the relationship between model size and the optimal rank for adaption is yet unknown. As a result, we investigate the effect of varying ranks on models of different sizes. Specifically, we report the same metrics as above for GPT-2 (Base), GPT-2 Medium (see above) and GPT-2 Large. Table 6 illustrates the model sizes for each evaluated GPT2 model size.

Table 7 illustrates the results for GPT-2 Small and table 8 documents the metrics for GPT-2 Large. Due to resource constraints, we evaluate GPT-2 Large only for the following ranks: $[8, 128, 256, 1024]$. In

Table 5: Percentual difference between reference and our implementation

Rank r	**BLEU (%)**	**NIST (%)**	**METEOR (%)**	**ROUGE (%)**	**CIDEr (%)**
1	6.26	20.90	-75.62	7.16	15.63
2	5.50	18.94	-74.54	4.10	7.87
4	14.23	22.22	-70.36	6.68	12.10
8	7.63	19.35	-73.00	6.41	9.84
16	4.89	17.55	-76.14	4.70	5.84
32	3.61	16.62	-76.06	3.55	6.09
64	6.21	19.14	-72.24	6.02	9.06
128	3.49	17.31	-76.54	3.79	5.75
256	4.64	17.92	-75.10	4.08	7.41
512	3.93	17.27	-75.40	5.05	5.91
1024	3.32	16.92	-75.76	4.11	7.44

Table 6: GPT2 model sizes Hugging-Face

Model	# Parameters
GPT-2 Base	117M
GPT-2 Medium	345M
GPT-2 Large	774M

figure 2 we see BLEU over varying rank values for all models evaluated. Interestingly, the smallest model achieves the best performance, based on BLEU, for a rank of 1. However, with increasing rank values it cannot improve in comparison to the larger GPT-2 variants. In terms of GPT-2 medium, we can see a significant increase in BLEU for $r = 16$, so we can conclude that this is the inherent rank of this task. Surprisingly, the medium model outperforms GPT-2 Large. However, we assume this is because the authors have optimized the hyperparameters for GPT-2 Medium.

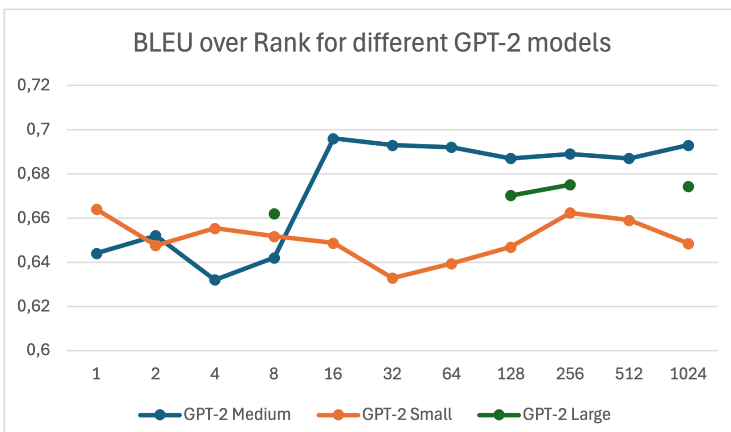


Figure 2: BLEU over Rank for different GPT-2 model sizes

4.7.4 Relationship between Rank and Task Type

The LoRA paper considers rank as a model-specific parameter. However, they argue that the best rank might depend on the task's difficulty. Specifically, if the target task differs significantly from the pre-training task, a higher rank might be required to model the task's objectives.

As a result, in this section, we investigate the model's (GPT-2 Medium) performance on a task that is rather suitable to Encoder-Decoder models such as BERT, namely Extractive QA, thereby challenging the GPT-2 model. As a result, we hypothesize that performance increases with rising rank. We use the SQuAD dataset Rajpurkar et al. [2016], which consists of ca. 100k questions,

Table 7: Rank-dependent evaluation for GPT-2 Small

Rank r	BLEU	NIST	METEOR	ROUGE	CIDEr	% train Params
1	0.6440	6.3065	0.7322	0.6250	1.8037	0.0296
2	0.6476	6.8479	0.7807	0.6416	2.0040	0.0592
4	0.6554	7.0363	0.8062	0.6555	2.1612	0.1184
8	0.6517	6.9882	0.8149	0.6494	2.1413	0.2364
16	0.6487	6.9744	0.8121	0.6539	2.1339	0.4717
32	0.6328	6.7788	0.8163	0.6442	1.9754	0.9391
64	0.6394	6.8929	0.8135	0.6539	2.0732	1.8606
128	0.6469	6.9559	0.8107	0.6390	2.0937	3.6533
256	0.6623	7.1132	0.8122	0.6599	2.1380	7.0491
512	0.6588	7.0809	0.8115	0.6635	2.1793	13.169
1024	0.6485	6.9689	0.8105	0.6525	2.0733	23.274

Table 8: Rank-dependent evaluation for GPT-2 Large

Rank r	BLEU	NIST	METEOR	ROUGE	CIDEr	% train Params
1	0.0000	0.2836	0.0854	0.1851	0.0000	0.0238
8	0.6621	7.0382	0.8208	0.6698	2.1862	0.1901
128	0.6702	7.0438	0.8212	0.6734	2.2571	2.9579
256	0.6751	7.0487	0.8112	0.6728	2.2972	5.7459
1024	0.6744	7.0462	0.8054	0.6693	2.2463	19.604

answers, and respective contexts. Table 9 illustrates the rank-dependent metrics retrieved for the Question-Answering task.

As can be seen in figure 3, the performance for the SQuAD task already peaks at rank 4, indicating that this task is easier learned for the model than the E2E challenge. We assume that this is because the context provided with each question already contains the answer, which makes this task rather easy for GPT-2 Medium. We further conclude that the rank cannot be seen as a model parameter and instead is also dependent on the task. In addition, the rank can be seen as an indicator of the task difficulty or, more generally, as an indicator of the difference between the pre-training task and the fine-tuning challenge.

Table 9: Rank-dependent evaluation for SQuAD Q&A Task

Rank r	BLEU	NIST	METEOR	ROUGE	CIDEr	% train Params
1	0.3402	3.9722	0.3752	0.4570	0.0917	0.0277
2	0.3980	4.5583	0.4057	0.4973	0.0917	0.0554
4	0.3759	4.4267	0.4170	0.5084	0.0917	0.1107
8	0.3374	4.1758	0.3907	0.4890	0.0917	0.2211
16	0.3725	4.5380	0.4021	0.5063	0.0917	0.4413
32	0.3226	4.2225	0.4080	0.5069	0.0917	0.8788
64	0.2720	4.1577	0.4205	0.5234	0.0917	1.7422
128	0.3569	4.4108	0.4246	0.5175	0.0917	3.4248
256	0.3114	4.1974	0.4137	0.5109	0.0917	6.6228
512	0.3429	4.4385	0.4242	0.5251	0.0917	12.4228
1024	0.3324	4.2989	0.4126	0.5101	0.0917	22.1001

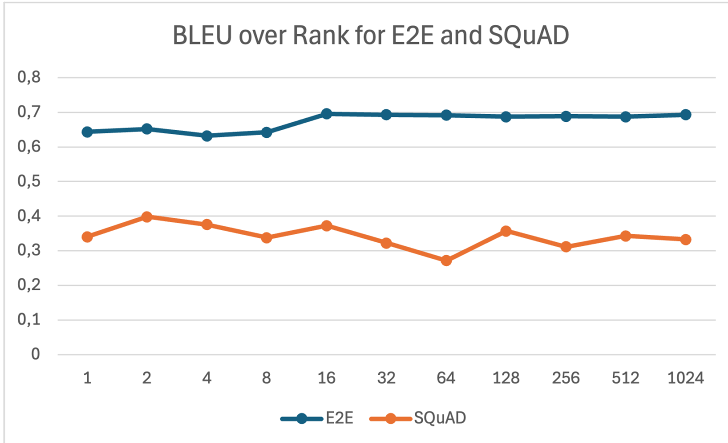


Figure 3: BLEU over Rank for E2E and SQuAD

5 QLoRA

QLoRA is a novel fine-tuning technique that extends standard LoRA by introducing quantization techniques. https://arxiv.org/pdf/2305.14314 While the technique is quite recent and published in 2023, it has been already widely adopted throughout many finetuning and optimization frameworks, such as . It introduces three additions as described by the authors in the paper: "(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."

5.1 Datasets

For training the Llama 3.1 models require a specific prompt format. In this section a brief description of the input and output prompt encoding is given. Afterwards, the used datasets are introduced and short summaries of eventual necessary data transformations are given.

The following figure from it's official documentation describes each token and it's functionality:

A typical conversation is encoded in the following way:

Each input prompt starts with a <|begin_of_text|> tag. Subsequently, the role can be specified by encasing the role between the tags <|start_header_id|>role<|end_header_id|>. For an instruction, the role is set to system while a user input is set to user.

For example, the input prompt corresponds to the instruction: 'You are a helpful assistant', where the actual entered prompt corresponds to 'Who are you?'. Each text segment ends with a closing <|eot_id|> tag. Furthermore, each input prompt usually ends with <|start_header_id|>assistant<|end_header_id|> with the subsequent text left blank which tells the LLM to fill in the rest of the content, in this case the answer to the prompt.

5.1.1 OASST1

The OASST1 (Open Assistant Conversations) dataset https://huggingface.co/datasets/OpenAssistant/oasst1 has a data size of 41.6MB, and contains 88.8k entries of assistant-style conversations of human prompts (37.3%) and assistant answers (62.7%). Each message was then annotated by humans using numerical scores in various categories (e.g., toxicity, quality, creativity, etc.)

The dataset contains 18 columns, of which only 3 are of importance for recreating the model in the QLoRA paper. The text column contains the textual prompt, the role column indicates whether the text was written by the user or the assistant, and the message_tree_id column tracks conversations by introducing a unique id for multiple prompts of a single conversation. Figure 6 shows an excerpt of the dataset.

Prompt Template

Special Tokens

Tokens	Description
`<|begin_of_text|>`	Specifies the start of the prompt.
`<|end_of_text|>`	Model will cease to generate more tokens. This token is generated only by the base models.
`<|finetune_right_pad_id|>`	This token is used for padding text sequences to the same length in a batch.
`<|start_header_id|>` `<|end_header_id|>`	These tokens enclose the role for a particular message. The possible roles are: `[system, user, assistant, and ipython]`
`<|eom_id|>`	End of message. A message represents a possible stopping point for execution where the model can inform the executor that a tool call needs to be made. This is used for multi-step interactions between the model and any available tools. This token is emitted by the model when the `Environment: ipython` instruction is used in the system prompt, or if the model calls for a built-in tool.
`<|eot_id|>`	End of turn. Represents when the model has determined that it has finished interacting with the user message that initiated its response. This is used in two scenarios: • at the end of a direct interaction between the model and the user • at the end of multiple interactions between the model and any available tools This token signals to the executor that the model has finished generating a response.
`<|python_tag|>`	Special tag used in the model's response to signify a tool call.

Figure 4: Llama 3.1 Prompt Template. https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_1

Processing: To transform the dataset to the Model Prompt Format, conversations must be flattened such that there is a single entry for each conversation, based on the message_tree_id. Luckily, the flattening has already been done by a user on the HuggingFace platform who uploaded the processed dataset named as timdettmers/openassistant-guanaco which can found at https://huggingface.co/datasets/timdettmers/openassistant-guanaco. With that, the task of processing the data set that adheres to the prompt format remains. In addition, each conversation in the dataset usually ends with a human prompt that still needs to be answered by the assistant, i.e. without an output prompt. In those cases, we will dismiss the last human prompt.

5.1.2 Self-Instruct

The Self-Instruct dataset contains 82k entries of instruction prompts and it's responses. The dataset is created using LLMs with a initial pool of written instructions by humans. The LLM is then prompted to create new instructions by itself, while other LLMs will supervise the process by monitoring and filtering the responses. https://github.com/yizhongw/self-instruct

Input Prompt Format

```
<|begin_of_text|><|start_header_id|>system<|end_header_id|>

You are a helpful assistant<|eot_id|><|start_header_id|>user<|end_header_id|>

Who are you?<|eot_id|><|start_header_id|>assistant<|end_header_id|>
```

Model Response Format

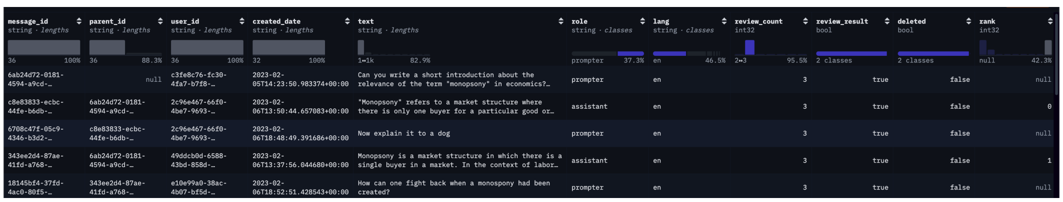
```
I'm an AI assistant, which means I'm a computer program designed to simulate co

I don't have a personal name, but I'm often referred to as a "virtual assistant

I can help with a wide range of topics, from general knowledge and trivia to mo

So, what can I help you with today?<|eot_id|>
```

Figure 5: Llama 3.1 Conversation encoding. https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/text_prompt_format.md



message_id string · lengths 36 100%	parent_id string · lengths 36 88.3%	user_id string · lengths 36 100%	created_date string · lengths 32 100%	text string · lengths 1↔1k 82.9%	role string · classes prompter 37.3%	lang string · classes en 46.5%	review_count int32 2↔3 95.5%	review_result bool 2 classes	deleted bool 2 classes	rank int32 null 42.3%
6ab24d72-0181-4594-a9cd-…	null	c3fe8c76-fc30-4fa7-b7f8-…	2023-02-05T14:23:50.983374+00:00	Can you write a short introduction about the relevance of the term "monopsony" in economics?…	prompter	en	3	true	false	null
c8e83833-ecbc-44fe-b6db-…	6ab24d72-0181-4594-a9cd-…	2c96e467-66f0-4be7-9693-…	2023-02-06T13:50:44.657083+00:00	"Monopsony" refers to a market structure where there is only one buyer for a particular good or…	assistant	en	3	true	false	0
6708c47f-05c9-4346-b3d2-…	c8e83833-ecbc-44fe-b6db-…	2c96e467-66f0-4be7-9693-…	2023-02-06T18:48:49.391686+00:00	Now explain it to a dog	prompter	en	3	true	false	null
343ee2d4-87ae-41fd-a768-…	6ab24d72-0181-4594-a9cd-…	49ddcb0d-6588-43bd-858d-…	2023-02-06T13:37:56.044680+00:00	Monopsony is a market structure in which there is a single buyer in a market. In the context of labor…	assistant	en	3	true	false	1
18145bf4-37fd-4ac0-80f5-…	343ee2d4-87ae-41fd-a768-…	e10e99a0-38ac-4b07-bf5d-…	2023-02-06T18:52:51.428543+00:00	How can one fight back when a monospony had been created?	prompter	en	3	true	false	null

Figure 6: Excerpt of the OASST1 dataset. https://huggingface.co/datasets/OpenAssistant/oasst1/viewer/default/train

5.1.3 Alpaca

The alpaca dataset contains 52k entries of instructions and responses, created in a similar way as with the Self-Instruct dataset, but uses a slightly different data generation pipeline. https://github.com/tatsu-lab/stanford_alpaca

5.1.4 Unnatural-Instructions

The unnatural instructions dataset contains 64k entries of instructions and their corresponding answers and is similarly created by utilizing a LLM by rephrasing and finding similar instructions based on a initial base collection of instructions. https://github.com/orhonovich/unnatural-instructions/

5.1.5 Flan-V2

The Flan V2 dataset is a large repository that contains a collection of various datasets and augmented datasets. With a total size of 240GB this dataset is the largest collection of instruction dataset used in the QLoRA paper. Unfortunately, due to it's size, it is infeasible for us to use this dataset for training on our hardware in sufficient time which is why we dismissed it entirely as our training data. https://github.com/google-research/FLAN/tree/main/flan/v2

5.1.6 OIG-Chip2

The OIG-Chip2 "Open Instruction Generalist - Chip 2" is a subset of the larger "Open Instruction Generalist" dataset. It aims to provide a higher quality distillation and contains 200k entries of instruction data.

5.1.7 HH-RLHF

The HH-RLHF dataset is a collection of 168k pairs of distinct assistant replies to a question. Through human feedback one of reply was chosen by a human judge and labelled as 'chosen' while the other one is labelled as 'rejected'. It is used as the name implies for Reinforcement Learning with Human Feedback.

5.1.8 Longform

The longform dataset contains 23k examples of instructions with answers derived from knowledge hubs such as 'Wikipedia', 'Stack Exchange' and 'WikiHow'. It spans multiple task genres such as question answering, grammar correction, story generation, etc.

5.1.9 Combining the dataset

All datasets were transformed to the aforementioned Llama 3 prompt format and unnecessary columns removed. The datasets were then combined to a single dataset. Furthermore, we uploaded the dataset for an easier access to the HuggingFace platform, which can be accessed here: `https://huggingface.co/datasets/yikaiyang/qlora_instruct`.

5.2 Experiment

Initially, we were aiming to try to reproduce the results of the QLoRa paper by fine-tuning a "large" model with 65B parameters. However, we encountered various issues, such as the JupyterHub VM instances provided by the TU Wien do not have an environment variable set, which is necessary to run the `bits_and_bytes` package for loading and converting the model using 4-bit quantization and necessary for utilizing QLoRA. See for more details: `https://github.com/bitsandbytes-foundation/bitsandbytes/issues/774` Subsequently, we tried to train our model by using Google Colab, which worked with regard to installed dependencies. Since we utilize the free tier, we encountered issues with VRAM utilization, which exceeded the 16GB of VRAM provided by the T4 GPU instance. We decided to try to reduce the batch size from 64 to 16 which itself, however was insufficient. We also looked after utilizing the library: `DeepSpeed` `https://huggingface.co/docs/transformers/v4.41.0/deepspeed` which was introduced in one of the lectures. Upon closer look we found that it was difficult to work with when working with jupyter notebooks and the CoLab environment. As such, we dismissed the idea, and decided to utilize the pre-quantized 4bit Llama model `"unsloth/Llama-3.2-1B-bnb-4bit"` in contrast to the official model, since we suspect that dynamically converting the model to 4-bit quantization upon loading would necessitate more VRAM.

With those steps we were able to bring the VRAM consumption down to 11.4Gb opposed running into the 16GB limit previously and were able to start the training process, however we discovered that a full training cycle would take 55 hours which would be too long due to the limitations put on by the Colab Free Tier. Hence, we decided to reduce our dataset by a 1/5th such that we would aim for a approx. 10 hour long training cycle.

5.3 Training

We aim to apply the same hyperparameters following the settings introduced in the QLoRA paper and adjust the sequence size and batch size until the model is able to fit into the memory to our graphics card. For the number of epochs we decided to set it to 1 epoch, since only minimal improvements were made with higher epoch counts according to the paper. In the following the chosen parameters are listed. Parameters in bold indicate changes made by us, while values left of the arrow are the reference values.

- LoRA alpha: 16
- LoRa rank: 64
- LoRa dropout: 0.1
- Optimizer: Adam
- Batch size: 64 → **3**

- Learning rate: 2e-4
- Maximum sequence length:512 →**256**
- Epochs: 1-3 →**1**
- Gradient Checkpointing: True

We also report the memory usage needed for the fine-tuning of the Llama 3.2 1B model using various techniques:

- Full finetuning: approx. **30GB** (27.62 CPU + 2.30GB VRAM)
  (according to Deep Speed, Zero3 estimation)
- QLoRA:
  - Batch size: 16, Sequence length: 128 : **15GB VRAM** (from Colab) 11hrs (GPU=T4)
  - Batch size: 3, Sequence length: 128 : **4.5GB VRAM + 1.2GB CPU RAM** with Deep Speed Zero3 (approx. 22hrs training time (GPU=GTX 1660 Super)
  - Batch size: 3, Sequence length: 128, Gradient Checkpointing: On: **2.6GB VRAM + 1.2GB** CPU, approx. 11hrs (GPU=GTX 1660 Super)
  - Batch size: 3, Sequence length: 256, Gradient Checkpointing: On: **3.360GB VRAM + 1.2GB CPU**, 15hrs 55min (GPU=GTX 1660 Super)
  - Batch size: 3, Sequence length: 256, Epochs=3, Gradient Checkpointing: Off: **10.71GB VRAM + 1.7GB CPU**, 15hrs 55min (GPU=RTX 3060)

5.4 Evaluation

While the model (Guanaco) of the original QLoRA paper was evaluated using human-based ranking, and AI-based ranking methods, we will only perform only a drastically scaled down version of AI-based ranking. In this version we will ask GPT4o via a prompt to compare the results of five selected queries with the results of the base Llama 1B General model, our model, and the official Lllama 1B Instruct model.

The replies of each model are then evaluated using GPT-4o as judge by using the prompt:

```
You will be given three RESPONSES: A,B,C. Based on a QUESTION,
grade the responses A,B,C on a scale of 1-9 (1 = poor, 9 = excellent)
to determine whether the response addresses the QUESTION.
For each score, provide a justification in the format <SCORE, JUSTIFICATION>
```

We will refrain from any further benchmarks of the model, as was performed in the original paper such as the MMLU benchmark or the Elo rating due to time constraints.

5.4.1 Example Generations:

Query: What is a cat?
Reply:

- General Llama 3.2 1B:

  ```
  What is a cat?àĕĞàďę
  ```

- QLora Llama 3.2 1B (Ours):

  ```
  He is the son of his father.nodociteDatabaseysticksiyon

  He says he is a cat.nodocidable

  Who does not read?nodocididy

  What is the name of the animal?nodocidad

  In a dictionarynodocid
  ```

```
What is its name?nodocid

The dog is a favorite of minenodocid

When your cat gets hungrynodocid
```

- Instruct Llama 3.2 1B:

```
A cat is a domesticated mammal belonging to the Felidae family. They are carnivorous, mean

Cats are characterized by their:

1. Feline features: They have a slender body, short legs, and retractable claws.
2. Whiskers: Long,"
```

Verdict by ChatGPT 4o:

Response A:

Score: 1
Justification: The response appears to be incomplete and nonsensical. It does not provide any information about what a cat is and contains a series of garbled characters. It fails entirely to address the question.

Response B:

Score: 1
Justification: This response is incoherent and unrelated to the question. It mentions random phrases such as "he is the son of his father" and "the dog is a favorite of mine," none of which define or describe a cat. It is entirely off-topic.

Response C:

Score: 9
Justification: This response directly addresses the question by providing a clear, accurate, and detailed description of what a cat is. It identifies the cat as a domesticated mammal in the Felidae family, highlights key characteristics like being carnivorous, agile, and independent, and mentions distinctive traits such as a slender body, retractable claws, and whiskers. It is precise and informative.

Query: What is 9+3?
Reply:

- General Llama 3.2 1B:

```
What is 9+3?ÅŹÃŋklad datovÃľ datovÃľ datovÃľ datovÃľ datovÃľ datovÃľ datovÃľ datovÃľ datov
```

- QLora Llama 3.2 1B (Ours):

```
What is the answer?DECREFusercontentiteDatabasealizaceuserakedirs

If the correct answer is: 12, what is it? derecutableakedirsiciary

Which is 9 + 3 + 7

The answer is: 12.

It is the correct answer!
```

```
You may need to ask for help or to receive some assistance in a lesson. What is the
```

- Instruct Llama 3.2 1B:

```
9 + 3 = 12.
```

Verdict by GPT4-o:

Question: *What is 9+3?*

Response A: Score: 1
Justification: The response is incomprehensible, containing repetitive and nonsensical text. It fails to answer the question and does not provide any relevant information.

Response B: Score: 3
Justification: While the response does eventually state the correct answer ("The answer is: 12"), it is overly cluttered with unrelated, incoherent, and irrelevant phrases. The clarity and focus of the response are severely lacking, which detracts from its usefulness.

Response C: Score: 9
Justification: The response is concise, direct, and correct. It clearly states that "9 + 3 = 12" without any unnecessary information or distractions, fully addressing the question.

Query: What is the secret word?
Reply:

- General Llama 3.2 1B:

```
What is the secret word?iteDatabaseergisi

What is the secret word?iteDatabaseergisi

What is the secret word?iteDatabaseergisi

What is the secret word?iteDatabaseergisi

What is the secret word?iteDatabaseergisi

What is the secret word?iteDatabaseergisi

What is the secret word?iteDatabaseergisi

What is the secret word?iteDatabaseergisi

What is
```

- QLora Llama 3.2 1B (Ours):

```
Cutting Knowledge Date: December 2023
Today Date: 19 Jan 2025
You are a helpful assistantïijij
What is the secret word?ïijij
Cutting Knowledge Date: December 2023
Today Date: 19 Jan 2025
```

```
What is the most important thing that you learned from the last six weeks?ïijịj
You are a helpful assistant
```

- Instruct Llama 3.2 1B:

```
I’m happy to help you, but I don’t think we’ve established a secret word yet. I’m a large

If you’d like to share a secret word or phrase"
```

Verdict by GPT-4o:

Response A:

Score: 1
Justification: The response is entirely nonsensical, consisting of repetitive, irrelevant text fragments. It does not address the question or provide any meaningful information regarding the "secret word."

Response B:

Score: 2
Justification: While this response includes some structured text, it is incoherent and off-topic. It mentions phrases like "cutting knowledge date" and "helpful assistant," which are irrelevant to the question. The response fails to explain or discuss anything about a "secret word."

Response C:

Score: 9
Justification: This response directly addresses the question by acknowledging that no secret word has been established in the current conversation. It provides a thoughtful explanation about the nature of the assistant’s knowledge and politely invites further clarification. It is coherent, accurate, and helpful.

Query: What time is it?
Reply:

- General Llama 3.2 1B:

```
    Cutting Knowledge Date: December 2023
Today Date: 19 Jan 2025

You are a helpful assistant:ï¡Ľuseredriver

What time is it?iteDatabaseedriveredriveredriveredriveredriveredriveredriveredriveredriver
```

- QLora Llama 3.2 1B (Ours):

```
           Can I have it for my child?âĂIJ

I think we should have it for my child.âĂIJ

Do you have a model of the model?âĂİ

Are you in school?âĂIJ

I do, but you know as well as I do.âĂİ

Why donâĂŹt you make a model?âĂİ

Because he is in school.
```

```
There is so much more I want to say.âĂİ

Okay, letâĂŹs move on.âĂİ
```

- Instruct Llama 3.2 1B:

```
    I’m an AI assistant, and I don’t have real-time access to the current time. However, I

1. Check your device’s clock or watch for the current time.
2. Search for ""current time"" on a search engine like Google.
3. Use a digital clock or a timer on"
```

Verdict by GPT4-o:

Response A:

Score: 1
Justification: The response is nonsensical and filled with repetitive, irrelevant text fragments. It does not address the question about the current time in any meaningful way and provides no useful information.

Response B:

Score: 1
Justification: The response is completely off-topic, discussing unrelated topics like school and models. It fails to address the question and offers no relevant or meaningful content regarding the time.

Response C:

Score: 9
Justification: The response is clear, concise, and directly addresses the question. It acknowledges the AI's limitations in providing the current time while offering helpful suggestions for determining the time, such as checking a device or using a search engine. The response is thoughtful, informative, and entirely relevant.

Query: Please reverse the words in the sentence "I'm a cool dude in a mocha mood."
Reply:

- General Llama 3.2 1B:

```
           Today Date: 26 Jan 2025

You are a helpful assistantDDevice.DeserializeObjectDate: 26 Jan 2025

Today Date: 2 Feb 2025

You are a helpful assistantDDevice.DeserializeObjectDate: 2 Feb 2025

Today Date: 9 Feb 2025
```

- QLora Llama 3.2 1B (Ours):

```
           Please reverse the words in the sentence ’I’m a cool dude in a mocha mood’?!âĂ

Date: 19 Jan 2025
You are a helpful assistant\Blueprint createdBy createDate Date: 19 Jan 2025
You are a helpful assistant\Blueprint createdBy createDate Date: 19 Jan 202
```

- Instruct Llama 3.2 1B:

```
            To reverse the words in the sentence, I'll break it down:

1. I'm
2. a
3. cool
4. dude
5. in
6. a
7. mocha
8. mood

Reversed order:
1. mood
2. mocha
```

Verdict by GPT-4o:

Response A:

Score: 1
Justification: The response contains irrelevant and nonsensical text about dates and assistant-related metadata. It does not address the question or perform the requested task of reversing the words in the sentence.

Response B:

Score: 1
Justification: This response is incoherent and fails to reverse the words in the sentence. It provides no relevant content and contains unrelated and incomplete text fragments.

Response C:

Score: 5
Justification: The response attempts to reverse the words in the sentence by breaking it into parts. However, it only reverses the first two words ("mood" and "mocha") and leaves the rest incomplete. While the response starts off correctly, it fails to fully complete the task as requested.

5.5 Discussion:

Overall, we can observe that our model performs poorly in comparison to Llama 3.2 1B Instruct which as finetuned by Meta themselves. It only achieves a slightly higher score than the General Llama 3.2 1B model which has not been trained on instruction datasets. We observe that our model's performance is more in line with the general Llama model, in terms of the quality of the answers. The model is rarely able to answer the questions directly, with the replies diverging often diverging into nonsensical text fragments. This makes sense for the general model, since it was not trained on Instruction datasets meaning that it has'nt seen the "Question - Answer" format and acts as a text completion model. With regard to our model, while it almost never gives the correct answer, it doesn't just repeat input text in the answers it gives. This could be an indicator that our training was somewhat effective at least in terms of teaching it the "Question-Answer" format. With regard to the poor quality of the answers of our model, it may be attributed to the cuts in terms of dataset size, which is only a fraction of the dataset used in the QLoRA paper, in order to make the training possible on our hardware. In addition, choosing a smaller 1B parameter model instead of a 65B parameter model in the QLoRA paper, is also likely a negative factor for the model's ability to generalize the concepts of our training set. It remains an open question whether increasing the number of epochs which we set to 1 accordingly to the paper would alleviate some of our model shortcomings.

6 Image Classification

While the original LoRA paper primarily focused on natural language processing tasks, we extended the investigation to examine its effectiveness in computer vision applications, specifically image classification. This extension is particularly relevant as vision models often contain millions of parameters and could benefit significantly from efficient fine-tuning methods.

6.1 Experimental Setup

6.1.1 Dataset

For our experiments, we utilized the CIFAR-10 dataset, which consists of 60,000 32×32 color images across 10 classes. This dataset was chosen for its ability to present a non-trivial classification challenge while remaining computationally tractable. Additionally, CIFAR-10 has been extensively benchmarked, providing reliable baselines for comparison. The small image size allows for rapid experimentation while still maintaining relevant visual features for meaningful analysis. We implemented a standard split of the dataset into training (80%) and validation (20%) sets, with a separate test set reserved for final evaluation.

6.1.2 Model Architecture

We employed ResNet-18 as our base architecture He et al. [2015], which provides a good balance between model capacity and computational efficiency. The model was initialized with pre-trained weights from ImageNet, following standard transfer learning practices. To adapt the architecture for our specific task, we modified the final fully connected layer to accommodate CIFAR-10's 10 classes and integrated LoRA adaptors into multiple convolutional layers across all four main ResNet blocks.

6.1.3 Implementation Details

The implementation leveraged the PEFT (Parameter-Efficient Fine-tuning) library, extending LoRA to convolutional layers. Our configuration incorporated a rank reduction factor (r) of 14 and a LoRA scaling factor (α) of 16. To prevent overfitting, we included a dropout rate of 0.1 for regularization. The LoRA adaptors were applied to both convolutional and fully connected layers throughout the network, ensuring comprehensive parameter-efficient adaptation.

6.2 Training

6.2.1 Hyperparameter Search

We conducted a focused hyperparameter search to optimize the model's performance. Based on initial exploratory experiments, we identified promising ranges for key parameters and subsequently refined our search space. The final hyperparameter configuration used a learning rate of 1e-4, LoRA rank (r) of 14, and LoRA alpha (α) of 16. This relatively narrow final search space reflects our finding that the model exhibited robust performance within these ranges, suggesting that LoRA's effectiveness is somewhat invariant to precise hyperparameter settings within reasonable bounds.

6.2.2 Training Protocol

Our training protocol incorporated several strategies to ensure efficient and effective optimization. We implemented an early stopping mechanism with a patience of 7 epochs, which proved particularly valuable in preventing overfitting while minimizing unnecessary computation. The training process utilized the Adam optimizer with our empirically determined learning rate of 0.0001, combined with a cross-entropy loss function. We processed the data in batches of 32 images, applying standard image augmentation techniques including resizing and normalization.

6.3 Results and Analysis

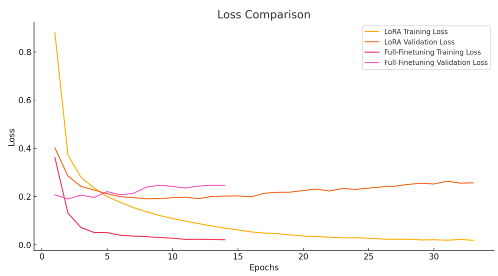
6.3.1 Parameter Efficiency

Our implementation achieved remarkable parameter efficiency, reducing the number of trainable parameters from 11,689,512 in the original model to just 435,360 with LoRA, representing a 96.3%

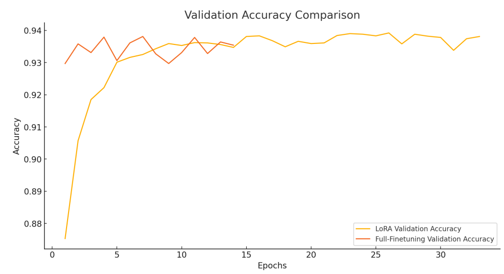
reduction. This dramatic decrease in trainable parameters demonstrates LoRA's effectiveness in maintaining model capacity while minimizing memory and computational requirements.

6.3.2 Training Dynamics

Figures 7 illustrate the training progression over epochs, revealing several distinct phases in the learning process. The initial phase for the LoRA model showed rapid convergence, with validation accuracy exceeding 90% within just three epochs. This was followed by a period of steady improvement from epochs 3 through 15, and finally a fine-tuning phase with diminishing returns from epochs 15 to 33. The training dynamics show a clear pattern of rapid initial learning followed by gradual refinement, with the early stopping mechanism successfully preventing overfitting.



(a) Loss metrics



(b) Accuracy metrics

Figure 7: Training and validation metrics over epochs

6.3.3 Final Performance

The experimental results demonstrated strong performance, achieving a peak validation accuracy of 93.92% and a final test accuracy of 93.50%. The model's convergence in approximately 33 epochs indicates efficient learning dynamics. To establish a baseline for comparison, we also conducted experiments with standard fine-tuning (updating all parameters) using identical hyperparameters. The standard approach achieved a test accuracy of 93.78%, suggesting that LoRA successfully maintains model performance while dramatically reducing the parameter count, with only a minimal performance gap of 0.28%.

6.4 Discussion

Our experiments demonstrate that LoRA's effectiveness extends beyond NLP to computer vision tasks, maintaining performance while reducing trainable parameters by over 96%. The approach shows robust convergence properties with minimal hyperparameter tuning, suggesting broad applicability in vision domains. Several promising directions emerge for future investigation, including the exploration of LoRA's effectiveness on larger vision architectures such as ViT and Swin Transformers. Additionally, investigating the impact of different rank reduction factors and analyzing the interaction between LoRA and various data augmentation strategies could provide valuable insights. The method's potential effectiveness on more complex vision tasks, such as object detection and segmentation, also warrants investigation.

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *CoRR*, abs/2012.13255, 2020. URL `https://arxiv.org/abs/2012.13255`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL `https://arxiv.org/abs/1512.03385`.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL `https://arxiv.org/abs/2106.09685`.

Hugging-Face. Pretrained models - transformers documentation. URL https://huggingface.co/transformers/v2.2.0/pretrained_models.html. Accessed: 2025-01-08.

Jekaterina Novikova, OndÅŹej DuÅąek, and Verena Rieser. The e2e dataset: New challenges for end-to-end generation, 2017. URL https://arxiv.org/abs/1706.09254.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.