Universiteit Leiden
The Netherlands

Opleiding Informatica

Transferring video captioning models from human to animal data

Giovanni Halevy

Supervisors:
Dr. Hazel Doughty

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

23/07/2024

Abstract

AI models are getting larger in size and creating one from scratch is becoming less appealing, instead a pre-trained model can be fine-tuned to succeed at a similar task. This falls under transfer learning and is the topic of this thesis. We will look at transfer learning techniques LoRA and AdaptFormer and compare these with regular fine-tuning for 2 video captioning models. These models are git-base-vatex and SpaceTimeGPT, both are already fine-tuned on human data from the VATEX dataset and next they will be fine-tuned on animal data from the Animal Kingdom dataset. The generated captions are then compared with the ground truth with the following metrics BLEU, ROUGE and METEOR to measure performance. The results show that LoRA outperforms AdaptFormer when both have few parameters, however when both techniques are given more parameters AdaptFormer performs significantly better than LoRA. The larger AdaptFormer fine-tune is either on par for the GIT model or better for the GPT model.

Contents

1 Introduction

When we look at a video we immediately have an understanding of what is going on in the video, and even if we do not understand exactly what we are seeing, we can still describe it. We built an understanding of the world around us from living in it and this is constantly adjusted to stay "up-to-date". However, AI models do not innately have this understanding of the world around us and so if we want the models to comprehend visual data such as videos we must first teach them. Video captioning is the process of generating a textual description or caption from a video. This requires understanding the individual frames and contextually creating a caption describing what was visible in the video.

While significant advancements have been made in video captioning for human activities, video captioning for animals has faded into the background. However, it can provide valuable insights for research in animal behavior and tracking, ultimately increasing our understanding and conservation of wildlife.

Current models are trained on datasets that are human-centric, so these datasets contain people doing human activities. Furthermore, these datasets also do not contain a wide range of animals so when such an model is shown an uncommon animal, like a sting ray for example, it would not know what it is looking at. Making it impossible for the model to accurately caption such a video. The current models struggle with animal data and it is clear we need a different model. However, this does not mean that we need to start from scratch.

In recent years AI models are becoming better but with this performance increase comes a cost. The size of the models is becoming larger and larger. Creating and training a model from scratch is not only becoming less economically viable but also computationally. Instead a pre-trained model trained on a similar task can be used as a base model to be fine-tuned with the new task. This is called Transfer learning and it can save a lot of computational power when training a model. There is various different techniques that can be used such as LoRA and AdaptFormer that will be used in this thesis. Both of these techniques freeze the original model and adds additional trainable parameters, only these new parameters will be changed. Thereby reducing the number of trainable parameters and thus with computational load and training time.
This study aims at investigating the following research questions:

1. Do video captioning models transfer well from human to animal data?

2. How does partial fine-tuning of a pre-trained model affect the performance of video captioning models in transferring from human to animal data?

3. How does training newly introduced parameters, using LoRA and AdaptFormer, into a frozen pre-trained model affect the performance of video captioning models in transferring from human to animal data?

To address these questions, this research evaluates two video captioning models, GIT-base-vatex and SpaceTimeGPT. Both models, originally fine-tuned on human data from the VATEX dataset, will be subsequently fine-tuned on animal data from the Animal Kingdom dataset using three transfer learning techniques. First a regular partial and full fine-tune, next a LoRA fine-tune will be executed on attention layers of both models and finally the models will be fine-tuned using the AdaptFormer technique. The performance of these models will be assessed using BLEU, ROUGE, and METEOR metrics to compare the generated captions with the ground truth.

1.1 Thesis overview

Here is a short thesis overview. This chapter contains the introduction; Section 2 includes the definitions; Section 3 discusses related work; Section 4 describes the experiments and their results; Section 5 concludes by answering the research question and proposes future work. This bachelor thesis was made by Giovanni Halevy under supervision of Dr. Hazel Doughty.

2 Definitions

Layers: Modern AI models work by taking an input and sending it through its various layers, in essence these layers are the model. Different layers alter the input differently before sending this processed input, or output, onto the next layer. layers often work by multiplying their input with their own weight matrix, adding a bias vector, and then applying an activation function to this result to obtain the output. Though the bias and activation function are important components, they are not relevant for thesis as we are more interested in the (trainable) parameter count.

Parameters & Trainable Parameters: Layers have weight matrices, and these matrices have dimensions. The parameter count is the number of elements in the weight matrix. So for an MxN matrix the parameter count would be MxN, technically the bias vector should be added of size N as well, so MxN + N.
The weights in the layers are learned during training, the values are adjusted with a technique called back-propagation. But when a model is trained or rather fine-tuned, one might not want to adjust all the weights. So only a subset of all parameters would be trainable, only the trainable parameters would then have their weight adjusted.

Attention: A concept that is emphasized in the transformer model [VSP+23], attention mechanisms help capture relationships within the input data by assigning different levels of importance to different parts of the input. This allows the model to focus on the most relevant pieces of information.

Note that git-base-vatex and SpaceTimeGPT are later often referred to as GIT and GPT for brevity.

3 Background & Related Work

3.1 Video Captioning

Video captioning in this context is the task of automatically describing the content of a video, which requires some level of understanding of the video and using this to generate text. Before the transformer model was introduced in "Attention is all you need" [VSP+23], combinations of CNN,RNN and LSTM's were often used. For example, here [CJ19] a CNN and a custom RNN with attention called GARU is used in a similar way to an encoder. Then, this would be decoded back to natural language using LSTMs. This model set the contemporary state of the art scores in 2019 for video captioning on MSVD and MSR-VTT datasets.

However, after the transformers introduction the model became more widely used because of its significant advantages over contemporary models. Specifically the benefits of its self-attention layers. The three main advantages are, as stated in the original paper, reduced computational complexity per layer, increased parallelization, and the ability to learn longer-range dependencies.
In the case of video captioning, these advantages are particularly beneficial. Transformers can process multiple frames simultaneously instead of consecutively, leading to a significant speedup in both training and inference. Additionally, the attention layers in transformers determine the importance of different parts of the input for each element of the output. This capability allows the model to identify which frames are most important for generating specific parts of the caption. For instance, a particular frame might be crucial for describing a specific action or object, enabling the model to generate more accurate and contextually appropriate descriptions.

3.2 Transfer Learning

By using transfer learning, high computational costs and the large data requirements associated with training a model from scratch can be avoided. Instead an already existing model with a similar task can serve as a starting point. This model can then be fine-tuned using various techniques, such as freezing certain layers or adding new parameters to allow for the model to adapt to the new task.

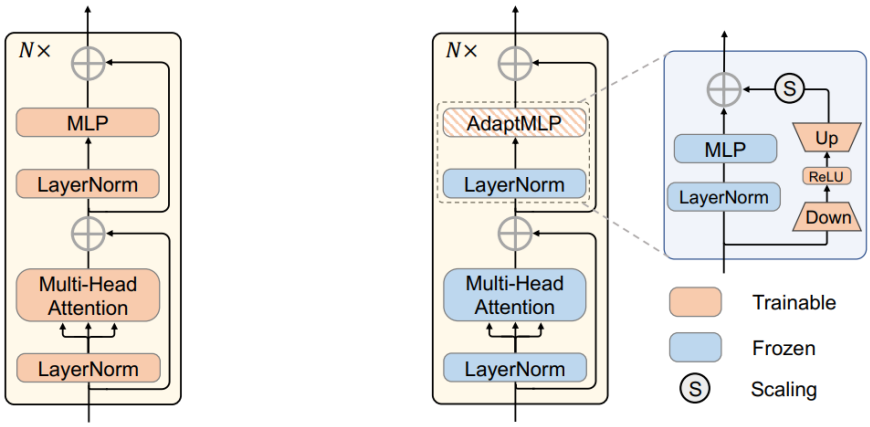
The most basic form of transfer learning begins by training a model on a large dataset to learn fundamental features like shapes and colours, which are broadly applicable. After this initial training, the model can be fine-tuned on a new smaller or more specialized dataset.

During this fine-tuning, the model's parameters are adjusted, typically with a lower learning rate. The early layers, which capture generic features, remain mostly unchanged, while the later layers are adjusted to adapt to the new task. This allows the model to reuse fundamental knowledge learned from the first dataset, requiring less data and computational cost to achieve high performance on a specialized task.

In this thesis, techniques like AdaptFormer [CGT+22] and LoRA (Low-Rank Adaptation) [HSW+21] are used to transfer knowledge from video captioning models trained on human data to instead work with animal data.

3.3 AdaptFormer

AdaptFormer, initially introduced in 2022 [CGT+22], addresses the challenges of adapting pre-trained vision transformers to various tasks. These significant challenges being: computational overhead and memory storage issues.
AdaptFormer introduces small, additional modules to the MLP layers of a transformer. These modules are designed to act as bottleneck structures, which include a down-projection layer, a non-linear activation layer (such as ReLU), and an up-projection layer. The dimensions of these layers are chosen to ensure that the middle dimension is smaller than the input and output dimensions, thus reducing the number of parameters and computational complexity.



(a) Full fine-tuning. (b) AdaptFormer fine-tuning

Figure 1: a: regular vision transformer, b: vision transformer with an AdaptMLP; Source: AdaptFormer: Adapting Vision Transformers for Scalable Visual Recognition, 2022

One of the key advantages of AdaptFormer is its efficiency. By adding less than 2% extra parameters to the model, it ensures that the increase in computational and memory requirements is minimal. This is achieved by freezing the original model parameters during fine-tuning and only updating the parameters of the added modules. As Figure 1 shows the output of the upscaling layer is then scaled by some scaling factor, we used a scaling factor of .1 as this is also used in their shown implementation in the appendix of their paper. The resulting tensor is then element-wise added to the output of the original MLP. Consequently, when the model is fine-tuned for a new task, only these additional parameters are updated, significantly reducing the computational load.
Additionally, as shown in the AdaptFormer paper, this method loses very little performance or even gains performance when compared to a full fine-tune. While, as said before, training just a fraction of the parameters.

3.4 Lora

LoRA [HSW+21], published one year prior of Adaptformer in 2021, works very similar to Adaptformer. They both address the same problem of fine-tuning larger pre-trained models. This fine-tuning becomes too computationally expensive. Lora mitigates this effect by freezing all original parameters and injecting trainable rank decomposition matrices into each layer of the transformer. This reduces the amount of trainable parameters and thus lowers the computational load.

The way this rank decomposition matrices work is like so, a weight matrix of MxN gets decomposed into two smaller LoRA weight matrices. These are called A and B, these matrices have dimensions MxR and RxN, where r is the rank of the decomposition. This r determines how many parameters A and B will contain.

$$\begin{aligned}
\#\text{param A} &= M \times r \\
\#\text{param B} &= r \times N \\
\#\text{param LoRA} &= r \times (M + N) \\
\#\text{param without LoRA} &= M \times N
\end{aligned}$$

As you can see, this only works with larger matrices, and preferably a low r (r smaller than M and N). And there is also a downside of using this rank decomposition, since A and B together might not exactly recreate the original matrix. But as shown in the original paper of LoRA, this effect is unnoticeable in the results. And it is mentioned in the LoRA paper that they suspect that this is because of this concept intrinsic dimension, it is thought that the relevant information that is learned in the model can be stored at a lower dimension without losing data.

3.5 Metrics

3.5.1 BLEU

In 2002 BLEU was proposed [PRWZ02] for automatic evaluation of machine translated text, initially this was done because human evaluations were slow and expensive and this automatic evaluation was fast and inexpensive. This metric was originally created to evaluate the quality of translations by measuring the similarity between candidate and reference sentences, but it can also be used to rate captions created by a captioning model by comparing its output to the ground truth.

BLEU works by counting n-gram precision, where precision counts the amount of n-grams of the candidate that also occur in the reference. An n-gram here is an in order sequence of n words. For example, "The cat is on the mat" becomes: "The cat is", "cat is on", "is on the" and "on the mat" in case of 3-grams. BLEU then sums n-gram precisions for n and below and averages them uniformly, followed by adding a brevity penalty. This is done so a short sentence with perfect precision, all its n-grams also in the reference, is punished relative to its length.

$$\log \text{BLEU} = \min\left(1 - \frac{r}{c}, 0\right) + \sum_{n=1}^{N} w_n \log p_n$$

Where r and c are the length of the reference and candidate sentences.

3.5.2 METEOR

Like BLEU, METEOR [BL05] was made for machine translation. METEOR works in a similar way but only works with unigrams (1-grams) and tries to take synonyms into account. METEOR works with precision and recall, whereas BLEU only used precision.
For METEOR, first the Fmean is calculated using the precision and recall.

$$F_{\text{mean}} = \frac{10PR}{R + 9P}$$

Then we count the amount of chunks in the system translation, chunks are created by grouping consecutive correct unigrams, with as few chunks as possible. Then, the penalty can be calculated, by taking the number of chunks and dividing it by the amount of matching unigrams and raising that to third power and finally multiplying that by .5. Now the final score can be calculated by taking the Fmean and multiplying that by (1-Penalty)

$$\text{Penalty} = 0.5 \times \left(\frac{\#\text{chunks}}{\#\text{unigrams\_matched}} \right)^3$$

$$\text{Score} = F_{\text{mean}} \times (1 - \text{Penalty})$$

3.5.3 ROUGE

ROUGE was proposed in 2004 [Lin04], with its purpose for automatically evaluating summaries. It is, however, also used to measure similarity between sentences, which is how it will be used in this thesis. This paper included 4 different ROUGE measures: ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S. However, we only use ROUGE-N, so only this one will be explained here.
ROUGE-N measures the overlap of n-grams between the candidate text and the reference text. Similarly to BLEU. For clarity, precision is the ratio of n-grams in the candidate that are also in the reference, and recall is the ratio of n-grams in the reference that are also in the candidate. The ROUGE-N F-measure is the harmonic mean of recall and precision, providing a balanced evaluation of both metrics.

$$F_1 = 2 \times \frac{PR}{P + R}$$

3.6 Methodology

For the experiments two different video models will be used, both were fine-tuned on the VATEX dataset. The VATEX dataset contains 41250 videos and 825000 captions in English and Chinese. With a 50/50 split of 10 english and 10 Chinese captions per video. The models, SpaceTimeGPT and git-base-vatex, were chosen so the VATEX dataset can be used as source for the transfer learning. Both models are available through huggingface. For all experiments the AdamW optimizer was used with a learning rate of 5e-7 and 100 epochs. All training of the models was done on the ALICE cluster of Leiden University.

The videos used for the VATEX [WWC+20] dataset are a subset of the Kinetics-600 dataset, the videos are taken from YouTube. These videos describe 600 human action classes, so Kinetics-600 is human-centric and VATEX using a subset of this dataset makes it also human-centric. This allows us to use models fine-tuned on this dataset as source of our transfer.

The SpaceTimeGPT model was created by "Neleacs" and is available on huggingface. This model was constructed by taking the TimeSformer made by Facebook [BWT21] and adding behind it a GPT2 model to convert this classification task to a captioning task. This model was chosen because both of its components prove to be strong models and the design of placing a LLM after a classification model peaked my interest. The TimeSformer is pre-trained on ImageNet-21K and then trained again on Kinetics-600. The model not only proves to perform well but is also shown to be computationally efficient. As shown in the original paper where it ranked 4th on their kinetics-400 experiments (not counting the larger TimeSformer models). With only a 1.6 difference between it and number 1 with scores 78 and 80.4 respectively. And above all, this was achieved with around one tenth of its computational power. Combining this model with the GPT2 LLM to make the SpaceTimeGPT video captioning model appears to be promising with a 67.3 CiDER score on the VATEX dataset as stated on the GitHub and huggingface page.

This second model was found after looking at video captioning benchmarks on paperswithcode, where I checked for the top models whether they had their model easily available on huggingface. This was the case for GIT2, however the original model was not made public, instead they made smaller base and large models public along with fine-tuned versions of these models on various datasets. Huggingface had these smaller models available and I chose to work with git-base-vatex model, because the SpaceTimeGPT model was also fine-tuned on the VATEX dataset. It is unclear what this smaller model was originally trained on, but it is likely to be trained on a subset of the GIT2's training data which is a mix of different datasets including COCO and Conceptual Captions. The git-base-vatex model scored a CiDER score of 60 as reported in its paper.

This dataset mainly held information for pose estimation, but it also had annotated videos for a video grounding task. Where a model would get a textual description and a video and it would need to identify the section of the video that the description applies to, however this can also be used for captioning since the dataset contains annotated sections snippets of the videos. These video clips ranged from 3 seconds to around 80 seconds. Though the annotations were not for the entire length of a video clip, and often there would be multiple annotations per clip, sometimes even multiple annotations for the same segment of a clip. In order keep the annotations more accurate we spliced out the clip which was annotated. This was not too difficult since the data look like this: `"AANNQNXN 28.7 30.9##The jellyfish is swimming."`. Where it shows the name of the video clip, followed by the start and end time of the annotated segment along with the actual annotation. Some clips were too short and needed to be extended in order to be usable, in those cases I lowered the start time by .3 seconds and increased the end by .5 seconds. This way the clips would be long enough to be used. In total there were 18744 captions across 4301 video clips totalling to 50 hours of video. For the different splits we took the train/test split that the Animal Kingdom [NOZ+22] dataset already provided, which was train/test of 80/20, next we took 10% of the train to create a validation set. So we ended up with 72/20/8 for train/test/val.
The git-base model takes 6 frames and the SpaceTimeGPT model takes 8 frames. From each spliced

video 8 frames are sampled linearly, the SpaceTimeGPT models gets all 8 and the git-base the first 6. This way the models roughly get the same data and as a bonus I had to change little to the pre-processing code.

4 Experiments

4.1 Zero-Shot

First, we evaluated the models on the test set of the Animal Kingdom dataset without any fine-tuning, this result will act as a zero-shot baseline for comparison with the fine-tuned models. The results are summarized in Table 1. The table also shows some of the previously mentioned metrics, as well as the parameter count for each model.

Notably, the SpaceTimeGPT model has approximately 55% more parameters than the GIT model, yet its zero-shot performance is lower. This holds throughout all experiments, indicating that the GIT model generally outperforms SpaceTimeGPT in video captioning. However, since we are specifically looking into transfer techniques and their performance, the results of the transfers should be evaluated separately for both models.
The zero-shot evaluation scores on the Animal Kingdom dataset are lower than the scores on the VATEX dataset which they were fine-tuned on. The BLEU4 and METEOR scores (not shown in the table) on VATEX are 37.9 and 24.4 for the GIT model and 8.1 and 27.2 for the GPT model. The BLEU4 score fell significantly for both models, however the METEOR scores are relatively the same. This is probably because this score uses unigrams and even accounts for synonyms, whereas BLEU4 uses n-grams ranging from 1-grams to 4-grams. The zero-shot results for BLEU4 dropping significantly indicates that the models might not transfer well from human to animal data.

Experiment	BLEU1	BLEU2	BLEU4	METEOR	ROUGE1	ROUGE2	Parameter Count
GIT0Shot	20.19	7.49	2.67	23.28	20.64	4.34	176,623,674
GPT0Shot	17.58	4.12	1.86	18.40	15.55	0.51	274,065,408

Table 1: Evaluation zero-shot results

In order to give these metrics some more visual context, Figure 2 shows a frame of 2 different videos with the generated zero-shot captions for both models along with the ground-truth.
As can be seen in the resulting captions, the SpaceTimeGPT model seems to consistently hallucinate people. The GIT model seems to caption the videos better, but it still it confuses a black bearded draco for a turtle. Although this is probably because it never had this animal in its previous dataset.

(a) GPT: A person is playing with a large white bird that is sitting on the ground.
GIT: A group of birds are standing in the sand and are walking around.
GroundTruth: The ardea alba egret is walking.

(b) GPT: A person is climbing up a tree with a chainsaw.
GIT: A turtle is sitting on a tree and then it is sitting on a tree.
GroundTruth: The black bearded draco is sensing its environment.

Figure 2: Zero-shot captions

4.2 Regular Fine-tune

Now that the baseline is set and we have an idea of what to expect of the captions, we can move on to the regular fine-tuning experiment. For this experiment we will fine-tune both models twice, once where we all parameters are fine-tuned and another where we only fine-tune the last block of the second part of the model. For the SpaceTimeGPT model this would mean the last of 12 GPT2blocks and below, and for the GIT model this would be the last of 5 GITLayers and below. The results can be seen below in Table 2.

We specifically chose to fine-tune the last block because the more complicated relationships between input and output are captured in the later layers of the model. By only fine-tuning the last block, we can train fewer layers while focusing on the most important parts of the model.

Experiment	BLEU1	BLEU2	BLEU4	METEOR	ROUGE1	ROUGE2	Trained Parameter Count
GITFT	56.61	38.96	25.69	52.32	58.40	35.25	176,623,674 (100%)
GITPT	50.34	29.09	17.12	42.36	51.14	23.19	31,156,026 (17.6%)
GPTFT	50.88	28.84	18.69	42.54	50.17	22.21	274,065,408 (100%)
GPTPT	48.57	25.66	16.53	39.70	47.25	18.74	48,050,688 (17.5%)
GIT0Shot	20.19	7.49	2.67	23.28	20.64	4.34	176,623,674
GPT0Shot	17.58	4.12	1.86	18.40	15.55	0.51	274,065,408

Table 2: Evaluation Results of Full fine-tune (FT) and Partial fine-tune (PT), previous zero-shot results are added for context.

As shown in Table 2, the metrics increased for all models after fine-tuning. This is to be expected as the zero-shot models were trained on a different dataset, so the resulting caption was matching output from the VATEX dataset. After fine-tuning the models, the output should be tuned more towards the Animal Kingdom dataset.
Out of the noted metrics, the longer n-grams are more important to look at, these are BLEU4 and ROUGE2, since in order to score higher on these metrics more consecutive words have to be correct. And like mentioned before these metrics increased as well, even for the partially trained models which had around 17.5% of the total trainable parameters.

(a) GPT FT: The common crane is sensing its environment.
GPT PT: The common crane is sensing its environment.
GIT PT: The common crane is attending.
GIT FT: The goose is flapping its wings.
GroundTruth: The greylag goose is flapping its wings.

(b) GPT FT: The black rhinoceros is walking.
GPT PT: The horse is sensing its environment.
GIT PT: The deer is walking.
GIT FT: The deer is keeping still.
GroundTruth: The spotted deer is keeping still.

Figure 3: Full and Partial fine-tune captions

In Figure 3 the captions are shown. Both models learned the format of the sentences. However, the GPT model continues to struggle with accurately identifying the animal and its action. The fully trained GIT model on the other hand often gets the animal and its action correct, but its partially trained counterpart can still incorrectly identify the animal and its action, like it did in Figure 3a where it calls a goose a “common crane” and incorrectly captions its action as “attending”.

4.3 LoRA

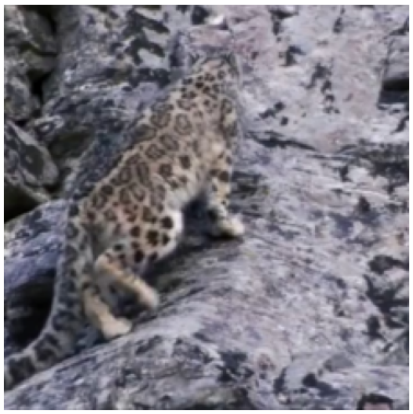
Next we use the LoRA technique to add extra parameters to both models. Where only these extra parameters will be trained and all original parameters are frozen. These new parameters will be injected in the attention layers as it was done in the original paper [HSW+21].
For this implementation LoRA the python package peft was used, where the extra parameters could easily be added. This required setting up a loraConfig that describes the dimensions of the newly added weight matrices along with the layers that should get the added matrices. For the initial experiment with LoRA, LoRA layers were added to all blocks or to just the last block using both middle dimensions 8 and 16. The LoRA paper stated that the relationship between optimal rank for adaptation and model size was still an open question but in their paper they did show a table for gpt2-medium, which is of similar size to this SpaceTimeGPT model, which indicate that the optimal rank for their model ranged from 4 to 16. So that is why 8 and 16 were chosen for LoRA. With the results below Table 3.

Experiment	BLEU1	BLEU2	BLEU4	METEOR	ROUGE1	ROUGE2	Trained Parameters
GITLora all 8	51.25	30.71	18.35	44.00	52.51	25.00	811,008 (0.45%)
GITLora all 16	51.22	30.78	18.52	43.99	52.51	25.10	1,622,016 (0.91%)
GITLora last 8	49.99	28.01	16.37	41.44	50.30	21.49	98,304 (0.06%)
GITLora last 16	50.01	28.09	16.32	41.49	50.24	21.57	196,608 (0.11%)
GPTLoRA all 8	46.53	20.78	12.39	35.17	44.02	13.14	998,024 (0.36%)
GPTLoRA all 16	46.61	20.87	12.49	35.23	44.20	13.24	1,996,048 (0.72%)
GPTLoRA last 8	45.52	19.32	12.07	33.77	42.81	11.62	457,352 (0.16%)
GPTLoRA last 16	45.55	19.32	11.98	33.72	42.93	11.55	914,704 (0.33%)
GITFT	56.61	38.96	25.69	52.32	58.40	35.25	176,623,674 (100%)
GITPT	50.34	29.09	17.12	42.36	51.14	23.19	31,156,026 (17.6%)
GPTFT	50.88	28.84	18.69	42.54	50.17	22.21	274,065,408 (100%)
GPTPT	48.57	25.66	16.53	39.70	47.25	18.74	48,050,688 (17.5%)

Table 3: Evaluation Results of LoRA with ranks 8 and 16, and LoRA applied to all or just last attention layers. Full fine-tune added for context.

The metrics for the LoRA-trained models are lower than those for the fully fine-tuned models. However, for the GIT model, the version where LoRA is applied to all attention layers outperforms the partially fine-tuned model, this is achieved with a fraction of the trainable parameters 0.9% or even 0.45% compared to the 17% of the partially trained GIT model. Additionally, doubling the dimension from 8 to 16 does not seem to have an effect on the performance.
For the GPT LoRA models, the results are quite low, this will be discussed in a later section.
Figure 4 shows the generated captions again for 2 videos, the captions of the GIT models are quite good when you take in account the few parameters that were added and trained. The GIT models get the animal right often, though the action can still be incorrect. For the GPT model, considering all its outputs, it almost always captions with "sensing its environment" and almost always calls the animal a bird. A potential explanation will be given later in the discussion section.

(a) GPT: The bird is sensing its environment.
GIT all 8, last8/16: The squirrel is keeping still.
GIT all 16: The squirrel is walking.
Ground Truth: The squirrel is sensing its environment.



(b) GPT all 8: The otter is sensing its environment.
GPT all 16: The otter is keeping its mouth open.
GPT last 8/16: The bird is sensing its environment.
GIT all 8/16: The leopard is walking.
GIT last 8/16: The cheetah is walking.
Ground Truth: The snow leopard is climbing.

Figure 4: LoRA captions, where "all" and "last" describes which attention layers got LoRA weights. Models that give the same captions are merged.

4.4 AdaptFormer

The next experiment uses AdaptFormer to add extra parameters to the MLP layers, here we will use the recommended middle dimension size of 64. And since we did two different sizes of LoRA we also added an AdaptFormer with a mid dimension which would correspond to the percentage of the added parameters using LoRA, this is 96 for the GIT model and 118 for the GPT model.

Experiment	BLEU1	BLEU2	BLEU4	METEOR	ROUGE1	ROUGE2	Trained Parameters
AdaptGIT mid96	47.52	23.74	12.15	37.83	48.83	17.15	1,631,520 (0.92%)
AdaptGIT mid64	47.38	23.53	12.11	37.59	48.74	16.83	1,090,496 (0.61%)
GPT118Adapt	32.22	15.41	8.98	24.24	26.22	8.93	2,003,474 (0.73%)
GPT64Adapt	30.94	13.84	7.47	22.49	25.29	7.12	1,090,496 (0.40%)
GITLora all 16	51.22	30.78	18.52	43.99	52.51	25.10	1,622,016 (0.91%)
GPTLoRA all 16	46.61	20.87	12.49	35.23	44.20	13.24	1,996,048 (0.72%)

Table 4: Evaluation Results for AdaptFormer with various middle dimensions. Corresponding LoRA results added for context.

The results in Table 4 are quite low compared to the LoRA models even though the models have around the same amount of added parameters. This could be because of the amount of trainable parameters or because of the location of trainable parameters. But because LoRA was able to do better with the same amount of parameters I think it is more likely to be the location of trainable parameters.

(a) GIT 96: The snake is walking on the ground.
GIT 64: The snake is walking on the ground.
GPT 118: A bird is keeping still.
GPT 64: A kangaroo is walking.
Ground Truth: The ashe's spitting cobra is keeping still.

(b) GIT 96: The eagle is eating.
GIT 64: The golden eagle is eating.
GPT 118: A bird is keeping still.
GPT 64: A bird is flying.
Ground Truth: The golden eagle is sensing its environment.

Figure 5: Captions for AdaptFormer models with various middle dimensions

As can be seen in Figure 5, for the GIT model, the animal is recognized correctly but its action is still wrong. The GIT model also appears to unlearn the specific animal name in Figure 5b, with middle dimension it said "golden eagle" and then with middle dimension of 96 it said "eagle" instead. Then the GPT model seems to get the animal name and action wrong, this model might need more trainable parameters to adjust properly to the Animal Kingdom dataset.

4.5 AdaptFormer vs LoRA

As the final experiment we give both the AdaptFormer and LoRA a significantly bigger middle dimension, this is done in order to give the models more trainable parameters. With these extra parameters it might be able to capture additional relationships between in and output.
Also for this experiment a AdaptIntermediate will also be added along with the previously used AdaptMLP, this was added so the AdaptFormer also gets trainable parameters in both parts of each model. Since both models can be seen as two main sections. The way LoRA was configured added extra parameters to all attention layers which included attention layers in the second. However, AdaptFormer only added to MLP layers which are only in the first section with the GIT model and only in the second part for the GPT model. So to ensure that both techniques get trainable parameters in both sections this AdaptIntermediate was added. Then for this final experiment the AdaptFormer was also placed in different configurations with middle dimension of 3072: all blocks, every other and only the last.
Since this was already done for the LoRA technique we only did 1 LoRA test with larger dimensions, where LoRA is applied to all attention layers with middle dimension of 1090 in order to match the trainable parameter percentage of AdaptFormer.

Experiment	BLEU1	BLEU2	BLEU4	METEOR	ROUGE1	ROUGE2	Trained Parameters
GITAdapt all	56.63	38.62	25.67	51.65	57.86	35.24	110,959,872 (38.58%)
GITAdapt half	55.61	37.22	23.64	50.26	57.14	33.50	63,742,464 (26.52%)
GITAdapt last	20.40	7.64	2.77	23.40	20.84	04.47	16,525,056 (8.56%)
GITLora all 1090	51.42	31.04	18.54	44.27	52.72	25.38	110,499,840 (38.49%)
GPTAdapt all	52.83	31.70	20.09	45.19	52.75	25.56	142,821,888 (34.26%)
GPTAdapt half	51.00	28.73	17.64	42.38	50.49	21.77	77,902,848 (22.13%)
GPTAdapt last	47.36	21.99	13.11	36.09	45.25	13.78	12,983,808 (4.52%)
GPTLoRA all 1150	46.22	20.39	12.18	34.91	43.69	12.81	143,465,950 (34.36%)

Table 5: Evaluation Results

The results for this final test Table 5 are quite interesting, since it appears that the AdaptFormer models benefited more from the extra parameters and adjusted placement of the added parameters. Although the GIT model with the larger middle dimension only in the MLP and Intermediate layer of the last block performs similarly to the zero-shot results, but this can be explained by the fact that parameters are only added two layers; 1 MLP and 1 intermediate.
Both the GPT and GIT model with a greater middle dimension either match or outperform the full fine-tune while training fewer parameters. However, note that the percentage is trainable parameters of total with the extra parameters added, so the 38% trainable parameters is around 62% of original model size.
The LoRA models are only barely better than its smaller counterpart, indicating that, at least for LoRA, the parameter count is less influential then its location.

(a) GIT all: The fish is keeping still.
GIT-A half: The fish is swimming.
GIT-A last: A fish is swimming in the water and then a fish is swimming around.
GIT-L 1090: The fish is swimming.
GPT-A all/half: The fish is swimming.
GPT-A last: The sea turtle is swimming.
GPT-L 1150: The fish is swimming.
Ground Truth: The pike perch fish is swimming.

(b) GIT-A all: The spider is flying.
GIT-A half: The spider is keeping still.
GIT-A last: A spider is on a branch and is being held by a spider.
GIT-L 1090: The spider is keeping still.
GPT-A all/half: The spider is keeping still.
GPT-A last: The bird is keeping still.
GPT-L 1150: The bird is sensing its environment.
Ground Truth: The golden orb spider is keeping still.

Figure 6: Captions for (A)daptFormer and (L)oRA in various configurations

Figure 6 shows some generated captions, the captions of the models are generally good with as exception AdaptFormer applied to only the last block but we knew this already from the metrics. The GPT models still struggle with identifying the animal yet it seems to get the action correct. The GIT models generally get the animal and action correct, however the GITAdapt all model misidentifies the spider as "flying" even though its fewer trainable parameter counterparts do identify the action correctly as "keeping" still.

4.6 Discussion

In this section I want to point out some important information before moving on to the conclusion:
The GPT models with LoRA were generally worse because I accidentally added the LoRA layers to only the attention layers of the encoder part of the model and not also the decoder part as I did with the GIT model. This might explain its poor results.
The dataset sometimes has very similar clips that have slightly different captions, for example sentences containing "attending", "keeping still" or "sensing its environment" often occur so getting a perfect score seems impossible.
There was also a problem mentioned in the data pre-processing, where too short clips were extended. This was done by extending the cut clip on both sides. So there might be some cases where the model is fed a frame from from outside its clip range, although I do think that this is only the case a handful of times since most clips were of sufficient length.

5 Conclusions and Further Research

How does training newly introduced parameters, using LoRA and AdaptFormer, affect a frozen pre-trained video captioning model in transferring from human to animal data?

The AdaptFormer model where all MLP and Intermediate layers were replaced performed better for the GPT model and equivalent for the GIT model compared to the regular full fine-tune. So the performance of the transfer here was better or at least equivalent, while training fewer parameters. For the LoRA models, they can do a lot with very few parameters. Outperforming the AdaptFormer models when both models train few parameters. With BLEU4 scores for LoRA of around 18 and 12 for the GIT and GPT models, compared to 12 and around 9 for AdaptFormer. However, when both models add and train more parameters AdaptFormer performs significantly better. The LoRA models were not able to outperform the regular full fine-tuned models. The GIT model did manage to outperform the partially fine-tuned model. However, LoRA might have performed better if it was given more layers instead of more parameters in the same layer, since increasing its dimension changed very little in performance. The problem here was likely that too few layers had LoRA weights in the first place.

Overall, AdaptFormer allowed for a better transfer for both models, though LoRA's performance might be improved upon with different configurations.

How does partial fine-tuning of a pre-trained model affect the performance of video captioning models in transferring from human to animal data?

The full fine-tune of both models outperformed the partial fine-tune, this is expected. For the GPT model, the partial train is not far behind on any metric. Both the full and partial fine-tune often get the animal wrong. Generally, the majority of the metric points appear to be obtained by getting the context or action correct so for the GPT model I would say that the partial fine-tune is just as good as the full fine-tune.

The fully fine-tuned GIT model generally does get the animal name correct which also reflects in its metrics, the highest of all experiments. However, the partial fine-tune performs significantly worse when comparing the resulting metrics. So for the GIT model I would say that the partial fine-tune, when compared to the full fine-tune, had a negative effect.

Do video captioning models transfer well from human to animal data?

I do no think that the models that I worked transfer well from their initial human data from the VATEX dataset to animal data from Animal Kingdom dataset. This shows from the poor zero-shot performance, and from the fact that the best performing models are the fully fine-tuned ones and the AdaptFormer with a large mid dimension; both of these changed/added a large amount of parameters. This is probably because the captioning human data task is too different from captioning animal data. This can be because the actions and actors are too dissimilar. The animals are likely never seen before and thus need to be learned from scratch. And the actions that the model was used to seeing were human actions, and do not occur in the Animal Kingdom dataset. A few example could be "attending", "keeping still" or "sensing its environment", these are very different to any of the actions in the VATEX dataset. Additionally the Animal Kingdom dataset is

not always consistent with how it captions its videos. For example, sometimes it uses a broad term for an animal like snake, and other times it uses its scientific name like "atheris squamigera". This also could have negatively impacted the transfer.

Further Research

Firstly, the performance of LoRA in my case was lower than expected. And like stated before, trying to apply LoRA with different configurations might give significant improvements. Specifically introducing more LoRA layers with a lower dimension, something like adding LoRA layers to all layers in the second half of both encoder as decoder with dimension 4 or 8 could be interesting.
Furthermore, adding preprocessing to the dataset and generalizing animal species to more basic classes to see how it impacts the model's ability to identify the actions correctly could be another direction of research. Another possibility could be focusing on the animal instead of its action. The loss can be calculated differently to take in account class imbalances caused by a long-tail distribution of animals as described here [CJL+19]. Calculating this class balanced loss could help the model deal with potential data imbalances.

References

[BL05] Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Jade Goldstein, Alon Lavie, Chin-Yew Lin, and Clare Voss, editors, *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics.

[BWT21] Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *International Conference on Machine Learning*, pages 813–824. PMLR, 2021.

[CGT+22] Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition, 2022. https://arxiv.org/abs/2205.13535.

[CJ19] Shaoxiang Chen and Yu-Gang Jiang. Motion guided spatial attention for video captioning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:8191–8198, 07 2019.

[CJL+19] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples, 2019.

[HSW+21] 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. https://arxiv.org/abs/2106.09685.

[Lin04] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[NOZ+22] Xun Long Ng, Kian Eng Ong, Qichen Zheng, Yun Ni, Si Yong Yeo, and Jun Liu. Animal kingdom: A large and diverse dataset for animal behavior understanding, 2022.

[PRWZ02] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[VSP+23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. https://arxiv.org/abs/1706.03762.

[WWC+20] Xin Wang, Jiawei Wu, Junkun Chen, Lei Li, Yuan-Fang Wang, and William Yang Wang. Vatex: A large-scale, high-quality multilingual dataset for video-and-language research, 2020.