Transformers for Dark Matter Morphology with Strong Gravitational Lensing

~ Archil Srivastava

1. Introduction

Since Dark Matter was discovered, physicists have been trying to understand its composition. Several models have been proposed for this, the most popular one being the Standard Model e.g. Weakly Interacting Massive Particles (WIMPs) [8] and Axions[6]. However, these have so far evaded detection, both by direct detection and colliders. These models also possess varying degrees of unexplainable flaws. Another paradigm is to consider condensate models using vortex substructure[7] of Dark Matter.

In practice, the best method to detect substructure is from strong gravitational lensing images. Given the huge success of deep learning algorithms in computer vision across various fields, it seems a promising path to use these algorithms on gravitational lensing images to classify the Dark Matter substructure. Attention-based networks[3] like Transformers[9] started as a breakthrough in Natural Language Processing and were quickly modified to successfully work on Computer Vision tasks as well, where they're called Vision Transformers (ViT)[4]. So far they have outperformed the previous computer vision tools like Convolutional Neural Networks (CNNs)[5].

DeepLense is a deep learning pipeline for particle dark matter searches with strong gravitational lensing. Deep learning methods like CNNs[2] and Variational Autoencoders[1] have already been implemented in this pipeline. This project will focus on implementing Vision Transformers in the DeepLense pipeline, which is expected to immensely boost the current performance of the system.

2. Background

What is attention?

The problem with machine translation using RNN encoder-decoder pair was that the whole information of the input sentence was bring compressed into a single context vector (Fig 1), leading to poor performance on long sentences. The attention mechanism solves this by dynamically highlighting the relevant features of the input data (Fig 2).

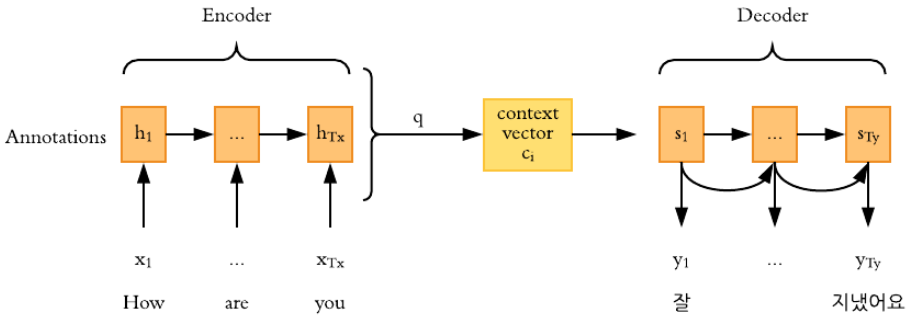

Fig 1 Machine Translation using RNNs compresses too much information into a single context vector*

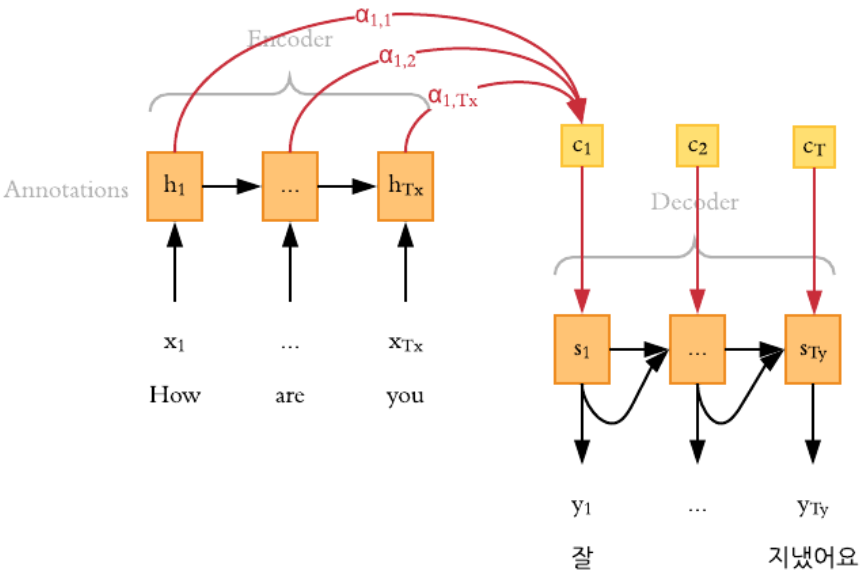

Fig 2 Attention dynamically highlights relevant input features for each output word*

What is self-attention?

Contrary to attention, which transfers better context information between different sequences, self-attention transfers context information within the same sequence. Hence, even if the decoder is missing, self-attention can operate on the encoder alone to produce better encodings.

* Images picked from https://medium.com/@edloginova/attention-in-nlp-734c6fa9d983

What is a transformer?

A transformer is a self-attention based network. It uses a 'Scaled Dot Product' attention mechanism that used query, key, and value triplets to achieve a better weightage of context and output. It also uses what is called 'Multi-Head Attention', which allows the model to attend to different 'representation sub-spaces' at different positions, similar to using different filters to create different feature maps in a CNN layer.

What is Vision Transformer?

Vision Transformer (Fig 3) is the adaptation of transformers to images, where it divides each image into patches (similar to words in NLP), and uses these patches to preserve the context in deeper layers. Vision Transformer uses only the encoder part of the original Transformer and outputs a single classification probability at the end.

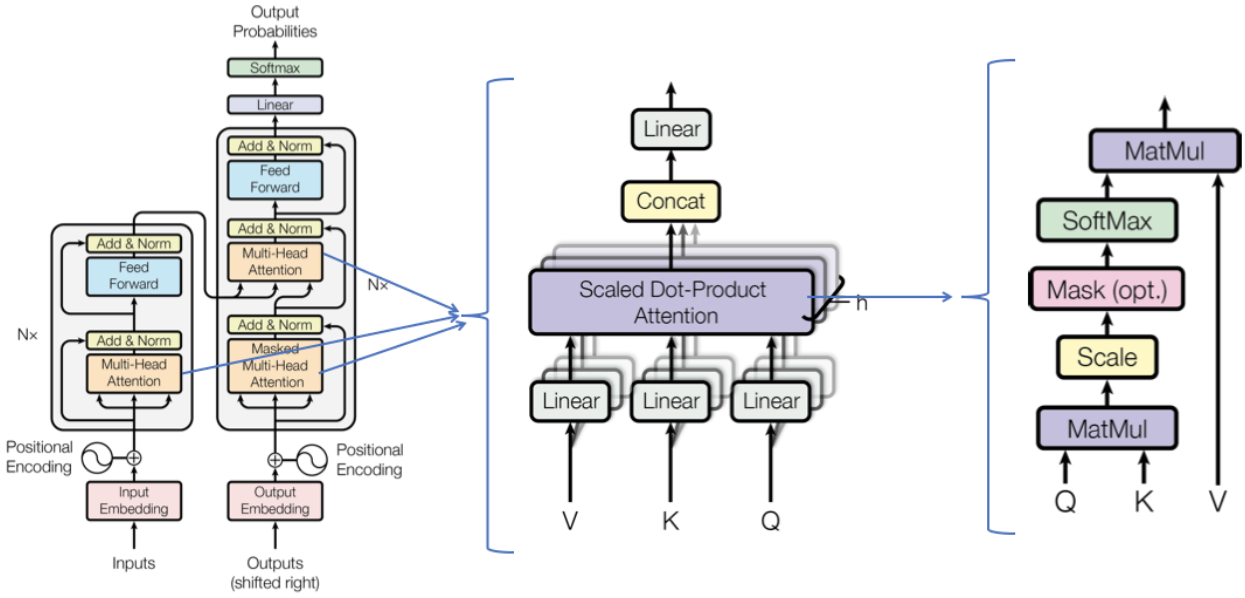

Fig 3: Architecture of Vision Transformer from "Attention Is All You Need" by Vaswani et al. [9]

3. Proposed Work

The project can be broken down into the following four deliverables:

a. Exploring and understanding the dataset (Real and Lenstronomy)
b. Implementing the Vision Transformer (ViT)
c. Optimizing and tuning the ViT implementation to work with real data.
d. Documenting, Testing, and Benchmarking the implementation.

a. Dataset Exploration

The first part of this phase would focus on getting a high-level understanding of the existing DeepLense codebase to get a better context. This will also help in quickly finalizing an initial high-level design of the project.

The second (and major) part would focus on carrying out exploratory data analysis on the Real (if available) and Lenstronomy datasets. This will help in preprocessing the data to achieve higher accuracy with the model.

b. Implementing the ViT model

The initial phase would focus on designing and implementing an end-to-end pipeline with a crude ViT model. This is to ensure that the code and data flow are correct. This will also include setting up any metrics or logs necessary to study the behavior of the ViT. Once this is done, we'll be able to focus better on the actual task at hand i.e. the Vision Transformer.

After this, the focus would be to build an efficient and extendable ViT implementation. While I'm comfortable building the new implementation in either TensorFlow or PyTorch, I feel we can go ahead with PyTorch because the existing DeepLense codebase[10] seems to be built on top of PyTorch. (However, I believe that Tensorflow 2 may prove to be more robust in a production environment if we aim to go for something like a real-time pipeline.)

c. Tuning and Optimization

This phase would involve optimizing and fine-tuning the ViT model to achieve state-of-the-art results on the Lenstronomy dataset (and also real data if available).

This phase would also involve getting rid of any computational bottlenecks in the code (using a profiler) and refactoring the code for better performance and extendability.

d. Testing, Documentation, and Benchmarking

The last stage would focus on testing, documenting, and benchmarking the codebase. The benchmarking over here would focus on understanding how much memory and time the model requires for standard training and inference tasks.

4. Deliverable, Timeline, and Conflicts

The project would last about 175 hours, and I aim to spend at least 20-25 hours per week on the project, spread evenly throughout the week. I can spend roughly 4 hours every day (including weekends) on project-related work.

On a high level, I aim to spend the first week understanding the existing codebase and the dataset. Then, in the next three weeks, I aim on wrapping up the preliminary implementation of the whole ViT pipeline. Thus, the first two deliverables specified in Section 3 should be completed before the first evaluation.

I would spend the next three weeks focussing on fine-tuning the implementation to achieve state-of-the-art results and removing any bottlenecks in the code. During this time, I'll also refactor the code, and write training/evaluation scripts and configs for improving the usability and extendability of the pipeline.

The next three weeks would be spent on testing, documenting, packaging the code, and working on any unforeseen tasks.

Deliverables for first evaluation (July 25):

- Exploratory data analysis of the dataset
- Code to preprocess the data
- Working implementation on the ViT with benchmarks on the standard datasets

Deliverables for final evaluation (September 5):

- Code to adapt the developed pipeline to real data (if available).
- A baseline model that demonstrates a proof of concept.
- Fine-tuned model and optimized code with benchmarks on performance.
- Tests and documentation.
- Packaged code for plug-and-play use in other pipelines.

Overall Project Deliverables

- A Python module implementing Vision Transformers (ViT).
- Training/evaluation scripts to train this ViT model on gravitational lensing dataset.
- Pre-trained, ready-to-use ViT models for other pipelines using lensing datasets.

Week	Objective
Community Bonding Period	- Discuss and refine the proposal with DeepLense mentors - Conduct a thorough literature survey of existing methods
Week 1 [June 13 - June 19]	- Going through the codebase and exploring dataset - Preliminary data analysis and preprocessing
Week 2 [June 20 - June 26]	- Write an initial boilerplate code. - Start working on the ViT implementation
Week 3 [June 27 - July 3]	- Get a working model ready in PyTorch - Start working on other parts of the pipeline (like metrics, plots, benchmark)
Week 4 [July 4 - July 10]	- Rearrange the code into a modular format - Bulk of the implementation would be ready by this week.
Week 5 [July 11 - July 17]	- Complete any remaining tasks to make the pipeline ready - Start removing bottlenecks from the code - Start fine-tuning the model.
Week 6 [July 18 - July 24]	- Finish removing all bottlenecks. - Write training/inference scripts. - Fine-tune the hyperparameters of the model further (try out different techniques from literature)
Evaluation 1	

Week 7 [July 25 - July 31]	- Train the model on real data (if available) - Wrap up fine-tuning the model and report results (could continue experimenting further)
Week 8 [Aug 1 - Aug 7]	- Refactor code - Profile the code for any lingering bottlenecks - Start working on documentation and writing test cases
Week 9 [Aug 8 - Aug 14]	- Wrap up test cases and documentation. - Package the code for a production-ready system. - Compile all results - Document the work done over the past 9 weeks
Week 10 [Aug 15 - Aug 21]	- Buffer for wrapping up any remaining task. - Look into future possibilities
Week 11 [Aug 22 - Aug 28]	- Publish the code - Write a blog detailing the whole process - Prepare slides and supplementary material for the future users/contributors of this project.
Week 12 [Aug 29 - Sep 4]	- Final touch-ups to the codebase before final evaluation. - Exploring further research prospects with mentors
Final Evaluation	

Time Commitments

I'm currently pursuing a Master's degree in Computer Science at University of California San Diego. I'll be working with ML4Sci during my summer break which starts from mid-May and goes up to late September. I have no other commitments during this time and will be able to devote my maximum time to the project.

As stated earlier, I am confident that I can comfortably devote at least 20-25 hours per week to the project.

5. About Me

Name	Archil Kumar Srivastava
Email	[University] asrivast@ucsd.edu [Personal] archil.kumar@gmail.com
Phone	+1 858 257 8178
Location	San Diego, USA
Timezone	GMT -7:00 (PDT)

School and Degree	Master's in Science, Computer Science UC San Diego (Expected: Spring'23)
Evaluation Tasks	https://github.com/archilk/ml4sci-gsoc22/tree/main/deeplense
Links	[CV] [GitHub] [LinkedIn]

Background

Education:

- **Bachelor's** in **Computer Science** from **Indian Institute of Technology Jodhpur**
- **Master's** in **Computer Science** from **University of California San Diego** (ongoing)

Programming Languages:

- Primary languages: Python, Java, JavaScript
- Also comfortable with: C++, MATLAB, MYSQL

Operating System:

- Currently using MacOS
- 6+ years of experience with Linux (command line and scripting)
- Thorough experience with using remote GPU clusters and GCP instances during my research here at the Nemati Lab UC San Diego.

Relevant Experience:

- **Extensive experience with Machine Learning** (both supervised and unsupervised) and Deep Learning, through coursework and **research projects**.
- **3 years of industry experience** as a Senior Software Developer at D.E. Shaw India Pvt. Ltd.
- **Strong theoretical knowledge** and intuition of mathematics, data structures, and algorithms.
- Recently submitted an article (under review) to the AMIA 2022 Annual Symposium analyzing the performance of deep learning models on various regions of the high dimensional space of clinical data.
- Experience with Computer Networks through an undergraduate project at IIT Jodhpur.

ML4Sci is the only organization that I am applying to for GSoC'22 as it aligns with my research interests. I would also like to pursue further research/projects with the organization even after GSoC'22 gets over.

Bibliography

[1] Alexander, S. et al. 2021. Decoding Dark Matter Substructure without Supervision. *arXiv:2008.12731 [astro-ph, physics:hep-ph]*. (Sep. 2021).

[2] Alexander, S. et al. 2020. Deep Learning the Morphology of Dark Matter Substructure. *The Astrophysical Journal*. 893, 1 (Apr. 2020), 15. DOI:https://doi.org/10.3847/1538-4357/ab7925.

[3] Bahdanau, D. et al. 2016. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv:1409.0473 [cs, stat]*. (May 2016).

[4] Dosovitskiy, A. et al. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at

Scale. *arXiv:2010.11929 [cs]*. (Jun. 2021).
[5] Krizhevsky, A. et al. 2012. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems* (2012).
[6] Preskill, J. et al. 1983. Cosmology of the invisible axion. *Physics Letters B*. 120, 1 (Jan. 1983), 127–132. DOI:https://doi.org/10.1016/0370-2693(83)90637-8.
[7] Rindler-Daller, T. and Shapiro, P.R. 2012. Angular Momentum and Vortex Formation in Bose-Einstein-Condensed Cold Dark Matter Haloes. *Monthly Notices of the Royal Astronomical Society*. 422, 1 (May 2012), 135–161. DOI:https://doi.org/10.1111/j.1365-2966.2012.20588.x.
[8] Steigman, G. and Turner, M.S. 1985. Cosmological constraints on the properties of weakly interacting massive particles. *Nuclear Physics B*. 253, (Jan. 1985), 375–386. DOI:https://doi.org/10.1016/0550-3213(85)90537-1.
[9] Vaswani, A. et al. 2017. Attention is All you Need. *Advances in Neural Information Processing Systems* (2017).
[10] 2022. *DeepLense*. https://github.com/ML4SCI/DeepLense.