Deep Regression Techniques for Decoding Dark Matter with Strong Gravitational Lensing

Personal Details

Name: Yurii Halychanskyi
University: University of Washington, Seattle, WA, United States
Email: renessmi2018@gmail.com
Phone: +13603564387
Github: https://github.com/Claussss
LinkedIn: https://www.linkedin.com/in/yurii-halychanskyi-a57590169/
Country of Residence: United States
Timezone: Pacific Standard Time

I am a junior Computer Science student at the University of Washington, Seattle, WA. My semester ends on Tuesday, May 31, therefore I will be able to work only part-time (~25 hours per week) the first week of the internship. Once the quarter ends, I will be able to dedicate 40+ hours per week to Google Summer of Code.

Technical Knowledge

I am a junior Computer Science student at the University of Washington. My main focus is Data Science, so I have already completed some classes that would help me to contribute to this project. Namely, I have completed Foundation of Computing 1 (discrete math, formal proofs), Foundation of Computing 2 (stats, probability theory), Data Structures and Parallelism, and I am currently taking a Machine learning class. I think those classes will help me to understand the theoretical part of the project better, which will lead to deliberated decisions from my side.

Besides my coursework, I have already completed a few internships in Data Science and Computer Science fields. In the summer of 2020, I participated in the program called FellowshipAI, where I worked on different data science problems for 3 months. Namely, I performed the following tasks:

- Implemented YOLOV3 for person re-identification on CCTV cameras. The model was able to identify people with the accuracy of 85%.

- Implemented an OCR algorithm for automatic analysis of workforce meeting records. The model recognized the name of the user to identify the current speaker.
- Researched, modeled, and benchmarked classifiers for the prediction of food doneness based on the images outputted by a camera embedded into an oven.

Also, I have worked for a year in SEH America, where I developed Windows Forms applications for operators to use. I performed the following task:

- Created a reporting system in PBI, which increased the reporting efficiency of the department and was able to automatically update the report with the data from the database.
- Created a C# Windows Form application for checking the condition of machines, which made the process less error-prone and organized the flow of the data to the database.
- Created an AutoHotKey script that automated the process of changing the parameters of machines.

In addition to that, I have already participated in the Google Summer of Code at ML4SCI organization in the summer of 2021. I implemented a modified version of ResNet to regress the axion mass of high-resolution images of galaxies in deep space. Also, I modified the strong lensing images simulator by adding parallelism, which reduced the image generation time by 4 times. Furthermore,I have participated in the following Kaggle competitions:

- “Catch Me If You Can” Kaggle Competition (anomaly detection on time series)
- Lyft Motion Prediction for Autonomous Vehicles (real-time segmentation, classification, and regression techniques to predict the motion of other cars)

Project

Previous work

Currently, DeepLense uses a specific type of ResNet that is trained on 25k of strong lensing simulations, which are grayscale images with the size of 150x150. CNNs are a good choice for extracting features from images, which are usually high dimensional, and reducing the number of parameters of the model. The ResNet architecture helps to speed up the learning rate of the model by allowing the model to train fewer layers in the initial stages of learning. The type of ResNet used as a part of the the current DeepLense pipeline is called XResNetHybrid. It

utilizes a novel self-regularized non-monotonic activation function called Mish, which can be mathematically defined as

$$f(x) = x \tanh(\frac{1}{\beta} \ln(1 + e^{\beta x}))$$

Mish provides better accuracy, lower loss, smoother and easy-to-optimize loss landscape compared to both Swish and ReLU. Also, the modified model contains some self-attention layers with CNNs, which allows using global information in addition to the local pixel information when performing the convolution operation. This approach is based on the covariance between a predicted pixel and every other pixel, where each pixel is considered a random variable.

Constraints of CNNs

Thanks to weights sharing, the features extracted from a convolution layer are translation invariant; they only determine whether the feature is present or not because they are not sensitive to the global position of the feature. Also, because of the nature of the convolution operator, the features extracted from a convolution layer are locality-sensitive, meaning every operation takes into account only a local region of the image. CNNs are great at extracting visual features but they are not able to model the dependencies between them. Large receptive fields are required in order to track long-range dependencies within an image, which in practice involves large kernels or long sequences of convolutional layers at the cost of losing efficiency making the model extremely complex, even impossible to train. Especially, we can observe this problem when regressing on images sampled from a new dataset, which is much harder because every image has 3 channels. The current model has a high bias on this new dataset.

How Transformers Can Help

Although we already use a self-attention mechanism in our model, using transformers will speed up the training process and allow us to train the model on higher-dimensional data (like images with 3 channels). Attention layers can directly replace the convolutions, then they will be able to attend to a larger receptive field than regular convolutions, hence being able to model dependencies between spatially distant features. The most basic approach consists of flattening the spatial dimensions of the feature map into a sequence of features with the shape HxWxF, where HxW represents the flattened spatial dimensions, and F represents the depts of

the feature map and uses self-attention directly over the sequence to obtain the updated representations.

Equivariant Transformers (ET)

The lensing signature for the vortex looks like a line, so to expand the dataset, we used to use some data augmentations like flips and rotations that would not change the properties of the line. Manual augmentation takes additional time when feeding batches to the model, also it allocates some memory, which could we used to increase the batch size, for example. To improve the performance of the regression pipeline, we could use an equivariant model. For instance, an equivariant transformer (ET) is a differentiable image-to-image mapping. All transformed versions of a base image are mapped to the same output image. The transformations are invertible, and differentiable with a real-valued parameter (it includes many common families of transformations, like translation, rotation, scaling, shear, perspective, etc). Canonical coordinates let us tailor Equivarient Transformer layers to specific transformation groups, and the image-to-image interface lets us compose ETs to handle more complicated transformation groups.

Scaling Up

The current regression pipeline cannot handle a dataset that does not fit into the RAM. It uses NumPy arrays to load and pass the data to the model. One of the ways to improve the pipeline is to use Numpy memory-map files. They are used for accessing small segments of large files on the drive, without reading the entire file into memory. Although, this subclass of ndarray has some unpleasant interactions with some operations because it does not quite fit properly as a subclass. Alternative for it will be using Dask arrays. They coordinate many NumPy arrays arranged into a grid. These arrays may live on disk or on other machines (clusters). Dask’s schedulers scale to thousand-node clusters, and it uses lazy initialization, which means that the computations will be performed only when we need them. Such an approach will speed up the DeepLense pipeline and allows us to generate bigger datasets.

Timeline

Pre GSOC

Learn more in-depth theory about strong lensing and different kinds of WIMP. Practice simulating different kinds of strong lensing images by using PyAutoLens.

Community Bonding

Discuss the ideas offered in the proposal with other DeepLense contributors. Maybe some of the approaches can be modified.

Week 1

Rewrite the data pipeline using Dask arrays to be able to feed bigger datasets into the model.

Week 2

Explore the performance of the current CNN model on the new dataset, and see what kind of error dominates (variance, bias, or noise) to be able determine what is needed to be modified in the current pipeline.

Week 3-5

Choose and implement a type of the Equivariant Transformer. Try a combination of the transformer and the CNN.

Week 6

Tune algorithms hyperparameters.

Week 7-8

Improve training loop (expand dataset, add augmentations, tune training hyperparameters, add Pytorch schedulers, experiment with optimizers).

Week 9

Edit the model based on the feedback from the community; document the code, and clean repositories.

Week 10

Get/give final feedback.