Hands-on

Ambre Visive, Karel de Vries, Liza Cherepanova

Meeting of Nikhef ML/AI group: Transformers

31 January 2025

Overview

4 notebooks:

- *Transformer.ipynb*
  - Classification task: distinguish ‘tZq’ from ‘ttZ’ states

  → We will go through this notebook all together

To solve a classification problem (common in various areas including particle physics)

- *Arithmetic.ipynb*
  - Predict the next number in arithmetic sequence
  - Learn how to visualise the output
- *Geometric.ipynb*
  - Predict the next number in geometric sequence
  - Do the batch training
- *Fibonacci.ipynb*
  - Predict the next number in Fibonacci sequence
  - Learn how to do a proper validation

To get the idea how to build your own transformer

How to run the notebooks

- Locally:
  - Run it with your favorite editor, e.g. Visual Studio Code
  - See the instruction on the slide 4

Way faster for Arithmetic.ipynb, Geometric.ipynb, Fibonacci.ipynb

- On Nikhef cluster Callysto:
  - Nikhef account is required
  - See the instruction on the slide 5

Easier (specially if you are used to stbc)

- On Google Colab:
  - Run everything in a browser
  - Google account is required
  - See the instruction on the slide 6

How to run locally

- Clone the code

  ```
  git clone https://gitlab.nikhef.nl/avisive/transformer-tutorial.git
  ```

- Or copy the directory from stbc

  ```
  scp -r username@stbc-i1.nikhef.nl:/project/atlas/users/kdevries/Workshop_Transformer/transformer-tutorial .
  ```

- Create two virtual (or conda) environments. One for TensorFlow (for Arithmetics .ipynb, Geometric .ipynb and Fibonacci .ipynb), another for PyTorch (Final_State_Transformer/Transformer.ipynb):
  - `python3 -m venv [venv_name]`
  - `source [venv_name]/bin/activate`
- Install the requirements:
  - `python3 -m pip install -r requirements_notebook.txt`
  - `python3 -m pip install -r Final_State_Transformer/requirements.txt`

For Arithmetic.ipynb, Geometric.ipynb, Fibonacci.ipynb

For Transformer.ipynb,

- Run it with your favorite editor, e.g. Visual Studio Code
  - You might need to install Jupiter kernel

    ```
    pip install ipython ipykernel
    ```

How to run on Callysto

- Go to the https://callysto.nikhef.nl
- Login through Nikhef SSO
- You will be in your user directory on stbc (e.g. /user/echerepa/)
- Open terminal (File -> New Launcher -> Terminal)
- Get the repository, there are 2 options:
  - Clone from the GitLab repo:

    ```
    git clone https://gitlab.nikhef.nl/avisive/transformer-tutorial.git
    ```

  - or copy from stbs:

    ```
    cp -r /project/atlas/users/kdevries/Workshop_Transformer/transformer-tutorial .
    ```

- The files will appear in the panel on the left side
- Click on the notebook to run it

How to run on Google Colab (1)

- Go to the git repository: https://gitlab.nikhef.nl/avisive/transformer-tutorial/-/tree/main?ref_type=heads
- Download the code: Code → **Download source code**
- Open Google Colab: https://colab.research.google.com/
  - Log in to your Google account if you are not yet logged in
- In the appeared window select 'Upload' and select a Jupiter notebook: *Arithmetic.ipynb*, *Geometric.ipynb*, *Fibonacci.ipynb* or *Final_State_Transformer/Transformer.ipynb*
- For *Transformer.ipynb* you will need to upload some files:
  - Go to the Files icon on the left
  - Upload all the files from the *Final_State_Transformer* folder
- You are all set to run the code! ☺

How to run on Google Colab (2)

By default Google Colab runs on CPU, but it is possible to run on GPU:

- Go to Runtime → Change runtime type
- Select T4 GPU

NB! Changing the runtime will delete all the files in the temporary directory and you will need to upload them again!

Tasks to do

- *Arithmetic.ipynb*
  - see the model predict the next number of your own sequence
  - change the num_epochs and compare the visual representation of the output
  - train with different common_difference and/or different maximal length of the sequence (n_max)
  - give a test sequence longer to anything that the model has seen before. Can it still predict?
  - try to make it predict a number outside of its library (bigger than the input size)

- *Geometric.ipynb*
  - compare the loss function per epoch to the one obtained without batch (with arithmetic.ipynb)
  - change batch size to try to see the impact on training
  - change the num of epochs, batch number and batch size and compare the visual representation of the output
  - train with different common_difference and/or different maximal length of the sequence (n_max)
  - give a test sequence longer to anything that the model has seen before. Can it still predict?
  - try to make it predict a number outside of its library (bigger than the input size)

- *Fibonacci.ipynb*
  - change batch size to try to see the impact on training
  - change the num_epochs, batch_size and compare the visual representation of the output and the loss functions
  - try to train with different ratio between training and testing
  - how good do you manage to make your model while changing the ratio between training and testing data, the number of epochs, the batch size...?