Overview of the code for 1907.03676

1 Introduction

This document consists of a description of the different files.

2 General comments

- The code was written for Python3 use.
- All units of energies are in GeV.
- All constants were chosen to match with those used in 1711.08437.
- The code is written for the diphoton channel. Changing it to the dilepton channel would be very easy. It also assumes a constant acceptance times efficiency.
- The code is an adaption of the actual code that was used to write the paper. Functions that were not used were deleted and scripts that were meant for implementation on our cluster were removed. All code designed to scan over large parameter spaces was also deleted.
- Notable required Python libraries include numpy, scipy, parton, pywt, keras, sklearn and tensorflow.
- For the pywt package, there is variable called "precision" in the "_cwt.py" file that can be changed. Depending on the signal, it's possible that its default value is too low which can result in a scalogram that appears somewhat grainy.
- The code includes a series of examples. These are the "main files" that combine all the source files to produce the final results. Each example file has a different purpose explained below. The structure is as follow:
  - Signal generation
    - A_ExampleA.py: Example of the event generation
  - Classifier
    - A_ExampleB.py: Example of the classifier to obtain constraints
  - Autoencoder
    - A_ExampleC1.py: Example of the autoencoder trainer
    - A_ExampleC2.py: Example of the autoencoder to obtain constraints
  - Region finder
    - A_ExampleD1.py: Example of the generation of many signals
    - A_ExampleD2.py: Example of the region finder trainer
    - A_ExampleD3.py: Example of the region finder to obtain constraints
  - Windowed Fourier transform
    - A_ExampleE.py: Example of the windowed Fourier transform analysis

3 Signal generation

The files described in this section are related to our generation of the signal. It seems that you will not need them, but we decided to include them to show how we implemented the neural networks. The files are presented in order of dependency and their order corresponds to the authors recommendation for reading them if the need were to arise.

3.1 Couplings.py

This is just a list of different known physical constants and some parameters related to the experiment. A definition of each of them is provided in the code directly. The file also contains some options for the data presentation.

3.2 Basic.py

This code contains functions to compute the most common functions of the CW/LD. A definition of each of them is provided in the code directly. It also contains functions that compute the volume of phase-space for the decays of the gravitons to SM particles.

3.3 KKcascade.py

This code contains a series of functions necessary for computing the decay widths of the gravitons to BSM particles.

3.4 PDFcalc.py

This code contains a function to compute the parton distribution functions (PDF). It uses the parton library.

3.5 Luminosity.py

This code contains functions to compute the luminosity associated to different processes taking into account the PDFs.

3.6 Graviton.py

This code computes the properties of a given graviton like cross section, branching ratios and decay widths. Some branching ratios are included for convenience but were never used. The contribution of decays to pairs of KK scalars is neglected to save computation time. Similarly, only the dominant contribution of the first generation quarks were incluced in the cross section computation.

3.7 Smearing.py

This code applies some rudimentary smearing based on 1711.08437.

3.8 Background.py

This code computes the background of the analysis.

3.9 SBLD.py

This code contains functions that can compute the exact signal and generate either backgrounds or signals with statistical fluctuations. With the proper options selected in Couplings.py, it will also produce some figures and a summary file.

3.10 Merger.py

This code contains a function to merge together bins in a scalogram to reduce the total number of bins. Signals with high frequency require bins to be small in mass space. This can result in a scalogram with a number of elements too high to practically serve as the input of a neural network. This function addresses this issue.

3.11 WT.py

This code contains a function to return the CWT of a given input. It uses the Morlet wavelet. It's possible to have it remove the cone of influence if wanted, but the option is turned off by default. The scales considered are equally spaced on a logarithmic axis such that there are 12 bins between two scales differing by a factor of two. This is to keep consistent with the Mathematica 10 convention, which we used to cross check some results.

3.12 A_ExampleA.py

This is an example for the event generation. To execute it, simply type in the terminal:

```
python3 A_ExampleA.py
```

The output is:

- GravitonProperties.txt: A list of the properties of each graviton (can be disabled by turning to False the option in Couplings.py)
- Signal.pdf: A pdf figure of the signal (can be disabled by turning to False the option in Couplings.py)
- Signal.txt: The exact signal in each bin.
- SBRandom.txt: The signal + background with statistical fluctuations.
- SBRandom.pdf: A CWT of the signal + background with statistical fluctuations.
- SFix.pdf: A CWT of the signal without statistical fluctuations.

Comments:

- Parton and matplotlib might give a few warnings. These depend on the versions of packages but are warnings only and not real problems.

4 Classifier

The files described in this section are related to the classifier. This is presented first as, in a lot of ways, this is the simplest approach to implement. All files necessary for the execution of this program have already been presented.

4.1 A_ExampleB.py

This is an example for the classifier. To execute it, simply type in the terminal:

```
python3 A_ExampleB.py
```

The output is:

- ExampleB_out.txt: A file that contains the values of M_5, k, the averaged expected significance and the median expected significance.
- ZExampleB_WandB.txt: A file that contains the weights and biases of the neural network.

Comments:

- Different combinations of numpy and tensorflow can result in warnings appearing on screen.
- For the sake keeping the execution time of this program relatively short, the number of events simulated was considerably reduced. In the simulation used in the paper, 4000 pseudo-experiments were generated to train the neural network, 2000 to obtain the test statistic distribution for the background and 2000 to obtain the test statistic distribution for the background + signal. Using very low statistics can result in the neural network not learning properly and hence results that vary from one execution of the program to another.
- The batch size was also reduced for this example from 1000 to 200, which can easily be changed back.
- The neural network options include an option for the number of threads to use to perform the training. For cluster use or older computers, it might be better to lower this value.

5 Autoencoder

The files described in this section are related to the autoencoder.

5.1 LSign.py

This file contains functions related to the evaluation of local p-values. It contains both functions to generate the pseudo-experiments and a function that maps a given scalogram to the corresponding p-value map.

5.2 TestS.py

This file contains functions to evaluate the test statistics for the autoencoder and the region finder.

5.3 A_ExampleC1.py

This is an example of a trainer for the autoencoder. To execute it, simply type in the terminal:

```
python3 A_ExampleC1.py
```

The output is:

- ZExampleC_WandB.txt: A file that contains the weights and biases of the neural network.

Comments:

- Different combinations of numpy and tensorflow can result in warnings appearing on screen.
- For the sake keeping the execution time of this program relatively short, the numbers of trial background was kept to 2000 for both the p-value mapping and the training. In the paper, these were set to 5000.
- Similarly, the number of epochs was lowered from 100 to 50.
- The neural network options include an option for the number of threads to use to perform the training. For cluster use or older computers, it might be better to lower this value.

5.4 A_ExampleC2.py

This is an example of how to use the autoencoder trained in A_ExampleC1.py to obtain constraints. To execute it, simply type in the terminal:

```
python3 A_ExampleC2.py
```

The output is:

- ExampleC2_out.txt: A file that contains the values of M_5, k, the averaged expected significance and the median expected significance.

Comments:

- This programs assumes that A_ExampleC1.py was run atleast once.
- Different combinations of numpy and tensorflow can result in warnings appearing on screen.
- For the sake keeping the execution time of this program relatively short, the numbers of trial background, the number of trial background and the number of trial background + signal were significantly reduced. In the article, these numbers were taken respectively as 5000, 2000 and 2000.

6 Region finder

The files in this section are related to the region finder.

6.1 A_ExampleD1.py

This is an example of how to generate a large number of possible signals without statistical fluctuation within a given parameter space. These will be used for training the neural network. To execute it, simply type in the terminal:

```
python3 A_ExampleD1.py
```

The output is:

- ZExampleD_Signals.txt: A file that contains a set of possible signals without statistical fluctuations.

Comments:

- Different combinations of numpy and tensorflow can result in warnings appearing on screen.

6.2 A_ExampleD2.py

This is an example of a trainer for the region finder neural network. To execute it, simply type in the terminal:

```
python3 A_ExampleD2.py
```

The output is:

- ZExampleD_WandB.txt: A file that contains the weights and biases of the neural network.

Comments:

- This programs assumes that A_ExampleD1.py was run atleast once.
- Different combinations of numpy and tensorflow can result in warnings appearing on screen.
- For the sake keeping the execution time of this program relatively short, the number of epochs was reduced from 200 to 100. Plus, some of the signals (without fluctuations) were kept the same. They were all different in the paper.

6.3 A_ExampleD3.py

This is an example of how to use the region finder trained in A_ExampleD2.py to obtain constraints. To execute it, simply type in the terminal:

```
python3 A_ExampleD3.py
```

The output is:

- ExampleD3_out.txt: A file that contains the values of M_5, k, the averaged expected significance and the median expected significance.

Comments:

- This programs assumes that A_ExampleD2.py was run atleast once.
- Different combinations of numpy and tensorflow can result in warnings appearing on screen.

7 Fourier analysis

The files in this section are related to the Fourier analysis.

7.1 FT.py

This file contains a function that computes the test statistic for the Fourier analysis. It will do so for different values of the upper limit of the windowed Fourier transform.

7.2 A_ExampleE.py

This is an example of how to apply the Fourier analysis. To execute it, simply type in the terminal:

```
python3 A_ExampleE.py
```

The output is:

- ExampleE_out.txt: A file that contains the values of M_5, k, the averaged expected significance and the median expected significance.

Comments:

- Different combinations of numpy and tensorflow can result in warnings appearing on screen.