KIT-Fakultät für Physik
Institut für Experimentelle Teilchenphysik
Dr. Matthias Schröder
Jan van der Linden

Karlsruher Institut für Technologie

Teilchenphysik 2 — W/Z/Higgs an Collidern

Sommersemester 2021

Exercises No. 7 Discussion on TODO

Exercises 7: Machine learning

In this exercise sheet, we will study and train neural networks (NN). We are going to apply some common machine learning methods to a particle physics task: the separation of jets originating from top quarks from jets originating from gluons in events recorded by the ATLAS detector.

This exercise was adapted from a previous exercise[1] with courtesy from Lisa Benato. The data used in this exercise was provided by Gregor Kasieczka et al[2].

Work on that exercise sheet by using the ETP Jupyter server[3]. Login with your KIT credentials (your account with username `uXXXX`). After that choose the **TP2-WZH - Tensorflow** image. This image includes all packages that are required for executing the code. This notebook has been tested with *Python 3.8.6* and *Tensorflow 2.4.1*.

The code for that exercise sheet as well as the instructions have been prepared in a *Jupyter* notebook. You can retrieve the exercise material (notebook, data and exercise sheet) using `git`. For that follow these steps:

- If you haven't done yet clone the repository. Login to the *ETP Jupyter Machine*. Go to the folder where you want to place the new repository. Select *Git > Clone a Repository* in the toolbar on the top of the *Jupyter Lab* web interface. Now a dialog that asks for a web address should open. Enter the URL[4] of the TP2 W/Z/Higgs exercise course. **TODO: ADAPT THE REPOSITORY'S URL ON THE SHEET.**

[1] `https://github.com/dkgithub/wuhan_DL_labs/tree/master/top-tagging`
[2] `https://arxiv.org/abs/1902.09914`
[3] `https://jupytermachine.etp.kit.edu/hub/login`
[4] `https://gitlab.etp.kit.edu/XXXXXX/TP2-WZH-XXXXXXX.git`

After having pressed the *Clone* button the download of the repository starts. A new folder with the name of the repository should appear in the file browser sidebar.

- To update the repository contents go to the repository folder that you have cloned. For retrieving the recent version of the repository go to *Git > Pull from Remote* in the toolbar on the top of the *Jupyter Lab* web interface.

1. Top tagging using deep neural networks

We want to separate jets that originate from top quarks from jets that originate from gluons.

- **Task 1.1.** What is the difference between jets originating from top quarks and gluons?

The separation of these jets is a difficult task and will be explained in the lecture. In this exercise we train a *deep neural network* (DNN) for classifying jets. The DNN is trained with kinematic information of these jets. In that way it learns to differentiate between jets as originating from a top quark or a gluon. This is a supervised learning approach where we need labeled data to train the classifier.

In our case, we use jets produced with the Pythia8 parton shower for a center-of-mass energy of 14 TeV. The detector response is simulated using Delphes simulating the ATLAS detector. The jets are clustered as fat jets using the anti-k_{T} algorithm with a radius parameter of $R = 0.8$ (called *AK8-jets*). Jets are only considered in a p_{T} range of $[550 - 650]\,\mathrm{GeV}$.

Due to the confinement of color charge at low energies, particles with color charge (i.e. gluons and quarks) form bundles of color-neutral particles during the hadronization and parton shower processes after the hard scattering. These color-neutral particles are referred to as jet constituents when clustered into jets. For each jet the four-momenta of the first 200 jet constituents (sorted by momentum) are stored.

The training dataset is saved in a `HDF` file and loaded below. In this notebook the dataset is represented as `pandas` data frame. The components of the four vectors are saved in columns with names `E_i`, `PX_i`, `PY_i`, `PZ_i` where `i` is a number that runs from 0 to 199. The column `is_signal_new` is the label that tells us whether a jet originates from a top quark decay (if `signal_new` has the value 1) or from a gluon (if `signal_new` has the value `0`).

First we want to look at the distributions of some kinematic variables. All events in the training dataset are labeled. Hence we can compare the distributions of top jets and gluon jets.

- **Task 1.2.** Plot the distributions for the transverse momentum p_{T} and the invariant mass m_{inv} of the jet, calculated with the 200 jet constituents with the highest p_{T}. The code has already been implemented. Compare the distributions for jets originating from top quarks and for jet originating from gluons.

- **Task 1.3.** Look at the following distributions:

  - The four-vector components of the constituents with indices 0, 10, 30 and 70 (indicating that the jet constituent is that with the highest, tenth-highest, ... p_{T}).
  - The transverse momentum of the constituents with indices 0, 10, 30 and 70.

  Also here, the code for adding the columns to the data frame and for plotting has already been implemented. What differences do you observe for the distribution of top jets and gluon jets?

A basic implementation for training a DNN has been prepared in the notebook.

- **Task 1.4.** Familiarize yourself with how the model is built and extract the following information from the code:

  - How many layers does this network have?
  - What is the reason for inserting the so-called dropout layers between the dense layers?
  - What activation functions are used?
  - What is the purpose of the loss function and which one is chosen for this task?
  - Where are the number of training epochs and the batch size defined? What is the meaning of these two parameters?

For answering the questions you can look at the official `keras` documentation pages of the used classes and methods:

- keras.Sequential class[5]
- keras.Input class[6]

[5] `https://keras.io/api/models/sequential/`
[6] `https://keras.io/api/layers/core_layers/input/`

- keras.layers.Dense class[7]
- keras.layers.Dropout class[8]
- keras.Model methods like compile and fit[9]
- **Task 1.5.** Now perform the training. After having finished the training, plot the loss and the accuracy history.
  - What do these curves tell you about the training?
  - How could over-training (e.g. over-fitting) be observed in these plots?

Finally we want to evaluate the separation power of the DNN we constructed above. For that we use a test dataset with labeled jets. The network output value is evaluated for each jet in the dataset. Finally the distributions of the network output value for jets that are labeled as top jets and for jets that are labeled as gluon jets.

An important measure for evaluate the separation power is the *ROC-AUC value* that is equal to the area under the *ROC curve*. The ROC curve is calculated as follows: Several cut values for the network output are considered. For each cut value events above that threshold are considered to be predicted as top quark jets and events below that threshold are considered to be predicted as gluon jets. Following these assumptions the true positive rate

$$\mathrm{TPR} = \frac{\text{top jets above cut}}{\text{top jets}}$$

and the false positive rate

$$\mathrm{FPR} = \frac{\text{gluon jets above cut}}{\text{gluon jets}}$$

are calculated. Here, the true positive rate is called *signal efficiency*, while 1 minus the false positive rate is callsed *background rejection*. For each cut value these two rates form a point of the ROC curve. The ROC-AUC value is simply obtained by computing the area that is included by the ROC curve and the two axes of the coordinate system.

- **Task 1.6.** Look at the distributions of the network output value and the ROC curve. What do these curves tell you about the network's ability to separate jets originating from a top quark from jets originating from a gluon? How would you rate the discrimination power of the given network?
- **Task 1.7.** Imagine how you can improve the network architecture in order to reach a higher ROC-AUC value. Test your ideas by modifying the present code in the notebook.

[7] `https://keras.io/api/layers/core_layers/dense/`
[8] `https://keras.io/api/layers/regularization_layers/dropout/`
[9] `https://keras.io/api/models/model_training_apis/`

2. Top tagging using convolutional neural networks

We will now try a different approach to tagging these top quarks, namely image recognition. The shape of a particle physics detector like ATLAS or CMS is basically a cylinder. The surface of the cylinder can be unrolled along the radial (ϕ) and longitudinal (η) coordinates. This surface is a rectangle and can be divided into pixels. Detected jet constituents can be filled into these pixels where the energies of the constituents can be transformed into intensities.

For the purpose of this top tagging exercise jet images are provided which were already preprocessed to appear homogeneous. How these images are processed will be described during the tutorial. (Note: A single jet covers only a small part of the whole detector cylinder surface, hence needs to be cut and rotated appropriately to yield comparable jet images.)

- **Task 2.1.** Plot some jet images with the provided code. This also produces an overlay of many jet images. What differences between top jet and gluon jet images do you observe? How can you explain these differences?

Image recognition works with a special type of neural networks, so-called *convolutional neural networks* (CNN). These networks have special convolutional layers which essentially convolute the pixelated jet images with learned filters. These filters can for example learn to detect edges, or sharpen or blur the images or even detect certain objects within the full picture.

- **Task 2.2.** The notebook contains code that is needed to prepare the dataset for training and that defines a CNN architecture. What are the functions of the convolutional layer, the pooling layer and the flatten layer? Train th network and look at the evaluation plots.

  For answering the questions you can look at the official `keras` documentation pages of the used classes and methods:

  - keras.layers.Conv2D[10]
  - keras.layers.MaxPooling2D[11]
  - keras.layers.Flatten[12]

[10] `https://keras.io/api/layers/convolution_layers/convolution2d/`
[11] `https://keras.io/api/layers/pooling_layers/max_pooling2d/`
[12] `https://keras.io/api/layers/reshaping_layers/flatten/`

3. Challenge

- **Task 3.1.** Your task is to try to construct the best neural network architecture to achieve the best ROC-AUC values possible.
  - Prepare your best performing NNs (at most 5). Use the code cells at the end of the two exercises 1 and 2 that contain the functions `predict_for_cnn_challenge` or respectively `predict_for_dnn_challenge`. These functions take a model (and in the case of DNNs the list of input features), evaluate the predictions on a secret test dataset and save the predictions to a `npy` file in the folder `submission_predictions`. Insert a name for your group into the filename. If you train more than one network also insert a name. This filename will label your submitted predictions for the ranking.
  - **TODO: Inform Michael/Jan (XXX@kit.edu). Send your `.npy` files with the predicted classes to him/them via mail.**
  - The predicted labels you submitted will be compared with the true labels of the jets.
  - The winner will be announced in class on **TODO: Date of last exercise class** and will be rewarded an epic prize.