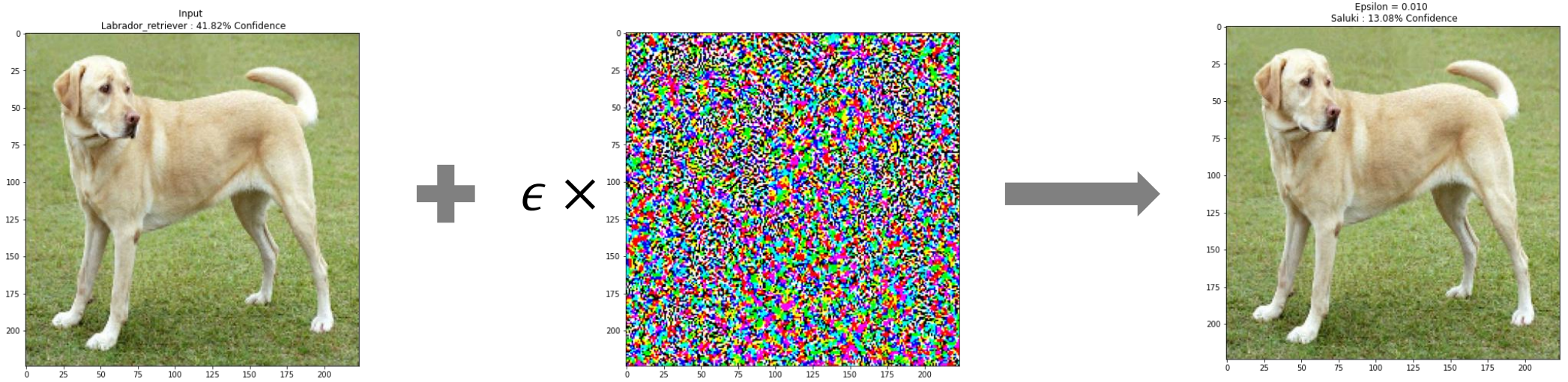
AI safety

Collection of materials to get started

- General concepts
- Technologies (deep learning, software / hardware…)
- What has been done, and some ideas for you

AI safety: example for image classification



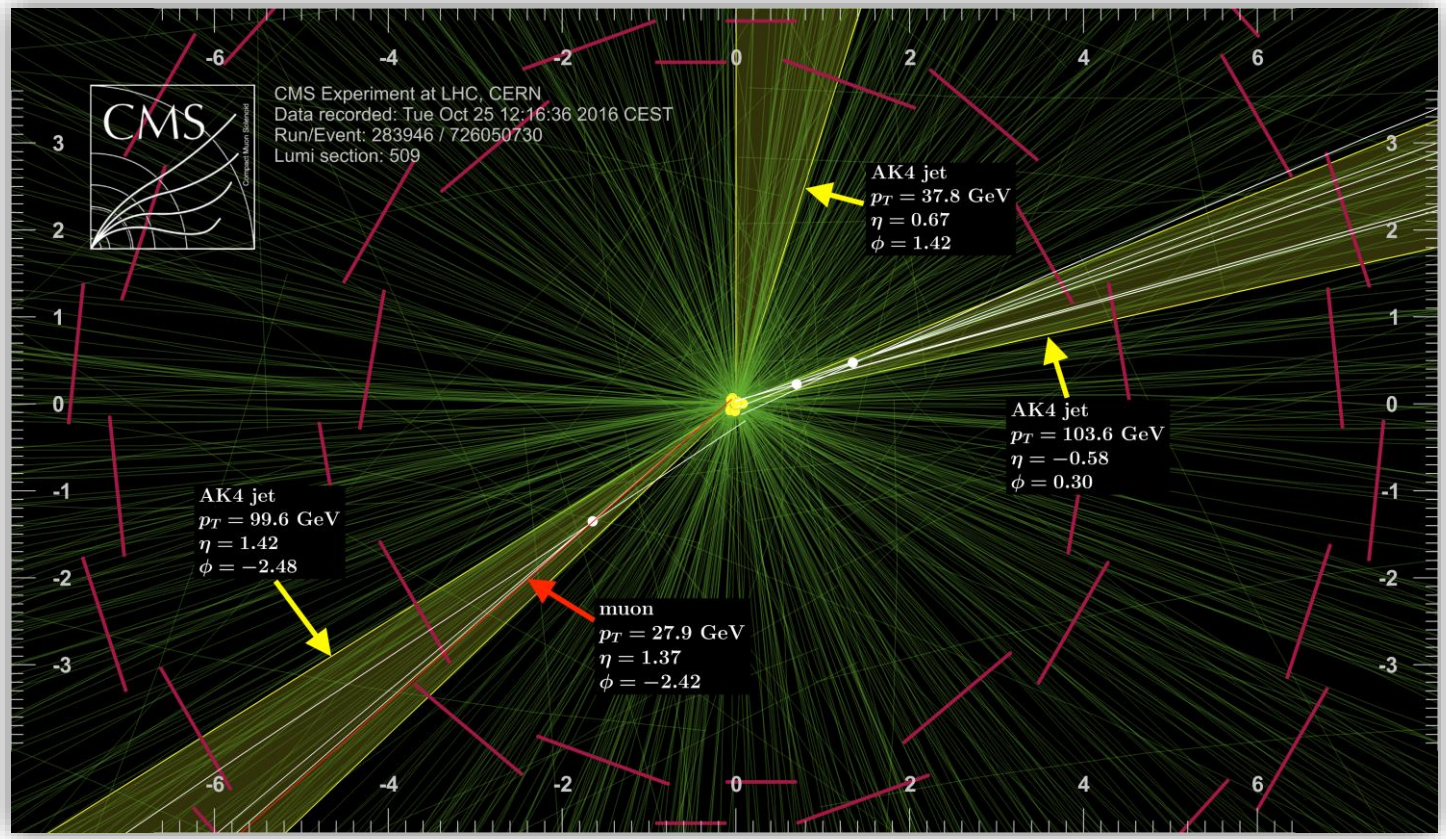
Classifier: labrador (breed of dog)

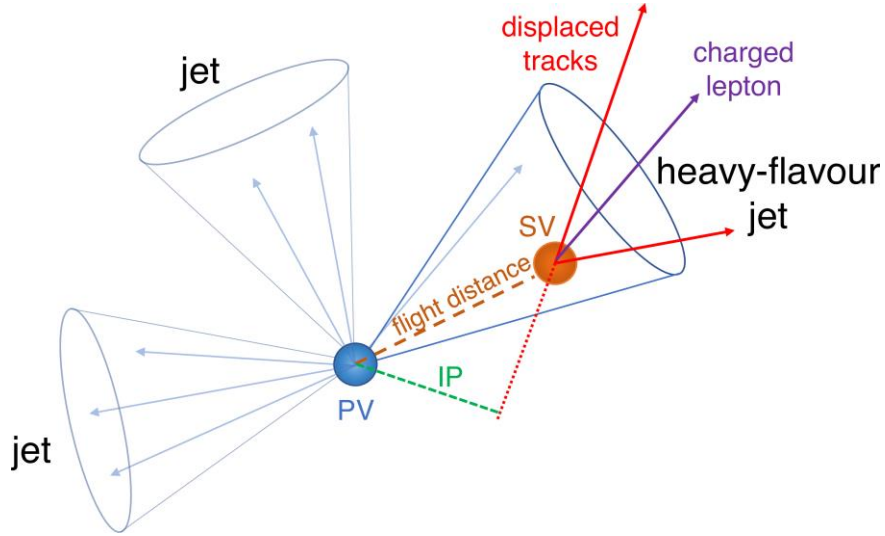
Classifier: saluki (breed of dog)
german: „Windhund“

→ Generate adversarial samples with perturbations that are not too easy to identify
→ Check their influence on the model performance

Recap: jet heavy-flavour tagging at CMS

- Goal: identify the flavour of the parton (hadron) from which the jet originates



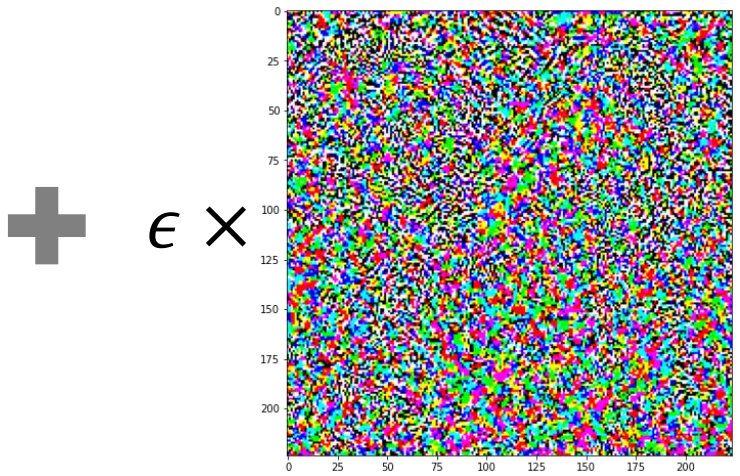


Heavy-flavour jets (b & c-jets)

- Long lifetime of b/c-hadrons → secondary vertex & displaced tracks
- Larger mass, harder fragmentation compared to light-jets
- (Soft) charged lepton in 20% (10%) of the cases for b- (c-jets)

AI safety: jet heavy-flavour tagging

Jet,
Track,
Secondary Vertex
properties of a **b-jet**



Slightly distorted
Jet,
Track,
Secondary Vertex
properties of a **b-jet**

Classifier: b-jet

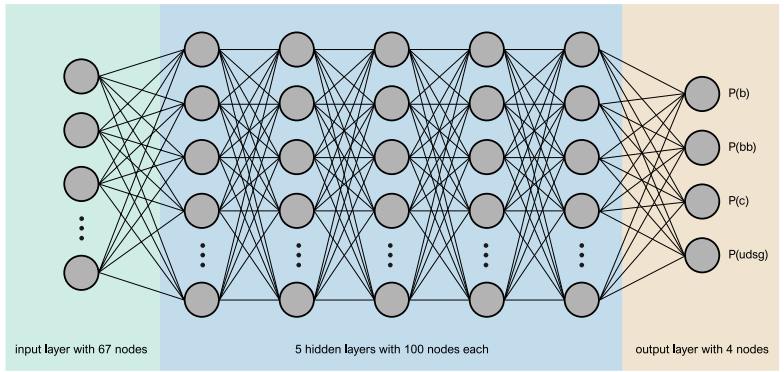
Classifier: light-jet

If one pixel alone can fool neural networks for image classification…

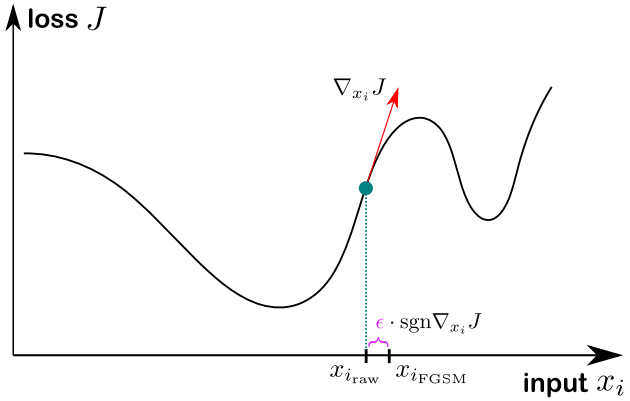
…could subtle mismodellings in our simulations cause wrong results in physics analysis?

Robustness of b-tagging algorithms: summary of the current status

1. Model: DeepCSV

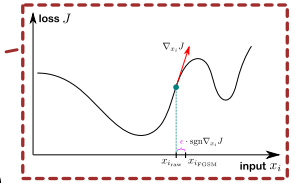


2. Attack: FGSM

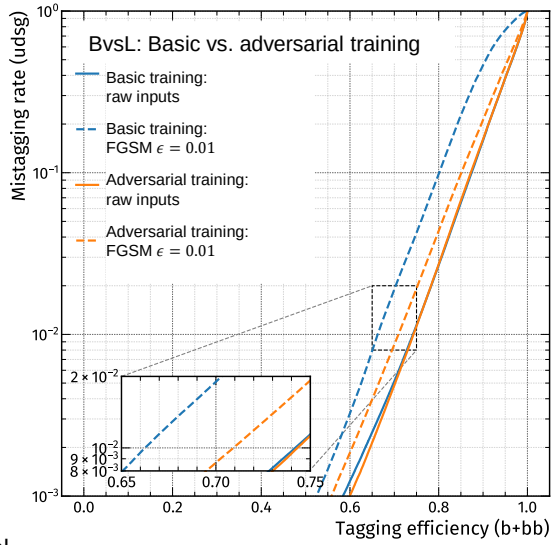


3. Defense: Adversarial training

FOR N EPOCHS:
- SPLIT WHOLE TRAINING SAMPLE INTO MINIBATCHES
- FOR EVERY MINIBATCH:
  - **DISTORT INPUTS (= APPLY FGSM)**
  - EVALUATE MODEL (FORWARD)
  - COMPUTE LOSS (AND APPLY LOSS WEIGHTING)
  - ACCUMULATE GRADIENTS OF LOSS (BACKWARD)
  - UPDATE MODEL PARAMETERS



4. Evaluation: e.g. ROC curves



More details: see next slides / additional material

Introduction: suggested material to get started

- CMS Induction Course (September 2021) – videos, slides, even more hyperlinks; choose some of them, some are more relevant than others, some repeat basics that you might already know; maybe have a look at the slides first and then decide which recordings you want to see

- Materials with which I started:
  - Reading:
    - BTV-16-002
    - AI safety for HEP
    - Bachelor thesis Nikolas Frediani
  - Pre-Exercises (get familiar with lxplus, CMSSW, possibly also git) / Intro flavour tagging
    Note: there might be updates – check status with Xavier, Andrzej, Spandan…
  - Intro deep learning / AI safety:
    - Machine Learning Mastery Tutorial (but it uses a different framework)
    - Nikolas‘ code

- Start setting up your programming environments – more details on additional slides

AI safety (HEP / b-tagging): suggested material

- AI Safety for High Energy Physics
  - Code (mlhep tutorial on github)

- Bachelor thesis Nikolas Frediani
  - Code (github, jupyter notebook that walks you from start to finish)
  - Subsidiary studies (git.rwth-aachen – use RWTH SSO)

- (Soon 😉 – like, this week) my thesis
  - Code (git.rwth-aachen, optimized to work on HPC RWTH)
  - Code (github, custom tagger version AI safety / scale factors on HTCondor, forked from Spandan's repo)
  - Talk (DPG, March 2021)
  - Talk (D-CMS, September 2021)

- T.B.A. (~ November?): more "hands-on" tutorial to understand the framework / code, and / or documentation; currently, there is only some text with links that I sent to Nikolas and Michael (see additional slides – sorry for all the text)

Note: all use PyTorch

Setup / getting started at RWTH Compute Cluster (HPC) [1]

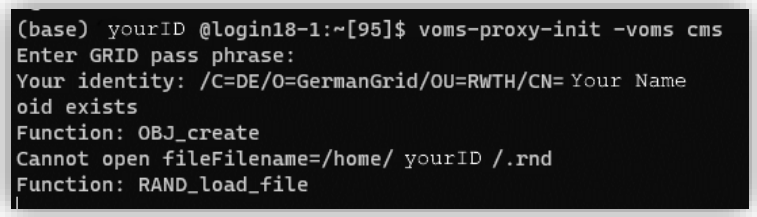
- Setup HPC (to be able to reproduce the results that we have so far – also see next slide(s))
  - Create an account via the RWTH Identity Management: add „Hochleistungsrechnen“ in the IdM and wait until it is activated
  - Ask Prof. Schmidt for your account to be added into a dedicated project (you will have better priorities and enough computation time…), for that, tell him your TIM-Id which goes like `xy123456`

  - The HPC is two things really, some login nodes where you can code and run some quick tests, run notebook servers and basically everything you do at lxplus. The major other thing is the usage of the batch system, which will in the end run the time- or memory-intensitive tasks. You do not actually access them, but instead you send so called jobs using dedicated .sh-scripts. For both the login and the batch system, there are different use cases when some nodes are better suited than others. Already the dialog system has dedicated nodes for cpu usage, gpu usage or file transfer (this will be interesting once you copy files over with larger bandwidths), see this link

  - On the practical side, to login via SSH, look here
    - Note: you need a VPN to the RWTH network, if you are not already in the network
    - Personally, I use Cisco AnyConnect VPN client and connect from Windows with a program called Cygwin / mintty
      - Setting up Windows Subsystem for Linux (WSL2) could be useful as well
    - If you use MacOS – others can provide you with details

Setup / getting started at RWTH Compute Cluster (HPC) [2]

- Once you have created your account and logged in, it would be useful to set everything up similarly to lxplus (e.g. conda, proxy - you might have looked at Andrzej's instructions already – walk through these instructions)
  1. https://hackmd.io/GkiNxag0TUmHnnCiqdND1Q
  2. https://twiki.cern.ch/twiki/bin/view/CMSPublic/WorkBookStartingGrid
  3. https://hackmd.io/@andrzejnovak/SJxscCZvD
     * When you encounter this problem (at HPC RWTH Aachen or NAF-DESY):



→ move to your home directory `cd /home/yourID` and do `touch .rnd` (create a dummy file)

- The packages I installed can be found here, but you don‘t have to copy that exactly, some are already outdated and you probably want to use the current versions; look at conda documentation and supplementary material (or ask ☺)
- One alternative to remote editing with VS Code or `jupyter notebook` is `jupyter lab` (it has a bit of both, you can edit .ipynb-notebooks, but it‘s also a bit like an IDE; start terminals, familiar directory structure)

Setup / getting started at RWTH Compute Cluster (HPC) [3]

- Then try to create your first jobscripts, this is described here
- Jobscripts are actually a bit complicated to understand when you first see them (e.g. one example is down below:)

```
#!/usr/local_rwth/bin/zsh

#SBATCH --ntasks=1

#SBATCH --ntasks-per-node=1

#SBATCH --mem-per-cpu=8G

#SBATCH --cpus-per-task=1

#SBATCH --job-name=MyTrainingJob

#SBATCH --output=output.%J.txt

#SBATCH --time=30:10:00

#SBATCH --account=rwth0583

# if you want to use a gpu, keep the next line as it is, if you don't need a gpu, change to # SBATCH (with a space between # and and SBATCH, making it commented out)

#SBATCH --gres=gpu:1

#SBATCH --mail-type=ALL

#SBATCH --mail-user=your.email@rwth-aachen.de

cd /home/um106329/aisafety
module unload intelmpi; module switch intel gcc
module load cuda/11.0
module load cudnn
source ~/miniconda3/bin/activate
conda activate my-env
python3 trL_weighted.py
```

Everything that **starts with a # and is not followed by a space is a command for the batch system**, the lines with # and space are comments, and every line that does not start with an # are shell commands like you would use them without the batch system. If you don't want to be informed via email on your job status, just add a space after the respective # commands (# SBATCH instead of #SBATCH). The important thing is: **don't mix the part of #-lines with the no-#-lines, everything with # should be placed before the shell-commands**. So to check whether your scripts work, you can first try all the shell-commands on a suitable login-node, e.g. one with gpus if you need them.

AI safety at RWTH Compute Cluster (HPC) [1]

The input files: to access them, you will need a valid grid proxy to work with conda, because we are going to clean them and store them. A basic script that does this can be found here: https://git.rwth-aachen.de/annika-stein/ai_safety_2021/-/blob/master/may_21/preparations/clean.py

- You can ignore the deprecated part (which is commented out). This script selects jets under given criteria (e.g. tracker acceptance), sets defaults if variables “make no sense” (e.g. are not correctly reconstructed), defines targets (the true hadron flavour = in our case four categories b, bb, c, udsg jets), and stores the result as numpy arrays. To later on have something to compare to, we also store DeepCSV (common tagger in CMS that also has the outputs b, bb, c and udsg probability).
- You will need additional files to make this script work. One are the default values: https://git.rwth-aachen.de/nikolas/ai-safety-default-value-studies/-/blob/master/default_value_studies/minima.npy (thanks to Nikolas) and the other are the file paths under which you can find the original .root files used for the training: https://raw.githubusercontent.com/andrzejnovak/nanocc/master/metadata/v2x17.json (thanks to Andrzej)
- This script currently runs interactively, e.g. you login to a copy.hpc node (the ones with number 18 inside seem to be very fast), and just perform this cleaning step (because shipping a proxy to SLURM is not as easy as shipping it to Condor jobs, at least I have not found out how to).
- To make the script really work in the end, you have to: have access to the additional files I linked above (downloading them is fine) and adjust all file paths in the script, e.g. the ones to the additional files plus the output paths. For that, you should be able to create a folder structure on `HPCWORK`, which will have enough disk space (as is the fastest with connection to the worker nodes). Something like the structure I used should be fine (`/hpcwork/yourtimID/october_21/cleaned_TT` for example).
- We will use a mixture of QCD and TTTosemileptonic files, and that means both have to be done. Start with e.g. TT (like you can see it in the script), and only when this is done, move on to QCD (you should update the output directory, otherwise you would overwrite files; also you have to change which paths are read by inverting what is commented out here in lines 30-49: https://git.rwth-aachen.de/annika-stein/ai_safety_2021/-/blob/master/may_21/preparations/clean.py#L30-49). And to finally run the script, everything has to be split up (running interactively has a timelimit). But you can just login to several nodes at a time, also the normal login18-nodes should work. Here are the splits I used that go into the argparser to run over all files, but stay in in the interactive timelimit: For TT: 0-499,500-999,1000-1499,1500-1999,2000-2446 and for QCD: 0-499,500-999,and so on,11000-11407 (yes, that's a lot, but the next steps will not take as long to set up and are less “manual” work). The third argument is the “default value” - this is actually a confusing name, the real definition is: by which number the average minima for all input variables are subtracted to obtain default bins. To understand this better, you might want to plot all inputs later with histograms.

AI safety at RWTH Compute Cluster (HPC) [2]

The preprocessing: Another very important step before we can start the training.

- We shift input distributions to be centered at 0, with standard deviation of 1. Scalers that do this need to be computed. But before that, we split our samples into training / validation / test sets. The scalers are then computed from all training samples only. They need to be stored, just like the scaled samples themselves. The current code that does this (plus a bit more for reweighting) can be found here: https://git.rwth-aachen.de/annika-stein/ai_safety_2021/-/blob/master/june_21/preparations/prep_new.py
- For the reweighting, download the eight .npy files from here (https://git.rwth-aachen.de/annika-stein/ai_safety_2021/-/tree/master/may_21) in order to point the script to your respective paths (i.e. update those lines in your preprocessing script: https://git.rwth-aachen.de/annika-stein/ai_safety_2021/-/blob/master/june_21/preparations/prep_new.py#L38-52). Also update all paths that read in the results from step 1 (cleaning) to your own paths.
- Uncomment these lines here: https://git.rwth-aachen.de/annika-stein/ai_safety_2021/-/blob/master/june_21/preparations/prep_new.py#L154-209 (necessary for reweighting later, same goes for lines 214,216,218 - they are needed as well, and 250,251,252 are also needed).
- This script has to be run several times, first one being with the `prepstep calc_scalers` (and there, you have to give a start and end index, because not all 67 inputs would fit into memory for the computation of the scalers). I guess something like packages of 6 or 7 variables at a time per node is feasible. Just with everything, make sure all paths are updated and directories exist before using the script. In this case, you want directories that store the scaled version of your inputs, so maybe give names like `/hpcwork/yourtimID/oct_21/scaled_TT` etc.

AI safety at RWTH Compute Cluster (HPC) [3]

Training: Finally, the part that runs a lot more automatic begins here.

- Currently, everything is located in the folders `train_models` and `attack` here: https://git.rwth-aachen.de/annika-stein/ai_safety_2021/-/tree/master/june_21 (so again, lots of code to download...)
- Important: the training can be done interactively before you submit jobs, that's nice if you want to test if everything works (which you should do, because of quota / core-hours). If you do it interactively, these lines can be used on a login-node with gpu access as well: https://git.rwth-aachen.de/annika-stein/ai_safety_2021/-/blob/master/june_21/train_models/training.sh#L28-30
- Again, you need to update every path that contains inputs, scalers, weights, source code... additionally, create directories that will hold checkpoints of your models (e.g. call them "`saved_models`"), namely where your scripts are placed, plus at the `HPCWORK` partition. (`HPCWORK` - fast, but no backup, `HOME` - slow, but backup available).
- Updates to paths also have to be done in the `jet_reweighting.py` file, in `submit_training.py`, `training.sh` and of course `training.py`. Then, you have LOTS of customizable options for the training, different parameters, setups, inputs, ...

Evaluation: lots of options – loss, tagger outputs, discriminators, ROC curves, AUC (over epoch), inputs, various supplementary scripts available that could be extended by you… but this will come later ☺

Possible next steps / project ideas

Tagger	Comment
DeepCSV (FCNN)	Current setup uses this one.
DeepJet (Convolutional + recurrent + dense layers)	Not studied yet in the context of AI safety. Could show higher susceptibility due to large number of (low-level) inputs.

Strategy / paradigm	Comment
Improve robustness against adversarial attacks (and hope that generalization to detector data improves as well)	That‘s the current strategy, which could be improved. Some keywords: FGSM attack, PGD attack, GANs, default and integer variables, keep distortions physical / compare with uncertainties, hyperparameter-tuning for **adversarial training**, reweighting; move to other attacks and other defenses, e.g. defensive distillation, combinations / hybrid-methods (see Michael‘s slides).
Transfer learning / domain adaptation	Not studied yet. Make use of / improve **scale factors** (SFs) while training your tagger (think of “coupling” the training with the evaluation).

Evaluation technique	Comment
Discriminators, ROC curves, AUC	Included in current setup, might be extended / split into different phase space regions.
Inputs (shapes, correlations, AUC ranking)	Code available for all three techniques, input shapes already part of detailed checks with current setup, others studied just as an aside.
Loss landscape	Could help understand the inner-workings of adversarial training.
Data / MC comparison, scale factors	Started already (code available), could be reused / extended / improved.

Deep learning: supplementary material

Blogs:

- Machine Learning Mastery (can confirm that this is a great blog for beginners 😇) – maybe look at one tutorial first and gradually read more of the related posts and references
- Towards Data Science (helpful individual contributions, nice ideas & implementation / code) – I used it mainly for “inspiration” and to get ideas for specific tasks like dataloading, reweighting, … not when first starting out

Books:

- Deep learning with PyTorch (free .pdf, informative illustrations, uses the PyTorch syntax, more practical)
- Deep Learning (free .html, also covers statistics, calculus, lin. alg. for ML …, more theoretical)

Miscellaneous:

- A Living Review of Machine Learning for Particle Physics (collection of more than 500 papers, sorted by topic)
- https://www.3blue1brown.com/topics/neural-networks (videos, interactive lessons, code…)

AI safety: supplementary material

- Explaining and Harnessing Adversarial Examples: http://www.arxiv.org/abs/1412.6572
- Intriguing properties of neural networks: https://arxiv.org/abs/1312.6199
- Towards Deep Learning Models Resistant to Adversarial Attacks: https://arxiv.org/abs/1706.06083
- Adversarial Attacks and Defences: A Survey: https://arxiv.org/abs/1810.00069
- Video (lecture on youtube)
- Michael‘s slides

Note: more references with specific ideas and concepts are presented in the thesis

Software: supplementary material

- Python general (numpy, scipy, scikit-learn, matplotlib, …)
- scikit-hep uproot awkward mplhep iris-hep coffea
- Git (git.rwth-aachen.de, Git CMSSW, gitlab.cern.ch, github.com)

Collection of tutorials:

- https://lpc.fnal.gov/programs/schools-workshops/hats.shtml
- https://hepsoftwarefoundation.org/training/curriculum.html
- https://indico.cern.ch/event/1019958/timetable/#20210705.detailed
- https://hsf-training.github.io/analysis-essentials/
- … I'm sure Andrzej knows even more resources 😁