Pattern matching with convolutional neural networks for the PANDA at FAIR experiment

UPPSALA UNIVERSITET

A project by:
Jakob Andersson
Kerstin Ersson

Supervisor:
Michael Papenbrock

PROJECT AIM

The aim of the project is to investigate the feasibility of neural networks as a method for the pattern matching for the PANDA at FAIR experiment. We limited the project to two scopes:

" Can neural networks be used to identify:

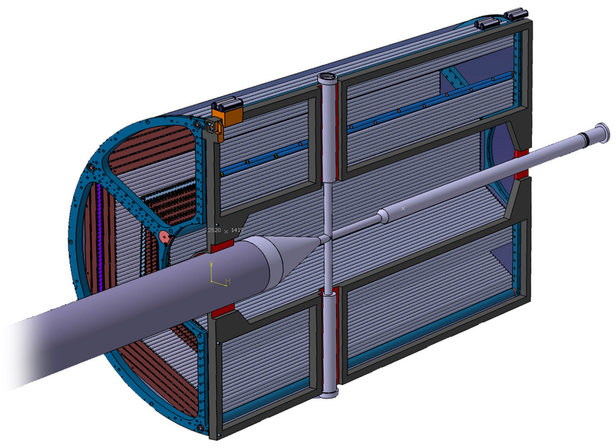
- *Particles type for a single track?*
- *The number of tracks in an event?"*

WHAT IS PANDA?

PANDA will be one of the most important experiments at the Facility for Antiproton and Ion Research (FAIR). The experiment will aim to investigate the strong and weak nuclear forces and the structure of composite particles called hadrons. The particle detector will use only software triggers, making it the first of its kind.

DATA COLLECTION

This project focuses on data from a part of the PANDA detector called the Straw Tube Tracker (STT), which is a part of the Central Tracker. It consists of 4224 gas filled tubes in a barrel shape, which will register hit signals when a charged particles passes through. The hits can then be clustered together to form trajectories.



SOFTWARE

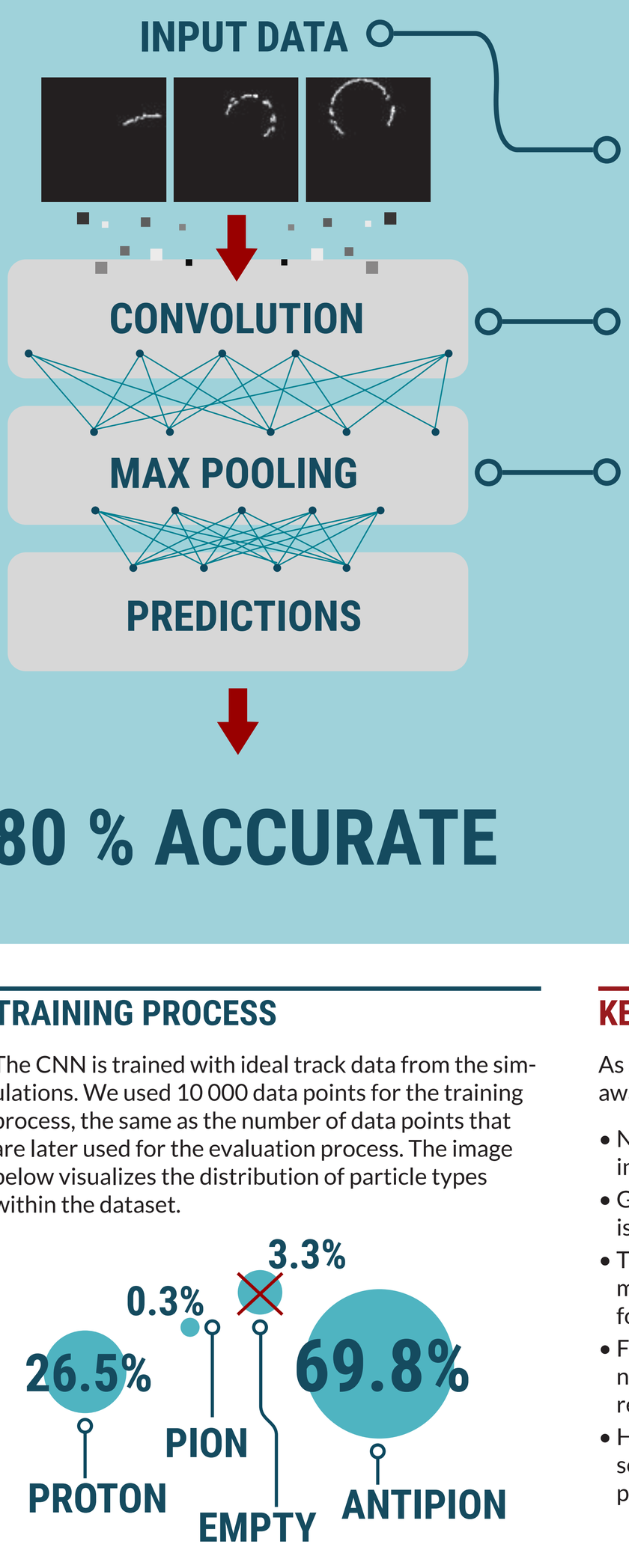
PandaRoot

PandaRoot
Simulations

Matlab
Data formatting

TensorFlow
CNN implementation

OUR CNN



The data was transformed from 1D tube hit arrays to 2D images using a weighted function. The images are 8-bit JPEG:s in black and white.

A ***convolutional layer*** acts like a filter on the image. It can extract different features, or shapes, used for the image classification.

The ***max pooling layer*** reduces the spatial size of the image, while storing the most important information.

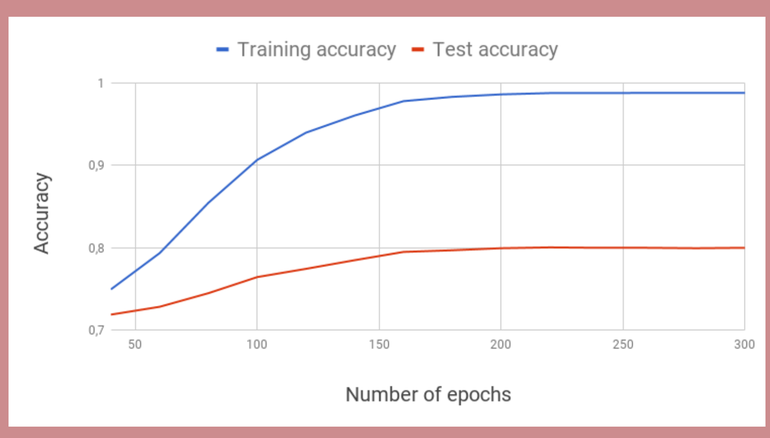
NETWORK DESIGN

Our network uses a relatively simple design; two alternating convolutional, amax pooling layers and a final dense (fully connected) layer where the predictions are made.

RESULTS

The network results from the project is presented in the graph below, showing the accuracy of the networks classification over a period of training. This is one example of a neural network and there are several different setups, with varying data sizes and network settings, that all are part of our results.

Higher accuracy for training than testing suggests that the network is being overfitted, which means that the network is more adapted to the specific training data set than to the general case. Due to unbalanced training data, this was to be expected.



THE COMPUTATIONAL ASPECT

Memory	Each image is ~1 kB and the total network model size is around 130 MB, so memory will not be a challenge in this case.
Run time	Evaluating one datapoint in this CNN takes approximately 2 ms, data formatting time excluded. For reference, the datastream of the detector is about 2 million events per second.

TRAINING PROCESS

The CNN is trained with ideal track data from the simulations. We used 10 000 data points for the training process, the same as the number of data points that are later used for the evaluation process. The image below visualizes the distribution of particle types within the dataset.

KEY TAKE-AWAYS

As a summary of our project, here are some key take-aways:

- Neural networks are often great at extracting information from data with unexpected varieties
- Good training data is key to success, make sure it is balanced!
- There are many parameters that affect the performance of a neural network, so finding the best setup for a certain problem can be time consuming
- For extremely high data rates a convolutional neural network might not be the best option because of the required computation times
- However, distributing the data processing across several nodes may make this kind of network a possbile alternative for the PANDA detector

FUTURE DEVELOPMENT

There are a number of ways this project could be developed further and improved upon. Below we list some of the alternatives we think are most promising:

- Using more realistic data to train the network
- Using a data set with a balanced representation of classes
- Investigate the effect of changing more parameters in the network such as number of filters, filter sizes, max pooling kernel sizes etc
- Optimize the code by e.g. porting the data formatting and network evaluation to C++
- Look into using a new kind of neural networks called capsule neural networks