JOURNAL OF
EMERGING INVESTIGATORS

Article

Evaluating TensorFlow image classification in classifying proton collision images for particle colliders

Rohan Nakra[1*], Arin Vansomphone[1*], Ryan Kimes[1]
[1] Huntington Beach High School, Huntington Beach, California
* These authors contributed equally to this work

SUMMARY

The Large Hadron Collider at the European Organization for Nuclear Research (CERN) is planned to increase the number of proton collisions 10-fold by 2025. To keep up with this rapid pace, novel and more efficient particle collision classification methods must be developed, which many physicists believe can be achieved via machine learning. Since particle collisions and trajectories are mapped onto images, we chose TensorFlow's image classification model to analyze and predict particle classes from these collision images. We aimed to evaluate the accuracy at which image classification could correctly identify a particle's category from its trajectory image. We hypothesized that image classification would improve its accuracy with larger training datasets and more training epochs. To test our hypotheses, we randomly partitioned 100 images for testing and reserved 3,400 for training. We ran numerous trials using different training set sizes, different training epoch values, and binary classification. Our results show that the model's accuracy increases with more training samples, and it increases its accuracy and consistency with more training epochs. In binary classification, the model distinguishes the electron class the best, with a 95.2% mean accuracy. We conclude that image classification has remarkable implementation potential in classifying particle collision outcomes because of its ability to improve with training sample size and in binary classification. Our work highlights one method collision classification algorithms should utilize, which can work effectively with the increased data output by particle colliders and give physicists a vital tool in accelerating their field.

INTRODUCTION

The Large Hadron Collider (LHC) at the European Organization for Nuclear Research (CERN) is the world's most powerful particle collider and is used by more than 100 countries to experiment in particle physics (1). The collider works by colliding proton bunches at high speeds to produce a plethora of new subatomic particles. These collisions occur in cylinder-shaped bodies of the collider known as detectors (2). The new particles shoot off at various trajectories, hitting silicon sensors within the detector that record their position. To analyze results and classify particles, physicists must reconstruct the particles' trajectory data from the raw sensor data. Currently, the LHC utilizes a multi-step process that includes first locating seeder hits, known as seeding, then applying layers of Kalman filters with algorithms to eliminate surrounding particle noise (3). Seeding acts as a preliminary step that tracks and stores data on the particle positions at different times (3). The Kalman filter, an algorithmic estimation tool, uses this data to produce a smooth trajectory of a particle by minimizing the effects of energy loss and scattering when tracking particles (3). It analyzes this data and uses matrix operations to estimate a more accurate position of the particle (3).

Unfortunately, current classification methods will soon be outdated due to increasing demands at the LHC. The number of proton collisions at the collider is expected to increase 10-fold by 2025, which will overload the Kalman filters with statistical noise and make it significantly more challenging to identify trajectories (2). Even with computer central processing unit improvements and the present algorithm software, it is estimated that there remains a more than 10-fold gap in the data processing capabilities required for the increased level of collisions (2). Therefore, novel techniques need to be developed to advance particle classification research at the LHC.

The most attractive field for finding new algorithms is machine learning, and CERN researchers are already exploring deep neural networks to simplify and speed up computer processing (4). In this study, we evaluate TensorFlow, an open-source library that contains tools for data preparation and deep learning models through the Keras Application Programming Interface. One of the types of deep learning models is image classification, which already plays a significant role in figure recognition in the technology, healthcare, and security fields (5). TensorFlow's image classification model takes in test images and, as the name suggests, categorizes them into thematic maps of their pixels, which are used to predict the classifications of test images (6). In training an image classification model, there are two methods: unsupervised and supervised classification (7). Unsupervised classification is when the model creates its own groupings without a user's designation of the classes or a user's designation of bounds for the sample images' pixels (7). Supervised classification is the opposite; the user designates a fixed number of classes and assigns groups of pixels for the model to analyze (7).

Since each particle type has similar trajectories and images within its class, image classification poses an enticing choice for predictive modeling. The trajectory images we focus on come in five different categories: electrons, kaons, muons, pions, and protons. These were the categories with trajectories available in our dataset (**Figure 1**) (8). Our research focused on answering three questions: Does the model's accuracy increase as we expand the sample size of training images it uses, does it increase as we increase the number of training epochs it uses, and which particle types does it classify best?

We hypothesized that the TensorFlow model would improve its accuracy as we increased its training sample size as well as the number of training epochs. Additionally, we hypothesized that the model would work best with the proton class, because protons are the heaviest and presumably have the most distinct trajectory out of the five classes we worked with. For our experiments, we randomly divided 100 samples for testing and allocated the other 3,400 for training. We then conducted multiple trials using varying training sample sizes, varying numbers of epochs, and binary classification. We found evidence to support our first and second hypotheses: the model's accuracy increased with larger training sets, as well as with more training epochs. Our results contradicted our third hypothesis and showed that the model instead excelled with electron classification with a 95.2% mean accuracy. This evidence demonstrates that image classification models have significant promise to determine particle collision outcomes with high success rates due to their ability to improve their accuracy with additional parameters.

RESULTS

The dataset we utilized contains roughly 1.2 million particle trajectory images, 10 pixels x 10 pixels, and their corresponding classification (8). The collisions were generated by the Acts Common Tracking Software (ACTS) simulation engine and converted to image form by the dataset creator to isolate individual trajectories and label classifications (8, 9). Each image represents a transverse section of the LHC detector, such that its cylinder shape was sliced parallel to the bases to produce two cylinders (9). The dataset creator also rotated the trajectories so that the collisions occur at the bottom right corner (**Figure 1**). An image covers 1.1 meters x 1.1 meters, making each pixel represent a 110 mm x 110 mm sector (**Figure 1**). The pixel shades detail the number of registered particle hits per sector, with a maximum of four. Since the dataset only had the image and its classification as parameters, we used both in our experiments.

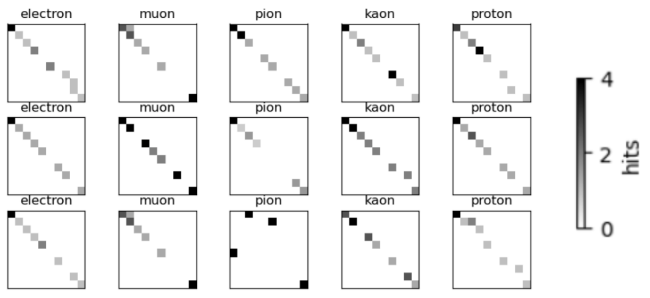

Figure 1. Samples from Dataset. Shown are the 10 x 10 pixel particle trajectory images of 15 different samples, 3 of each classification. An image covers a 1.1 meter x 1.1 meter transverse section of the LHC detector, such that the body was sliced parallel to the bases. The trajectories are rotated so that the collisions occur at the bottom right corner. Differing shades of the pixels indicate the number of particle hits detected in each 110 mm x 110 mm sector. Collisions were generated by the ACTS simulation engine, then converted to image form to isolate individual trajectories and designate classifications.

Positive Correlation Between Training Sample Size and Model Accuracy

As observed through studies such as Chu et. al, expanding sample sizes for image classification models can yield higher performance accuracies (10). Similarly, measuring the accuracy at increasing training sample sizes would allow us to determine whether our model could improve in its ability to correctly classify particle collision outcomes. To do this, we tested our model accuracy on 100 images while increasing the training set gradually from 400 to 3,400 images. The accuracies increased with larger training set sizes, demonstrating a positive correlation (Linear regression t-test, $p < 0.001$, **Figure 2**).

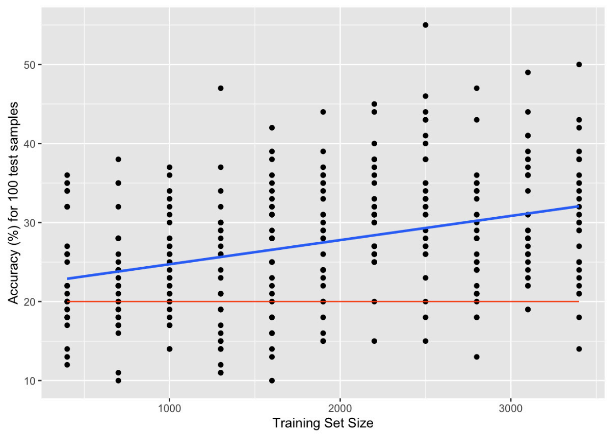

Figure 2. Training set size and model accuracy are positively correlated. We recorded model accuracy using 3 training epochs at varying training sample sizes (30 trials per sample size). Sample sizes ranged from 400 training samples to 3,400 training samples, in increments of 300. The resulting scatter plot comparing the accuracy vs. the training set size is shown. Each trial is denoted by a black dot. Some trials are not visible due to overlapping accuracy values with other trials. The least-squares line of best fit is shown in blue while random guessing is shown by the red line. Linear regression T-test, $p < 0.001$.

Positive Correlation Between Training Epochs and Model Accuracy

The number of epochs represents the number of times the model reviews the training dataset. In our previous experiment, we set our model to the default three epochs. Numerous studies have demonstrated that increasing the number of training epochs in neural networks improves test accuracy, so we hypothesized that the same adjustments would improve our model's performance (11). Using a dataset split of 3,400 training and 100 testing samples, we assessed our model accuracy while incrementally increasing the number of epochs.

Similar to the training set experiment, the model's accuracy increased with more training epochs (Linear regression t-test, $p < 0.001$, **Figure 3**). This time, however, we were not limited by the amount of data we had and could increase the training epochs as much as we desired. So, we ran trials with 100, 200, 300, 400, 500, and 1,000 training epochs to attempt to find the limit of improvement through epoch adjustments (**Table 1**).

At these extreme trials, the model reaches accuracies near

Epochs	Mean Accuracy (%)	Standard Deviation (%)	Maximum (%)
100	55.8	2.57	59
200	57.5	2.32	61
300	58.4	1.35	61
400	60.4	3.06	65
500	61.3	2.71	66
1000	61.8	1.93	65

Table 1. Model reaches an accuracy improvement limit between 500 and 1,000 training epochs. We recorded model accuracy using 3,400 training samples at varying numbers of training epochs, ranging from 100 to 1,000 (10 trials per number of training epochs). The table shows the numbers of training epochs and their corresponding mean accuracy, standard deviation, and maximum as a percent. Mean accuracy, referring to the statistical mean, and standard deviation were calculated within each group of 10 trials. 95% confidence interval. Two-sample T-test, $p = 0.320$.

Particle Classification	Mean Accuracy (%)	Standard Deviation (%)	Maximum (%)
Electron	95.2	7.15	100
Kaon	78.4	13.3	96
Muon	90.1	9.25	100
Pion	80.8	10.3	93
Proton	84.6	11.2	98

Table 2. There are statistically significant differences between the accuracies of the various binary models. We recorded Model accuracy using 1,300 training samples and 7 training epochs for every classification's binary model (30 trials per classification). The table shows the particle type of interest and their corresponding mean accuracy, standard deviation, and maximum as a percent. Mean accuracy, referring to the statistical mean, and standard deviation were calculated within each group of 30 trials. One-way ANOVA, $p < 0.001$. Tukey post hoc test, see Results section for p values.

60% consistently. For instance, 1,000 training epochs gives the model a mean accuracy of 61.8%, with a margin of error of 1.38% (95% confidence interval, **Table 1**). The accuracies are significantly improved and more consistent compared to the trials with smaller epoch values. Unfortunately, this enhancement appears to have a limit; there is no significant improvement in the 1,000 epoch trials from the 500 epoch trials (two-sample T-test, $p = 0.320$, **Table 1**).

Improved Model Accuracy Through Binary Classification

We attempted to boost the accuracy of our model even more by narrowing the classifications to two outcomes. Similar to expanding the number of samples or training epochs, simplifying multiclass learning into binary classification has been demonstrated in studies to enhance an algorithm's accuracy (12). Instead of the five original particle types, we tested one type against all other types. In doing so, these experiments also gave evidence on which particle type is the easiest to distinguish. For each classification test, we relabeled the samples in accordance with binary code ("1" for the particle of interest and "0" for all the others), then evaluated our model on 100 test samples and 1,300 training samples.

All binary models achieved higher mean accuracies than the original five-outcome model (**Table 2**). All maximums were at least 90%, and the electron and muon models each reached 100% multiple times. We found a statistically significant difference between the accuracies of the different binary models (One-way ANOVA, $p < 0.001$, **Table 2**). Further analysis revealed significant pairwise differences between the accuracies of the kaon and electron (Tukey post-hoc test, $p < 0.001$, **Table 2**), pion and electron (Tukey post-hoc test, $p < 0.001$, **Table 2**), muon and kaon (Tukey post-hoc test, $p < 0.001$, **Table 2**), and pion and muon groups (Tukey post-hoc test, $p = 0.007$, **Table 2**). Electrons performed with the highest mean accuracy, followed by muons, protons, pions, then kaons (**Table 2**).

DISCUSSION

From our experiments, the model accuracy demonstrated a positive correlation with training set size. It also showed a positive correlation with the number of training epochs, and its accuracy became more consistent with more epochs, illustrated by the decrease in standard deviations from 3 to 1,000 epochs (**Figure 3, Table 1**). Epoch improvement became negligible after roughly 500 epochs. Finally, our model performed more accurately in binary classification with individual particle types than in classification with all five particle types. The electron model ran the best, with a 95.2% mean accuracy, followed by the muon model, then the proton model, then the pion model, and the kaon model. All binary classification models reached maximums of at least 90%. Our results lead us to conclude that image classification can distinguish and learn particle collision outcomes. Its accuracy improves with larger training datasets and more training epochs, and in both parameters, the model does not overfit the data. Image classification works best in classifying binary outcomes of particle collisions, where it can achieve accuracies of 90% and greater. The method performs better with certain particle classes over others, and we found that it works best in distinguishing electrons and muons.

We acknowledge several limitations, primarily in our dataset, that could have affected our conclusions. The 10x10 images of the dataset could have been too oversimplified, eliminating more distinct features of the trajectories (8). The five particle classes we had are not the complete set of elementary particles, so we had a lack of representation of other particle classes (13). Additionally, many deep learning models require hundreds of thousands or millions of samples to predict effectively. We were limited to 3,500 samples in total, preventing our ability to observe trends with huge amounts of data. In a hypothetical situation where a particle is not one of the possible targets, the model would misclassify the particle, showing the need for more data not just on the particles already present in the dataset, but also those that are excluded.

These constraints emphasize numerous variables to explore in future experiments. Seeing that Hirahara et al. concluded through their study that optimized image scaling increases accuracy in neural network models, we would like to experiment with more pixelated (and thus more detailed) collision images as data (14). We predict that this adjustment will boost our model's accuracies, in alignment with Hirahara et al., Since there are more than five elementary particles,

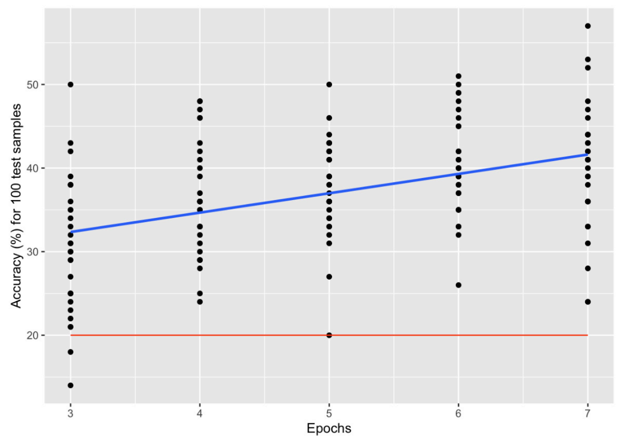

Figure 3. Training epochs and model accuracy are positively correlated. We recorded model accuracy using 3,400 training samples at increasing numbers of training epochs (30 trials per number of training epochs). Training epochs ranged from three to seven. The resulting scatter plot comparing the accuracy vs. the number of epochs is shown. Each trial is denoted by a black dot. Some trials are not visible due to overlapping accuracy values with other trials. The least-squares line of best fit is shown in blue while random guessing is shown by the red line. Linear regression T-test, $p < 0.001$.

we are interested in acquiring data for additional particle categories to examine our model's performance with more classes. Image classification has achieved higher accuracies with other datasets that have more categories, such as handwritten digits (15). Even when the handwritten digits (MNIST) dataset, which has 10 categories for the 10 digits, are shortened to 3,500 samples, 500 epochs, and a 3:1 ratio of train and test data (to match the parameters for the particle collision model), it achieves an accuracy of 96.5% (15). The difference in accuracy between the MNIST dataset and particle collision dataset is most likely due to the images in the particle collision dataset not being detailed enough and the MNIST dataset having a more straightforward and predictable pattern for the machine learning model. However, we still anticipate that TensorFlow will maintain similar accuracies to those recorded in our experiments with more particle categories, and that the accuracies will increase by expanding the data and training runs. Typically, researchers use all 70,000 samples of the MNIST dataset, as shown in Hasanpour et al., to allow the model to analyze the dataset further in order to increase its accuracy and understanding of the dataset (16). Therefore, we are interested in training our model on thousands more trajectory images. As shown in our epoch experiment, we found a limit for epoch improvement of the accuracy. There were also long run times for each trial, which discourages further exploration and implementation into the model. We have not found a limit in improvement with increasing the training set size, and more data samples will allow us to uncover the complete potential of this method. Moreover, we would also like to explore the effects of using unbalanced distributions of particle classes to train our model, because the types of output particles following LHC particle collisions are rarely evenly-distributed (17). When we trained the binary classification model on uneven representation and tested on a handful of runs, the accuracies (data not shown) were lower compared to the accuracies we show here (**Figure 4**). We hypothesize that the accuracies will fluctuate if the training set splits between different particle types are changed. We reason that the frequency at which the model examines one particle will be different compared to that of another particle, resulting in a higher comprehension for one and a lower comprehension for the other. Furthermore, we look forward to integrating other particle characteristics, namely charge, momentum, and spin (other types of raw data measured by the LHC silicon sensors), into classification models (2). This could be achieved through a multivariate model analyzing the collision output image along with the added parameters to determine the resulting particle from the collision. This multivariate analysis adds more potential to boost our algorithm's accuracies, because it would increase the number of features available to the model and provide insights on the model's ability to comprehend other particle characteristics.

In summary, TensorFlow's image classification model improves its accuracy in classifying particle collision outcomes with larger sample sizes and more training epochs. Image classification worked the best in binary classifications, especially with electrons and muons, which both achieved mean accuracies above 90%. Our work indicates the promising potential that TensorFlow image classification algorithms have compared to other neural network methods. For example, Tsaris et al. utilized Long Short-Term Memory recurrent neural networks in their hit classification model and recorded a 70% overall accuracy (18). Tsaris et al. used a similar ACTS dataset composed of millions of collision events. With our model recording 62% mean accuracy training with only 3,400 samples, the TensorFlow algorithm has the potential to exhibit higher accuracies with the larger training datasets. Additionally, as reported in Tüysüz et al., Graph Neural Network models face the issue of extensive training times, taking around one week to process the 10,000 event TrackML dataset (19). Seeing that the TensorFlow algorithm takes roughly 4 minutes to train and test on 3,500 samples, we predict that image classification can have exceptional

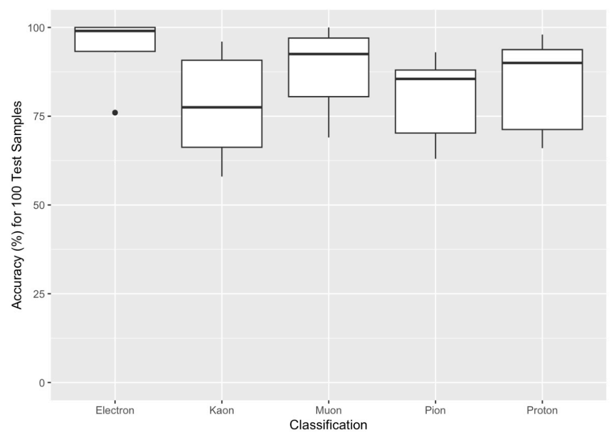

Figure 4. Electron model performed with the highest accuracy among binary models. We recorded a binary model accuracy using 1,300 training samples and 7 training epochs for all 5 particle classifications (30 trials per classification). The resulting box plots displaying the accuracies for each model is shown. The black dot in the Electron column represents one trial.

scalability with larger datasets. Its ability to quickly recognize and strengthen patterns with rapidly increasing amounts of data suits implementation in the LHC and other particle accelerators. TensorFlow is also an open-source program, which allows more users everywhere to experiment with and record data on particle collisions. We hope our work progresses the field of optimizing particle classification techniques, allowing particle physicists to accelerate their research on elementary particles.

MATERIALS AND METHODS

We used 350 simulated particle collisions programmed to mimic the conditions of LHC silicon detectors as data (as explained in the Results section). The original dataset is found at: www.kaggle.com/datasets/stephenmugisha/particle-collisions/discussion (8). Each collision event produced around 3,000 particle trajectories, totaling roughly 1.2 million 10x10 images, in '.pkl' format.

In the total of 1.2 million samples, 3,000 were electron instances, 700 were muon instances, 980,000 were pion instances, 160,000 were kaon instances, and 114,000 were proton instances. Clearly, the sample size was not consistent for each class. To account for this, we reduced the number of samples for each particle to 700 samples, the size of the muon class. The resulting dataset contained 3,500 samples. Reducing the dataset to balanced distributions of each classification ensured equal representation, which allowed us to more accurately evaluate our model. When we trained and tested the model on the full, unbalanced dataset it performed well with the pion class, but not on the other classes, resulting in a misleading overall accuracy (data not shown). Balancing the dataset enabled us to obtain an accuracy that reflected the model's ability to understand all of the classes.

All tests scaled the data through standardization to limit bias of certain features of the dataset. The target/labels used the 'to_categorical' function from TensorFlow to convert the labels into a format that the model can understand.

We used Python version 3.9.12, with the libraries 'NumPy' and 'sklearn' to tidy, split, and rescale the data into formats that TensorFlow could process (20, 21). We used the 'pickle' library to change the file from '.pkl' format to a '.csv' file (22). The data was downloaded as 350 pickle files, each of which contained approximately 3,000 labeled images. The files were also "unpickled" through the 'pickle' module in Python. Then, we changed the target value range to 0 - 4, which was more practical than the initial target values of 11, 13, 211, and more. We used the 'Matplotlib' library to visualize the 10 x 10 sample images (23). The complete script for these operations can be found at our GitHub repository: github.com/rohannakra/AI-Predicts-Particle-Collisions.git, in the file "main.ipynb".

Our model was a TensorFlow Keras model with two input layers, two hidden layers, and one output layer. We flattened the data into a single vector to be processed, the hidden layers predicted each batch of samples given, and stochastic gradient descent (the optimization function) updated the weights based on the hidden layers' performance on the batch. We calculated model accuracy by the number of samples the model correctly identified divided by the total number of samples we tested the model on. We used TensorFlow's library on Python as well to create the model (24). The code can also be found in the "main.ipynb" file in our GitHub repository.

To analyze trial results, we used the R language, version 4.2.2., on the Rstudio integrated development environment. We used the R packages 'readr' and 'tidyverse' to integrate '.csv' result files and calculate statistics (25, 26). We also used the R package 'ggplot2' to produce the graphs of results (27). The trial results and the R analysis script are found in the GitHub repository as well, under the folder "R Data Analysis".

Positive Correlation Between Training Sample Size and Model Accuracy Procedure

From our dataset of 3,500 images, we started by randomly selecting 100 images of each particle classification, totaling 500 samples. We randomly partitioned the 500 samples into 100 testing images and 400 training images. Then, we ran 30 trials (to satisfy the Central Limit Theorem and create normality) and recorded the model's accuracy, resetting the model's memory after every trial. We repeated this process in increased increments of 300 training samples, 60 from each classification until we reached the maximum training set size available of 3,400.

Positive Correlation Between Training Epochs and Model Accuracy Procedure

We randomly split our dataset into 3,400 training samples and 100 test samples. We started by altering the model to complete 4 training epochs and ran 30 trials and recorded the accuracies. We then repeated the procedure for five, six, and seven epochs. All trials used the same training and test images, and we wiped the memory of the model after every trial to create control in our experiment. When we experimented with 100 training epochs up to 1,000 epochs, we also used the same data split and cleaned the memory after every trial. These trials had long run-times, reaching 4 minutes each at 1,000 training epochs, so we only gathered 10 trials per epoch value.

Improved Model Accuracy Through Binary Classification Procedure

We designated a particle class as "1" and labeled the rest as "0" in our 3,500-sample dataset. We took all 700 "1" particles and randomly selected 700 "0" particles to put into a separate data pool. From the pool, we randomly selected 100 samples for testing and assigned the rest to training. We ran and recorded 30 trials, using 7 training epochs, then repeated the entire process for the other 4 particle types.

Received: January 17, 2023
Accepted: May 3, 2023
Published: August 20, 2023

REFERENCES

1. CERN. "Our Member States." *CERN*, home.cern/about/who-we-are/our-governance/member-states. Accessed 29 December 2022.
2. Rousseau, David et al. "The TrackML challenge." *NIPS 2018 - 32nd Annual Conference on Neural Information Processing Systems*, Dec. 2018, pp. 1-23.
3. Sguazzoni, Giacomo. "Track reconstruction in CMS high luminosity environment." *Nuclear and Particle Physics Proceedings*, 31 May 2016. https://doi.org/10.1016?j.nuclphysbps.2015.09.437.
4. Coelho, Claudionor N. et al. "Automatic heterogeneous

quantization of deep neural networks for low-latency interference on the edge for particle detectors." *Nature Machine Intelligence*, vol. 3, no. 1, Jun. 2021, pp. 675-686. https://doi.org/10.1038/s42256-021-00356-5.
5. Schmidt, Johnathan et al. "Recent advances and applications of machine learning in solid-state materials science." *Computational Materials*, vol. 5, Aug. 2019. https://doi.org/10.1038/s41524-019-0221-0.
6. Abu, Mohd Azlan et al. "A study on Image Classification based on Deep Learning and Tensorflow." *International Journal of Engineering Research and Technology*, vol. 12, no. 4, 2019, pp. 563-569.
7. Raschka, Sebastian and Vahid Mirijalili. *Python Machine Learning*. 3rd ed., Packt Publishing, 2019. *Falksangdata*, falksangdata.no/wp-content/uploads/2022/07/python-machine-learning-and-deep-learning-with-python-scikit-learn-and-tensorflow-2.pdf. Accessed 30 June 2023.
8. Mugisha, Stephen. "Particle collisions." *Kaggle*, www.kaggle.com/datasets/stephenmugisha/particle-collisions. Accessed 21 June 2023.
9. Amrouche, Sabrina et al. "The Tracking Machine Learning Challenge : Accuracy phase." *The NeurIPS '18 Competition*, 30 Nov. 2019, pp. 231-264. https://doi.org/10.1007/978-3-030-29135-8_9.
10. Chu, Carlton et al. "Does feature selection improve classification accuracy? Impact of sample size and feature selection on classification using anatomical magnetic resonance images." *NeuroImage*, vol. 60, no. 1, Mar. 2012, pp. 59-70. https://doi.org/10.1016/j.neuroimage.2011.11.066.
11. Gbenga Ajayi, Oluibukun and John Ashi. "Effect of varying training epochs of a Faster Region-Based Convolutional Neural Network on the Accuracy of an Automatic Weed Classification Scheme." *Smart Agricultural Technology*, vol. 3, Feb. 2023. https://doi.org/10.1016/j.atech.2022.100128.
12. Dietterich, Thomas G. and Ghulum Bakiri. "Solving Multiclass Learning Problems via Error-Correcting Output Codes." *Journal of Artificial Intelligence Research*, vol. 2, Jan. 1995, pp. 263-286. https://doi.org/10.1613/jair.105.
13. Griffiths, David J. *Introduction to elementary particles*. 2nd ed., Wiley, 2008. *CERN*, cds.cern.ch/record/111880. Accessed 26 July 2023.
14. Hirahara, Daisuke et al. "Effects of data count and image scaling on Deep Learning training." *PeerJ Computer Science*, vol. 6, no. 312, Nov. 2020. https://doi.org/10.7287/peerj-cs.312v0.2/reviews/2.
15. Nakra, Rohan. "AI Predicts Handwritten Digits." *Github*, github.com/rohannakra/AI-Predicts-Hand-Written-Digits. Accessed 21 June 2023.
16. Hasanpour, Seyyed Hossein et al. "Lets keep it simple, Using simple architectures to outperform deeper and more complex architectures." *arXiv*, Apr. 2023. https://doi.org/10.48550/arXiv.1608.06037.
17. Aamodt, Kjeld et al. "Production of pions, kaons and protons in pp collisions at √s=900 GeV with ALICE at the LHC." *The European Physical Journal C*, vol. 71, no. 1655, Jun. 2011. https://doi.org/10.1140/epjc/s10052-011-1655-9.
18. Fantahun, Kebur et al. "Classification of Pixel Tracks to Improve Track Reconstruction from Proton-Proton Collisions." *SMU Scholar*, vol. 6, no. 2, Sep. 2022.
19. Tüysüz, Cenk et al. "Hybrid quantum classical graph neural networks for particle track reconstruction." *Quantum Machine Intelligence*, vol. 3, no. 2, Dec. 2021. https://doi.org/10.1007/s42484-021-00055-9.
20. Harris, Charles R. et al. "Array programming with NumPy." *Nature*, vol. 585, Sep. 2020, pp. 357-362. https://doi.org/10/1038/s41586-020-2649-2.
21. Buitinck, Lars et al. "API Design for Machine Learning Software: Experiences from the Scikit-Learn Project." *arXiv*, Sep. 2013. https://doi.org/10.48550/arXiv.1309.0238.
22. Van Rossum, Guido. *The Python Library Reference, Release 3.8.2*. Python Software Foundation, 2020. *Massachusetts Institute of Technology*, py.mit.edu/_static/spring21/library.pdf. Accessed 30 June 2023.
23. John D. Hunter. "Matplotlib: A 2D Graphics Environment." *Computing in Science & Engineering*, vol. 9, no. 3, May 2007, pp. 90-95. https://doi.org/10.1109/mcse.2007.55.
24. Abadi, Martín et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems." *arXiv*, Mar. 2016. https://doi.org/10.48550/arXiv.1603.04467.
25. Wickham, Hadley et al. "readr: Read Rectangular Text Data." *CRAN*, cran.r-project.org/package=readr. Accessed 19 June 2023.
26. Wickham, Hadley et al. "Welcome to the Tidyverse." *Journal of Open Source Software*, vol. 4, no. 43, Nov. 2019, pp. 1686. https://doi.org/10.21105/joss.01686.
27. Wickham, Hadley. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. *ggplot2*, ggplot2.tidyverse.org. Accessed 19 June 2023.