Using Physics, Monte Carlo Techniques and Machine Learning in the NOvA Project

Candidate Number: 167006
Supervisor: Dr. Lily Asquith
Word Count: 9640
Last Updated: May 19, 2020

University of Sussex

Abstract

A website was made to introduce students to practical techniques and knowledge required to work on a neutrino particle physics experiments. A focus was placed on NOvA, a modern neutrino experiment with aims to understand many of the mysteries of neutrinos. NOvA heavily uses Monte Carlo techniques and machine learning in its process, this was represented here. This was motivated by a desire to introduce students who previously only had a physics background to computing and informatics. Code used as examples and the website can be found at the Github repository `https://github.com/nlc11/Neutrino-Physics-Student-Website`.

Keywords: Neutrino, NOvA, Monte Carlo, machine learning, deep learning, convolutional neural network

Contents

1 Introduction

Modern particle physics is a multi-disciplinary field. In order to do anything we first need to have a background in theoretical particle physics, then simulate this (often using Monte Carlo techniques), then analyse this simulation (increasingly using machine learning, especially deep learning, techniques), before finally constructing, running, analysing, interpreting and theorising some more about the experiment. Students and physicists hoping to enter into particle physics experiments generally require knowledge around these topics. This is a large amount of knowledge, and a scientist with a physics background is not guaranteed to have ever studied Monte Carlo techniques or machine learning (or even particle physics in sufficient detail).

The author's supervisor had a recurring problem of introducing new students to NOvA - it required a polymath approach that few physics students had: deep understanding of neutrino physics, backgrounds on other neutrino experiments, machine learning and simulation. At present, no known resource was able to quickly and efficiently give a student a sufficient background to begin work on NOvA. As such a requirement for an easy to approach, easy to use, portable, comprehensive tool to teach students required skills for operating on NOvA was required.

By far, the best choice for this was a website, written in simple HTML. This format has several advantages - being a standard format, all computers are able to view HTML out of the box and it is unlikely that this will change. Updates will be very unlikely to cause compatibility issues. It is easy to edit when knowledge about neutrinos inevitably changes. It can be accessed in multiple ways - either hosted online for a wide audience to view or given to a student as .html files to view offline.

Anecdotally, students need to spend four or five months reviewing text on neutrino physics, machine learning or particle simulation techniques before approaching NOvA; this time could be reduced significantly with use of this website. Common hurdles that a student will inevitably face such as setting up Tensorflow can also be addressed, saving significant time in experimenting with uncooperative technology.

NOvA is the experiment most closely worked with. This is a Fermilab experiment in America. Neutrinos are produced in Chicago, where they are initially detected by the near detector. These neutrinos then travel through the earth to Northern Minnesota, where they are then again detected again. In this way, neutrino flavour oscillation can be accurately measured. Broadly NOvA has three goals: to understand neutrino oscillation, to understand neutrino mass organisation and to understand the symmetry between matter and anti-matter [1]. More finely, its goals are to understand the mass hierarchy by determining if ν_3 has more or less mass than the other two states. Measure the amount of CP violation present, a measurement of δ. Measure the PMNS mixing angle θ_{13}. Determine if the ν_3 state has more ν_μ or ν_τ admixture (whether $\theta_{23} > \frac{\pi}{4}$ or $< \frac{\pi}{4}$) [2]. The goal is to create a tool that summarises the physics up to this point and allows a student to aid in the discovery of new physics.

First a literature review of neutrinos, particle physics detector technology, deep learning and Monte Carlo Methods is given. Then a discussion of the website and design choices is given. The .html files can be found at `https://github.com/nlc11/Neutrino-Physics-Student-Website`.

2 Background

2.1 Neutrino

2.1.1 Early History

Neutrinos are the lightest fundamental particle currently known, and are the key to physics beyond the standard model. The neutrino was proposed in 1930 by Pauli to solve the problem of observing a continuous energy spectra for electrons in the β decay [3]. An electron emitted from a β decay should have a fixed kinematic energy equal to the released energy, but it actually has a continuous spectrum of energy, with the "missing" energy being carried away by the neutrino. In figure 1 the expected energy spectrum can be seen, but we observe the continuous spectrum instead. Note how the continuous spectrum ends at the expected energy.

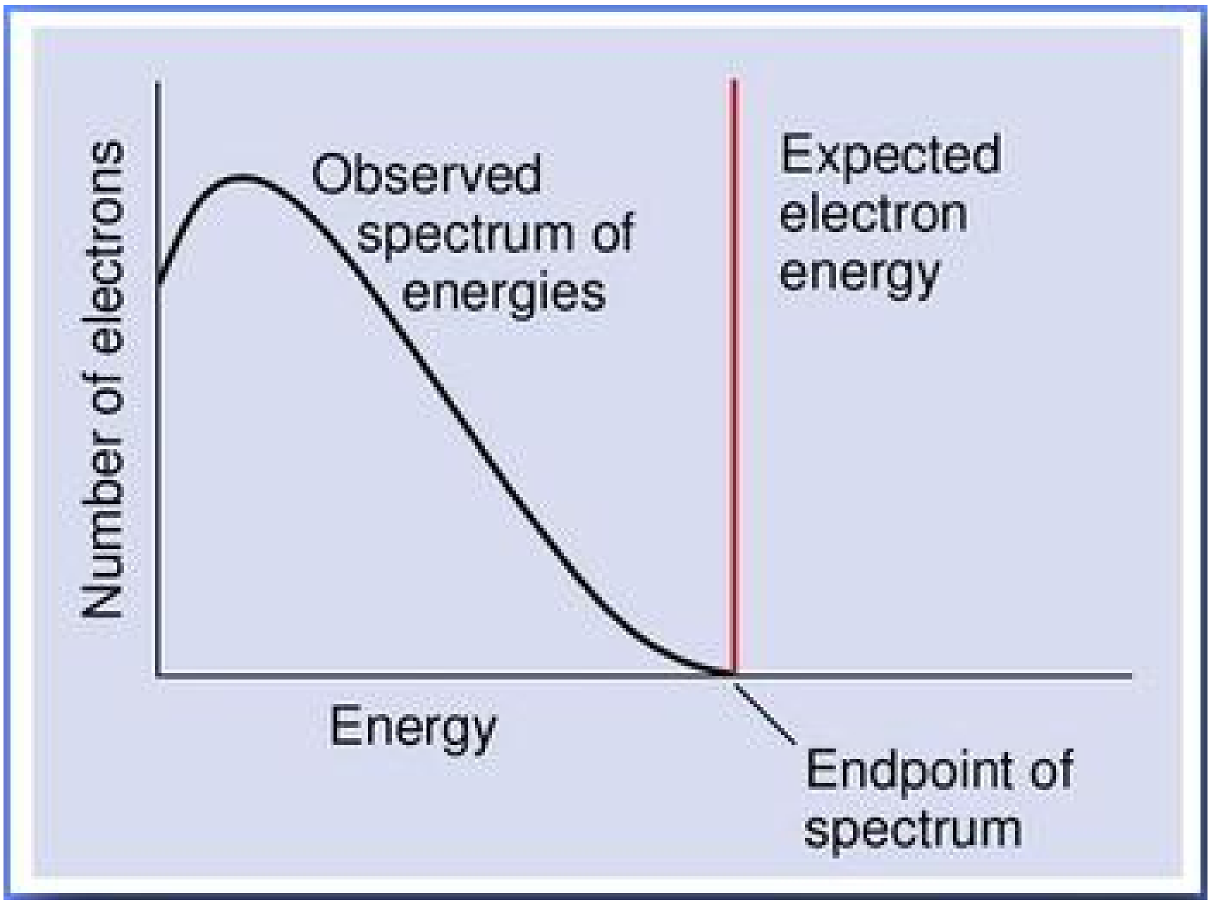


Figure 1: Energy spectrum of expected and observed β decays [4]

Fermi then developed this theory further, and formalised that in β decay, an electron and a neutral, light, spin 1/2 particle was emitted like so

$$n \to p + e^- + \bar{\nu}.$$

2.1.2 Oscillations

ν_e, ν_μ, ν_τ do not have definite masses but they are a linear combination of ν_1, ν_2, ν_3. ν_1, ν_2, ν_3 are the wave equations which oscillate, their interaction and mixing give rise to the neutrino particle as we know it. They do have definite masses, m_1, m_2, m_3. For brevity we will only consider the interaction of two, but this extrapolates to three.

$$\nu_\alpha = \nu_i \cos\theta_{ij} + \nu_j \sin\theta_{ij}$$

$$\nu_\beta = \nu_i \sin\theta_{ij} + \nu_j \cos\theta_{ij}$$

If $\theta_{ij} \neq 0$ then ν_i, ν_j have different energies due to different masses and the mixing waves will have different frequencies. This gives rise to a "beats" phenomenon that results in a beam of original ν_α developing a ν_β component [5].

Neutrino oscillation was first proposed by Pontecorvo in 1957 [6], but it was the Homestake experiment and the *solar neutrino problem* that began the modern understanding of neutrino oscillations. The Homestake experiment began in 1968 and lasted over 20 years. Solar neutrinos were observed through the reaction $\nu_e +^{37} Cl \rightarrow e^- +^{37} Ar$ inside a huge tank of C_2Cl_4 which produced argon every few days. It aimed to find the solar neutrino flux in the unit of SNU (solar neutrino unit), which is one capture event per second for every 10^{36} target atoms. The flux was finally determined to be $2.55 \pm 0.17 \pm 0.18 \;\; SNU$ [5]. This is significantly lower than the expected $7.3 \pm 2.3 \;\; SNU$ predicted from the solar neutrino model. Thus began the solar neutrino problem. Since, at the time, the only measurement of solar neutrinos were from the Homestake experiment, it was presumed that there might not be a problem and instead this was a discrepancy in uncertainty of calculation, measurement or both. However, later experiments of SAGE, GALLAX, Kamiokande and Super-Kamiokande have confirmed said deficit.

Later, SNO was able to prove that neutrino oscillations had indeed occurred. SNO used a large amount of heavy water which was used as both a target and a medium for Cherenkov radiation. The detector was sensitive to following reactions [7]:

$$\nu_x + e^- \rightarrow \nu_x + e^-$$

$$\nu_e + d \rightarrow p + P + e^-$$

$$\nu_x + d \rightarrow p + n + \nu_x{}'$$

By observing electron neutrino flux and flavour independent flux SNO was able to prove the existence of neutrino oscillations, thus solving the solar neutrino problem [6].

There has also been work done on atmospheric neutrinos. Atmospheric neutrinos are produced in the reactions $\pi^+ \rightarrow \mu^+ + \nu_\mu$ followed by $\mu^+ \rightarrow e^+ + \bar{\nu}_\mu + \nu_e$ [8]. Super-Kamiokande used the fact that we expect to see 2 ν_μ for every ν_e as the basis of its investigation. We actually find 1.3 ν_μ for every ν_e, implying a possible oscillation from muon neutrinos into other flavours. This was confirmed by exploiting the fact that the experiment can measure the direction of incoming neutrinos, and explore the relation between the ratio of muon to electron neutrinos and the azimuthal angle. The transition probabilities of a neutrino is given by [8]

$$P(\nu_\alpha \rightarrow \nu_\beta) = \sin^2 2\theta \sin^2 1.27 \frac{L}{E_\nu} \Delta m^2$$

where θ is the mixing angle, L is the distance from the neutrino source (in KM), E_ν is the energy of the neutrino (in GeV) and Δm^2 is the difference of mass squares. Being able to detect neutrinos from above and below the detector is valuable, as this varies the distance from the neutrino source.

2.1.3 Mass

Neither the mass nor mass hierarchy of neutrinos are known. While ν_e, ν_μ, ν_τ do not have definite masses, they are composed of ν_1, ν_2, ν_3 which have mass m_1, m_2, m_3. Neutrino mass experiments are sensitive to the squared difference of masses ($\Delta m_{ij}^2 = m_i^2 - m_j^2$). At current, we have $\Delta m_{21}^2 \approx 7.6 \times 10^{-5}\ \ eV^2$ and $\mid \Delta m_{31}^2 \mid \approx 2.5 \times 10^{-3}\ \ eV^2$ [9]. Only the absolute value of Δm_{31}^2 is known, as atmospheric mass splitting is only measured by neutrino oscillations in a vacuum.

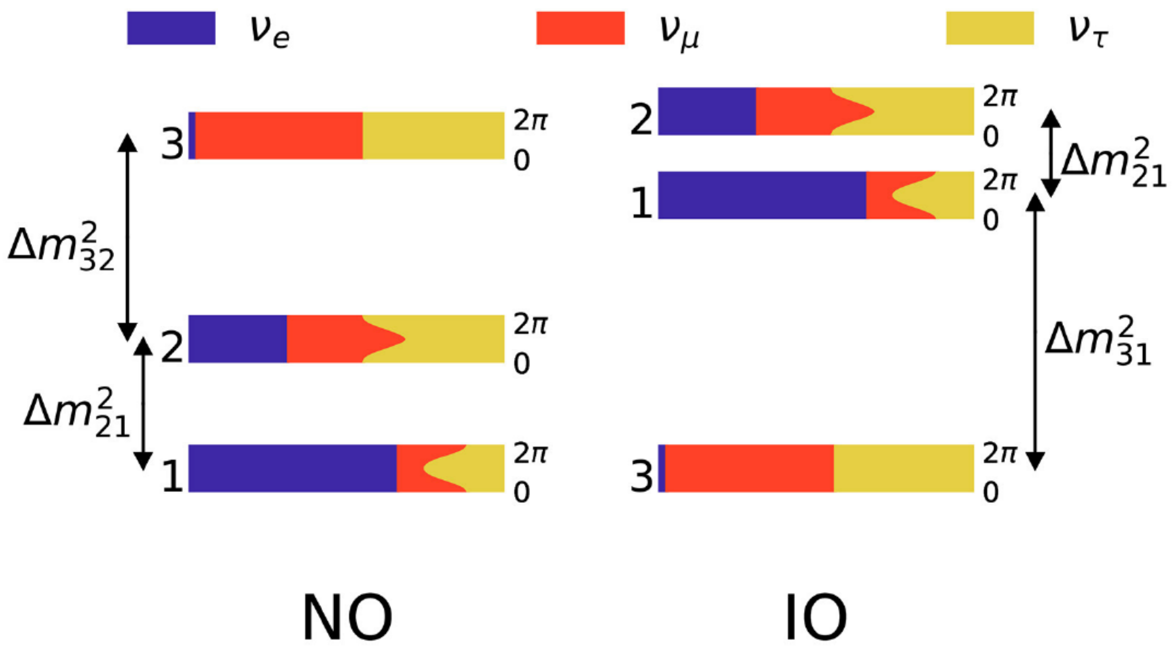


Figure 2: Possible mass hierarchy solutions with probability of finding a neutrino flavour in a given mass eigenstate [9]

The consequences of only having the magnitude of Δm_{31}^2 means we have a hierarchy problem. These are the normal mass hierarchy of $m_3 > m_2 > m_1$ or the inverted mass hierarchy of $m_2 > m_1 > m_3$. While there has been a mild experimental preference for the normal mass ordering, more recent data from Super-Kamiokande has placed a strong favour for the normal mass ordering [9]. There is also a minimum mass sum of all three mass states ($\sum m_\nu$) of 0.06 eV for normal mass ordering or 0.1 eV for inverted mass ordering. Therefore, there are two possible ways to find mass ordering, either by finding the sign of Δm_{32}^2 or by determining if the minimum sum mass is below or above 0.1 eV.

2.1.4 Dirac vs Majorana Particle and the Double β Decay

It is possible for a small number of elements to undergo a neutrinoless double β decay in the process

$$(Z, A) \rightarrow (Z + 2, A) + 2e^-,$$

however, this is only possible if the neutrino is its own antiparticle (Majorana neutrino). Dirac neutrinos are forbidden as it violates lepton number conservation [10]. Double β decay has been observed since 1987, and can only occur if single β decay is forbidden. A neutrinoless double β decay has not yet been observed, but SNO+ hopes to observe one (assuming that the neutrino is Majorana).

Neutrinoless double β decay can be distinguished from double β decay since the electrons will not have a continuous energy spectra, but rather have a single peak; since there is no other particle to take away the energy.

There is a significant problem with neutrinoless double β decay being that it has a very long expected half life. For example, the estimated half life of ^{130}Te which is being used in the SNO+ detector is over 9×10^{25} years [11].

2.1.5 Dark Matter, Early Universe and Sterile Neutrinos

Dark matter and early universe neutrino physics revolves around the concept of the sterile neutrino. The sterile neutrino is a hypothesised neutrino particle that does not interact with the weak force, may interact with neutrinos in some way, be much more massive than neutrinos, and there could be an unknown number. In general, the postulation of the sterile neutrino is to answer some of the following questions

- What is dark matter composed of?
- How was dark matter made? How did it come to permeate the universe?
- What causes the baryonic asymmetry between matter and anti-matter, leading to the modern matter universe?
- Why has a right handed neutrino (or left handed anti-neutrino) not been observed?

Dark matter as a concept originated when Zwicky noticed a discrepancy between rotational velocity of a galaxy and the mass as inferred by visible light [12]. However it was not until later when the difference between the predicted and observed rotational velocities was observed that the problem became something of major consideration [13]. A more complete history of dark matter can be found in [14].

The matter density of neutrinos when non-relativistic in the universe at large is determined by the sum of neutrino masses given by

$$\rho_\nu \approx n_\alpha \sum m_i$$

where m_i represents the neutrino mass eigenstates and n_α is the number density of neutrinos given by

$$n_{alpha} = \frac{6\zeta(3)}{4\pi^2} T_\nu^3$$

where ζ represents the interaction strength and T_ν represents the decoupling temperature. This sum must have a mass of at least 11.5 eV for conventional neutrinos to be all of dark matter [15]. However, the absolute upper limit on neutrino eigenstate mass sum is 0.15 eV [9], implying that there is more dark matter to be found. One solution is to postulate the existence of a massive sterile neutrino; if such a particle exists with masses in the keV range, then this would adequately fulfil all dark matter needs.

Sterile neutrinos may also be able to help solve the "mass puzzle" that m_i are many orders of magnitude smaller than any other fermion mass. Λ is introduced as a matrix of sterile neutrino masses, it is analogous to m_i, but since the number of sterile neutrinos is unknown it is unlabelled (in principle it can have one entry or more entries than are current known fermions). However, the existence of the mass puzzle and the possibility of the sterile neutrino has lead to the postulation of the "seesaw" mechanism for neutrino masses. This is the idea that neutrino masses are so light because they do not get mass from the Higgs interaction like other fermions but rather interact with the sterile neutrino in some way. The seesaw name is given as an increase in Λ denotes a decrease in m_i.

The production of sterile neutrinos in the early universe is possible in one of three ways [16], bearing in mind that they do not feel the weak, strong or EM forces. Firstly they could mix with ordinary neutrinos and actually be produced by the weak interaction during this mixing. Secondly, higher energy states of neutrinos and sterile neutrinos may have different gauge interactions, and if the temperature of the early universe was high enough, sterile neutrinos could have been produced thermally by these new gauge interactions. Finally, sterile neutrinos could have been produced from the decay of heavy particles in the early universe.

2.2 Particle Physics Detector Technology

2.2.1 Natural Sources of Neutrino Radiation

There are several naturally occurring sources of radiation available to a particle physicist. While these tend to be abundant and free to use, the energy ranges might be inconvenient or they are unpredictable. Early particle experiments tend to use naturally occurring sources of radiation, but as time goes on, more precise measurements with neutrinos produced in labs or reactors are used.

Cosmic rays are not yet fully understood, but we do understand that 90% are protons, 9% alpha particles and the rest have much heavier nuclei. They are usually relativistic and some have energies many orders of magnitude greater. Most of these appear to come from outside the solar system - therefore, only stable or very long life particles and isotopes may arrive [19]. These cosmic rays interact in the atmosphere and produce a whole plethora of particles, many of these eventually have a neutrino or anti-neutrino as the decay product, for example

$$\pi^+ \to \mu^+ + \nu_\mu$$

$$\mu^+ \to e^+ + \nu_e + \bar{\nu}_\mu [8]$$

Solar neutrinos have been a useful source of neutrinos for many experiments. The sun's primary neutrino producing reaction is

$$4p \to He + 2e^+ + 2\nu_e$$

which releases 26.7 MeV per reaction, which neutrinos remove approximately 1%. This gives a flux at earth of $\Phi \approx 6 \times 10^{10}$[17]. This is a huge number of neutrinos, and these can be utilised in experiments, for example the Homestake experiment. Since neutrinos are unaffected as they travel through space by forces, and this gives information about the neutrino and the sun.

Super-Nova can provide a source of neutrinos, and neutrino detector experiments often form part of the SuperNova Early Warning System (SNEWS) [18]. In order to allow the formation of a neutron star, the energy radiated needs to be

$$\mathcal{E} = G_N \frac{M_{ns}^2}{R_{ns}} \, const \times 10^{53} erg$$

where G_N is the gravitational constant, M_{ns} is the neutrino star mass, R_{ns} is the neutrino star radius and 'const' is estimated to be around 3. Thus we estimate the fluence per neutrino type at earth to be

$$F = 2 \times 10^{11} cm^{-2} [17]$$

.

Beta decay was where the discovery of the neutrino began, and it still provides a source for neutrinos today. In general this is of the form

$$^A_ZX \rightarrow ^A_{Z\pm1} Y + e^-(e^+) + \bar{\nu}_e(\nu_e)$$

where either an electron and anti-neutrino or positron and neutrino are produced. These reactions are very common, but there is a rarer double β decay

$$^A_ZX \rightarrow ^A_{Z+2} Y + 2e^- + 2\bar{\nu}_e$$

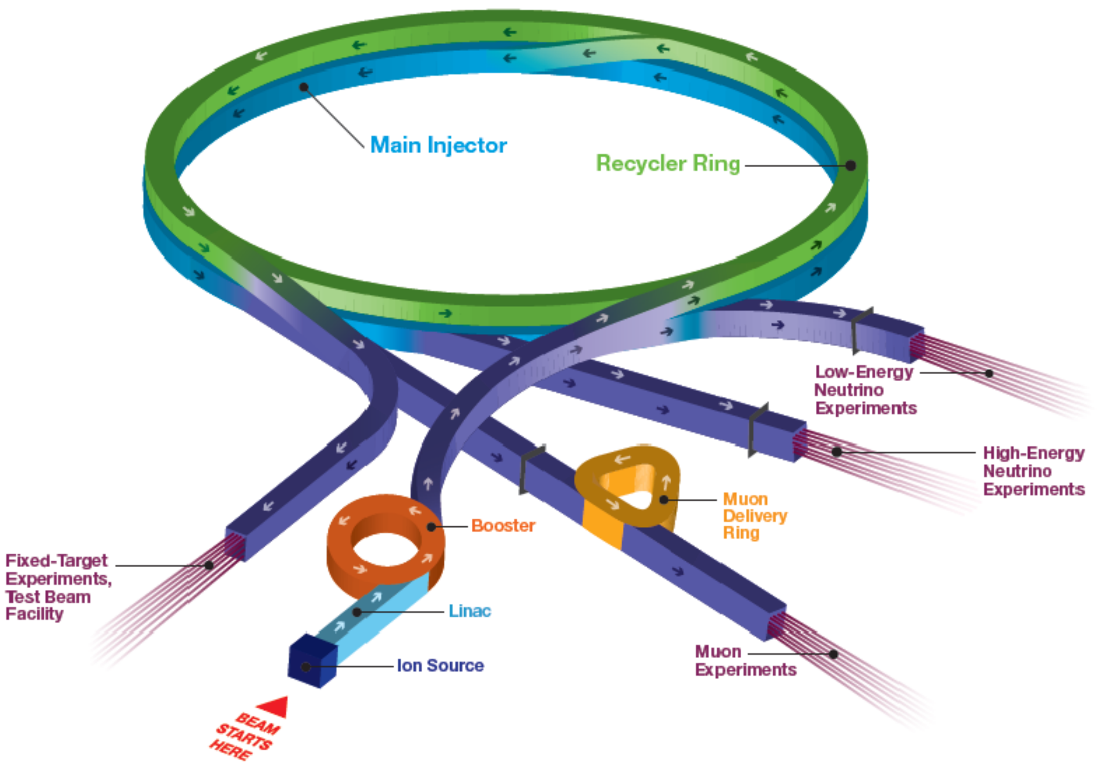
These decays are very rare, and so aren't used as a source of neutrinos but as something studied themselves. There is also a theorised neutrinoless version of the double $\beta\beta$ decay, which if possible shows that the neutrino is its own anti-particle.

2.2.2 Accelerators

Even though we have many sources of natural radiation available to us, we still need to be able to create our own sources. This is because we generally need higher energy particles that have a specific momentum rather than the slightly lower energy, more random sources we have naturally present. One of the early ways to do this was with a cyclotron.

There are several accelerator designs. Early accelerators were cyclotrons, where particles are accelerated in a circle by a voltage and a magnetic field. These were useful up to a point, but if the particle was made too fast, it would become relativistic, and become "out of step" with the magnet. Thus, their use was severely limited once particle experiment energy became relativistic. There are several things that can be done to alleviate this problem, mainly creating a cyclotron with a varying magnetic field. However, at a certain point the practical problem of creating magnetic fields strong enough becomes just a bit too much.

A good solution to this is to produce a synchrotron. They usually feature a linear pre-accelerator, and then a storage ring. While it may appear like a continuous accelerating ring, it is actually a series of alternating focusing and bending magnets. The focusing magnets keep the particles on a narrow path and the bending magnets make the particles follow the curved path. While the particles bend, they can produce a very high intensity beam of high energy radiation - usually in the X-ray spectra [20].



Main Injector
Recycler Ring
Low-Energy Neutrino Experiments
High-Energy Neutrino Experiments
Muon Delivery Ring
Booster
Fixed-Target Experiments, Test Beam Facility
Linac
Ion Source
Muon Experiments
BEAM STARTS HERE

Figure 3: Synchrotron present at Fermilab that produces neutrinos for NOvA, and other experiments [21]

In figure 3 we can see the synchrotron that produces neutrinos for NOvA. It features a linear accelerator and a storage ring, with four channels that particles can come out of. Recently, it was upgraded and can now deliver a 700kW beam to the NOvA experiment [21]. This will allow the NOvA experiment to meet its goals.

2.2.3 Energy Deposition

The detection of particles is possible because of the energy deposited in the detector. Consider a heavy particle of mass M and electric charge ze approaches an electron of mass m_e with speed v and a distance of closest approach b. The formula was derived semi-classically by Bohr and a full treatment can be found in [20].

Several assumptions need to be made:

- the electron is free,
- the electron is initially at rest,
- the movement of the electron is slow during the interaction time; the electric field is not disturbed,
- the path of the heavy particle is undisturbed.

Bohr's formula for this was

$$-\frac{dE}{dx} = \frac{z^2 e^4 N_e}{4\pi\epsilon_0^2 v^2 m_e} \ln\left(\frac{m_e v^2 \gamma^2}{2\pi\hbar\nu_e}\right)$$

where $\frac{dE}{dx}$ is the energy deposited in the material per length, N_e is the electron density of the material, ϵ_0 is the permittivity of free space, γ is the Lorentz factor, ν_e is the Compton frequency of an electron.

This is a good approximation for very heavy particles, and predicts that energy loss is proportional to the inverse square of the velocity, indicating a sharp drop off when the energy decreases, and then a logarithmic rise. However, this equation is only an approximation, and this was improved on with a quantum mechanical approximation.

The Bethe-Bloch formula describes how a heavy charged particle interacts with an electron in a medium. The quantum mechanical derivation is too complex for this work but can be found at [22] and the correction terms in [23], [24] and [25]

$$-\frac{dE}{dx} = 2\pi N_a r_e^2 m_e c^2 \rho \frac{Z}{a} \frac{z^2}{\beta^2} \left[\ln \left(\frac{2 m_e \gamma^2 v^2 W_{max}}{I^2} \right) - 2\beta^2 - \delta - 2\frac{C}{Z} \right]$$

where $\frac{dE}{dx}$ is the energy deposited in a distance, N_a is Avogadro's number, r_e is the classical electron radius (which of course is a nonsense, but the *true* classical mechanical equation is yet to be found, and this is a remarkable approximation), m_e is the mass of an electron, c is the speed of light, Z is the atomic number, A is the atomic mass number, ρ is the density of the material, z is the charge of the particle, γ is the Lorentz factor of the particle, v is the velocity of the particle, W_{max} is the maximum energy transfer in a single interaction, I is the mean excitation potential (minimum quantum mechanical energy level), β is the ration of velocity to c, δ is the density correction and C is the shell correction. W_{max} is given by

$$W_{max} = \frac{2 m_e c^2 (\beta\gamma)^2}{1 + 2\frac{m_e}{M}\sqrt{1 + (\beta\gamma)^2} + \left(\frac{m_e}{M}\right)^2}.$$

The density correction is for when the material is polarised by the incident particle, is only relevant at very high energy, and the shell correction is for slow moving particles when the assumption that the electron is not moving breaks down. Curiously, there is a minimum ionisation energy that will always take place - but the exact energy depends on the incident particle and the material.

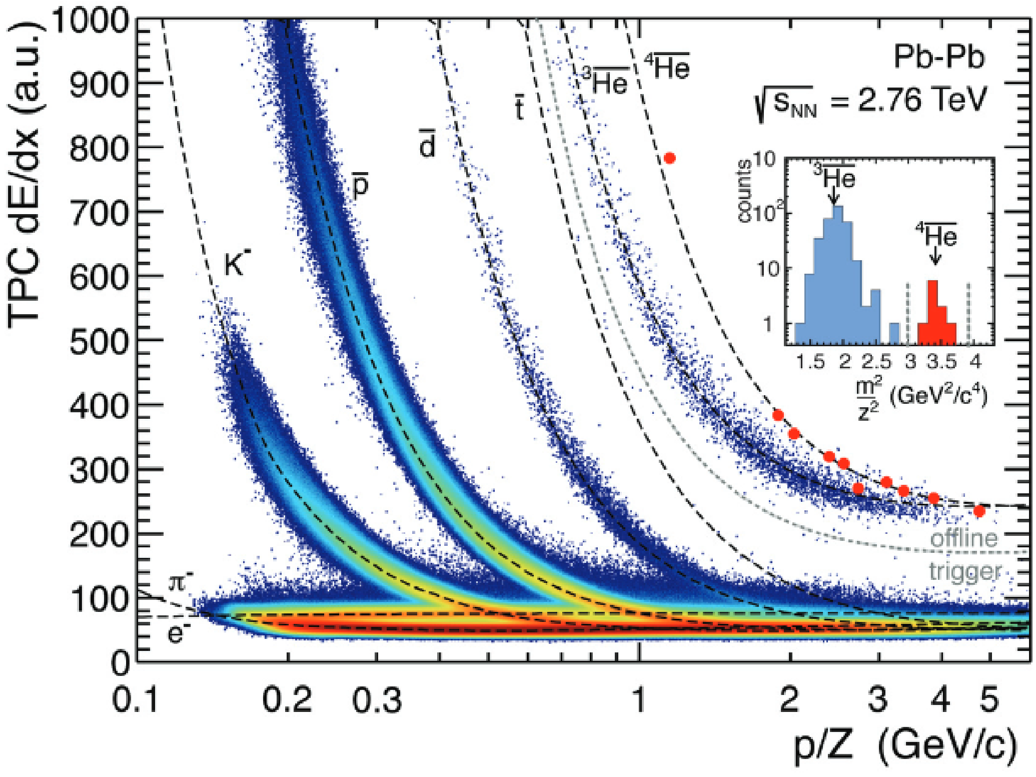


Figure 4: Ionisation energy plot at the ALICE detector at CERN [26]

Since

$$\beta\gamma = \frac{p}{Mc}$$

if the amount of deposited energy is measured and a measurement of momentum made, then the mass can be found. This was done exactly at ALICE as can be seen in figure 4, each curve is an ionisation energy representing a different particle in the detector.

We have so far described heavy particles in matter (e.g. protons, muons, alpha particles), but now we shall consider electrons. In this case the incident particle and the targets are of the same size. There are two effects - energy loss by collision (which we shall model with a modified Bethe-Bloch equation [27]) and Bremsstrahlung, which is the emission of photons. The Bethe-Bloch equation for electrons is

$$-\frac{dE}{dx} = 2\pi N_a r_e^2 m_e c^2 \rho \frac{Z}{a}\frac{1}{\beta^2}\left[\ln\left(\frac{\tau^2(\tau+2)}{2(I/m_e c^2)^2}\right) + F(\tau) - \delta - 2\frac{C}{Z}\right]$$

where τ is the kinetic energy of the particle (in units of $m_e c^2$ and $F(\tau)$ represents the difference between electrons and positrons. Also notice that z drops out of the equation (since electrons always have charge e).

The bremsstrahlung effect takes place when electrons are accelerated in the electric field of the nucleus, and emit a photon. It is proportional to the inverse square mass, and therefore is only relevant for the lightest particles (electrons, positrons and ultra-relativistic muons ($\beta > 0.99$). The effect is approximated by (for a more full discussion on bremsstrahlung see [28])

$$E(x) = E_0 x^{-x\rho/X_0}.$$

Since ionisation is dominant at low energies and bremsstrahlung for high energies, there is a critical energy E_c which they are equal, which can also be used to identify particles.

2.2.4 Scintillators

Scintillation converts energy from particles to light, usually near visible or visible light. The underlying phenomenon is fluorescence, which is the fast emission of light with a delay on the order of 10ns from the time of deposition of energy. Scintilators generally release photons in a pulse shape, described by the fast and slow parts of

$$N_{photons}(t) = Ae^{-t/\tau_A} + Be^{-t/\tau_B}$$

where A and B are normalised constants, τ_A and τ_B are the decay constants and $N_{photons}$ is the number of photons released, as a function of time. This gives scintillators a pulse shaped emission of photons, which varies depending on the incoming particle, allowing more information to be gathered from these peaks. Scintillators are also fast emitters of light.

Typical materials used for scintillation include inorganic crystals, organic plastics and gasses. Inorganic scintillators are crystals, such as NaI and CsI. Often the impurity is as important as the crystal in light production, so a small amount of Tl is added to facilitate this. The emission of photons occurs when electrons are excited into the conduction band. A more full description of the band structure of such crystals can be found in [29]. Organic scintillators are usually hydrocarbon compounds such as naphthalene or antracene. Here the light production is due to molecular processes, when electrons are excited and "drop" to emit a photon; a more thorough description of this can be found in (the excellently written) [30]. Finally, gas scintilators operate in much the same way as organic scintillators, but are obviously a gas rather than a liquid, usually an inert noble gas like argon.

NOvA uses a liquid scintillator in both the near and far detectors. 8.8 kt of the scintillator was used [31]. The scintillator is 95% mineral oil by mass, and blended in there is 1,2,4-trimethylbenzene, 2,5-diphenyloxazole and 1,4-bis-(o-methyl-styryl)-benzene (the active sctinillator), as well as small amounts of anti-static agents (for fire safety) and antioxidant (to prevent yellowing). As can be seen, quite a few chemicals go into a well optimised scintillator fluid. This scintillator converts energy from the particle interactions into photons that can be detected by the many, many photodetectors.

2.2.5 Cherenkov Radiation

Cherenkov raidation is similar in that it emits light, but this time it can take place in any liquid. If the speed of the particle is larger than the speed of light in the medium, then light is emitted. This process is analogous to the shockwave from a super-sonic jet. It produces a cone of light with an angle of

$$\cos\theta = \frac{1}{\beta n}$$

and the energy deposited per unit area is

$$\frac{dN}{dx} = 2\pi\alpha \sin^2\theta \left(\frac{1}{\lambda_1} - \frac{1}{\lambda_2}\right) [20].$$

There is therefore great capacity to differentiate between different particles based on cone angle and energy deposited rate.

2.3 Machine Learning and Deep Learning

Machine learning and deep learning are two current buzzwords used frequently. Indeed, they are frequent techniques used in the modern world, such as image recognition, speech recognition, translation, video suggestions, detecting fraudulent bank transactions and so on. These terms are often confused. Deep learning is a subset of machine learning where the algorithm is allowed to determine which features are used. This definition will be expanded. Weather a specific machine learning network has progressed enough to be considered "artificial" intelligence is a philosophical task far beyond they scope of this work, and to decide on this is left as an exercise to the reader

The different types of tasks a machine learning program might be expected to carry out include (but are not limited to) [32]

- **Classification**: In this task the program needs to specify which category an entity belongs. This is usually done by learning some function f that transforms data to one of k categories. This could be done in a supervised manner (with labelled training data) or unsupervised (no data labels - the program must create its own categories). Further complications can be made when some inputs are missing, and the program must learn n functions for each possibility of missing data. Common examples of classification tasks include categorising animals into dogs, cats, horses, &c,
- **Regression**: In this task the program is required to produce a numerical output from an input, usually by finding a function f that maps the input to the output. Common examples include predicting a salary based on education and experience,
- **Transcription**: In this task the program is required to change the representation of the data in some way, usually from some unstructured form to a more regular or standardised form. Common examples include character recognition from images. These tasks are now, usually, done with deep learning (for reasons that will later be discussed),
- **Translation**: The program is presented with a series of characters and must convert it to another series of characters. The most obvious example is natural language translation e.g. English to French,
- **Structured output**: A broad category, but involves produces a vector that contains relationships between data elements. The most common example is parsing where the program takes a sentence and labels words as nouns, verbs, the subject and so on. Another might be to label the roads in a satellite image,
- **Anomaly detection**: The program is shown a set of data and must identify which of these are anomalous in some way. The most used application by far is detecting fraud in banking,
- **Synthesis**: The program must generate samples that are close to those in the training sample. Examples include generating textures in video games rather than have an artist hand paint them,
- **Missing values**: In this task the program must predict values that are missing in data. For example trying to figure out education based on experince and salary in the salary prediction example above,
- **Denoising**: This task involves trying to eliminate noise or corruption from data. Examples include trying to deblur a photograph.

This isn't an exhaustive list of course, but aims to give some scope of the sorts of problems that machine learning techniques can solve. For the purposes of particle physics, at the time of writing, the most important aspect is classification, and we will limit the discussion to that. Classification is relevant to particle physics because a detector might get thousands of events per second, and hand classifying these events would be almost impossible.

2.3.1 Over and Under Fitting or Bias/Variance Trade off

An important concept to remember in machine learning is that the model is required to generalise to data that the model has not yet seen. Therefore, we have to balance bias and variance, or in other words, prevent over or under fitting.

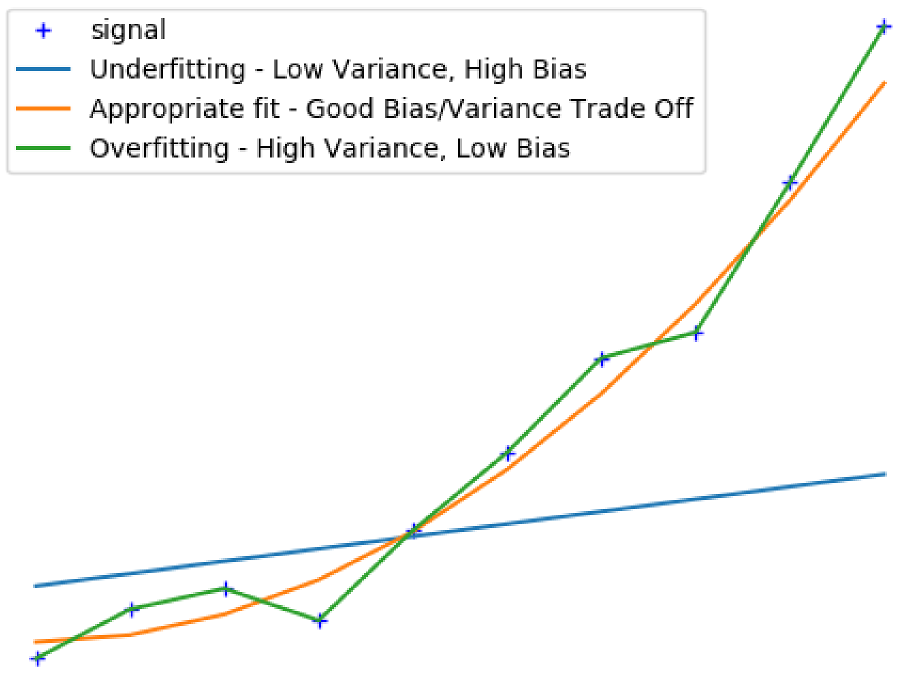


Figure 5: Simple Bias/Variance trade off example. The signal is quadratic with some noise. The blue line is an attempt at a linear fit, and largely ignores the data. The green line is highly overfitted function so that it traverses every data point. The orange line is an appropriate quadratic fit and will generalise well, despite not matching the training data exactly

High bias means that the model largely ignores the training data. This can be seen in figure 5. The blue line is a linear fit to an obviously quadratic signal. This will clearly not be a good representation of the data, and be a poor predictor. This is also called underfitting. Equally, high variance means that the data is represented very well by the model. Again in figure 5 the green line perfectly fits the signal. Unfortunately, this will not generalise to data that has not yet been seen by the model. This is also called overfitting. When overfitting has occurred, the pattern along with the noise has been learned by the model however the model should aim to only learn the pattern. Obviously in order to reduce the variance and thus generalise the model, we must introduce some bias. This can be seen again in figure 5 where the orange line has a good bias variance tradeoff. While it does not represent the data perfectly, it will be able to generalise into data not yet seen. This is an appropriate fit. This topic is developed in Neal et al [33].

For reasons for reasons outlined above it is important that over and under fitting are avoided when building a machine learning network. This can be achieved by showing the network a large sample of training data. Also by making sure that the sample is representative of what is being modelled. For example, if a neural network is built to separate cats from dogs, and the only example of dogs it is given is German shepherds, when shown a chihuahua it may misclassify as a cat. It is also important to not repeatedly train with the same data outside of an appropriate number of epochs to reach an error minimisation.

2.3.2 K-means

K-means is an unsupervised classification algorithm to group similar data. Imagine we have a series of points on a graph, we must begin by generating a series of "centroids" [34]

$$\mu_1^{(0)}, \mu_2^{(0)}, ..., \mu_K^{(0)},$$

where μ represents a centroid, the subscript represents a label and the superscript of 0 indicates that these are the 0^{th} iteration. We must then assign each datapoint to the nearest "centroid" by

$$C^{(t)}(j) \leftarrow arg \underset{i \epsilon 1,...,K}{min} \parallel \mu_i - x^j \parallel^2$$

Then we must move the centroids so that they are in the centre of the new clusters by

$$\mu_i^{(t+1)} \rightarrow \frac{1}{\mid \{j : C(j) = i\} \mid} \sum_{j:C(j)=i} x^j \;\; for \;\; all \;\; i\epsilon\{1, ..., K\}.$$

This algorithm is performed repeatedly until no change occurs.

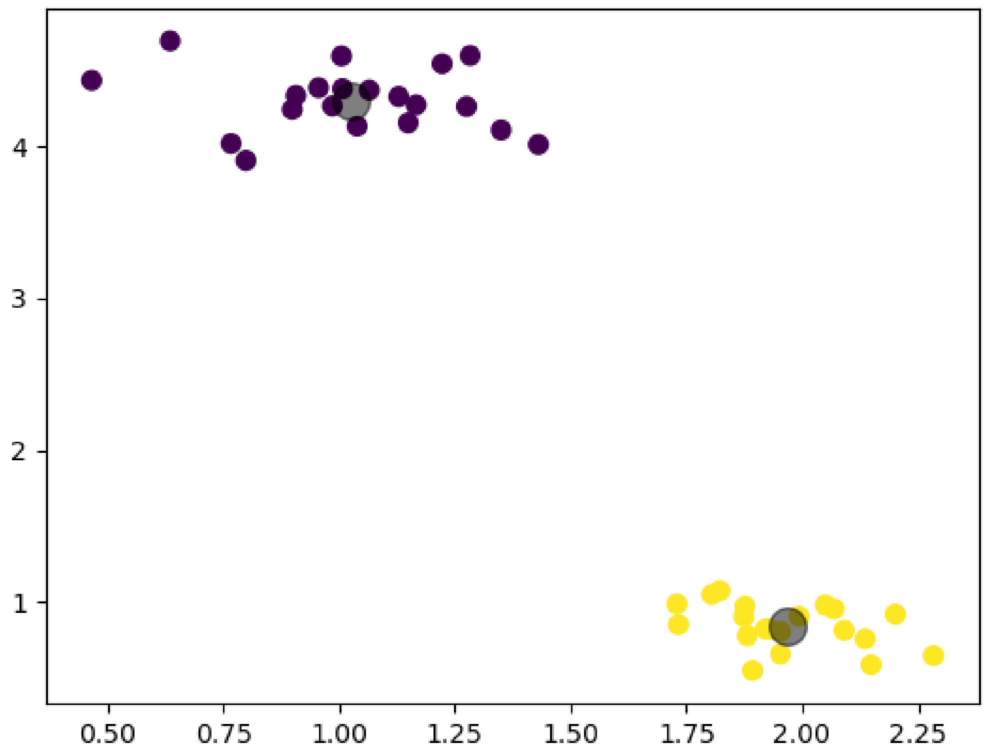
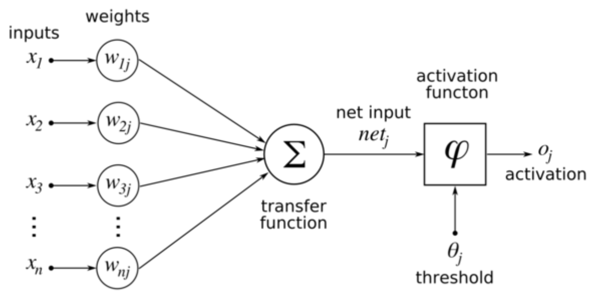


Figure 6: A simple example of a solved K-means algorithm

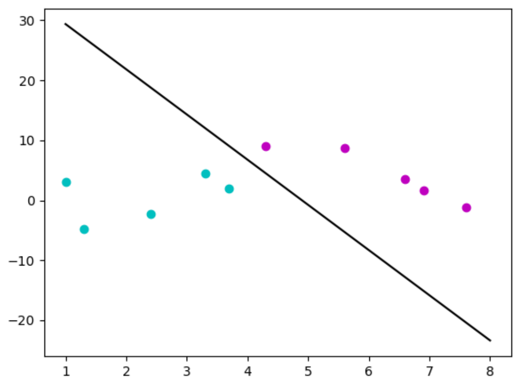
Figure 6 shows a solved K-means algorithm. The transparent grey circles represent the centroids, and the two clusters have been coloured. The colour only represents the solution; the data is unlabelled.

2.3.3 Perceptrons and Muli-Layers

Perceptrons were an early attempt at a classifier. Work was abandoned on them as they fail to tackle more complex tasks, but interest arose again when they were stacked, producing the multi-layered perceptron. The single layered perceptron is a linear classifier, meaning that it can classify between objects so long as there is a single straight line that can be drawn between them. The first layer of figure 7a is the input. This contains an initial value and a series of weights and biases. The weights assign importance to that value, and the bias allows the perceptron to have a separation boundary that does not pass through the origin of the feature space. There is then a sum function, taking the weighted sum of the weights, biases and inputs and then passing this to an activation function. If the sum is above some value, then the activation returns a value of 1, and below a value of zero, corresponding to the categories (for a two category classification problem). Figure 7b shows data that has been linearly separated with a single layered perceptron.



(a) Perceptron structure [35]



(b) Linearly separated data

Figure 7: Single layered perceptron and its solution to classification problems

There are really two mathematical steps in a perceptron. The first is to calculate the weighted sum of the inputs by

$$y_j(t) = f(\mathbf{w}(\mathbf{t}) \cdot \mathbf{x_j})$$

where $y_j(t)$ is the weighted sum of the input, $\mathbf{w}(\mathbf{t})$ is the weights and $\mathbf{x_j}$ are the features. And then we update the weights

$$w_i(t+1) = w_i(t) + (d_j - y_j(t))x_{j,i} \;\; for \;\; all \;\; features \;\; 0 \leq i \leq n$$

Where d_j is the desired output.

However, the linear perceptron has a fundamental flaw, in that it is impossible to differentiate data that is not linearly separable. Consider the XOR problem or two concentric circles, no one straight line can be drawn to categorise them. Therefore, a more complex mode is required. This prompts the multi-layered perceptron (MLP), which is capable of utilising multiple separation boundaries to build an arbitrary boundary. However, this introduces a new problem, that deciding how to modify the weights and biases becomes more difficult. The solution to this is a process known as backpropagation, and relies on using a cost function. The cost function is the square difference between the output and the desired output. A more detailed analysis of the backpropagation algorithm can be found in [36].

The main concept with "learning" when applied to machine learning is gradient descent. In effect, this is a calculus problem of minimisation, and the solution is the minimisation of the cost function. However, to differentiate the function would in general be far too complex and computationally taxing, as it might have several thousand variables that it will need to be differentiated with respect to (the weights and biases). Instead, the gradient is found and then a step in the opposite direction is taken (that is, in the direction of steepest descent). This gradient descent is taken on all possible training examples. A small step is taken, to prevent overshooting, and therefore the gradient descent must be done in several epochs in order to minimise the function. However, since training sets may be very large (containing at least several thousand examples for real world applications) keeping all data examples in computer memory becomes impossible. Instead, the training data is split into batches, and once all batches have had the gradient descent applied one epoch has passed. This is now stochastic gradient descent - since the path taken is slightly random each time depending on how the batches are made. This is why each time a neural network is trained with stochastic gradient descent, slightly different accuracies are found. While the path taken is not optimally efficient, it is a good approximation and after suitable epochs will still minimise the cost function.

2.3.4 Deep Learning

Deep learning is a specific type of machine learning that dramatically improves the performance of machine learning algorithms. Traditionally computers are used to do tasks that humans find challenging: large number multiplication and fast, high accuracy memory recall. However, computers tend to struggle at easy human tasks like understanding natural speech and recognising faces. This is because these tasks require a huge amount of "intuitive" knowledge that is very difficult for a computer to "learn" - indeed, programming the ability for a computer to recognise faces by hand would be an insurmountable task [37].

The solution is a computer program that is capable of teaching itself the intuitive knowledge by looking at many already known, and well indexed, cases, that is structured like the brain of an animal. That is, it has "neurons" in layers that modify the data in stages to detect something, maybe edges in images or patterns in data. The neural network is capable of learning by modifying how these layers interact with the data given to them. For our purposes we are using a convolutional neural network (CNN). In a CNN, the image is split into all possible N by N matrices and each of these is dot multiplied by the same N by N matrix filter to produce an output pixel. This constructs a new image that is in some way modified compared with the original.

2.3.5 CNN

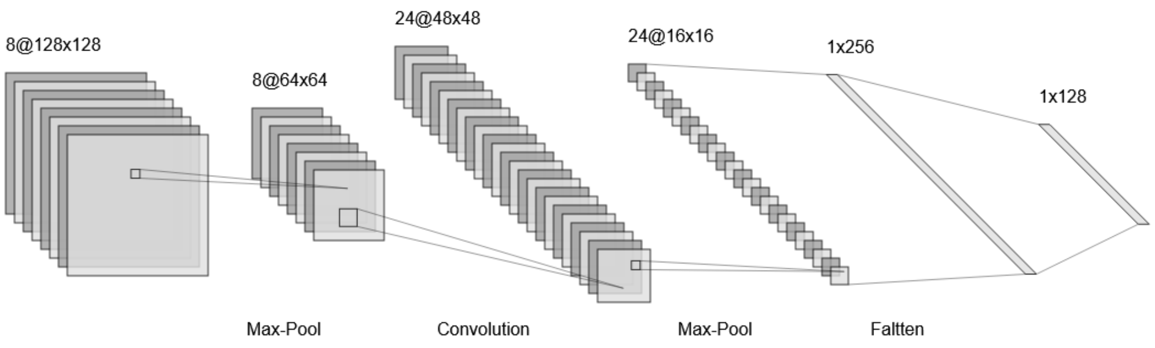


Figure 8: A Convolutional Neural Network [38]

A convolutional neural network (CNN) is one of the most common deep learning algorithms used for image recognition. Figure 8 shows the structure of a CNN, and it has a more complicated structure than a MLP. We start with an 8 channel, 128x128 pixel image. The first process performed is a max pooling layer. Max pooling is the process of placing a grid on the image, and selecting the largest value pixel from each cell. This creates a maximum pixel value image, but with a sharp dimensionality reduction. This process can be seen in figure 9. Several assumptions are made here about the types of pixels that contain information, which might not always be true [39], and so sometimes other pooling types are used like average pooling (where the average of each cell is taken).

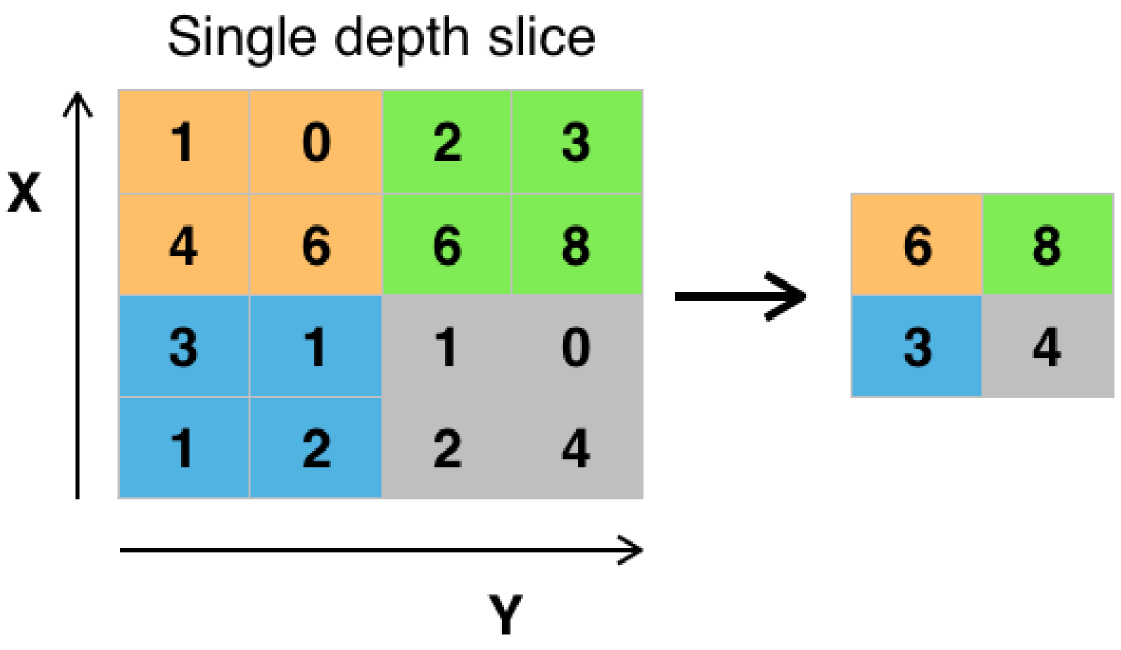


Figure 9: The process of max pooling [40]

There are also convolutional layers present, where the network gains its name. Convolutional layers actually allow the algorithm to detect patterns, by using a series of filters. Convolution is really another phrase for dot multiplication if the images are seen as a matrix.

Figure 10 demonstrates the convolution operation. A "filter" which is really an image which is really a matrix, is placed on top of the source image. These pixels are then convoluted (dot product taken) to produce one output number. This number is then placed in a new matrix which is smaller than the original image, in the corresponding place. The output of this operation is a new image, but modified in some way. For example, an "edge detector" filter might make pixels part of edges have a value very close to 1, while making non-edge pixels close to 0. In this way the edges have been "detected" by the convolution operation.

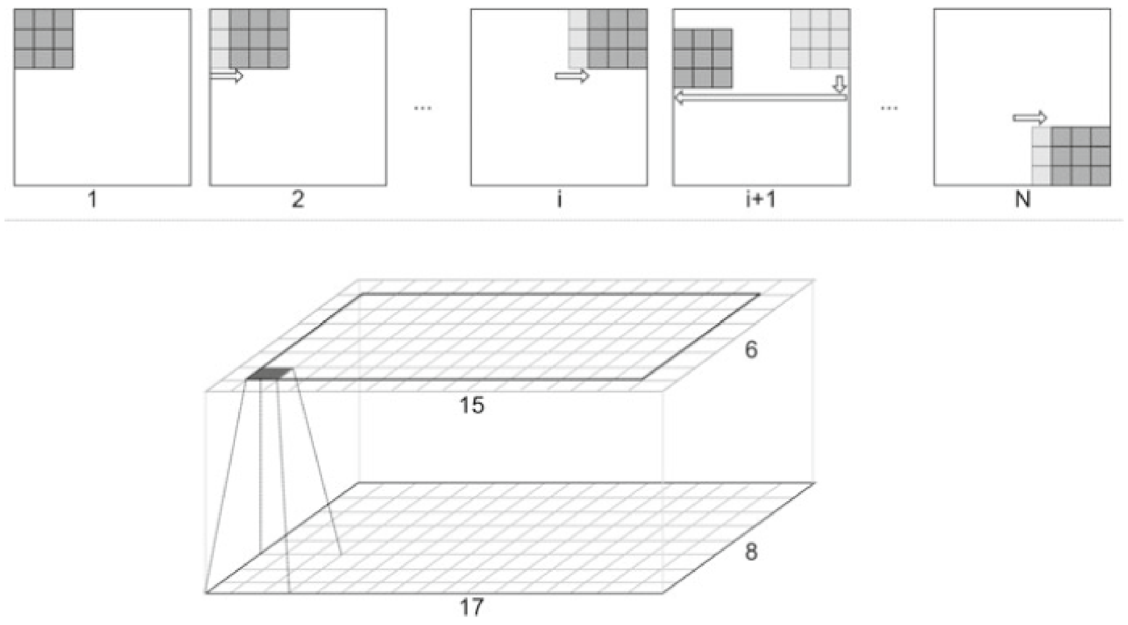


Figure 10: The process of convoluting filters with an image in a CNN [39]

Usually several stages of convolution and max pooling are performed, until the image is small, but has many channels. At this point the image is flattened (sometimes referred to as "dense"). This turns the images into a single vector. Now we fully connect it, and in some sense it begins to resemble a MLP again. At this point, ReLU activation is applied. ReLU is the rectified linear unit, and it simply takes an input, and if the input is less than 0, sets it to 0, and if the input is greater than 0, does not change it. Mathematically, this is represented as $f(x) = max(0, x)$. Finally, some kind of activation function is applied, usually softmax, which then allows classification of the original input. The softmax function essentially turns the fully connected layer into a set of probabilities. The node with the highest probability is the one the network as selected as the class of the original input.

Despite a MLP being able to theoretically do any task, CNNs are still used as they are easier to train. First, there are far, far fewer weights and biases. For example, a fully connected three layer MLP for MNIST would contain some 13,000 weights and biases, a five layer CNN with 3 by 3 filters has only 45 weights and 5 biases [39]. Also, it is possible to train CNNs in parallel, meaning that GPU technology can be utilised to speed up the training process significantly.

2.3.6 Machine Learning Methods for Particle Physics

The core problem with modern high energy physics is the correct identification of events within the detector. Even with shielding, we might have thousands of events per second, and sorting through all of these by hand is a cumbersome chore. The use of CNNs can make this much faster and easier. Also, since machines can perform these identification tasks at high rate, spending significant amounts of effort on shielding might become less and less necessary. Indeed, many neutrino experiments are very far underground, but this is not necessary for NOvA, as it has a machine find the relevant data from within the cosmic ray mess. For example, the UKs contribution to CERN per year is in excess of £150 million [41], and finding ways to reduce the costs of running experiments will become paramount as they grow in size and complexity. This also allows the detector to be multi-functioning for example also being able to act in a supernova early warning system, or any other number of unrelated physics goals. Again, this allows for a reduction in cost of operation and an increase in research.

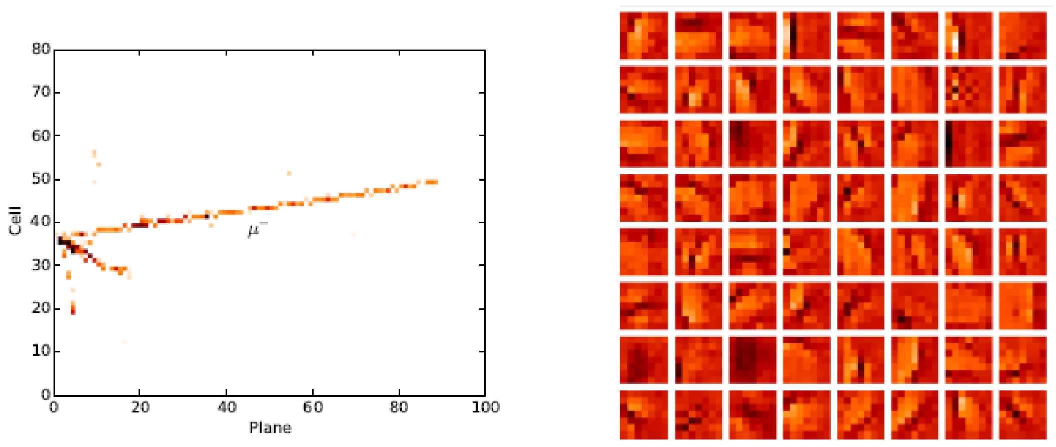


Figure 11: An example neutrino interaction (left) and convolutional filters (right) in a NOvA CNN training example [42]

For example, at NOvA, a CNN was trained on data generated by Monte Carlo simulations. This was done in the same way as has been described here; several thousand examples of interactions used to improve the filters. An example of a neutrino interaction and the filters used to convolute with that image can be seen in figure 11. An accuracy of nearly 70% was achieved, which is remarkable for such a complex task. This could be improved in future by using data from the detector rather than simulated data and continued optimisation of the structure of the CNN for this problem.

2.4 Monte Carlo Methods

Monte Carlo methods were first conceptualised by Stan Ulam when he attempted to find the probability that a game of solitaire is solvable [43]. His initial attempts at an analytic solution were unsuccessful due to the complexity of the problem. Instead, he simply played hundreds of games of solitaire to build the probability distribution. This is the core of the Monte Carlo method - to find probabilities by simulation rather than analytic calculation. Von Neumann thought about using the method with newly developed "fast computing machines" of the time. This was first applied to analysing the behaviour of neutrons in nuclear chain reactions. This research was used in the Manhattan project in the second world war, and was not publicised until the paper by Metropolis [44] in 1953.

2.4.1 Generating Random Numbers

Generating random numbers is very important for Monte Carlo methods. There are many times when it is necessary to generate random numbers, for example when estimating the value of pi, or in a particle interaction simulation. It is also very likely that numbers should be selected from some arbitrary distribution. Fortunately, it is relatively simple to do so. This begins by generating uniform random numbers between 0 and 1.

One such algorithm that will be used in this work is an older random number generator: the IBM RANDU. This generator produces randomly distributed numbers between 0 and 1. RANDU has a flaw as numbers generated in the tuple tend to align into planes [45], however, for the purposes of this work it is more than suitable and has the benefit of being very simple to implement. Please don't use it for cryptography though! The RANDU algorithm begins with a "random" seed number - this could be anything from a dice roll, the data and time, number of people in Times Square, a wall of lava lamps or solar neutrino data. The algorithm is repeatedly applied to continually generate random numbers.

$$n_{i+1} = (n_i * 65539) \;\; mod \;\; 2^{31}$$

$$x_i n_i / 2^{31}$$

where n is the index number and x is the randomly generated number [46]. A shrewd observer will notice that these numbers are not random but pseudo random. In general, all computer generated numbers are pseudo random; once we have the "seed", we have the whole series.

While uniformly distributed numbers are useful, an arbitary distribution is even more useful. Fortunately this is actually quite simple to do once the uniform numbers are generated. It is as simple as inverting the cumulative distribution function (CDF) of the desired distribution [47]. This works as the CDF is a one to one function between 0 and 1, which describes the probability distribution function.

2.4.2 Markov Chain

The Markov chain is probably one of the most powerful Monte Carlo techniques. Markov chains are defined as those that have no memory other than the current state. Let us imagine a scenario where we have states: A and B. If A, then there is a $\frac{1}{2}$ probability to remain A and a $\frac{1}{2}$ probability to become B. If B, then there is a $\frac{1}{2}$ probability to remain B and a $\frac{1}{2}$ probability to become A. In this very simple example, after many iterations we expect half of the time to be at A and half of the time at B. This becomes more difficult to calculate by hand if there are many states. Therefore to solve this problem it is common to use a "walker". A walker starts at some poistion and randomly moves around the system with the described probabilities. The locations of the walker are recorded and this builds the probability distribution for the system. This is a Markov process as the walker has no memory and the transition probabilities are determined only by the current location [48].

2.4.3 Particle Physics Applications

Monte Carlo use in particle physics normally comes in the form of several pre-built simulation packages like Pythia [49] for interactions in the LHC at CERN, GEANT4 [50] for the simulation of particles in solid objects and GENIE [51] for the simulation of neutrinos. These can often be used to generate data in a wide array of forms, useful for understanding what processes are to be looked for before the experiment begins. It is common to use data output from these simulations to train machine learning algorithms [42]. The Monte Carlo method in particle physics is often done in two stages [52] - event generation and detector simulation. If a theory that the probability of some event occurring is linked to a measurable quantity, first vectors of the measurable quantities of the interaction are produced, and then these are used as input for the detector. In the case of NOvA, this would be neutrinos leaving the Fermilab accelerator and then simulated inside the main detector - the output being an image so that it can train a CNN.

3 Methods

3.1 Website

In the modern world, website production is very simple, which allows for more time to be focused on content rather than developing. The open-source software Brackets was used for the website itself. An open source HTML and CSS template was used to form the backbone of the project, but it was modified extensively of course. The website was made in a Wiki style, since many popular websites use this style and would be familiar to users. Finally MathJax was used to allow LaTeXto be used in the website. Thankfully MathJax implementation is simple as inserting two lines at the top of the HTML document. Initially, mathematics was being implemented by use of images made in LibreOffice Math, but this method was far too clunky and resulted in a larger file size.

3.2 Particle Physics Detector Technology

Unfortunately, due to the nature of this topic, no practical tutorials could be performed. A general overview of section 2.2 was given, with the hope of introducing the topic of particle detectors to the student. Topics that are directly relevant to neutrino experiments were included so that an understanding of these specific topics could be given in a short overview. In general, this section is less relevant to a final year physics student, as they tend to work with data from established experiments rather than design a new one from scratch.

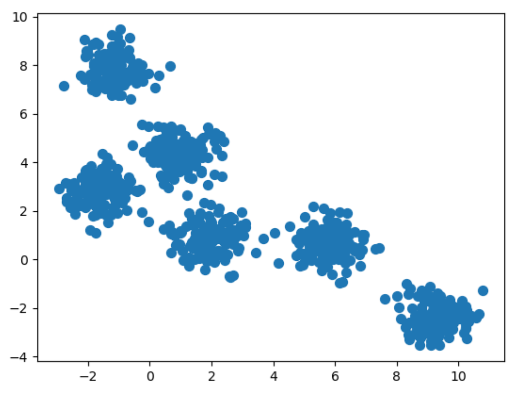
3.3 Machine Learning

Machine learning is a rather large field with more aspects than most people imagine. While the ability for machine learning to drive our modern world can clearly be seen online in image recognition and product suggestions, machine learning has many more techniques available to us. For example we might like to cluster data into groups or draw line boundaries between datasets using machine learning techniques. Machine learning was performed in Python, using Anaconda. Python and Anaconda have several advantages. Firstly, Python is easy to learn and beginner friendly, and many physicists working with machine learning have this as their choice of language. Secondly, Python has several packages that are powerful and easy to learn that make tasks related to machine learning simple. Namely Scikit-learn for machine learning applications and Matplotlib for making graphs. Anaconda is a Python package that makes package management much easier. While installing and using Tensorflow and Keras (two deep learning Python packages) is very complex and took roughly one week to get working on the first attempt, Anaconda can install these packages in one command. The ease of use is staggering compared with manual attempts.

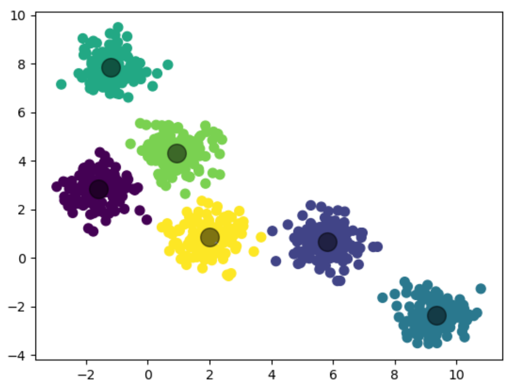
3.3.1 K-Means

A K-means tutorial was created. It began with images of data points, and asking the reader how they would be grouped. Some potential example groupings are shown; the point is to show that grouping is potentially somewhat subjective and sensitive to what K parameter is used. The K parameter is free, and is a prior. Potentially the problem might show what K should be, such as clustering genders (male/female). However, this might not always be apparent, and it might have to be determined through trail and error, reason or prior research. Then a graphical demonstration of the technique is provided, to allow the user to understand the principle concept before attempting to actually perform one. This, anecdotally, seems to be the way physics and mathematics textbooks are made. Then the mathematical background is given. This is because this is not strictly necessary to follow the tutorial, but students have a tendency to skip important background mathematics in favour of getting to the primary results. Then the actual K-means tutorial is presented. The structure of the tutorial is the student is provided a snippet of code at a time, and then explain *what* and *why*. This allows the user to see what code is needed to perform a K-means test, but also why that code is necessary and does what it does. This allows them to not only copy, but develop an intuition for K-means to be able to use the technique on their own.

The task that is used is to generate some clusters of data using Scikitlean, produce a plot, perform the correct K-means algorithm on that data, and produce a plot with the points colour co-ordinated based on which grouping they belong to.



(a) Raw data. The clusters here are obvious, but the exercise is rewarding



(b) Solved K-means. The grey circles represent the centroids. Data points are coloured based on which cluster they belong to

Figure 12: K-means tutorial goals

After, the user is encouraged to change variables in the data such as as standard deviation and K values, to explore how this changes the plots. The purpose of this is to show that if data is generated with a K of e.g. 4 and has a large distribution, the wrong K e.g. 6 can be made to fit, as can be seen in figure 13.

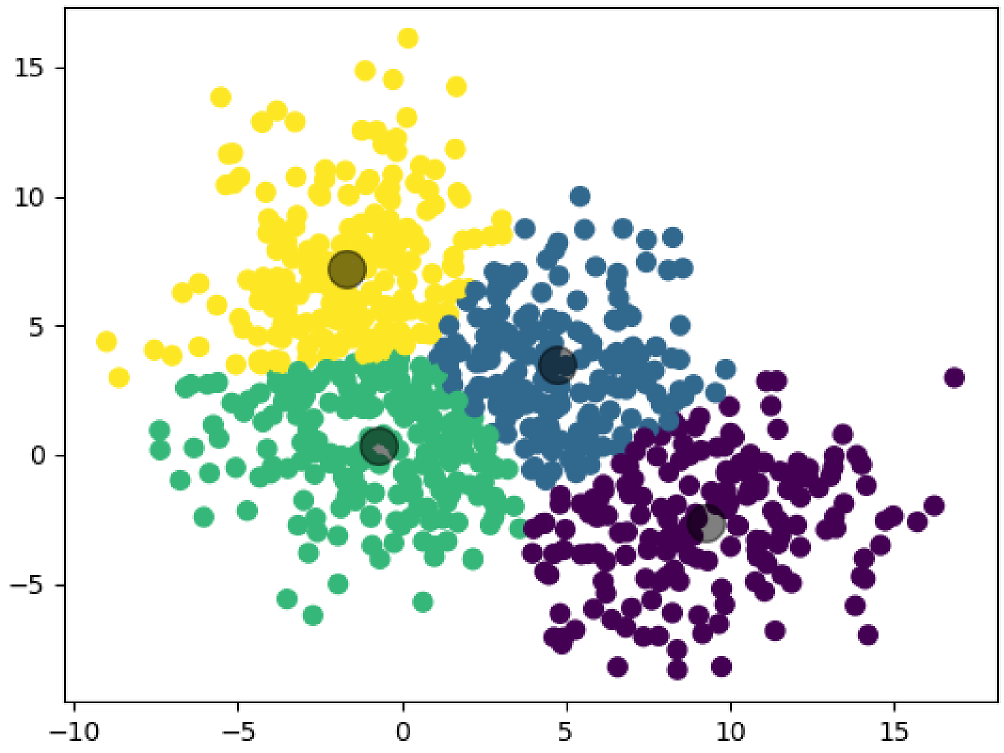


Figure 13: K-means generated with K=6 and a a high standard deviation, but fitted with K=4. Notice how no apparent clusters can be seen, and K=n would appear to fit

3.3.2 Perceptron

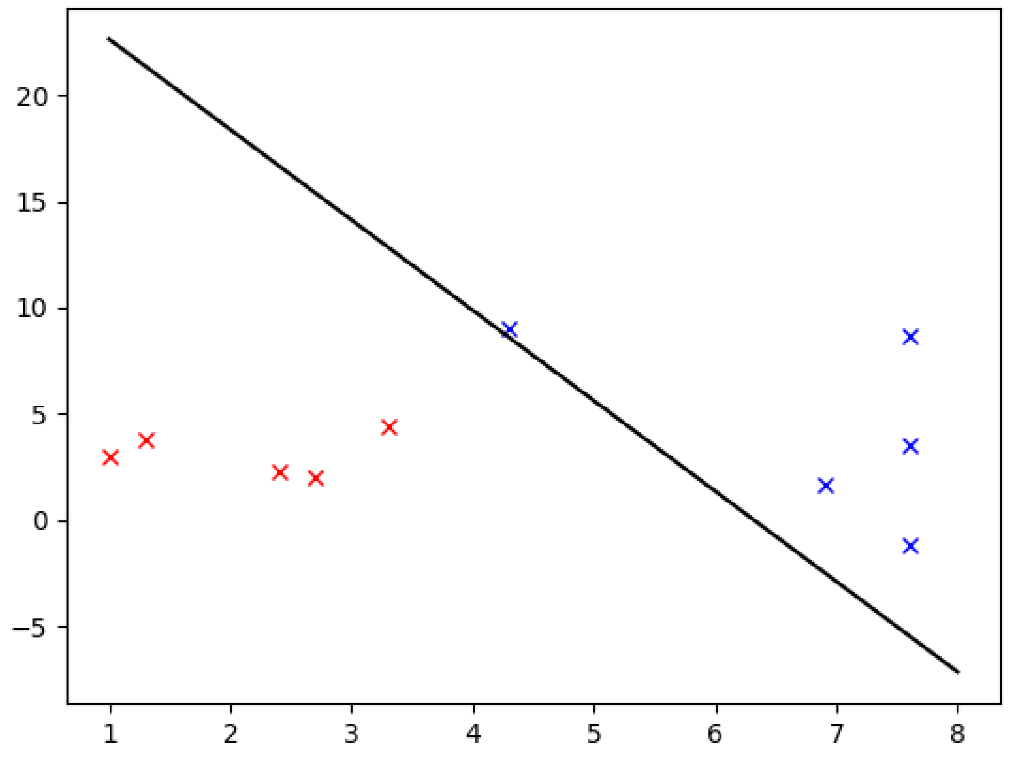


Figure 14: Linear separation of two classes with a single layered perceptron

The student is shown the basic idea of how perceptrons work and the mathematical background. A focus is placed on a perceptron's limitation of only being a linear separator, which will later prompt the creation of more advanced programs, for example to make an XOR solver. The perceptron is the most basic element of machine learning, and later the student will understand how to stack them. The student is then shown how to code one from scratch in Python. Some example data is created to demonstrate the model working. The student is shown how to make a graph like the one in figure 14. This graph shows two classes which have a straight line separating them, the line has been determined by the algorithm, and it neatly classifies all points.

3.3.3 Multi Layered Perceptron

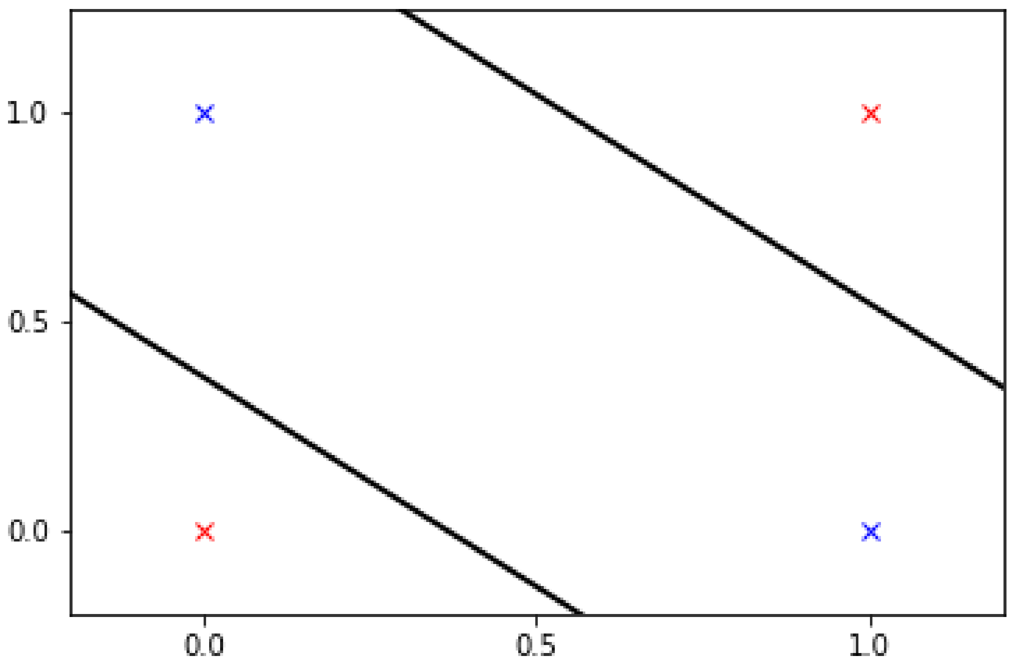


Figure 15: One possible solution to the XOR problem found with a MLP

Building on from the single layered perceptron the student is shown how to produce the multi layered perceptron. The student is shown why a multi layered perceptron is able to tackle the XOR problem that the single layered perceptron is not able to. Now, Tensorflow is introduced as even coding a full two layered perceptron from scratch is a lengthy task and outside the scope of this website. The XOR problem is set up and the student shown how to solve this using Tensorflow. The student is shown how to produce graphs like those in figure 15. In the graph, it can be seen that there are now two lines found by the algorithm, with one class being inside the boundary of the lines and the other class being outside the boundary. In this way we have overcome the limitations of the single layered perceptron.

3.3.4 Convolutional Neural Network

CNNs are more complicated than perceptrons by a significant degree, so some text in the website is dedicated to explaining how they differ. The convolution function is explained, the concept of filters, and the process of max pooling. It is then exaplained why CNNs are suited to image recognition, and why they are used.

The student is introduced to Keras, a Tensorflow API that makes creating machine learning programs even easier, which is essential for concise tutorials considering how complicated a CNN can be. The student is introduced to the digit recognition problem and the MNIST data set, which contains tens of thousands of labelled examples of handwritten digits. The student is shown how to import and process this data and how to build a CNN. In the end, a CNN with an accuracy of 99% is made and the student is shown how to use it to recognise their own handwritten digits.

3.3.5 NOvA Theory

NOvA uses CNNs to sift through their massive amounts of data and identify neutrino events. The problem is much more complicated than image recognition. For several reasons recreating the same or similar CNN to that used by NOvA is impossible or undesirably. First, large amounts of graphical particle interactions labelled in a database are generally unavailable. Second, generating such data would be a mammoth task outside the scope of this website. This task would require GEANT and GENIE packages, and at NOvA these simulations are done by numerous persons together, as the scope and complexity of the operation is huge. Finally, training such a CNN would be very long indeed. Even with two GPUs specifically designed for the task of machine learning, it took NOvA over a week to train it. As such, the process of [42] is explained, making reference to tasks performed already in the website.

3.4 Monte Carlo

3.4.1 Random Number Generators

One of the most fundamental processes used in Monte Carlo techniques is the generation of random numbers. Since Monte Carlo techniques largely reduce to estimating a distribution by building a histogram, the ability to create any distribution randomly is obviously very useful and powerful. Random number generation techniques have been used to estimate the value of π repeatedly, for example. The Monte Carlo techniques section was largely taught in R - the most powerful statistical computation programming language. Coming with many built in packages for random number generating and other Monte Carlo techniques, as well as being very beginner friendly (although less so than Python) makes it an obvious and attractive choice.

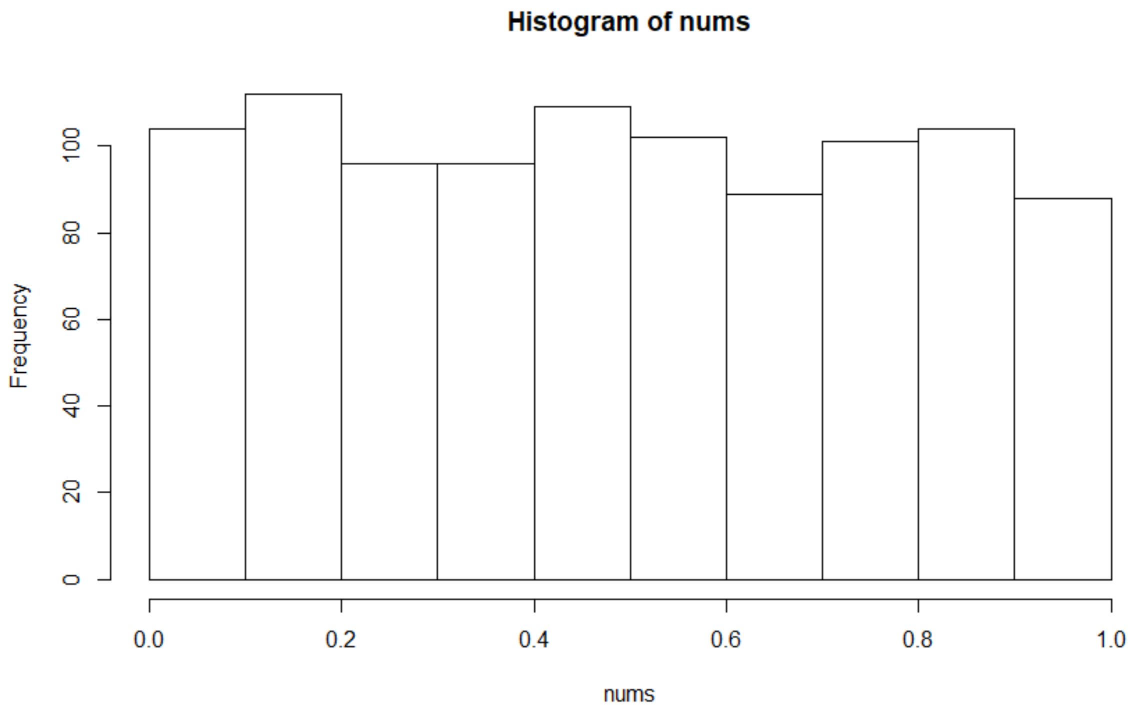


Figure 16: Histogram of uniformally distributed random numbers between 0 and 1 produced with the IBM RANDU algorithm

Before any random distribution can be made, first we must be able to produce random uniform numbers between 0 and 1. This was done with the IBM RANDU algorithm, a relatively old but easy to implement algorithm for generating random uniform numbers. The student is then taught how to put this into a histogram, which can be seen in figure 16. Note that the histogram will look slightly different each time it is run, since it is random.

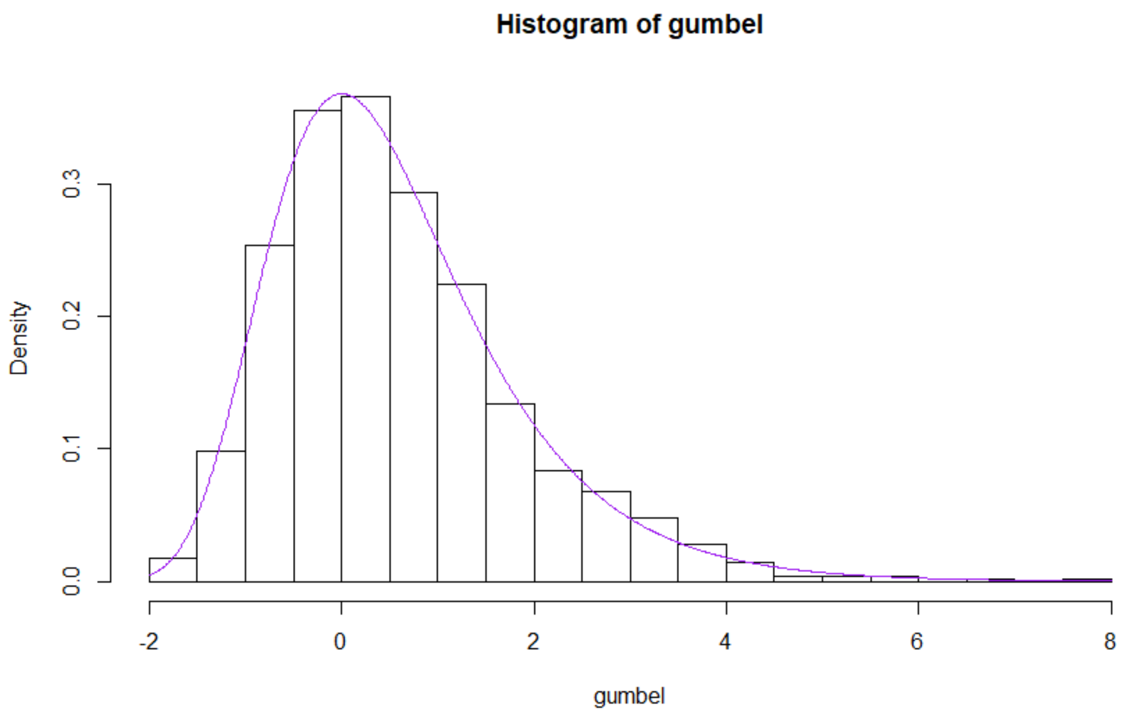


Figure 17: Histogram of Gumbel distribution with a theoretical distribution curve superimposed

The student is then walked through how to invert the CDF of the Gumbel distribution. By following this example, they could invert any CDF to perform this procedure. They are then taught how to produce a histogram of this, with a line showing the theoretical Gumbel distribution, which can be seen in figure 17. Again, this will be slightly different each time due to the random nature of the operation. By eye a very close approximation to the theoretical curve can be observed, but it would be nice to know how to. The students is taught how to use a χ^2 test to find how good a fit it is. When this algorithm was run, a value of 0.00399 was found for χ_0^2, indicating a very agreeable fit. Again, this number will vary from run to run.

3.4.2 Markov Chains

Markov Chains are a very powerful Monte Carlo technique and a very useful skill to know. We begin by using an example of how they could be used in the real world by imagining tourists moving between islands and a tourist board wanting to know where they might expect tourists to be at any given time. We then abstract this idea to a mathematical construction, and then teach the student how to produce a transition diagram using the *markovchain* and *diagram* R packages. The student is shown how to produce the transition diagram in figure 18

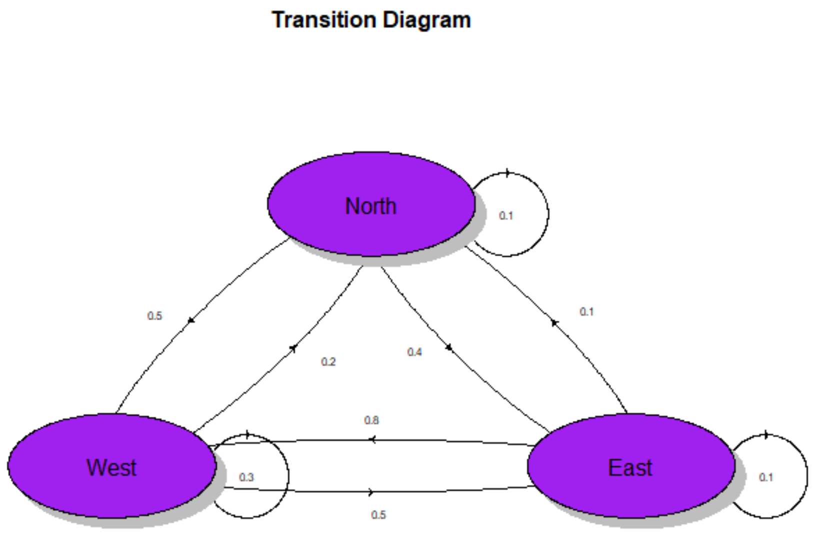


Figure 18: Abstracted transition diagram of tourists moving between islands

Finally, the student is shown how to use a "walker" to move around the islands and then record the distribution of the results. For this problem we end up with a tourist distribution that can be seen in table 1

Table 1: Distribution of tourists on islands

Island	North	West	East
Probability	0.15	0.50	0.35

3.4.3 Pythia

Pythia is a very common particle physics simulation program. While it isn't used by NOvA, GEANT and GEANIE were found to be far too complex for a beginner to particle physics simulation to use. Even so, Pythia is challenging and requires a very detailed step by step tutorial. This is especially prevalent if the student is using Microsoft Windows as an operating system, which Pythia does not officially support, but we have found a suitable workaround for this.

We begin by showing the student how to set up a Linux subsystem on Microsoft Windows. This is newer technique that Microsoft has only implemented in recent versions of Windows 10. Older operating systems will not support this method. The author notes the power of having Linux available on Windows and how many technical problems this could potentially solve. The student is then given a short tour of how the Linux subsystem works and choices of which to use. In this instance, Ubuntu is used for it is the simplest, easiest and most popular. While it theoretically does not have as much functionality as other Linux distributions, those would certainly not be needed.

The student is then shown how to run a Pythia simulation. The author was disappointed in the general lack of instruction when using Linux, not just from Pythia but all Linux tutorials. In general, it appears as though knowledge of Linux is assumed and that short abstractions would cue in the steps to be taken. This is why it was important to only use very clear, very precise step by step guides. There is a good chance the student has never been exposed to Linux before this point - with Microsoft Windows and Apple Macintosh being vastly more popular systems.

Pythia fortunately has a very large library of examples included. These are used as a springboard from which to teach the student. We begin by running one, and then change it to become a more realistic simulation. This is done by including more types of particle interactions. The student is shown how to use Pythia to make a histogram of charged events.

4 Conclusions

In conclusion, an educational tool was created aimed at final year physics students to introduce them to the main points of the work flow of modern day particle physics experiments was created. This tool took the form of a website in order to be both portable and accessible. Topics were selected on their relevance to the NOvA experiment and other similar neutrino experiments. An introduction to particle physics detector technology was given, a background of neutrino theory, an introduction to Monte Carlo methods and deep learning.

Particle physics detector technology focused on commonly used techniques and background in neutrino experiments, especially natural sources of neutrinos, accelerators, scintillators and Cherenkov radiation. These are all techniques used in neutrino experiments (scintillator experiments like NOvA and SNO+ and Cherenkov radiation experiments like Super-Kamiokande). Neutrino background included that most relevant to NOvA: mass, oscillations and early universe with the motivation of understanding the matter/anti-matter asymmetry found in the universe. Monte Carlo techniques used pseudo-random number generation and Markov chains to build up to using Pythia. While using GEANT and GEANIE would have been a far better representation of the Monte Carlo techniques used by NOvA, it was deemed far too complicated for use in an introductory project like this. Finally, a section on machine learning was developed. This primarily focused on deep learning, convolutional neutral networks and image recognition since these are techniques used in NOvA.

References

1. Fermi Research Alliance *NOvA Experiment* available from `https://novaexperiment.fnal.gov`

2. R. B. Patterson *The NOvA Experiment: Status and Outlook* Nuclear Physics B, 2013

3. S. M. Bilenky *Neutrino. History of a Unique Particle*, 2012, arXiv:1210.3065

4. Lily Asquith *My favourite particle: the neutrino* The Guardian, 2011, available from `https://www.theguardian.com/science/life-and-physics/2011/apr/02/1`

5. B. R. Martin, G. Shaw *Particle Physics* 3^{rd} *Edition* Wiley, 2008

6. G. Bellini, L. Ludhova, G. Ranucci, F. L. Villante *Neutrino Oscillations* Hindawi Advances in High Energy physics, 2014

7. A. L. Hallin *Neutrino Physics from SNO* Progress in Particle and Nuclear Physics

8. Massayuki Nakahata *Super-Kamiokande* Nuclear Physics B, 2000

9. Pablo F. de Salas, Stefano Gariazzo, Olga Mena, Christioph A. Ternes and Mariam Tortola *Neutrino Mass Ordering From Oscillations: 2018 Status and Future Prospects* Frontiers in Astronomy and Space Sciences, 2018

10. B. R. Martin *Nuclear and Particle Physics* 2nd *Edition* Wiley, 2009

11. F. Descamps *Neutrino Physics with SNO*+ Nuclear and Particle Physics Proceedings, 2015

12. Fritz Zwicky *Die Rotverschiebung Von Extragalaktischen Neblen* Helvetica Physica, 1933 (an English translation can be found by H. Andernach *The Redshift of Extragalactic Nebulae* 2017, arXiv:1711.01693)

13. S. M. Faber, J. S. Gallagher *Masses and Mass-to-Light Ratio of Galaxies* Annual Review of Astronomy and Astrophysics, 1979

14. Jaco de Swart, Gianfranco Betone, Jeroen van Dongen *How Dark Matter Came to Matter* 2017, arXiv:1703.00013v2

15. Scott Dodelson, Lawrence M. Widrow *Sterile Neutrinos as Dark Matter* Physical Review Letters, 1994

16. A. Boyarsky, M. Drewes, T. Lasserre, S. Merthens, O. Ruchayskiy *Sterile Neutrino Dark Matter* 2018, arXiv:1807.07938v2

17. Francesco Vissani *Neutrino Sources and Properties* 2014, arXiv:1412.8386v2

18. Pietro Antonioli, Richard Tresch Fienberg, Fabrice Fleurotk, Yoshiyuki Fukuda, Walter Fulgione, Alec Habig, Jaret Heise, Arthur B McDonalda, Corrinne Millsb, Toshio Nambac, Leif J Robinson, Kate Scholbergd, Michael Schwendenerk, Roger W Sinnott, Blake Staceyd, Yoichiro Suzukic, Réda Tafiroutkg, Carlo Vigorito, Brett Virene, Clarence Virtuek, and Antonino Zichichi *SNEWS: The SuperNova Early Warning System* 2004, arXiv:astro-ph/0406214v2

19. Thomas K. Gaisser, Ralph Engel and Elisa Resconi *Cosmic Rays and Particle Physics* Cambridge University Press, 2016

20. Lucio Cerrito *Radiation and Detectors* Springer, 2017

21. M. E. Convery, *Fermilab's Accelerator Complex: Current Status and Outlook* Proceedings of Science, 2016

22. H. Bethe *Zur Theorie des Durchgangs schneller Korpuskularstrahlen durch Materie* Annalen der Physik, 1930 (Originally published in German, and English translation can be found at *Theory of the Passage of Fast Corpuscular Rays Through Matter* Translated by American Meteorological Society, 1958)

23. F. Bloch *Zur Bremsung rasch bewegter Teilchen beim Durchgang durch Materie* Annalen der Physik, 1933 (This article does not appear to have ever been translated, if you have a translation please contact me at nathaniel.curnick@gmail.com)

24. D. N. Makarov, V. I. Matveev, K. A. Makarova *An Analytical Formula for the Barkas Correction to the Theory of Ion Stopping* Technical Physics Letters, 2015

25. V. I. Matveev, D. N. Makarov *Nonpertubative Shell Correction to the Bethe-Bloch Formula for the Energy Losses of Fast Charged Particles* Journal of Experimental and Theoretical Physics, 2011

26. The ALICE Collaboration *Performance of ALICE Experiment at the CERN LHC* International Journal of Modern Physics, 2014

27. Hidetake Morimoto *Approximate Formulas for Electron Penetration* Japan Science and Technology Information Aggregator, Electronic, 1961

28. H. W. Koch and J. W. Motz *Bremsstrahlung Cross-Section Formulas and Related Data* Reviews of Modern Physics, 1959

29. Charles Kittel *Introduction to Solid State Physics* 7^{th} *Edition* Wiley, 1996

30. Jacob Dunningham and Vlatko Vedral *Introductory Quantum Physics and Relativity* World Scientific, 2018

31. S. Mufson, B. Baugh, C. Bower, T. E. Coan, J. Cooper, L. Corwin, J. A. Karty, P. Mason, M. D. Messier, A. Pla-Dalmau, M. Proudfoot *Liquid Scintillator Production for the NOvA Experiment* 2015, arXiv:1504.04035v2

32. Ian Goodfellow, Yoshua Bengio and Aaron Courville *Deep Learning* MIT Press, 2016 `http://www.deeplearningbook.org`

33. Brady Neal, Sarthak Mittal, Aristide Baratin, Vinayak Tantia, Matthew Scicluna, Simon Lacoste-Julien, Ioannis Mitliagkas *A Modern Take on the Bias-Variance Tradeoff in Neural Networks* 2019, arXiv:1810.08591v4

34. Anil K. Jain *50 Years Beyond K-means* Pattern Recognition Letters, 2010

35. User: Jjitss common-swiki *Rosenblatt Perceptron* Wikimedia Commons, 2012 available from `https://commons.wikimedia.org/wiki/File:Rosenblattperceptron.png`

36. David E. Rumelhart, Geoffrey E. Hinton, Ronald J. Williams *Learning Representations by Back-Propagating Errors* Nature, 1986

37. Yann LeChun, Yoshua Bengino, Geoffrey Hinton *Deep Learning* Nature, 2015

38. NN-SVG, available from `https://alexlenail.me/NN-SVG`

39. Sandro Skansi *Introduction to Deep Learning* Springer, 2018

40. User: Aphex34, *Max Pooling* Wikimedia Commons, 2015 available from `https://commons.wikimedia.org/wiki/File:Max_pooling.png`

41. CERN *2020 Annual Contributions to CERN budget* available from `https://fap-dep.web.cern.ch/rpc/2020-annual-contributions-cern-budget`

42. A. Aurisano, A. Radovic, D. Rocco, A. Himmel, M. D. Messier, E. Niner, G. Pawloski, F. Psihas, A. Sousa, P. Vahle *A Convolutional Neural Network Event Classifier* 2016, arXiv:1604.01444v3

43. Roger Eckhardt *Stan Ulam, John Von Neumann and the Monte Carlo Method* Los Alamos Science, 1987

44. Nicholas Metropolis, Arianna W. Rosenbluthh, Marshall N. Rosenbluth, Augusta H. Teller, Edward Teller *Equation of State Calculations by Fast Computing Machines* The Journal of Chemical Physics, 1953

45. M. Donald MacLaren, George Marsaglia *Uniform Random Number Generators* Journal of the Association for Computing Machinery, 1965

46. D. M. Fellows *Comments on "A General Fortran Emulator for IBM 360/370 Random Number Generator 'RANDU'"* INFOR Journal: Information System and Operations Research, 1976

47. Luc Devroy *Non-Uniform Random Variate Generation* Springer-Verlag, 1986 (The book is long since out of print but is available from `http://luc.devroye.org/rnbookindex.html`)

48. J. B. Norris *Markov Chains* Cambridge University Press, 1997

49. Torbjörn Sjöstrand, Stefan Ask, Jesper R. Christiansen, Richard Corke, Nishita Desai, Philip Ilten, Stephen Mrenna, Stefan Prestel, Christine O. Rasmussen, Peter Z. Skands *An Introduction to Pythia 8.2* 2014, arXiv:1410.3012

50. S. Agostinelli, J. Allison, K. Amako, J. Apostolakis, H. Araujo, P. Arce, M. Asai, D. Axen, S. Banerjee, G. Barrand, F. Behner, L. Bellagamba, J. Boudreau, L. Broglia, A. Brunengo, H. Burkhardt, S. Chauvie, J. Chuma, R. Chytracek, G. Cooperman, G. Cosmo, P. Degtyarenko, A. Dell'Acqua, G. Depaola, D. Dietrich, R. Enami, A. Feliciello, C. Ferguson, H. Fesefeldt, G. Folger, F. Foppiano, A. Forti, S. Garelli, S. Giani, R. Giannitrapani, D. Gibin, J.J. [Gómez Cadenas], I. González, G. [Gracia Abril], G. Greeniaus, W. Greiner, V. Grichine, A. Grossheim, S. Guatelli, P. Gumplinger, R. Hamatsu, K. Hashimoto, H. Hasui, A. Heikkinen, A. Howard, V. Ivanchenko, A. Johnson, F.W. Jones, J. Kallenbach, N. Kanaya, M. Kawabata, Y. Kawabata, M. Kawaguti, S. Kelner, P. Kent, A. Kimura, T. Kodama, R. Kokoulin, M. Kossov, H. Kurashige, E. Lamanna, T. Lampén, V. Lara, V. Lefebure, F. Lei, M. Liendl, W. Lockman, F. Longo, S. Magni, M. Maire, E. Medernach, K. Minamimoto, P. [Mora de Freitas], Y. Morita, K. Murakami, M. Nagamatu, R. Nartallo, P. Nieminen, T. Nishimura, K. Ohtsubo, M. Okamura, S. O'Neale, Y. Oohata, K. Paech, J. Perl, A. Pfeiffer, M.G. Pia, F. Ranjard, A. Rybin, S. Sadilov, E. [Di Salvo], G. Santin, T. Sasaki, N. Savvas, Y. Sawada, S. Scherer, S. Sei, V. Sirotenko, D. Smith, N. Starkov, H. Stoecker, J. Sulkimo, M. Takahata, S. Tanaka, E. Tcherniaev, E. [Safai Tehrani], M. Tropeano, P. Truscott, H. Uno, L. Urban, P. Urban, M. Verderi, A. Walkden, W. Wander, H. Weber, J.P. Wellisch, T. Wenaus, D.C. Williams, D. Wright, T. Yamada, H. Yoshida, D. Zschiesche *GEANT4 - A Simulation Toolkit* Nuclear Instruments and Methods in Physics Research Section A, 2003

51. C. Andreopoulos, A. Bell, D. Bhattacharya, F. Cavanna, J. Dobson, S. Dytman, H. Gallagher, P. Guzowski, R. Hatcher, P. Kehayias, A. Meregaglia, D. Naples, G. Pearce, A. Rubbia, M. Whalley, T. Yang *The GENIE Neutrino Monte Carlo Generator* Nuclear Instruments and Methods in Physics Research Section A, 2010

52. Glen Cowan *Statistical Data Analysis* Oxford University Press, 2002