COSMIC QUEST

Mentors:

Tanish Singla[1], Amar Sathwik[2] and Ayush Baghel[3]

[1] *tanishs22@iitk.ac.in,* [2] *thogiti22@iitk.ac.in,* [3] *ayushba22@iitk.ac.in*

Mentees:

Aditya , Aman , Harshit , Mayank , Mrinal , Pearl , Pragya , Palak , Parv , Rishabh , Sarvesh , Sowbarnika

Acknowledgments:

We are deeply grateful to our mentors for their guidance and support throughout the project. We would like to express our appreciation for the valuable learning experience they provided. We also extend our heartfelt thanks to the Astronomy Club, IITK, and the Science and Technology Council, IITK, for granting us the opportunity to work on this project.

Objective:

Combining cosmological studies and powerful tools of machine learning to develop a neural network model capable of classifying and detecting strong gravitational lenses based only upon images taken by space telescopes.

Contents

END OF PAGE

1. INTRODUCTION

Gravitational lensing, predicted by Einstein's general theory of relativity, is a very powerful tool for probing the distribution of dark matter, understanding the large-scale structure of the universe, measuring cosmological parameters (Hubble constant and density of universe) and understanding the formation and evolution of galaxies in the early universe by magnifying them.
The traditional methods of detection of gravitational lenses are tedious and time-consuming, It has become imperative to find better methods using machine learning algorithms. This project introduces the various datasets and the different computational methods and algorithms employed by astrophysicists to classify probable gravitationally lensed candidates. It also inspires to find new ways to classify such objects and to solve the newer challenges arising in the astrophysical community.

2. SUMMARY

The traditional methods for detecting gravitational lensing often rely on manual inspection that require significant resources, expertise and much time due to the extensive number of possible candidates in the form of galaxies mostly (DR10 database alone which we will be working on has around 1 million galaxy candidates). However, with the onset of machine and deep learning, particularly Neural Networks (NNs), has opened new ideas for automating and enhancing the detection process of such objects. NNs have presented us with remarkable success in various image recognition tasks, making them well-suited for analyzing astronomical data where gravitational lensing signatures are present.

In this project, we will develop and implement a deep learning framework model to identify and analyze the probable gravitational lensing candidates from the newly released dataset DR10 (Data Release 10) from the DESI (Dark Energy Spectroscopic Instrument) Legacy Survey. By using the complex model, we seek to automate and optimise the detection process, to improve the accuracy, and reduce the time required to analyze these vast datasets.

A brief overview of this project can be given as:

(i) **Data Collection and Preprocessing:** Gathering a comprehensive dataset of astronomical images from the DR10 of DESI (Dark Energy Spectroscopic Instrument) Legacy Survey, KiDS (Kilo-Degree Survey) by ESO (European Space Organization), SDSS (Sloan Digital Sky Survey) including both lensed and non-lensed samples, and applying necessary pre-processing techniques to prepare the data for training and evaluation.

(ii) **NN Architecture Design:** Designing an efficient NN architecture designed for the detection of gravitational lensing, incorporating layers and techniques that enhance feature extraction and classification performance.

(iii) **Model Training and Evaluation:** Training the NN model on the prepared dataset, tuning the parameters to optimize performance, and evaluating the model's accuracy and robustness using appropriate metrics.

Through these steps, this project aims to contribute to the field of computational astrophysics by providing a reliable and efficient tool for detecting gravitational lensing events. This will not only enhance our understanding of the universe in general but also pave the way for further future research and discoveries in cosmology and gravitational studies.

Here in this documentation we will summarise the work put into the project which will be divided into 6 parts starting with basic concepts of cosmology like CMB (Cosmic Microwave Background), Dark Matter and Energy etc, introducing gravitational lensing and its types, going through the detection methods and neural network architecture employed, presenting the week wise proceedings of the project and ending with a conclusion. This document has been penned in a way that a beginner can implement the project on his own with the help of this documentation.

3. COSMOLOGY

Cosmology is the study of the origin, development, structure, history, and future of the entire universe. These are the things that we discussed about cosmology during the course of our project:

3.1. Astronomy and Cosmology:

No branch of science can have a wider expanse than cosmology for it is the study of the universe and by definition, the universe contains everything. Astronomy began with the study of the properties of various planets and stars and now includes the limits of the Milky Way Galaxy.

Cosmology is concerned mainly with the objects beyond the Galaxy, study of a large-scale model of our universe covering billions of light years and the overall physical and dynamic behaviour of all the galaxies within that expanse. Although our techniques of observing the universe are not even close to perfect yet, we are at a point where we can make some sense out of what we see and observe in our universe.

3.2. Galaxy Types:

Scientists have categorized galaxies based on their shapes and physical features. Some of them listed below:

3.2.1. Spiral Galaxy:

These galaxies resemble giant rotating pinwheels with a pancake-like disk of stars and a central bulge or tight concentration of stars. Based on this central bulge or nucleus, spiral galaxies are classified into Sa, Sb, Sc, and so on.

Fig. 1: Spiral galaxy

Our galaxy and M31 are of type Sb. Some spirals have bars in the central region. These are called barred spirals and are categorized as SBa, SBb, SBc, and so on. Spiral galaxies are surrounded by halos, mixtures of old stars, star clusters, and dark matter. The youngest stars form in gas-rich arms, while older stars can be found throughout the disk and within the bulge and halo.

3.2.2. Elliptical Galaxy:

Elliptical galaxies have shapes that range from completely round to oval. They are less common than spiral galaxies. Unlike spirals, elliptical galaxies usually contain little gas and dust and show very little organization or structure. The stars orbit around the core in random directions and are generally older than those in spiral galaxies since little of the gas needed to form new stars remains. Scientists think elliptical galaxies originate from collisions and mergers with spirals.

3.2.3. Lenticular Galaxy:

Lenticular galaxies or S0 are a kind of cross between spirals and ellipticals. They have the central bulge and disk common to spiral galaxies but no arms. But like ellipticals, lenticular galaxies have older stellar populations and little ongoing star formation.

Scientists have a few theories about how lenticular galaxies evolved. One idea suggests these galaxies are older spirals whose arms have faded. Another proposes that lenticulars formed from mergers of spiral galaxies.

Fig. 2: The Spindle Galaxy (NGC 5866), a lenticular galaxy in the constellation Draco.

3.2.4. Seyfert Galaxy:

Seyfert galaxies, first identified in 1943 by American astronomer Carl Seyfert, are the most common active galaxies and also exhibit the lowest energies. All Seyferts look like normal galaxies in visible light, but they emit considerable infrared radiation. When observed in the infrared, some reveal bright emission from the donut-shaped torus. Some also emit X-rays. Seyfert galaxies tend to have lower radio luminosities, although some produce radio jets. Scientists divide

Seyferts into two classes. Type I Seyfert galaxies display unusual features in their visible light that imply rapid motion near the accretion disk. Type II Seyferts show features that imply much slower motion. However, this distinction may result from different viewing angles into the centers of these galaxies.

Fig. 3: The Circinus Galaxy, a Type II Seyfert galaxy

3.3. Quasars:

Quasi-stellar radio source or quasars are rare and extremely luminous active galactic nuclei (AGN) that's powered by a supermassive black hole. They are among the most luminous objects in the known universe and can emit thousands of times more light than the Milky Way and are distinguished from other AGNs by their enormous luminosity and distance from Earth. They were first discovered in 1963 as a result of the optical identification programme. When the Moon happens to cross the line of sight of a source, the source is said to be occulted. The radio position of the quasar 3C273 was accurately determined by lunar occultation. There was a noticeable drop in the intensity of a radio source when it is blocked by the Moon and a rise when the Moon has moved out of its way gives an accurate indication of when occultation takes place and hence where it is located in the sky.

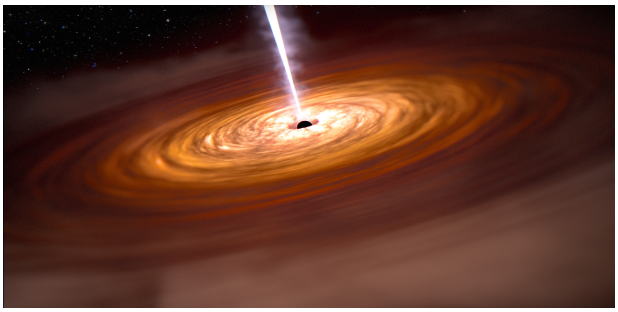


Fig. 4: Quasars are the blazing centers of active galaxies and are powered by a supermassive black hole feeding on huge quantities of gas.

The optical identification of this object and of another radio source, 3C48, gave us the assumption that these were radio stars in the galaxy. But when their spectra were carefully observed, it became clear that their wavelengths were quite redshifted.**If the wavelength of an emission line in the laboratory is $\lambda 1$ and if the observed wavelength is $\lambda 2$, then the line is said to be redshifted by a fraction z given by:**

$$z = \frac{\lambda 2 - \lambda 1}{\lambda 1}$$

It is usual to call z the redshifted of the object. For 3C273, z=0.158 and for 3C48, z=0.367. The word redshift (z) is used to define a shift to the red end of the visual spectrum.

These values of z for stars in the galaxy were pretty high. The possible reasons for this were that they were either high-velocity stars ejected from the galaxy or that these redshifts arose form the expansion of the universe. If the second reasoning was correct, this implies that quasars are very distant objects, and since from such large distances, they are bright enough to be mistaken for stars, they must be very powerful. Many quasars show rapid variation in their light and radio output. This fact places a limit on the physical size of a quasar since if an object shows variability on a characteristic time scale T, its size cannot be greater than cT where c is the speed of light due to constraints from special relativity(nothing can travel faster than light in the universe). By now more than 5000 quasars are known yet only a few percent of the total quasar population emits radio waves.

3.4. Expansion of Universe:

The expansion of the universe is the increase in distance between gravitationally unbound parts of the observable universe with time. It is an intrinsic expansion, so it does not mean that the universe expands "into" anything or that space exists "outside" it. To any observer in the universe, it appears that all but the nearest galaxies recede at speeds that are proportional to their distance from the observer, on average. While objects cannot move faster than light, this limitation applies only with respect to local reference frames and does not limit the recession rates of cosmologically distant objects.The universe at the largest scales is observed to be homogeneous (the same everywhere) and isotropic (the same in all directions), consistent with the cosmological principle. These constraints demand that any expansion of the universe accord with Hubble's law, in which objects recede from each observer with velocities proportional to their positions with respect to that observer. That is, recession velocities

The velocity $\vec{v}$ scales with (observer-centered) positions $\vec{x}$ according to

$$\vec{v} = H_0 \vec{x},$$

where the Hubble rate H_0 quantifies the rate of expansion. The Hubble rate H_0 is a function of cosmic time.

The expansion of the universe can be understood as a consequence of an initial impulse (possibly due to inflation), which sent the contents of the universe flying apart. The mutual gravitational attraction of the matter and radiation within the universe gradually slows this expansion over time, but expansion nevertheless continues due to momentum left over from the initial impulse. Also, certain exotic relativistic fluids, such as dark energy and inflation, exert gravitational repulsion in the cosmological context, which accelerates the expansion of the universe. A cosmological constant also has this effect. Mathematically,

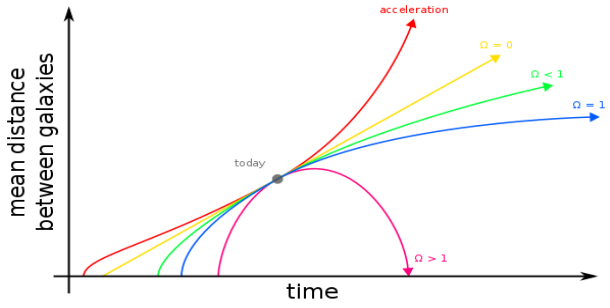


Fig. 5: The expansion history depends on the density of the universe. Ω on this graph corresponds to the ratio of the matter density to the critical density, for a matter-dominated universe. The "acceleration" curve shows the trajectory of the scale factor for a universe with dark energy.

the expansion of the universe is quantified by the scale factor, a, which is proportional to the average separation between objects, such as galaxies. The scale factor is a function of time and is conventionally set to be $a = 1$ at the present time. Because the universe is expanding, a is smaller in the past and larger in the future. Extrapolating back in time with certain cosmological models will yield a moment when the scale factor was zero; our current understanding of cosmology sets this time at 13.787 ± 0.020 billion years ago. If the universe continues to expand forever, the scale factor will approach infinity in the future. It is also possible in principle for the universe to stop expanding and begin to contract, which corresponds to the scale factor decreasing in time.

The scale factor a is a parameter of the **Friedmann–Lemaître–Robertson–Walker** metric, and its time evolution is governed by the Friedmann equations. The second Friedmann equation,

$$\frac{\ddot{a}}{a} = -\frac{4\pi G}{3}\left(\rho + \frac{3p}{c^2}\right) + \frac{\Lambda c^2}{3},$$

shows how the contents of the universe influence its expansion rate. Here, G is the gravitational constant, ρ is the energy density within the universe, p is the pressure, c is the speed of light, and Λ is the cosmological constant. A positive energy density leads to deceleration of the expansion, $\ddot{a} < 0$, and a positive pressure further decelerates expansion. On the other hand, sufficiently negative pressure with $p < -\rho c^2/3$ leads to accelerated expansion, and the cosmological constant also accelerates expansion. Nonrelativistic matter is essentially pressureless, with $|p| \ll \rho c^2$, while a gas of ultrarelativistic particles (such as a photon gas) has positive pressure $p = \rho c^2/3$. Negative-pressure fluids, like dark energy, are not experimentally confirmed, but the existence of dark energy is inferred from astronomical observations.

3.5. Redshift:

Astronomers can learn about the motion of cosmic objects by looking at the way their color changes over time or how it differs from what we expected to see. For example, if an object is redder than we expected we can conclude that it is moving away from us, and if it is bluer we can tell that it is moving towards us. But how do we know this? Hubble established that almost all galaxies are moving away from us, and each other. Hubble also observed that the further away the object, the greater the amount of redshift. This

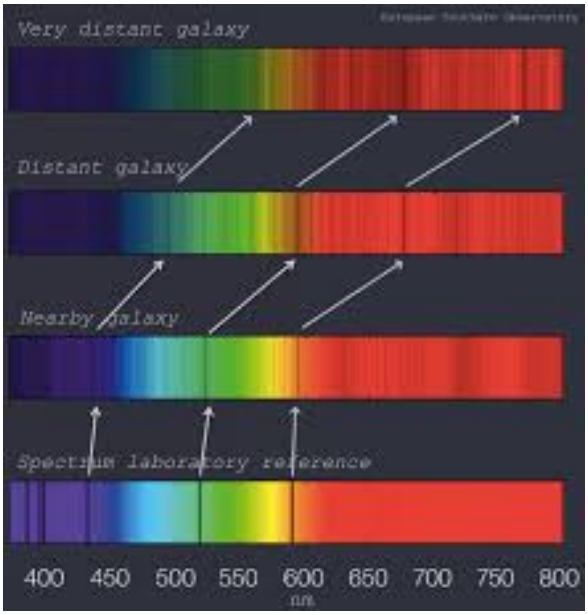


Fig. 6: Astronomers can look at the spectra created by different elements and compare these with the spectra of stars. If the absorption or emission lines they see in the star's spectra are shifted, they know the object is moving either towards us or away from us.

is expressed in Hubble's law: **The velocity with which galaxies recede from us (recessional velocity) is proportional to their distance from us**. Scientists now believe that redshift observed for

very distant objects is only partly due to the speed of recession of the object from the Earth. Most of it is caused by expansion of the Universe and consequent 'stretching' of space.

Redshift is an example of the Doppler Effect. As an object moves away from us, the sound or light waves emitted by the object are stretched out, which makes them have a lower pitch and moves them towards the red end of the electromagnetic spectrum, where light has a longer wavelength. As space itself is stretched, light travelling across it is also stretched, causing wavelengths to increase and the light to be redshifted. Astronomers use the observed shift in wavelength of light from distant celestial objects to calculate how far away objects are. The further away an object is, the longer it has taken for light to reach us, and the more that light has been stretched out due to expansion of the Universe.

3.6. Hubble constant and its evaluation methods:

The Hubble constant, H_0 , is a measure of the rate at which the universe is expanding. It is defined as the proportionality constant between the recessional velocity of galaxies and their distance from us, expressed by Hubble's law:

v= H_0 d

where:

v is the recessional velocity of a galaxy,

d is the distance to the galaxy,

H_0 is the Hubble constant.

The value of the Hubble constant is a critical parameter in cosmology as it influences estimates of the age and size of the universe. Various methods are used to evaluate H_0, each with its own strengths and potential sources of error. Here are the primary methods:

1.Cepheid Variable Stars: These stars vary predictably in luminosity, allowing astronomers to calculate distances by comparing their absolute brightness with their apparent brightness, aided by measuring the host galaxy's redshift.

2.Type Ia Supernovae: Known for their consistent peak luminosity, these supernovae serve as reliable "standard candles" for distance measurements. Observing their apparent brightness and the redshift of their host galaxy enables calculation of distances and thus H_0

. 3.Tully-Fisher Relation: This relation links a spiral galaxy's luminosity to its rotational velocity. By measuring the velocity and comparing intrinsic luminosity with observed brightness, distances can be derived alongside redshift data to estimate H_0.

4.Surface Brightness Fluctuations (SBF): Analyzing brightness fluctuations in elliptical galaxies due to stellar distribution provides distance estimates. When combined with redshift data, this method contributes to calculating H_0.

5.Gravitational Lensing Time Delays: In systems where distant quasars are gravitationally lensed by foreground galaxies, time delays between brightness variations in different images help estimate distances. This data, combined with redshift measurements, aids in determining H_0

6.Cosmic Microwave Background (CMB): Detailed measurements of the CMB's early universe snapshot allow inference of cosmological parameters, including H_0 . Experiments like Planck provide precise values, integrated with standard cosmological models.

7.Baryon Acoustic Oscillations (BAO): Regular density fluctuations in visible baryonic matter reveal insights into the universe's large-scale structure. Measuring these oscillations in galaxy distributions, alongside redshift data, offers another approach to estimate H_0.

Current Discrepancies: Measurements of H_0 from different methods show a notable discrepancy. Local distance ladder methods (like Cepheids and Type Ia supernovae) suggest higher values (∼73-74 km/s/Mpc), while CMB and early universe observations (e.g., Planck) indicate lower values (∼67-68 km/s/Mpc). Resolving this tension is a major challenge in cosmology, potentially pointing to new physics or systematic measurement errors.

3.7. Theories Of Origin Of The Universe:

2.7.1 Steady State Theory:

The steady state theory was developed by Bondi, Thomas, Gold German and Fred hoyle. According to this theory, the number of galaxies in the observable universe is constant and new galaxies are being created continuously out of the empty space, which fill up the gaps caused by those galaxies which have crossed the boundary of the observable universe. So by this theory it can be concluded that the overall size of mass of the observable universe is constant therefore the steady state of the universe is not disturbed at all. Therefore this theory is named as steady state theory.

2.7.2 Pulsating Theory:

Pulsating theory gives another explanation for the evolution of the universe. This theory states that the universe is pulsating, which means that our universe is supposed to be expanding and contracting alternatively. As now we know that our universe is expanding at present. This theory also accepts this fact that at the present time our universe is expanding but it will not keep on expanding all the time. According to this theory, at certain time it is possible that this expansion of the universe will be stopped and then it will start contracting due to the gravitational pull. With the time this contraction will come to a point where the explosion of the universe again take place and again the expansion of the universe will start. So

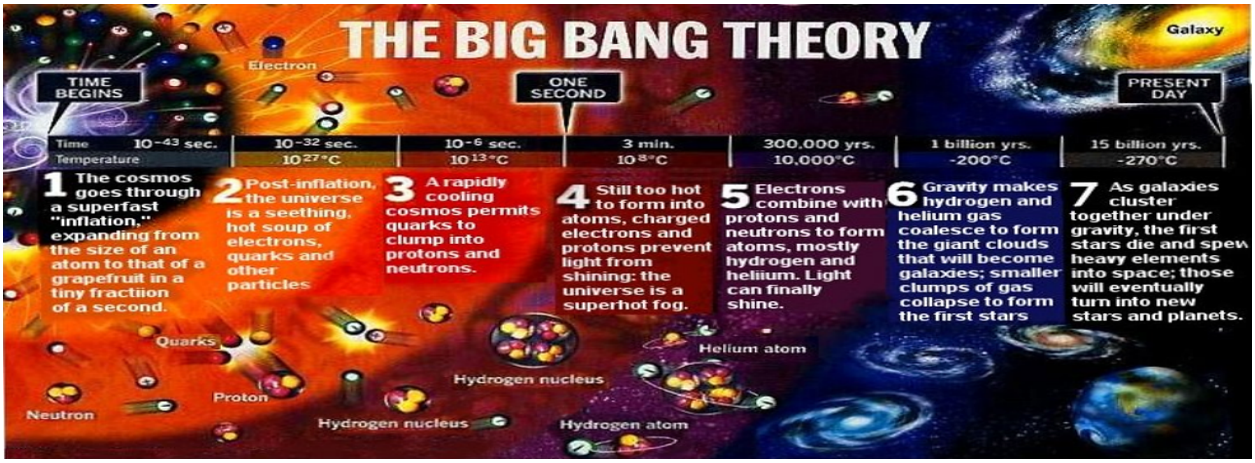


Fig. 7: Timeline of the Universe

in this way this alternate expansion and contraction will be continued in the pulsating universe.

2.7.3 Big Bang Theory:

The big bang theory was proposed by Le Maitre and Gammow. This theory says that the whole matter of the universe was once concentrated in an extremely dense and hot fireball at the beginning of the universe. And then a very big explosion occurred nearly 20 billion years ago. This big explosion was named "the Big Bang". This big explosion caused the matter to break into pieces. The highly concentrated matter was thrown out with high speed in all the directions that later formed stars and galaxies. These stars and Galaxies are still moving away from each other.

3.8. The Early Universe:

2.8.1 Singularity:

Without a perfect cosmological principle, scientists can't explain the universe's past. They use general relativity to trace its expansion backwards. This leads to infinite density and heat at a specific point in time. Some Big Bang models call this the initial singularity. People often label this event the Big Bang. It marks when our universe started following known physics laws. These include general relativity and the Standard Model of particle physics. The early universe was incredibly dense - denser than what forms black holes. Yet, it didn't collapse back into a singularity. It also didn't form many black holes. This suggests matter spread very at first, with tiny differences in density.

2.8.2 Inflation and Baryogenesis:

The Big Bang's earliest stages are uncertain, but most theories suggest the universe began as a superhot soup that expanded and cooled at breakneck speed. The Planck epoch, which occurred for 10^{-43} seconds, saw the merging of four basic forces: electromagnetism, strong nuclear, weak nuclear, and gravity. The universe reached a temperature of 1032 degrees Celsius and spanned approximately 10^{-35} meters. The grand unification epoch followed, where gravity split from other forces and cosmic inflation began. This inflation froze quantum fluctuations, causing Heisenberg's uncertainty principle. The electroweak epoch began around 10^{-36} seconds, and the strong nuclear force split off, while the electromagnetic and weak nuclear forces remained together. The universe reheated and reached the required temperatures for the production of quark-gluon plasma and elementary particles. At one point, the universe underwent a new interaction called baryogenesis, leading to a small excess of quarks and leptons compared to antiquarks and antileptons, resulting in more matter than antimatter.

2.8.3 Cooling:

The universe's density and temperature decreased, causing a decrease in particle energy. Symmetry-breaking phase transitions separated the electromagnetic force and weak nuclear force at 10^{-12} seconds. After 10^{-11} seconds, particle energies dropped to values achievable in particle accelerators. Quarks and gluons combined to form baryons, protons, and neutrons. A mass annihilation followed, leaving only one in 108 original matter particles and none of their antiparticles. The energy density of the universe was dominated by photons, with neutrinos contributing a minor contribution. At a billion kelvin temperature and density, neutrons combined with protons to form deuterium and helium nuclei, known as Big Bang nucleosynthesis (BBN). As the universe cooled, the rest energy density of matter gravitatedally dominated photon radiation. The recombination epoch began after 379,000 years, when electrons and nuclei

combined into atoms, emitting radiation known as the cosmic microwave background.

2.8.4 Structure Formation:

After the recombination epoch, the slightly denser regions of the uniformly distributed matter gravitationally attracted nearby matter and thus grew even denser, forming gas clouds, stars, galaxies, and the other astronomical structures observable today. The details of this process depend on the amount and type of matter in the universe. The four possible types of matter are known as cold dark matter (CDM), warm dark matter, hot dark matter, and baryonic matter.

2.8.5 Cosmic Acceleration:

Independent lines of evidence from Type Ia supernovae and the CMB imply that the universe today is dominated by a mysterious form of energy known as dark energy, which appears to homogeneously permeate all of space. Observations suggest that 73 % of the total energy density of the present day universe is in this form. When the universe was very young it was likely infused with dark energy, but with everything closer together, gravity predominated, braking the expansion. Eventually, after billions of years of expansion, the declining density of matter relative to the density of dark energy allowed the expansion of the universe to begin to accelerate. Dark energy in its simplest formulation is modeled by a cosmological constant term in Einstein field equations of general relativity, but its composition and mechanism are unknown. More generally, the details of its equation of state and relationship with the Standard Model of particle physics continue to be investigated both through observation and theory.

3.9. Quantum fluctuations and formation of large scale structures:

Quantum fluctuations are temporary changes in the amount of energy in a point in space, arising from Heisenberg's uncertainty principle. During the inflationary epoch of the early universe, these fluctuations were stretched to macroscopic scales, providing the seeds for the formation of large-scale structures like galaxies and galaxy clusters.

2.4.1 Historical Background:

- **Inflation Theory:** Proposed by Alan Guth in 1981, inflation is a period of rapid exponential expansion in the early universe. This theory helps solve several problems in cosmology, such as the horizon and flatness problems.
- **Seeding of Structures:** Quantum fluctuations during inflation were stretched to cosmic scales, becoming the initial density perturbations that later grew into large-scale structures under gravitational instability.

2.4.2 Key Concepts:

- **Density Perturbations:** Small variations in the density of matter in the early universe, quantified by the density contrast

$$\delta(\vec{x}, t) = \frac{\rho(\vec{x}, t) - \bar{\rho}(t)}{\bar{\rho}(t)},$$

  where $\rho(\vec{x}, t)$ is the local density and $\bar{\rho}(t)$ is the average density.
- **Power Spectrum:** The statistical distribution of the density perturbations is often described by the power spectrum $P(k)$, where k is the wave number. The power spectrum is related to the two-point correlation function of the density field.

2.4.3 Mathematical Framework:

- **Friedmann Equations:** These govern the expansion of the universe. The first Friedmann equation is given by

$$\left(\frac{\dot{a}}{a}\right)^2 = \frac{8\pi G}{3}\rho - \frac{k}{a^2} + \frac{\Lambda}{3},$$

  where $a(t)$ is the scale factor, G is the gravitational constant, ρ is the energy density, k is the curvature parameter, and Λ is the cosmological constant.
- **Growth of Perturbations:** The linear growth of matter perturbations in a matter-dominated universe is given by the equation

$$\ddot{\delta} + 2H\dot{\delta} - 4\pi G\rho\delta = 0,$$

  where δ is the density contrast, $H = \frac{\dot{a}}{a}$ is the Hubble parameter, and ρ is the background density.

2.4.4 Observational Evidence:

- **Cosmic Microwave Background (CMB):** Measurements of the CMB, such as those by the Planck satellite, show small temperature anisotropies that correspond to density fluctuations in the early universe. These fluctuations match the predictions of inflationary theory.
- **Large Scale Structure Surveys:** Surveys like the Sloan Digital Sky Survey (SDSS) map the distribution of galaxies, providing a three-dimensional picture of large-scale structures and supporting the growth of perturbations from initial quantum fluctuations.

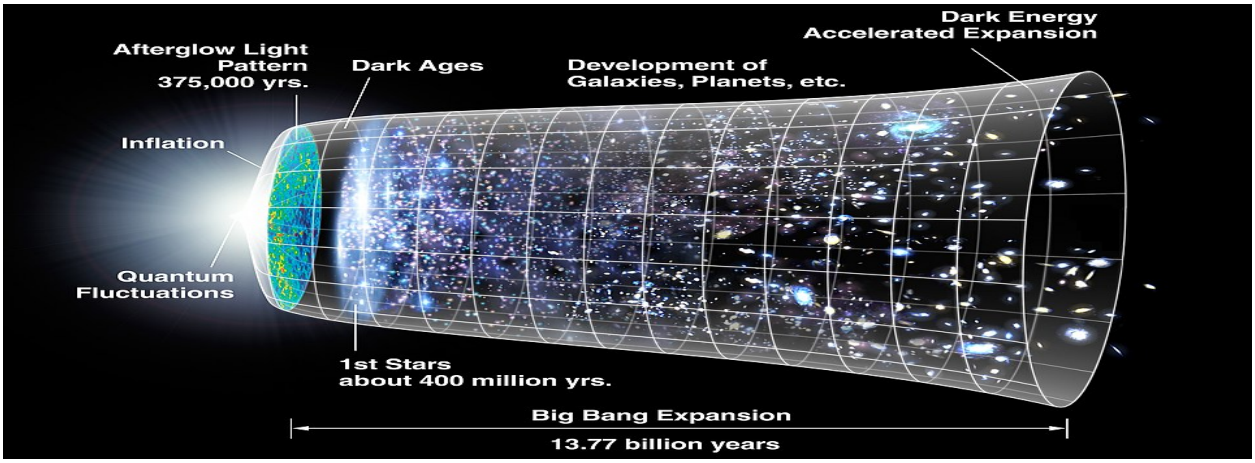


Fig. 8: Quantum fluctuation and formation of large scale structures

2.4.5 Current Understanding:

- **ΛCDM Model:** The standard cosmological model, which includes cold dark matter (CDM) and a cosmological constant (Λ), explains the formation and growth of large-scale structures from initial quantum fluctuations.

- **N-body Simulations:** These computational simulations model the evolution of dark matter and baryonic matter under gravity, starting from initial conditions provided by the power spectrum of primordial fluctuations.

3.10. Cosmic Microwave Background:

The Cosmic Microwave Background (CMB) is the cooled remnant of the first light that could ever travel freely through space. This "fossil" radiation, the most distant that any telescope can see, was released soon after the "Big Bang". Scientists think of it as an echo or "shock wave" of the big bang. In time this primordial light cooled and weakened considerably; today we detect it in the microwave domain. The CMB radiation was discovered by accident in 1965. Penzias and Wilson, two radio astronomers in the United States, registered a signal in their radio telescope that could not be assigned to any precise source in the sky. Apparently it came from everywhere with equal intensity, day or night, summer or winter. They concluded that the signal must have come from outside our Galaxy. It came almost from the beginning of the Universe.

Scientists considered their discovery to be solid proof of the "big bang" theory. This theory predicted that the "shock wave" of this primeval explosion would still be detectable as a subtle "wallpaper" coming from anywhere beyond all the galaxies, quasars, and galaxy clusters. Today, the Big Bang model is still the only model capable of convincingly explaining the existence of the CMB. According to this model, the universe began with a very dense and hot phase that expanded and cooled; for several hundred thousand years the temperature was so high that neutral atoms could not form. Matter consisted mainly of neutrons and charged particles (protons and electrons). Electrons interacted closely with light particles and therefore light and matter were tightly coupled at that time (meaning light could not travel long distances in a straight line). Therefore, light could not propagate and the universe was opaque.

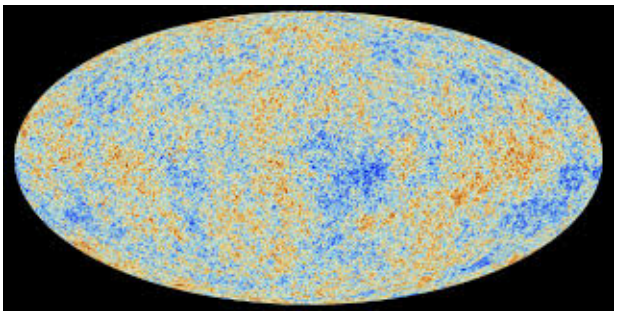


Fig. 9: Heat map of temperature fluctuations in the cosmic microwave background

It took the universe about 300,000 years to cool to a temperature where atoms can form (about 3000°C). Matter then became neutral and allowed light to travel freely: the Universe became transparent. The remnant of this "first light" is the CMB. Since this radiation was released, the universe has expanded and gotten cooler and cooler at the same time. The cosmic background was affected by the same process: it expanded and cooled. The space "stretched" and with it all the lengths. After all, light is a wave, just like waves on the sea, and when you stretch a wave, its characteristic length scale (and also its 'frequency') changes. Today, we can detect the CMB at microwave frequencies or length scales that are much longer than, for example, the length scales that our

eyes can see. Because of this, human eyes cannot see microwaves from the CMB (or X-rays or infrared). However, with the use of specially designed detectors such as those Planck is to carry, we can.

The CMB is the most distant and oldest light that a telescope can detect. It is not possible to see beyond the time of its release, because then the universe was completely "opaque". The CMB brings astronomers as close as possible to the Big Bang and is currently one of the most promising ways to understand the birth and evolution of the universe in which we live.

3.11. Dark Matter and Dark Energy:

In astronomy, **dark matter** is a hypothetical form of matter that appears not to interact with light or the electromagnetic field. This means it does not absorb, reflect or emit light, making it extremely hard to spot. In fact, researchers have been able to infer the existence of dark matter only from the gravitational effect it seems to have on visible matter. Dark matter seems to outweigh visible matter roughly six to one, making up about 27 percent of the universe.The matter we know and that makes up all stars and galaxies only accounts for 5 percent of the content of the universe. Researchers speculates that it could contain "supersymmetric particles" – hypothesized particles that are partners to those already known in the Standard Model. Experiments at the Large Hadron Collider (LHC) may provide more direct clues about dark matter.

Dark energy makes up approximately 68 percent of the universe and appears to be associated with the vacuum in space. It is distributed evenly throughout the universe, not only in space but also in time – in other words, its effect is not diluted as the universe expands. The even distribution means that dark energy does not have any local gravitational effects, but rather a global effect on the universe as a whole. This leads to a repulsive force, which tends to accelerate the expansion of the universe. The rate of expansion and its acceleration can be measured by observations based on the Hubble law. These measurements, together with other scientific data, have confirmed the existence of dark energy and provide an estimate of just how much of this mysterious substance exists.

4. GRAVITATIONAL LENSING

Detecting dark matter and probing the contents of the universe is an age-old concern of cosmology. But in recent years, this previously baffling problem has found a solution. The missing piece to this puzzle seems to be gravitational lensing. The various types of lensing aid to fathom different aspects of cosmology which were earlier inexplicable. Therefore , it is necessary to understand gravitational lensing scrupulously.

The following section explains this crucial phenomena along with its types.

Gravitational lensing is a phenomenon that occurs when a massive object, such as a galaxy or a cluster of galaxies, lies between a distant light source (like a star or another galaxy) and an observer on Earth. The massive celestial bodies act like a lens, bending and magnifying the light from the distant source. Such massive bodies are called gravitational lenses.

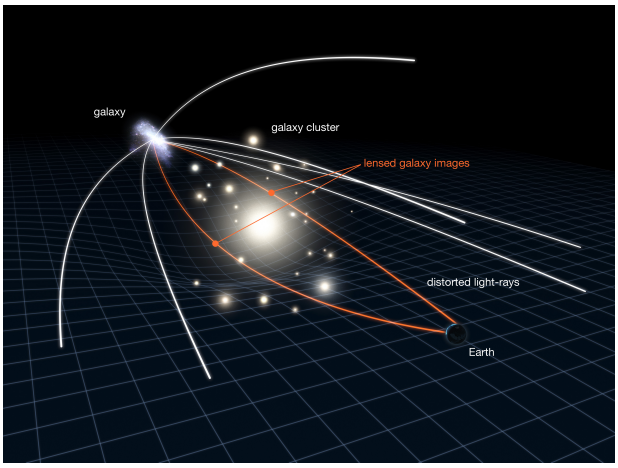


Fig. 10: This is an illustration of gravitational lensing.

According to Einstein's general theory of relativity, such objects cause the spacetime to curve. Gravity is simply a curvature of spacetime. The more massive is the object, stronger is the gravitational pull, hence only massive objects lead to gravitational lensing. When light from a distant source passes by a gravitational lens, path of light gets curved leading to the formation of ring or multiple images around the gravitational lens known as 'Einstein ring' or 'Einstein cross'. The bending of light results in light being focused, creating a magnified and brighter image of the lens to an observer on Earth. This helps us to observe objects that are very far and faint. With Hubble Telescope's sensitivity and high resolution, we can see faint and distant galaxies which otherwise would be impossible to observe. Gravitational lensing also provides valuable information about distribution of dark matter as it can bend light but not emit light.

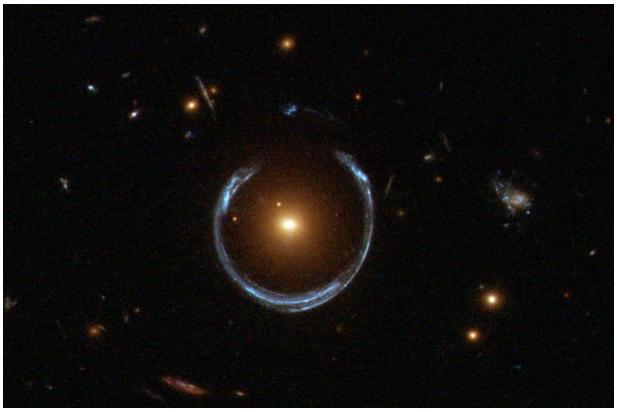
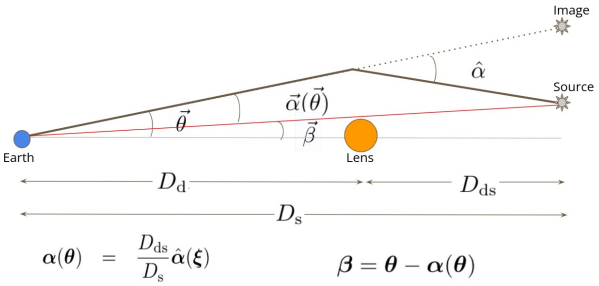


Fig. 11: Einstein ring around gravitational lens.

4.1. Gravitational Lensing by a Thin Lens

In a thin lense, the extent of the lens is much smaller than the distances between the lens and the source or the lens and the observer.An extended lens can be characterized by its surface density in the plane of the sky $\Sigma(\vec{\xi})$

Consider a small mass element $\Sigma(\vec{\xi'})d^2\xi'$. This small mass element will cause a deflection in its direction. The total deflection for light rays arriving at a position $\vec{\xi}$ will be the vector summation of all deflections due to all such mass elements in the plane of the lens.



The bold line shows the path of the light ray from a source which is at a true angular position $\vec{\beta}$. The light ray eventually reaches the observer on Earth at an angle $\vec{\theta}$. The total deflection angle $\vec{\alpha}$ is the angle between the light ray paths before and after the deflection as shown in the above figure. We also show the angular diameter distance between us and the deflector lens as D_d, that between us and the source as D_s , and that between the deflector and the source as D_{ds} .

One can define the scaled deflection angle defined to be the angle between the true position of the source and the observed image as shown in the above figure. The scaled deflection angle is given by

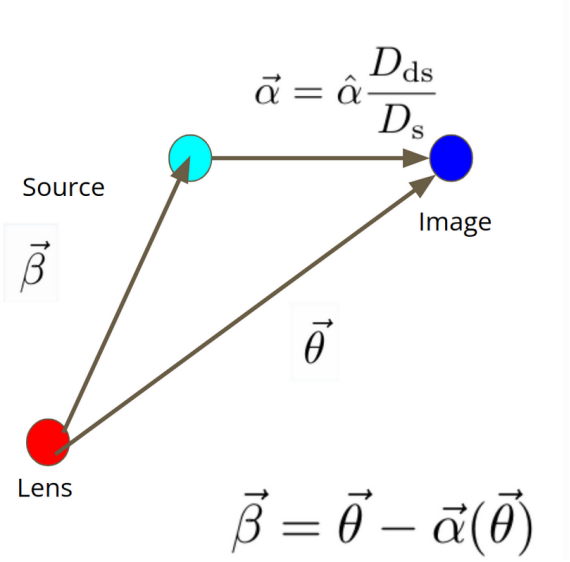
$$\vec{\alpha}(\vec{\theta}) = \frac{D_{ds}}{D_s}\,\hat{\vec{\alpha}}(\vec{\theta})$$

From geometry we also obtain the lens equation

$$\vec{\beta}=\vec{\theta}\text{-}\vec{\alpha}(\vec{\theta})$$

Note that here we are using vector signs to denote all angles. This is because the deflection angle from an extended deflector will be a vector not always pointing towards the center of mass of the deflector, and it depends upon the entirety of the mass distribution in the lens plane.

The easiest way to consider this is to consider the above diagram but as seen in the plane of the sky.



The lens equation is a non-linear equation which relates the true position of the source to the observed position of the image as seen by an observer via the deflection angle. Given the mass distribution in the lens plane, and a true source position, one can solve the above equation to obtain the location $\vec{\theta}$ where the image will be formed.

Note also that the above equation is non-linear. Although we get one value of the source position $\vec{\beta}$ for a given position of the image $\vec{\theta}$,we can have multiple values of $\vec{\theta}$ for a given true source position.

4.2. Types of Lensing

Gravitational lensing is observable in three different types, i.e, strong, weak and micro. The types are differentiated based on the size of lens, the kind of images formed and the distance between source and lens.

The types are described below:

1. Strong:-

This type of gravitational lensing is observed when the lens is massive and the source is considerably close to the lens. The source-observer geometry in this case is favourable for lensing. Light takes multiple paths due to bending around the lens, so as a result, more than one image of the source is observed. These images sometimes take the form of arcs or rings. If the source varies with time, the multiple images will vary with time as well. However, the light doesn't travel the same distance to each image, due to the bending of space. So there will be time delays for the changes in the images. These time delays can be used to calculate the hubble constant H_0.

Phenomena such as Einstein ring, Einstein cross, etc. arise due to strong gravitational lensing as they occur due to formation of multiple images of the source.

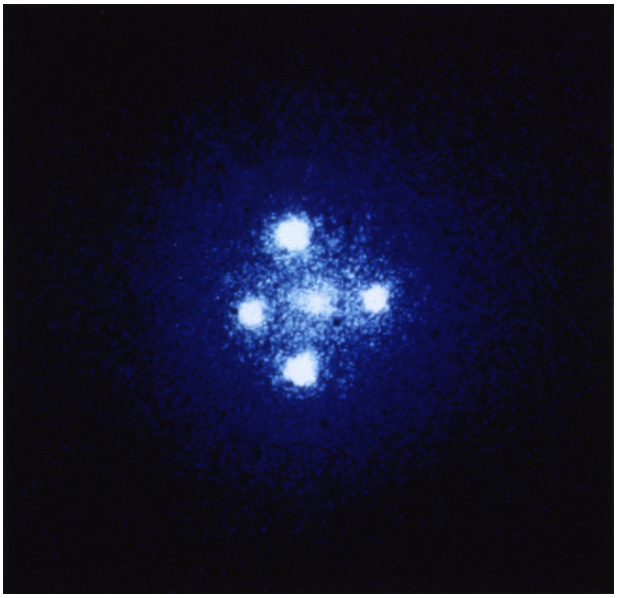


Fig. 12: Einstein Cross(Q2237+0305): 4 images of a single quasar. An example of strong lensing.

2. Weak:-

This type of lensing is observed when the lensing system is not strong enough. This implies that though the lens is a large mass, the geometry is not favourable in terms of distance between the source and lens. In this case,multiple images or arcs are not formed rather the image formed is distorted. It is stretched,called shear, and magnified,called convergence.Weak lensing is a useful complement to measures of the distribution of luminous mass such as galaxy surveys. Lensing measures all the mass, in particular the dark matter as well as the luminous matter.

3. Micro:-

This type of lensing is observed when the lens has a small mass and the geometry favourable in terms of the relative distance between the source and lens. In this case, no multiple images are formed and neither is the image so formed distorted. Due to the additional light reaching the observer due to bending, the object just seems to be brighter than it actually is.

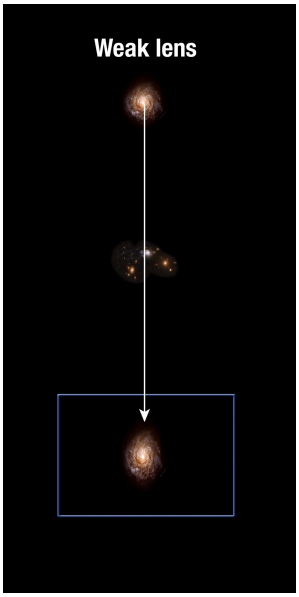


Fig. 13: Depiction of weak lensing.

Micro lensing is being employed to detect a type of dark matter called MACHOs (Massive Compact Halo Objects).Although MACHOs, as dark matter, cannot be seen themselves, if they pass in front of a source (e.g. a star nearby), they can cause the star to become brighter for a while, e.g. days or weeks. This effect has been observed, but determinations of the dark matter are not yet conclusive.

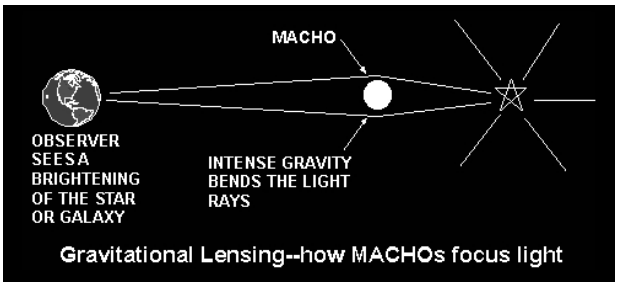


Fig. 14: MACHOs are a type of dark matter and are being detected by microlensing.

5. DETECTION OF STRONG GRAVITATIONAL LENSES

This is the main computational part of the project. It involves the sourcing of the database catalogues, preprocessing of the same and the development and training of the ResNet model for the predicability of the possible lensed candidates and its analysis.

5.1. A brief description about different space surveys (DESI, SDSS AND KiDS)

Gravitational lensing occurs when a massive object, like a galaxy cluster, bends the light from a more distant object. Strong gravitational lensing, characterized by multiple images of the background object or highly distorted arcs, provides a powerful tool for probing the mass distribution of lensing objects and the large-scale structure of the universe. Several surveys have been instrumental in the detection and analysis of strong gravitational lenses, including the Dark Energy Spectroscopic Instrument (DESI), the Sloan Digital Sky Survey (SDSS), and the Kilo-Degree Survey (KiDS). This section provides an overview of these surveys, focusing on their contributions to the field of strong gravitational lensing.

5.1.1. Dark Energy Spectroscopic Instrument (DESI)

DESI is a state-of-the-art spectroscopic survey designed to map the large-scale structure of the universe and measure the effect of dark energy. It aims to collect spectra for over 35 million galaxies and quasars, making it one of the most ambitious projects of its kind.

DESI's high-resolution spectra and extensive sky coverage allow for the precise identification and characterization of strong gravitational lenses. Key contributions include:

- **Spectral Identification:** DESI's ability to obtain detailed spectra enables the identification of lens and source galaxies. By distinguishing between foreground and background objects, DESI helps confirm lensing systems.
- **Mass Distribution Analysis:** The survey's data aids in modeling the mass distribution of lensing galaxies and clusters, providing insights into dark matter and galaxy evolution.
- **Large Sample Sizes:** With its vast dataset, DESI significantly increases the number of known strong lenses, facilitating statistical studies and rare lens discovery.

DESI has already identified numerous strong lenses, contributing to our understanding of galaxy mass profiles and the distribution of dark matter. Its findings help refine models of galaxy formation and evolution.

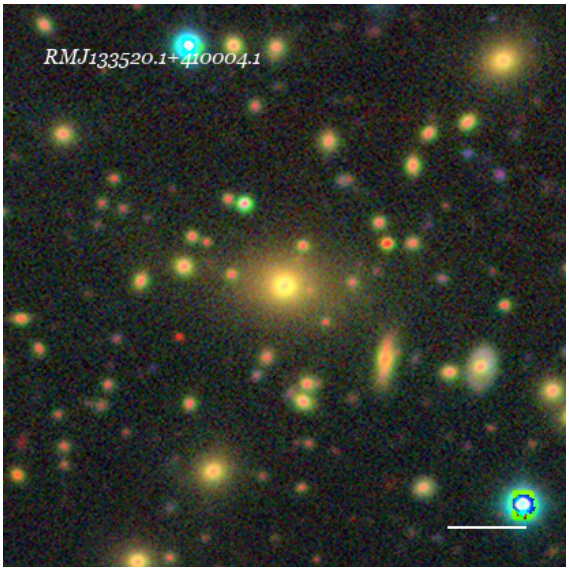


Fig. 15: Example of a strong gravitational lens identified by DESI.

5.1.2. Sloan Digital Sky Survey (SDSS)

The SDSS has been one of the most influential astronomical surveys since its inception in 2000. Covering a third of the sky, SDSS has created detailed maps of the universe, including images, spectra, and redshifts for millions of objects.

SDSS has made significant strides in the detection and study of strong gravitational lenses through its extensive imaging and spectroscopic data:

- **Imaging Data:** SDSS's high-quality imaging facilitates the visual identification of lensing features such as arcs and multiple images.
- **Spectroscopic Data:** The survey's spectra allow for precise redshift measurements, confirming the lensing nature of observed systems.
- **Citizen Science:** Projects like Galaxy Zoo have involved the public in identifying potential lenses, leveraging human pattern recognition to complement automated searches.

SDSS has discovered hundreds of strong lenses, providing valuable data for lens modeling and cosmological studies. Its contributions include detailed mass maps of lensing galaxies and clusters, enhancing our understanding of dark matter distribution.

5.1.3. Kilo-Degree Survey (KiDS)

KiDS is a European Southern Observatory (ESO) survey that uses the VLT Survey Telescope to image 1500 square degrees of the sky. It focuses on weak and strong gravitational lensing to study dark matter and dark energy.

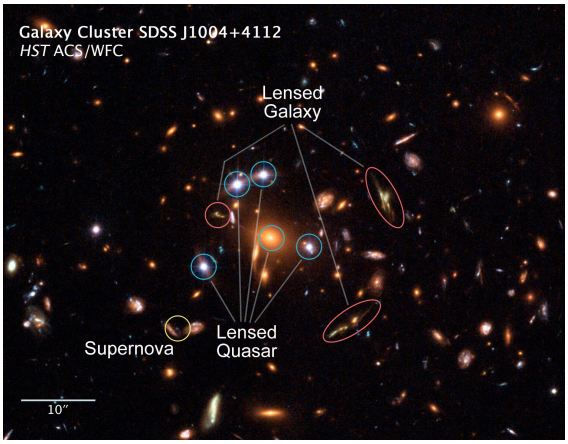


Fig. 16: Example of a strong gravitational lens identified by SDSS.

KiDS excels in detecting strong lenses due to its high-resolution imaging and dedicated lensing analyses:

KiDS has identified numerous strong lensing systems, contributing to the study of dark matter halos and the distribution of baryonic matter. Its high-quality data allows for precise lens modeling, essential for accurate mass reconstructions.

Fig. 17: Example of a strong gravitational lens identified by KiDS: Cluster of galaxies forming a question mark in $KIDS_2$14.0_0.5

5.1.4. Comparative Analysis

Survey Strengths:

- **DESI:** Excels in spectroscopic analysis, providing detailed redshift measurements and mass distribution data.
- **SDSS:** Combines extensive imaging and spectroscopy with public engagement, resulting in a large number of identified lenses.
- **KiDS:** Offers deep, high-resolution imaging ideal for detailed lensing studies and mass modeling.

Scientific Impact: Each survey contributes uniquely to the field of strong gravitational lensing:

- **DESI:** Advances our understanding of the universe's large-scale structure and dark energy.
- **SDSS:** Provides comprehensive maps of the cosmos, with significant contributions to galaxy evolution and dark matter research.
- **KiDS:** Enhances lens modeling techniques and collaborates widely to integrate findings with theoretical frameworks.

  Redshift vs Absolute Magnitude Plot: We have plotted the redshift vs absolute magnitude curve for the objects present in the datasets of these three surveys for their detailed analysis and comparison.

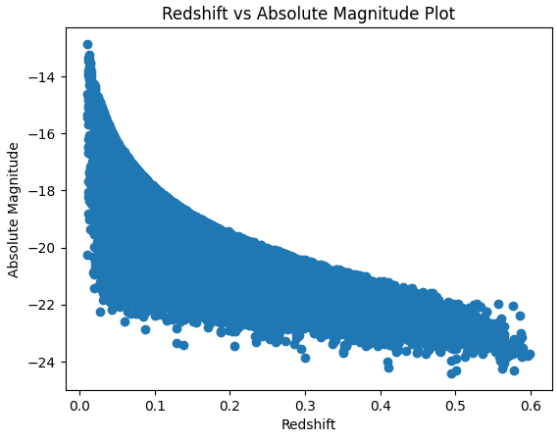


Fig. 18: **DESI Survey :** Plot of Redshift vs Absolute Magnitude for the objects in the dataset of DESI survey

5.2. Python and it's Libraries (Tractor and SExtractor, Numpy, Pandas, Astropy etc)

For handling astronomical data in this project, we used Astropy, Tractor, TensorFlow, NumPy, Pandas, Matplotlib and SciPy.

5.2.1. Tractor and Source Extractor (SeX)

Tractor is a algorithm for optimizing or sampling from models of astronomical objects. The approach is generative: given astronomical sources and a description of the image properties, the code produces

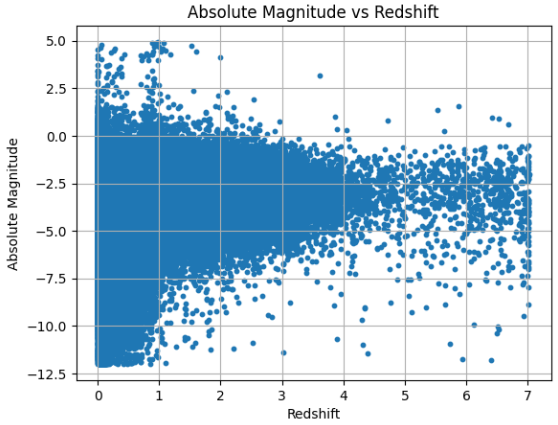


Fig. 19: **SDSS Survey :** Plot of Redshift vs Absolute Magnitude for the objects in the dataset of SDSS survey

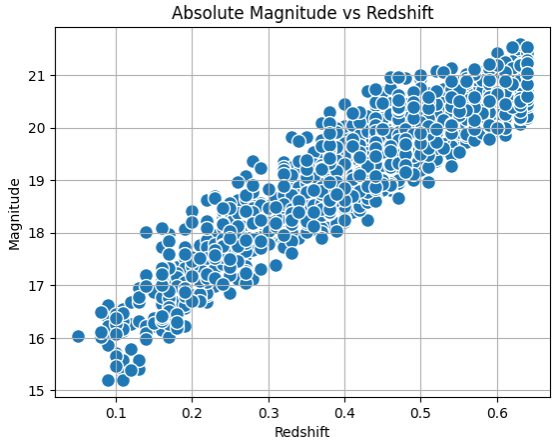


Fig. 20: **KiDs Survey :** Plot of Redshift vs Absolute Magnitude for the objects in the dataset of KiDs survey

pixel-space estimates or predictions of what will be observed in the images. We use this estimate to produce a likelihood for the observed data given the model: assuming our model space actually includes the truth (it doesn't, in detail), then if we had the optimal model parameters, the predicted image would only differ from the actually observed image by noise. It is the upgraded version of Source Extractor(SeX).

SExtractor or Source-Extractor is a program that builds a catalog of objects from an astronomical image. It is particularly oriented towards the reduction of large scale galaxy-survey data, but it also performs well on moderately crowded star fields. It excels at identifying and isolating sources in an image, such as stars, galaxies, and other astronomical objects. It uses a sophisticated algorithm to distinguish between genuine sources and background noise, ensuring accurate detection even in crowded fields. The software performs both aperture and profile-fitting photometry, providing precise measurements of the flux (brightness) of detected sources.

5.2.2. Numpy

NumPy (Numerical Python) is a fundamental library for optimized scientific computing in Python which it does by providing by powerful array objects, which allows for efficient operations on large multi-dimensional arrays and matrices. NumPy offers a wide range of mathematical functions such as mathematical functions, random number generators, linear algebra tools, different types of transformation algorithm and much more to operate on these arrays, enabling complex calculations and manipulations with ease.

One of the core features of NumPy is its ability to perform element-wise operations and broadcasting, which allows arithmetic operations on arrays of different shapes. This feature is particularly useful for data manipulation and mathematical computations, making code more concise and faster compared to traditional loops.

These capabilities make it an essential tool for data analysis, machine learning, and scientific research. Additionally, NumPy arrays serve as the primary data structure in many other scientific libraries and are compatible with other libraries and modules, including Pandas, SciPy, and Scikit-learn.

5.2.3. Pandas

Pandas is a powerful and versatile library for data manipulation and analysis in Python. Built on top of NumPy, Pandas introduces two primary data structures: Series (one-dimensional) and DataFrame (two-dimensional). These structures are designed to handle labeled data and facilitate a wide range of data manipulation tasks with ease.

One of the key strengths of Pandas is its ability to handle diverse data types, including integers, floats, strings, and complex database objects. This versatility makes it ideal for various data science tasks, from simple data cleaning and transformation to complex statistical analyses. Moreover, Pandas provides powerful group-by functionality, enabling the aggregation and summarization of data based on categorical variables.

It integrates seamlessly with other data science libraries in Python, such as Matplotlib for plotting and visualization, and TensorFlow alongwith Scikit-learn for machine learning (which we've used here).

5.2.4. AstroPy

Astropy is an exclusive library for astronomy and astrophysics in Python, designed to support the extensive needs of the astronomical community in computational astronomy. It provides a wide range of tools for data manipulation and structures, time analysis, visualization and FITS file support. It can be inte-

grated easily with other Machine Learning libraries to model different aspects of astronomical research.

5.2.5. *TensorFlow*

TensorFlow is an open-source machine learning framework developed by Google, widely used for building and deploying machine learning models. It provides a comprehensive ecosystem of tools, libraries, and community resources that facilitate the development and training of deep learning models.

It is extensively used in various domains, including computer vision, natural language processing, and time series analysis. It powers various applications ranging from image and speech recognition to predictive analytics and automated decision-making systems including a diverse range of applications in the astronomy community. Its versatility and robust features make it a preferred choice for academic research, industry applications, and cutting-edge AI development.

5.3. Obtaining Catalogs

We must first create a catalogue of images which we can use in turn to train our model to detect lenses.

5.3.1. *KiDS catalogue*

Data from KiDS catalogue is first downloaded in form of FITS images form the KiDS data release website. Run the files through SExtractor which helps us visualise the FITS images and create a catalogue.

5.3.2. *DESI survey*

For the DESI survey it's best to mass download all the images.

```
    import requests
import os

# Function to download a single image
def download_image(url, save_path):
    response = requests.get(url)
    if response.status_code == 200:
        with open(save_path, 'wb') as f:
            f.write(response.content)
        print(f"Downloaded: {save_path}")
    else:
        print(f"Failed to download {url}")

# Base URL for DESI Legacy Imaging Surveys
base_url = "http://legacysurvey.org/viewer/
    fits-cutout"

# Directory to save the downloaded images
save_dir = "desi_images"
os.makedirs(save_dir, exist_ok=True)

# Define the list of coordinates and other
    parameters for the cutouts
coordinates = [
    {'ra': 150.0, 'dec': 2.2, 'layer': 'ls-
        dr9', 'pixscale': 0.262, 'bands': '
        grz'},
    {'ra': 150.1, 'dec': 2.3, 'layer': 'ls-
        dr9', 'pixscale': 0.262, 'bands': '
        grz'},
    # Add more coordinates as needed
]

# Loop through the coordinates and download
    images
for coord in coordinates:
    ra = coord['ra']
    dec = coord['dec']
    layer = coord['layer']
    pixscale = coord['pixscale']
    bands = coord['bands']
    url = f"{base_url}?ra={ra}&dec={dec}&
        layer={layer}&pixscale={pixscale}&
        bands={bands}"
    save_path = os.path.join(save_dir, f"
        desi_image_ra{ra}_dec{dec}.fits")
    download_image(url, save_path)
```

Listing 1: Code for mass download

This images are the processed via tractor which creates our catalog

```
import numpy as np
from astropy.io import fits
from tractor import Tractor, Image, Catalog
from tractor.galaxy import ExpGalaxy,
    DevGalaxy
from tractor.psf import PixelizedPSF
from tractor.sky import ConstantSky
from tractor.ellipses import EllipseE

# Load the configuration
import yaml
with open('tractor_config.yaml', 'r') as file
    :
    config = yaml.safe_load(file)

# Load the image data
image_hdu = fits.open(config['image_file'])
image_data = image_hdu[0].data
image_wcs = image_hdu[0].header

# Load the PSF model
psf_hdu = fits.open(config['psf_file'])
psf_data = psf_hdu[0].data
psf = PixelizedPSF(psf_data)

# Load the sky model
sky_hdu = fits.open(config['sky_file'])
sky_data = sky_hdu[0].data
sky = ConstantSky(sky_data)

# Create the image object
image = Image(data=image_data, wcs=image_wcs,
     psf=psf, sky=sky)

# Define the initial source catalog (this can
    be from a previous SExtractor run or
    other sources)
# Here we use a dummy catalog for
    illustration purposes
catalog = Catalog([
```

```
    ExpGalaxy(ra=150.0, dec=2.0, flux=1000.0,
        shape=EllipseE(1.0, 0.5, 0.1)),
    DevGalaxy(ra=150.1, dec=2.1, flux=500.0,
        shape=EllipseE(1.0, 0.5, 0.1)),
])

# Create the Tractor object
tractor = Tractor([image], catalog)

# Optimize the model parameters
tractor.optimize()

# Save the output catalog
catalog.write_fits(config['output_catalog'],
    overwrite=True)

# Close the fits files
image_hdu.close()
psf_hdu.close()
sky_hdu.close()
```

Listing 2: Code to run tractor

But the data set and computing resources required for testing and identifying new lenses is really huge. Data from each source easily takes up multiple TBs and the processing is also a very large process

5.4. Some Important Concepts in Machine Learning:

Before we dive into the application of the CNN model on our dataset, we need to understand a few basic terminologies and processes for a better understanding of it:

5.4.1. Neuron:

A neuron is the basic building block of a neural network. It is a mathematical function modeled on the working of biological neurons. One or more inputs are separately weighted, inputs are summed and passed through a nonlinear function to produce output. Every neuron holds an internal state called activation signal. Each connection link carries information about the input signal. Every neuron is connected to another neuron via connection link.

5.4.2. The Simple Linear Unit:

- **Input Features:** The columns that are inputted into our model. They serve as the input variables (independent variables) that the model uses to make predictions or classifications.
- **Target Variable:** The prediction target in machine learning, also known as the dependent variable or outcome variable, is the specific variable that a model aims to predict based on input features.
- **Weights:** Each input neuron is associated with a weight, which represents the strength of the connection between the input neuron and the output neuron.
- **Bias:** A bias term is added to the input layer to provide additional flexibility in modeling complex patterns in the input data.
- **Output:** The output is the value that is predicted by the neuron in our model.
- **Activation Function:** An activation function is simply some function we apply to each of a layer's outputs.The most common is the rectifier function max(0,x). The output becomes $max(0, w * x + b)$

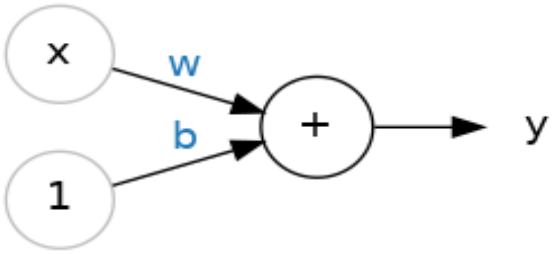


Fig. 21: The Simple Linear Unit

Now suppose the input is x. Its connection to the neuron has a weight which is w. Whenever a value flows through a connection, you multiply the value by the connection's weight. For the input x, what reaches the neuron is w times x. A neural network "learns" by modifying its weights. The b is a special kind of weight we call the bias. The bias doesn't have any input data associated with it; instead, we put a 1 in the diagram so that the value that reaches the neuron is just b (since 1 times b = b). The bias enables the neuron to modify the output independently of its inputs. The y is the value the neuron ultimately outputs. To get the output, the neuron sums up all the values it receives through its connections. **This neuron's activation is $y = wx + b$.** article titlesec

5.4.3. Loss Functions

Loss functions, also known as cost functions, are a crucial component in training machine learning models, especially in supervised learning. They measure the difference between the predicted output and the actual output, guiding the optimization process to improve model performance.

Types of Loss Functions

There are various types of loss functions, each suited for different kinds of tasks. Some of the most commonly used loss functions are:

Mean Squared Error (MSE): Mean Squared Error is widely used for regression tasks. It calculates the average of the squared differences between the predicted and actual values.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{1}$$

```python
from tensorflow.keras import backend as K

def mean_squared_error(y_true, y_pred):
    return K.mean(K.square(y_pred - y_true),
        axis=-1)

# Example usage:
import numpy as np
y_true = np.array([3.0, -0.5, 2.0, 7.5])
y_pred = np.array([2.5, 0.0, 2.1, 7.8])
loss = mean_squared_error(y_true, y_pred)
print(f'MSE Loss: {K.eval(loss)}')
```

Listing 3: Mean Squared Error in Keras

Cross-Entropy Loss: Cross-Entropy Loss is commonly used for classification tasks. It measures the performance of a classification model whose output is a probability value between 0 and 1.

$$\text{Cross-Entropy} = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(\hat{y}_i) \tag{2}$$

```python
from tensorflow.keras.losses import
    categorical_crossentropy

# Example usage:
y_true = np.array([[0, 1], [1, 0]])
y_pred = np.array([[0.1, 0.9], [0.8, 0.2]])
loss = categorical_crossentropy(y_true,
    y_pred)
print(f'Cross-Entropy Loss: {K.eval(K.mean(
    loss))}')
```

Listing 4: Cross-Entropy Loss in Keras

Huber Loss: Huber Loss is used for regression tasks and is less sensitive to outliers in data than the MSE. It combines MSE and Mean Absolute Error (MAE) to handle outliers effectively.

$$L_\delta = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{for } |y_i - \hat{y}_i| \le \delta \\ \delta|y_i - \hat{y}_i| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases} \tag{3}$$

```python
from tensorflow.keras.losses import Huber

huber_loss = Huber()

# Example usage:
y_true = np.array([3.0, -0.5, 2.0, 7.5])
y_pred = np.array([2.5, 0.0, 2.1, 7.8])
loss = huber_loss(y_true, y_pred)
print(f'Huber Loss: {K.eval(loss)}')
```

Listing 5: Huber Loss in Keras

Binary Cross-Entropy Loss: Binary Cross-Entropy Loss is used for binary classification tasks. It evaluates the performance of a classification model whose output is a probability value between 0 and 1 for a single class.

$$\text{Loss} = -\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] \tag{4}$$

```python
from tensorflow.keras.losses import
    binary_crossentropy

# Example usage:
y_true = np.array([1.0, 0.0, 1.0, 0.0])
y_pred = np.array([0.8, 0.4, 0.7, 0.2])
loss = binary_crossentropy(y_true, y_pred)
print(f'Binary Cross-Entropy Loss: {K.eval(K.
    mean(loss))}')
```

Listing 6: Binary Cross-Entropy Loss in Keras

Choosing a Loss Function

Choosing the appropriate loss function is critical for the success of a machine learning model. The selection depends on the type of problem, the nature of the data, and the specific requirements of the application.

Regression Tasks For regression tasks, Mean Squared Error (MSE) and Huber Loss are commonly used. MSE is preferable when the data does not have many outliers, whereas Huber Loss is more robust to outliers.

Classification Tasks For classification tasks, Cross-Entropy Loss and Binary Cross-Entropy Loss are widely used. Cross-Entropy Loss is used for multi-class classification problems, while Binary Cross-Entropy Loss is used for binary classification problems.

Handling Imbalanced Data In cases of imbalanced data, loss functions can be modified to account for class imbalance. This can be done by assigning different weights to different classes, which helps the model to pay more attention to the minority class.

```python
from tensorflow.keras.losses import
    CategoricalCrossentropy

weights = np.array([0.7, 0.3])
loss_fn = CategoricalCrossentropy()
y_true = np.array([[0, 1], [1, 0]])
y_pred = np.array([[0.1, 0.9], [0.8, 0.2]])
```

```
# Manually apply weights
loss = loss_fn(y_true, y_pred)
weighted_loss = K.mean(loss * weights)
print(f'Weighted Cross-Entropy Loss: {K.eval(
    weighted_loss)}')
```

Listing 7: Weighted Cross-Entropy Loss in Keras

In conclusion, loss functions are fundamental to the training and optimization of machine learning models. Selecting the right loss function tailored to the specific task and data characteristics is essential for achieving high performance.

5.4.4. Gradient Descent:

Cost function or Loss Function measures the performance of a machine learning model for a data set. Cost function quantifies the error between predicted and expected values and presents that error in the form of a single real number. The purpose of cost function is to be minimized.

Gradient Descent is a widely used optimization algorithm in machine learning and deep learning that minimises the cost function of a neural network model during training. It is a numerical optimization algorithm that aims to find the optimal parameters—weights and biases—of a neural network by minimizing a defined cost function. It works by iteratively adjusting the weights or parameters of the model by moving in the direction of the steepest decrease in the cost function until the minimum of the cost function is reached. The algorithm calculates gradients, representing the partial derivatives of the cost function(measures change in output for small change in input) concerning each parameter. These gradients guide the updates, ensuring convergence towards the optimal parameter values that yield the lowest possible cost.

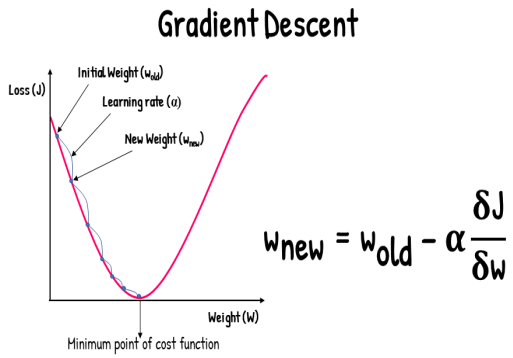


Fig. 22: Gradient Descent Algorithm

Learning rate is a hyperparameter in machine learning that controls the step size at which the weights of a neural network are updated during training. It specifies the amount by which the model's parameters are altered in the direction opposite to the gradient of the loss function. The learning rate is crucial in determining the speed at which a model learns. An efficient learning rate is one that is low enough for the network to converge at some point of time while also being high enough to train in a reasonable amount of time. Adjusting the learning rate is crucial to balance convergence speed and avoiding overshooting the optimal solution.

5.4.5. Backpropagation:

In machine learning, backpropagation is an effective algorithm used to train artificial neural networks. Backpropagation is an iterative algorithm, that helps to minimize the cost function by determining which weights and biases should be adjusted. During every epoch, the model learns by adapting the weights and biases to minimize the loss by moving down toward the gradient of the error. Computing the gradient in the backpropagation algorithm helps to minimize the cost function and it can be implemented by using the mathematical rule called chain rule from calculus to navigate through complex layers of the neural network. It is like fixing mistakes in a network. You adjust the connections between neurons to reduce errors in predictions. By doing this, the network gets better at its job as it learns from its mistakes over time.

Neural network training is about finding weights that minimize prediction error. We usually start our training with a set of randomly generated weights. Then, backpropagation is used to update the weights in an attempt to correctly map arbitrary inputs to outputs. Backpropagation accelerates the learning

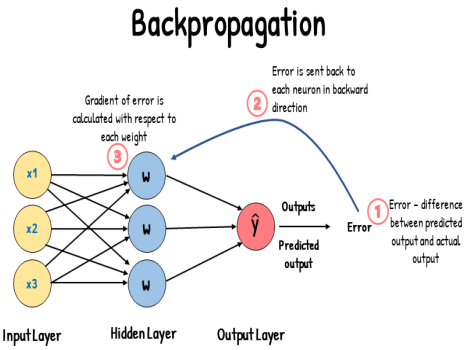


Fig. 23: Backpropagation

process by directly updating weights based on the calculated error derivatives. This efficiency is particularly advantageous in training deep neural networks, where learning features of a function can be time-consuming. It also enables neural networks to generalize well to unseen data by iteratively adjusting weights during training. This generalization ability is crucial for developing models that can make accurate predictions on new, unseen examples. Backpropaga-

tion also scales well with the size of the dataset and the complexity of the network.

5.4.6. Convolutional Neural Networks:

A Convolutional Neural Network (CNN) is a type of deep learning algorithm that is particularly well-suited for image recognition and processing tasks. It is made up of multiple layers, including convolutional layers, pooling layers, and fully connected layers. The architecture of CNNs is inspired by the visual processing in the human brain, and they are well-suited for capturing hierarchical patterns and spatial dependencies within images. Convolutional neural networks ingest and process images as tensors, and tensors are matrices of numbers with additional dimensions.

Key components of a Convolutional Neural Network include:

- **Convolutional Layers:** These layers apply convolutional operations to input images, using filters (also known as kernels) to detect features such as edges, textures, and more complex patterns. Convolutional operations help preserve the spatial relationships between pixels.

  The result of a convolved feature depends on certain parameters that we need to define before the convolution is performed:

  1. Stride: Stride is the number of pixels by which we slide our filter matrix over the input matrix. When the stride is 1 then we move the filters one pixel at a time. When the stride is 2, then the filters jump 2 pixels at a time as we slide them around. Having a larger stride will produce smaller feature maps.

  2. Padding: Padding involves adding extra border pixels around input images or feature maps in CNNs. Mostly padding would be 1. It helps in retaining the information of the image to subsequent layers for processing. Types of Padding include valid padding and same padding.

  3. Depth: The depth of an output volume controls the number of neurons (i.e., filters) in the layer that connect to a local region of the input volume. Each filter produces an activation map that "activates" in the presence of oriented edges or blobs or color. For a given layer, the depth of the activation map will be K, or simply the number of filters we are learning in the current layer.

- **Pooling Layers:** Pooling layers down-sample the spatial dimensions of the input, reducing the computational complexity and the number of parameters in the network. The pooling layer works by dividing the input feature map into a set of non-overlapping regions, called pooling regions. Each pooling region is then transformed into a single output value, which represents the presence of a particular feature in that region. The most common types of pooling operations are max pooling and average pooling.

  In **Max pooling,** the output value for each pooling region is simply the maximum value of the input values within that region. This has the effect of preserving the most salient features in each pooling region, while discarding less relevant information. Max pooling is often used in CNNs for object recognition tasks, as it helps to identify the most distinctive features of an object, such as its edges and corners.

  In **Average pooling,** the output value for each pooling region is the average of the input values within that region. This has the effect of preserving more information than max pooling, but may also dilute the most salient features. Average pooling is often used in CNNs for tasks such as image segmentation and object detection, where a more fine-grained representation of the input is required.

- **Activation Functions:** Non-linear activation functions, such as Rectified Linear Unit (ReLU), introduce non-linearity to the model, allowing it to learn more complex relationships in the data. ReLU is an element wise operation (applied per pixel) and replaces all negative pixel values in the feature map by zero. The purpose of ReLU is to introduce non-linearity in our CNN, since most of the real-world data we would want our CNN to learn would be non-linear (Convolution is a linear operation — element wise matrix multiplication and addition, so we account for non-linearity by introducing a non-linear function like ReLU).

- **Fully Connected Layers:** These layers are responsible for making predictions based on the high-level features learned by the previous layers. They connect every neuron in one layer to every neuron in the next layer. The term "Fully Connected" implies that every neuron in the previous layer is connected to every neuron on the next layer. The output from the convolutional and pooling layers represent high-level features of the input image.

  The purpose of the Fully Connected layer is to use these features for classifying the input image into various classes based on the training dataset. The sum of output probabilities from the Fully Connected Layer is 1. This is ensured by using the **Softmax** as the activation function in the output layer of the Fully Connected Layer. The Softmax function takes a vector of arbitrary real-valued scores and squashes it to a vector of values between zero and one that sum to one.

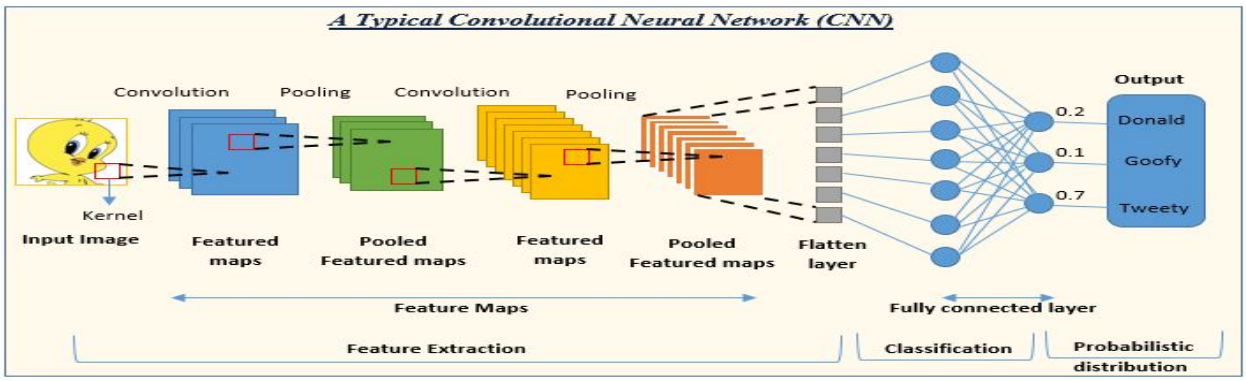


Fig. 24: Convolutional Neural Network

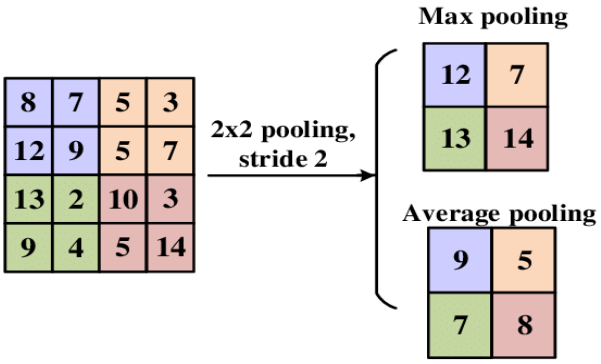


Fig. 25: Max Pooling and Average Pooling

5.4.7. Optimizers:

In machine learning, optimizers and loss functions are two components that help improve the performance of the model. By modifying the model's parameters to reduce the loss function value, the optimizer contributes to its improvement. RMSProp, ADAM, and SGD are a few examples of optimizers. The optimizer's job is to determine which combination of the neural network's weights and biases will give it the best chance to generate accurate predictions.

Before we proceed, it's essential to acquaint yourself with a few terms:

1. Epoch: The number of times the algorithm iterates over the entire training dataset.

2. Batch weights: The number of samples used for updating the model parameters.

3. Sample: A single record of data in a dataset.

4. Learning Rate: A parameter determining the scale of model weight updates

5. Weights and Bias: Learnable parameters in a model that regulate the signal between two neurons.

- **Gradient Descent:** Gradient Descent has been discussed above and can be used as an optimization technique but there are a few disadvantages to using Gradient Descent as an optimizer.The major problems are saddle points and the local minima of the loss function. At the saddle points and the local minima the loss function becomes flat and the gradient at this point goes towards zero. A gradient close to zero in a saddle point or in a local minimum does not improve the weight parameters and prevents the whole learning process. Also if the dataset is too large it becomes computationally expensive and requires large memory.

- **Stochastic Gradient Descent(SGD):** It's a variation of the Gradient Descent algorithm. In Gradient Descent, we analyze the entire dataset in each step, which may not be efficient when dealing with very large datasets. In Stochastic Gradient Descent, we process just one example at a time to perform a single step. So, if the dataset contains 10000 rows, SGD will update the model parameters 10000 times in a single cycle through the dataset, as opposed to just once in the case of Gradient Descent.

  The process is as follows:

  1. Select an example from the dataset.

  2. Calculate its gradient.

  3. Utilize the calculated gradient from step 2 to update the model weights.

  4. Repeat steps 1 to 3 for all examples in the training dataset.

  5. Completing a full pass through all the examples constitutes one epoch.

  6. Repeat this entire process for several epochs as specified during training.

  Its advantages include that it requires less memory and we may get a new minima. But its disadvantage is that SGD algorithm is noisier and takes more iterations as compared to gradient descent.

- **SGD with Momentum:** In Stochastic Gradient Descent, we don't calculate the precise

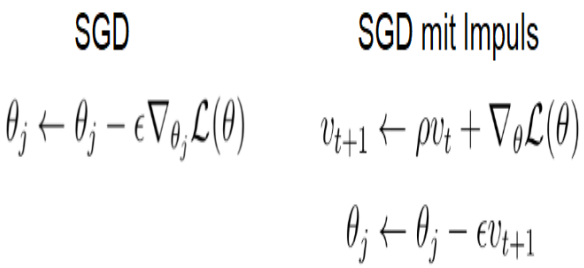


Fig. 26: SGD vs SGD with Momentum

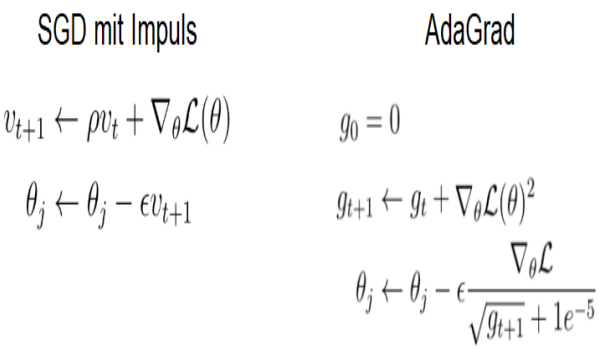


Fig. 27: SGD with Momentum vs AdaGrad

derivative of our loss function. Instead, we estimate it using a small batch. This results in "noisy" derivatives, which implies that we don't always move in the optimal direction. To address this issue, Momentum was introduced to mitigate the noise in SGD. It speeds up convergence towards the relevant direction and diminishes fluctuations in irrelevant directions.

The concept behind Momentum involves denoising the derivatives by employing an exponential weighting average by assigning more weight to recent updates compared to previous ones.

Its advantages are that it mitigates parameter oscillations and reduces parameter variance. It also achieves faster convergence compared to standard gradient descent. Its disadvantage is that it introduces an additional hyper-parameter that must be chosen manually and with precision.

- **AdaGrad:**AdaGrad, short for adaptive gradient, signifies that the learning rates are adjusted or adapted over time based on previous gradients. A limitation of the previously discussed optimizers is the use of a fixed learning rate for all parameters throughout each cycle. This can hinder the training features which often exhibit small average gradients causing them to train at a slower pace. While one potential solution is to set different learning rates for each feature, this can become complex . AdaGrad addresses this issue by implementing the concept that the more a feature has been updated in the past, the less it will be updated in the future. This provides an opportunity for other features, such as sparse features, to catch up. AdaGrad, as an optimizer, dynamically adjusts the learning rate for each parameter at every time step 't'.

  Its advantages include adaptive learning rates that facilitate effective training of all features. Disadvantages include that with a large number of iterations, the learning rate diminishes to extremely small values, causing slow convergence.

- **RMSProp:** There is a slight modification of AdaGrad called "RMSProp". This modification is intended to solve the previously described problem that can occur with AdaGrad. In RMSProp, the running sum of squared gradients g_{t+1} is maintained. However, instead of allowing this sum to increase continuously over the training period, we allow the sum to decrease.

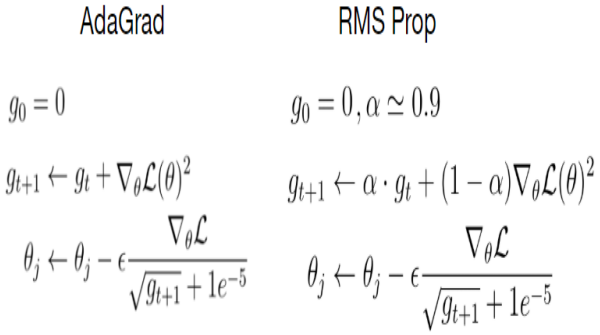


Fig. 28: RMSProp vs AdaGrad

For RMSProp, the sum of squared gradients is multiplied by a decay rate and the current gradient – weighted by (1- decay rate) – is added. The update step in the case of RMSProp looks the same as in AdaGrad. Here we divide the current gradient by the sum of the squared gradients to get the nice property of speeding up the updating of the weights along one dimension and slowing down the motion along the other.

Its advantages are that it prevents the learning rate from decaying, allowing continuous training without premature stopping. Its disadvantages are that it involves higher computational complexity due to increased parameter, making it more computationally expensive.

- **Adam:** So far, we have used the momentum term to determine the velocity of the gradient and update the weight parameter in the direction of that velocity. In the case of AdaGrad and

RMSProp, we used the sum of squared gradients to scale the current gradient so that we could update the weights in each space dimension with the same ratio.

These two methods seemed like pretty good ideas. Why don't we just take the best of both worlds and combine these ideas into a single algorithm? That's exactly the concept behind the final optimization algorithm I'd like to introduce. This algorithm is called ADAM.

So far, ADAM has incorporated the nice features of the previous two optimization algorithms. However, there is a small problem with ADAM, and that is what happens at the beginning of training.

At the very first time step t=0, the first and second pulse terms m_0 and v_0 are set to zero. After the first update of the second momentum v_1, this term is still very close to zero. When we update the weight parameters, we divide by a very small second momentum term v_1. This leads to a very large first update step.

$$m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * \nabla w_t$$

$$v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * (\nabla w_t)^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} * \hat{m}_t$$

Fig. 29: Adam

This very large first update step is not the result of the geometry of the problem, but an artifact of the fact that we initialized the first and second momentum to zero. To address the problem of large update steps happening at the beginning of training, ADAM includes a correction clause. After the initial update of the first and second pulses, we make an unbiased estimate of these pulses by considering the current time step. With the so-called bias correction, we obtain the corrected first and second impulses.

These correction cause the values of the first and second impulse to be higher at the beginning of the training than without this correction. As a result, the first update step of the neural network weight parameters does not become too large. Thus, the training is not already messed up at the very beginning. With the additional bias corrections, we obtain the complete form of the ADAM optimizer.

5.5. Neural Network Model

5.5.1. Introduction

In this section, we delve into the specifics of our Convolutional Neural Network (CNN) model, designed to detect gravitational lensing in astronomical images. The model leverages the Tractor algorithm for preprocessing and data augmentation, ensuring accurate and robust lensing detection.

5.5.2. Data Preparation

The first step involves preparing the data, which includes normalization, resizing, and augmentation of images from surveys such as DESI, SDSS, and KiDS. We use various libraries for data handling and preprocessing.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from astropy.io import fits
from astropy.table import Table
from tractor import Tractor, Image
from tensorflow.keras.preprocessing.image
    import ImageDataGenerator

# Define image properties
image_height, image_width = 128, 128  #
    Example dimensions
batch_size = 32

# Data augmentation and preprocessing
train_datagen = ImageDataGenerator(
    rescale=1./255,        # Normalize pixel
        values to [0, 1]
    rotation_range=20,     # Randomly rotate
        images
    width_shift_range=0.2, # Randomly shift
        images horizontally
    height_shift_range=0.2, # Randomly shift
        images vertically
    shear_range=0.2,       # Shear
        transformations
    zoom_range=0.2,        # Zoom
        transformations
    horizontal_flip=True, # Randomly flip
        images horizontally
    fill_mode='nearest',  # Fill mode for
        newly created pixels
    validation_split=0.2  # Split for
        validation data
)

# Load training and validation data
train_generator = train_datagen.
    flow_from_directory(
    'path/to/data',
    target_size=(image_height, image_width),
    batch_size=batch_size,
    class_mode='binary',  # Assuming binary
        classification for lensing
    subset='training'
)
```

```
validation_generator = train_datagen.
    flow_from_directory(
    'path/to/data',
    target_size=(image_height, image_width),
    batch_size=batch_size,
    class_mode='binary',
    subset='validation'
)
```

Listing 8: Data Preparation

5.5.3. Model Architecture

The architecture of our CNN model consists of multiple convolutional layers, pooling layers, and dense layers designed to extract and interpret features from the images. We use TensorFlow and Keras for building the neural network.

```
from tensorflow.keras.models import
    Sequential
from tensorflow.keras.layers import Conv2D,
    MaxPooling2D, Flatten, Dense, Dropout,
    BatchNormalization

# Define the CNN model
model = Sequential([
    # First convolutional block
    Conv2D(32, (3, 3), activation='relu',
        input_shape=(image_height,
        image_width, 3)),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    # Second convolutional block
    Conv2D(64, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    # Third convolutional block
    Conv2D(128, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),

    # Fully connected layers
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')  #
        Assuming binary classification
])

# Compile the model
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

Listing 9: Model Architecture

5.5.4. Training the Model

The model is trained using the training data generator, with early stopping to prevent overfitting. We utilize the callback functionality in Keras to implement early stopping.

```
from tensorflow.keras.callbacks import
    EarlyStopping

# Define early stopping
early_stopping = EarlyStopping(monitor='
    val_loss', patience=10,
    restore_best_weights=True)

# Train the model
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples
        // batch_size,
    validation_data=validation_generator,
    validation_steps=validation_generator.
        samples // batch_size,
    epochs=50,
    callbacks=[early_stopping]
)
```

Listing 10: Training the Model

5.5.5. Evaluation

After training, the model is evaluated on a separate test set to assess its performance. We plot the training and validation accuracy and loss to visualize the model's performance.

```
# Evaluate the model
test_loss, test_accuracy = model.evaluate(
    validation_generator)
print(f'Test accuracy: {test_accuracy:.2f}')

# Plotting training history
plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='
    Train Accuracy')
plt.plot(history.history['val_accuracy'],
    label='Validation Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='
    Train Loss')
plt.plot(history.history['val_loss'], label='
    Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.title('Training and Validation Loss')

plt.show()
```

Listing 11: Evaluation

6. Results

The results of the model demonstrate its capability to accurately detect gravitational lensing in astronomical images. The model achieved high accuracy on the validation set, indicating its robustness and effectiveness. Further details, including confusion matrices and ROC curves, are presented to illustrate the model's performance comprehensively.

7. Future Prospect

We have already understood the importance of finding and cataloguing gravitational lenses. We have also pointed out the problems in finding gravitational lenses using traditional methods. The model we have made is capable of classifying a given image of a galaxy field into lens or non-lens. But still a lot of potential is left in the project. Terabytes and terabytes of astronomical data are generated daily, and to detect new gravitational lenses, we need much more computational resources, which the computing cluster at IIT Kanpur could provide. The following steps can be taken to realize the true potential of our project:

- **Improving the model-**Train our model on a lot more images that could be generated by the simulation method discussed above. The number of images could be in the range of a hundred thousand. This will reduce the chance of missing a potential strong lens and make our model more efficient
- **Finding undetected lenses-**Finding new lenses in the latest data release of KiDS and DESI by running our model on the data. The data combined is more than a few terabytes. Running our model on this massive amount of data would only be possible on a supercomputer. We plan to present our model to the SPASE department of IIT Kanpur and request computational resources. Combined with a better CNN model, this could lead to the publishing of many new lenses in a research paper.
- **Creating a universal astronomical tool-** The long-term vision of this project is to set up a pipeline or standard method in the form of a package or Python library that could automatically detect new lenses continuously as the data is being generated from any survey. Till now, the gravitational lens detection sector is very cluttered, and there is no standard model to detect lenses. Each student or researcher has to train their own model, the information of which is very scarce in the first place. This leads to many inefficiencies in the detection process. If we set up this pipeline in the future, then it would be a very big achievement in astronomy and cosmology research.

8. Bibliography

- Storfer, C., et al. "New Strong Gravitational Lenses from the DESI Legacy Imaging Surveys Data Release 9." ArXiv, 2022, /abs/2206.02764. Accessed 11 Jul. 2024.
- Davies, Andrew, et al. "Using Convolutional Neural Networks to Identify Gravitational Lenses in Astronomical Images." Monthly Notices of the Royal Astronomical Society, vol. 487, no. 4, 2019, pp. 5263-5271, https://doi.org/10.1093/mnras/stz1288. Accessed 11 Jul. 2024.
- Wilde, Joshua, et al. "Detecting Gravitational Lenses Using Machine Learning: Exploring Interpretability and Sensitivity to Rare Lensing Configurations." Monthly Notices of the Royal Astronomical Society, vol. 512, no. 3, 2022, pp. 3464-3479, https://doi.org/10.1093/mnras/stac562. Accessed 11 Jul. 2024
- Bertin, E. & Arnouts, S. 1996: SExtractor: Software for source extraction, Astronomy & Astrophysics Supplement 317, 393