

Copyright is owned by the Author of the thesis. Permission is given for a copy to be downloaded by an individual for the purpose of research and private study only. The thesis may not be reproduced elsewhere without the permission of the Author.

DEEP LEARNING FOR ASTEROID DETECTION IN LARGE ASTRONOMICAL SURVEYS

A THESIS PRESENTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN
COMPUTER SCIENCE
AT MASSEY UNIVERSITY, ALBANY,
NEW ZEALAND.

Preeti Cowan

2022

Abstract

The MOA-II telescope has been operating at the Mt John Observatory since 2004 as part of a Japan/NZ collaboration looking for microlensing events. The telescope has a total field of view of 1.6×1.3 degrees and surveys the Galactic Bulge several times each night. This makes it particularly good for observing short duration events. While it has been successful in discovering exoplanets, the full scientific potential of the data has not yet been realised. In particular, numerous known asteroids are hidden amongst the MOA data. These can be clearly seen upon visual inspection of selected images. There are also potentially many undiscovered asteroids captured by the telescope. As yet, no tool exists to effectively mine archival data from large astronomical surveys, such as MOA, for asteroids. The appeal of deep learning is in its ability to learn useful representations from data without significant hand-engineering, making it an excellent tool for asteroid detection. Supervised learning requires labelled datasets, which are also unavailable.

The goal of this research is to develop datasets suitable for supervised learning and to apply several CNN-based techniques to identify asteroids in the MOA-II data. Asteroid tracklets can be clearly seen by combining all the observations on a given night and these tracklets form the basis of the dataset. Known asteroids were identified within the composite images, forming the seed dataset for supervised learning. These images were used to train several CNNs to classify images as either containing asteroids or not. The top five networks were then configured as an ensemble that achieved a recall of 97.67%. Next, the YOLO object detector was trained to localize asteroid tracklets, achieving a mean average precision (mAP) of 90.97%. These trained networks will be applied to 16 years of MOA archival data to find both known and unknown asteroids that have been observed by the telescope over the years. The methodologies developed can also be used by other surveys for asteroid recovery and discovery.

Acknowledgements

I would like to thank my supervisor, Ian Bond, for his guidance and support all through this research. Thank you to my co-supervisors: Napoleon Reyes for advising the deep learning aspects of the research and Nicholas Rattenbury for providing the physics perspective. Thank you also to Michele Bannister for welcoming me into the community of planetary scientists. Special thanks to our post-graduate administrator Linh Mills who always made the administrative tasks as painless as possible.

It takes the proverbial to bring a PhD to fruition and I must extend thanks to my friends and colleagues through the years. You made this journey more rewarding.

None of this would have been possible without the unending support of my family. Thank you to my parents for always believing in me, to my children for making me better, and to my husband whose love and encouragement were a constant guiding light.

Contents

Abstract	ii
Acknowledgements	iii
1 Introduction	1
1.1 Research Goal	4
1.2 Report Structure	5
2 Review of Asteroid Detection Techniques	6
2.1 Asteroids in Our Solar System	7
2.2 From Early Detectors to MOPS: An Overview	10
2.3 Deep Learning for Discovering Small Bodies in our Solar System .	13
2.3.1 Meteor Detection	13
2.3.2 Euclid	14
2.3.3 DeepStreaks	15
2.3.4 ATLAS	17
2.3.5 Tails	19
2.3.6 Hubble Asteroid Hunter	20
2.4 Summary	22
3 MOA-II Telescope: Building The Dataset	23
3.1 Difference Imaging Analysis for MOA-II	29
3.2 Asteroids in MOA-II	33
3.2.1 MOA-II Archival Data	35
3.2.2 Known Asteroids	36
3.2.3 Building The Dataset	37
3.3 Summary	50

4	Tracklet Classification With CNNs	51
4.1	Data	51
4.1.1	Images with Tracklets	52
4.1.2	Images without Tracklets	54
4.2	Network Architecture	56
4.2.1	Stacked Architectures	56
4.2.2	Inception-ResNet	61
4.3	Training	71
4.4	Results	74
4.4.1	Ensemble	83
4.5	Discussion	87
5	Tracklet localization with YOLO	89
5.1	Background	89
5.1.1	YOLOv1	90
5.1.2	YOLOv2	93
5.1.3	YOLOv3	94
5.2	Data	95
5.3	YOLOv4	99
5.4	Training	104
5.5	Results	105
5.6	Discussion	110
6	Tracklet Classification with CNN-LSTM	112
6.1	Background	112
6.2	Data	116
6.3	CNN-LSTM Model	119
6.4	Training	122
6.5	Results	123
6.6	Discussion	124
7	Conclusions	126
7.1	Dataset	127
7.2	Classification CNN	128
7.3	YOLO	129
7.4	CNN-LSTM	131

7.5	Future Work	131
7.6	Conclusion	134
A	Neural Networks Primer	135
B	Minor Planet Database Schema	141
C	More Metrics for the Classification CNNs	143
D	More Detections from YOLOv4 and YOLOv4-tiny	151
	Bibliography	156

List of Tables

3.1	Imaging data numbers	47
4.1	Summary of data from GB5-R5 used for classification	54
4.2	The Train/ Validation/ Test set before (a) and after (b) augmentation.	72
4.3	The number of trainable parameters in each network and the time taken to train them.	73
4.4	Evaluating the effectiveness of the various CNNs at finding images with tracklets in the GB5-R5 test set at the 0.5 threshold	78
4.5	Evaluating the effectiveness of the various CNNs at finding images with tracklets in the GB-All (28-06-2013) test set at the 0.5 threshold	80
4.6	Evaluating the predictions made by the ensemble of classifiers at the 0.5 threshold	83
5.1	mAP achieved when training YOLOv4 and YOLOv4-tiny to detect asteroid tracklets with both SetA and SetB while applying both default and custom anchor boxes	107
6.1	Train/ Validation/ Test set for the CNN-LSTM	123

List of Figures

2.1	Inner solar system diagram with the positions of all numbered asteroids (main belt and Jupiter’s trojans) and all numbered comets on the 1st of January 2018. Asteroids are yellow dots and comets are symbolized by sunward-pointing wedges. <i>Credit: P. Chodas, NASA/JPL-Caltech</i>	8
2.2	Comparing the orbits of a typical NEA and PHA. <i>Credit: NASA/JPL-Caltech</i>	9
2.3	The torus (doughnut-shaped) Kuiper belt, which extends beyond the orbit of Neptune and is home to Pluto. <i>Credit: NASA/JPL-Caltech</i>	10
3.1	The MOA-II telescope at the Mt John Observatory	24
3.2	CCD arrangement for MOA-cam3	24
3.3	Single exposure of a field surveyed by MOA-II	25
3.4	Fields surveyed by the MOA-II telescope towards the Galactic Bulge	26
3.5	Dataflow for the MOA-II telescope	28
3.6	Image processing pipeline for the MOA-II telescope	28
3.7	Difference imaging analysis for generating an image (c) that represents the changes seen in a new observation image (a) since the reference image (b) was taken (Bond et al. (2001)).	29
3.8	Various artefacts present in difference images: a) and b) saturated stars and imperfect subtractions, c) satellite streak, d) possible imperfections on the camera	31
3.9	Consecutive observations of the same region on the night of 9 June 2007.	32
3.10	Six consecutive observations of the asteroid (78153) 2002 NX24 on 23-June-2006	33

3.11	Tracklet for the main-belt asteroid (78153) 2002 NX24 on 23-June-2006 as seen in the MOA-II data. a) is the tracklet based on the observations in Figure 3.10 and b) is the tracklet from all the observations on the same night. Most of the observations are 10-12 minutes apart but there is an interval of 26 minutes between observation 5 and 6.	34
3.12	Results from an MP Checker query to find all known minor planets visible from the Mt John Observatory within the search radius of one CCD chip at a specific date-time.	37
3.13	Stacking all of a night's observations on one chip in one field gives us a clear tracklet for asteroid (78153) 2002 NX24. In (a) the observations are stacked by brightest pixel; in (b) they are stacked by the median pixel; and in (c) we see the result of subtracting (b) from (a).	37
3.14	Pseudocode for creating brightest and median pixel stacks.	39
3.15	Pseudocode for creating the subtracted stack image.	39
3.16	A stack of all 51 observations from the night of 15-May-2008 reveals 5 asteroid tracklets clearly visible in a sub-region of the stack image. Clockwise from left to right, these are: (148657) 2001 SX124 ($V = 19.6$); (338789) 2005 SZ154 ($V = 20.4$); (103842) 2000 DQ33 ($V = 19.5$); (152083) 2004 RH30 ($V = 19.9$); 2016 AT221 ($V = 20.4$). The line/streak on the top left is from a satellite.	41
3.17	Tracklets seen for 3 different asteroids over 3 consecutive nights.	42
3.18	In addition to the tracklets seen in Figure 3.17, asteroid tracklets can also appear to move left to right or vice versa across the field of view.	43
3.19	The Cohen-Sutherland line clipping algorithm was used to reject tracklets outside the area of interest and determine intersection points of the ones partially inside.	45
3.20	Tracklets from observations with low (a) and high (b) cadence.	46
3.21	A selection of 128 x 128 tiles with tracklets of known asteroids.	48

3.22	What a difference the reference image makes. On the left are images from the GB5-R5 dataset and on the right are the corresponding sub-regions from the same night, field (GB5) and, chip (R5) from the GB-All dataset. Differences are due to a different reference image being used.	49
4.1	A tracklet image (centre) with its augments. Clockwise from top left: low contrast, horizontal flip, darkened, brightened, vertical flip, high contrast, rotated, blurred.	53
4.2	Images with no tracklets. Each row is from the same sub-region, but different nights.	55
4.3	Architecture of the original AlexNet (a) and the custom version (b)	58
4.4	Architecture of VGG16(a) and the custom architectures MOA-12(b) and MOA-14(c). VGG19 has one more 256 conv layer and two more 512 conv layers than VGG16. MOA-13 has one more 512 conv layer than MOA-12 and MOA-15 has one more 32 conv layer than MOA-14.	60
4.5	Composition of an Inception module	62
4.6	Number of filters in each layer of five Inception modules. The “reduce” columns refer to the 1x1 filters before the 3x3 and 5x5 convolutions.	62
4.7	Custom 15 layer Inception architecture using the Inception module seen in Figure 4.5	63
4.8	Residual block when the previous and current layer have the same (a) and different (b) filters	65
4.9	Custom 18 layer ResNet	66
4.10	Hybrid module combining salient features of a ResNet block and an Inception module	69
4.11	Custom architectures using the hybrid module.	70
4.12	Confusion matrix for the MOA-15 custom stacked network	75
4.13	Confusion matrix of the predictions made by the top six CNN architectures on the GB5-R5 test set at the 0.5 threshold	77
4.14	Confusion matrix of the predictions made by the top six CNN architectures on the GB-All (28-06-2013) test set at the 0.5 threshold	79
4.15	False negatives shared by the models MOA-12, MOA-14, MOA-15, Hybrid(a), and Hybrid(b) in the GB5-R5 test set	82

4.16	False negatives shared by the models MOA-12, MOA-14, MOA-15, Hybrid(a), and Hybrid(b) in the GB-All test set.	82
4.17	Confusion matrix of the predictions made by the ensemble of classifiers at the 0.5 threshold	84
4.18	PR curve for the ensemble of classifiers	85
4.19	ROC curve for the ensemble of classifiers	86
5.1	Tracklet image with both the ground truth and the predicted bounding box	91
5.2	YOLO's loss function (Redmon et al. (2016))	93
5.3	Images with faint tracklets in a noisy field	95
5.4	Images from Figure 5.3 with tracklets of known asteroids highlighted with bounding boxes	96
5.5	Breaking down the YOLO data format	98
5.6	Architecture of YOLOv4	100
5.7	A single block in a DenseNet where the input for a layer is the feature maps of all preceding layers.	100
5.8	Cross stage partial connection	101
5.9	Predictions made with the trained YOLOv4-tiny	108
5.10	Predictions made with the trained YOLOv4	109
6.1	A simple uni-directional RNN	113
6.2	Composition of a standard RNN module	114
6.3	Composition of a LSTM module	115
6.4	Structure of the dataset for the CNN-LSTM	117
6.5	All observations for a sub-region on one night (a to s) followed by the corresponding subtracted stack image. The asteroid is visible from frames k to r.	118
6.6	An overview of the CNN-LSTM architecture for asteroid detection	120
6.7	Simple CNN-LSTM model (left) and CNN-LSTM model with a pre-trained CNN (right)	121
6.8	Confusion matrix (left) and ROC plot (right) for a simple CNN-LSTM trained to identify asteroids	124
A.1	A fully connected artificial neural network with two hidden layers and two output nodes.	136

A.2	A fully connected artificial neural network with two hidden layers and two output nodes.	136
A.3	A simple CNN with one convolutional layer	140
B.1	Database for storing metadata about minor planets in MOA fields	142
C.1	PR plot for classification models with GB5-R5 test set	145
C.2	ROC plots for classification models with GB5-R5 test set	146
C.3	Confusion matrix for some classification models with GB5-R5 test set	147
C.4	Confusion matrix for some classification models with GB-All (28-06-2013) test set	148
C.5	Confusion matrix for AlexNet and Custom AlexNet with GB5-R5 test set	149
C.6	Confusion matrix for AlexNet and Custom AlexNet with GB-All (28-06-2013) test set	149
C.7	Evaluation metrics sorted by PR AUC, F2 Score, and Recall, highlighting the top five architectures	150
D.1	Detections from YOLOv4 with default anchor boxes	152
D.2	Detections from YOLOv4 with custom anchor boxes; only one to find both faint tracklets in the leftmost image on the last row. . .	153
D.3	Detections from YOLOv4-tiny with default anchor boxes	154
D.4	Detections from YOLOv4-tiny with custom anchor boxes	155

Chapter 1

Introduction

Our fascination with the night sky spans the length of human civilization. Witnessing a celestial body moving against the backdrop of this vast, star-studded darkness offers us a chance to interact with the universe at our timescale and perhaps enrich our understanding of it in the process. These moving objects are almost always bound to the Sun and thus part of our solar neighbourhood. Our solar system is teeming with millions of small worlds, the detritus from its formation 4.6 billion years ago. These include asteroids, comets, centaurs, trojans, and trans-Neptunian objects and are collectively called small solar system objects.

The mention of asteroids generally evokes images of giant space rocks impacting Earth and causing mass extinction events. Indeed, we see evidence of this with massive impact structures such as the Chicxulub crater, the Sudbury basin, as well as the multi-cratered surface of the moon. While such collisions were common in the early years of our solar system, the reality now is that asteroids lead largely uneventful lives, orbiting their star just as we do. A significant proportion of these asteroids could have once coalesced into a planet except for the disruptive influence of Jupiter in that region. In fact, it is close encounters with nearby planets that are thought to nudge main belt asteroids into orbits that bring them close to Earth. Observing and tracking asteroids, then, gives us a better understanding of their complex orbital dynamics, which in turn aids with determining the hazard posed by one in Earth's neighbourhood. Further, as asteroids have remained virtually unchanged since the early days of the solar system, studying them informs our understanding of its evolution.

While early asteroid discoveries were often providential, the recent years have seen several large astronomical surveys dedicated to discovering small solar system objects, especially near-Earth asteroids. These include LINEAR, Pan-STARRS, ATLAS, and the space telescope NEOWISE. Astronomy is an observational science and much of the historical work has been done by eye and based on the intuitions of the astronomer. But this type of human involvement is no longer feasible or even remotely practical with the sheer amount of data collected every night by these modern observatories. For example, the Hubble Space Telescope has generated about 150TB of data in 28 years of operation and the MOA telescope has 170TB of observational data in 10 years. However, this pales in comparison to the volume of data that will be generated by future surveys. The Roman Telescope and the Vera Rubin Observatory, two major next generation telescopes, are expected to generate 2.4TB and 30TB of data per day, respectively. Automated software tools are necessary to obtain the maximum scientific return from the data collected by past, present, and future surveys. The aim of this thesis is to offer a deep learning solution to aid in both the recovery and discovery of objects moving in our solar system.

The idea of artificial intelligence has also long intrigued humanity with much philosophical discourse on whether machines can ever match humans. The advent of modern computers provided a means to investigate this further, with Alan Turing’s “Can Machines Think” essay (Turing (1950)) often considered the foundational challenge. Neural networks were first proposed as a possible representation of how a human brain processes information. However, at the time, neural nets proved slow to train and it was not until the computational speed of computers improved that they were considered with any seriousness. In the 60 years since Turing’s axiom, we have seen enormous advances in computer hardware, with a trend towards ever greater storage and computational speeds. Computers can now store petaflops of data with supercomputers processing this data in milliseconds. The speed of the hardware is matched with the availability of large volumes of data that are required to train these algorithms. Neural networks have evolved over the years to the sophisticated architectures we see today, with the goal to achieve human-level performance in every field that they are applied to. The main challenge facing these algorithms, and indeed the broader artificial intelligence community, is solving the problems that are intuitive and trivial to most people but difficult to formally describe to a machine, such as

tracking movement.

Big data is the siren call for modern deep learning algorithms and with astronomical data set to get bigger still, the time is ripe for harnessing the power of deep learning for astronomy research. While work began in this direction in 1992 with star-galaxy classification (Odewahn et al. (1992)), it has been slow to pick up pace in the field. Indeed, Odewahn et al. (1992) noted that while their network was reasonably accurate, building a suitable test set for supervised learning and choosing a good parameter representation for an image posed a significant challenge. The latter problem can now be solved with a convolutional neural network (CNN), which takes an entire image as its input. In 2012, AlexNet (Krizhevsky et al. (2012)) won the prestigious ImageNet Large Scale Visual Recognition (ILSVR) challenge¹ with its state-of-the-art CNN, heralding the start of the current era in deep learning research. Since then, we have had neural nets that have surpassed human-level performance in tasks such as face recognition and natural language processing. Thus, there is currently a lot of interest in training and applying advanced neural network architectures to everything from biomedical research to detecting gravitational waves.

Along with star or galaxy classification (Naim (1995); Bazell and Peng (1998); Ball (2001); Dieleman et al. (2015); Kim and Brunner (2017); Cavanagh et al. (2021)), artificial neural networks have also been applied to other astronomy research such as light curve classification (Djorgovski et al. (2016); Charnock and Moss (2016); Mahabal et al. (2017); Armstrong et al. (2017); Naul et al. (2018); Pasquet-Itam and Pasquet (2018); George and Huerta (2018); Shallue and Vanderburg (2018)) and transient detection (Cabrera-Vives et al. (2017); Gieseke et al. (2017); Sedaghat and Mahabal (2018); Duev et al. (2019a)). The impetus in each case was the same - developing a robust pipeline for supporting digitised sky surveys. In each case the researchers noted that while the network performed well once trained and was of value to the data pipeline, the labelled dataset required for supervised learning was the weakest link. A trained model is only as good as the training data used and having an accurate and relevant dataset is of utmost importance for the robustness of the network. The availability of such a store of labelled data is by far the greatest hurdle facing the astronomical research community keen to apply deep learning to its analysis pipeline.

¹<https://image-net.org/challenges/LSVRC/>

Deep learning research for computer vision has benefited greatly from the availability of vast repositories of labelled data such as the ImageNet and MS COCO datasets. Despite being data-rich, no such repository exists for astronomical research. Part of the reason for this is that all observatories have highly personalised image processing pipelines and the data is rarely shared publicly. The configuration of the telescope and other instrumentation used by surveys can also have a significant impact on the images produced. Further, just as there are surveys that specialize in hunting asteroids, other surveys have a different focus. MOA-II, for example, exclusively scans its data for microlensing events. However, just like artificial satellites and cosmic rays, asteroids are part of the landscape of our night sky and will thus be present in the imaging data from most surveys. Consequently, archival survey data offers another avenue for both discovery of small solar system objects as well as an untapped resource to build a data repository that can be used for supervised learning. This informs another goal for this research, which is to create a repository of labelled data using techniques that can be applied to other surveys.

1.1 Research Goal

The gaps identified were the lack of labelled datasets and no viable solutions for finding asteroids in archival data. Thus, the goal of this research is two-fold:

- Construct labelled datasets consisting of astronomical images suitable for training deep neural networks by developing a methodology that can be replicated by other surveys;
- Find deep learning solutions for recovering and discovering asteroids in survey data.

At the conclusion, pre-trained weights will be available and can be applied towards expanding the dataset as well as towards further research.

1.2 Report Structure

We have discussed the motivations and goals for this research. Next (Chapter 2), we take a look at the families of asteroids, early and current detection techniques, followed by a review of the deep learning research undertaken in the field to date. Chapter 3 offers a brief overview of the MOA project and the MOA-II telescope before describing how archival difference images are converted into a dataset suitable for supervised learning. Chapter 4 presents CNN-based networks for classifying images as either containing or not containing asteroid tracklets. Chapter 5 extends CNNs past classification and into localising tracklets in difference images. Chapter 6 introduces a different paradigm that could leverage the temporal aspect of the data to discover moving objects. Additionally, four appendices are included with a primer about neural networks (Appendix A), schema of a database constructed for ease of access to the minor planet data (Appendix B), the ROC and PR plots from the various classification CNNs (Appendix C), and further examples of tracklet detections (Appendix D) by YOLOv4. We conclude (Chapter 7) with a review of the work undertaken and a look to the future.

Chapter 2

Review of Asteroid Detection Techniques

Mapping the motions of stars has captivated humankind for generations. Close observation reveals that some of those tiny points of light in the night sky move appreciably with respect to the more static seeming stars. Early Greek astronomers called these “wandering stars” or “*astēr planētēs*” and we now know them as planets. The most important consequence of this early study of the motions of planets was establishing the heliocentric model of the solar system. While many had long suspected that the Sun was the centre of our solar system with planets including Earth revolving around it, it was not officially accepted until 1543, when Copernicus published “*On the Revolutions of the Celestial Spheres*”. The first asteroid was discovered by Giuseppe Piazzi in 1801, who carefully and laboriously tracked its path across the celestial sphere for many weeks. Several other asteroids were discovered in the following years, but it was with the advent of large observatories that the rate of asteroid discovery picked up significantly. Since then, thousands of asteroids have been discovered in our solar system. In the 1980s, scientists theorised (Alvarez et al. (1980)) that an asteroid was the likely cause of the extinction of the dinosaurs, which renewed efforts to track and catalogue the orbits of all near-Earth asteroids so as to avoid the same fate. These Earth-crossing asteroids are, however, only one part of the families of ancient planetesimals that inhabit our solar neighbourhood, as we will see next.

2.1 Asteroids in Our Solar System

Asteroids are small rocky, sometimes metallic, worlds that orbit the Sun but are too small to be planets. They are also called minor planets and are the left-over debris from the formation of our solar system 4.6 billion years ago. Just as studying fossils tells us more about how life evolved on Earth, studying asteroids teaches us about the early history of our own solar system.

The current known minor planet count is 1,166,054 ¹. The largest known asteroid - Vesta - is about 530 kilometres in diameter and smallest are less than around 2 meters in diameter.

The vast majority of asteroids in our solar system orbit about 2 to 4 AU (300 million to 600 million kilometres) from the sun, between Mars and Jupiter. This region is known as the asteroid belt and the asteroids there in are referred to as main belt asteroids. It is estimated to contain between 1.1 to 1.9 million asteroids that are larger than one kilometre, including Vesta, and likely millions more that are smaller. The main belt also contains the dwarf planet Ceres, which is around 950 kilometres in diameter. The asteroids seen in this work are predominantly from the main belt.

Another family of asteroids, known as Trojans, share the orbit of a larger planet without colliding with it. The trojans gather at two special regions in a planet's orbit called the L4 and L5 Lagrangian points, where the gravitational pull of the sun and the planet are balanced so that the asteroid can maintain a stable orbit. Trojans have been discovered sharing the orbits of Jupiter, Neptune, Mars, and Earth, with Jupiter having the largest population of trojan asteroids. It is estimated that there are nearly as many trojan asteroids as main belt asteroids.

Figure 2.1 shows the view of the main belt asteroids and Jupiter's trojans² as well as the orbits of Mercury, Venus, Earth, Mars, and Jupiter.

¹<https://minorplanetcenter.net/about>

²https://ssd.jpl.nasa.gov/?ss_inner

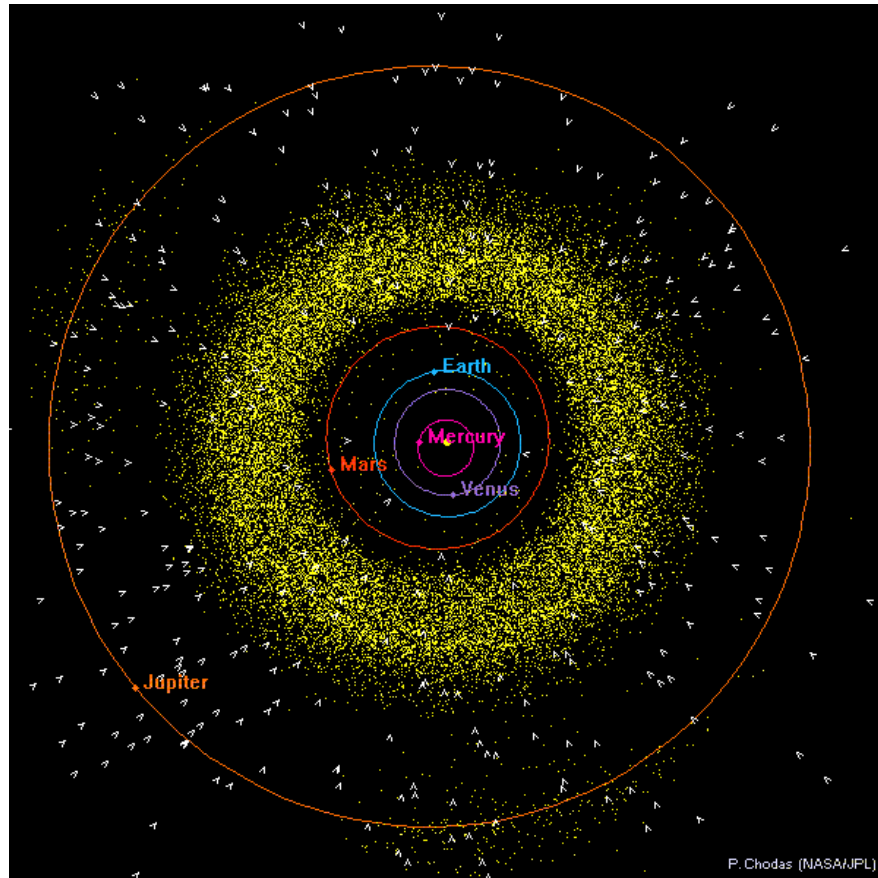


Figure 2.1: Inner solar system diagram with the positions of all numbered asteroids (main belt and Jupiter’s trojans) and all numbered comets on the 1st of January 2018. Asteroids are yellow dots and comets are symbolized by sunward-pointing wedges. *Credit: P. Chodas, NASA/JPL-Caltech*

Other asteroids orbit closer to the sun and thus to the Earth. If their orbit brings them closer than 1.3 AU (195 million kilometres) of the Earth, they are classified as Near Earth Asteroids (NEA). Some of these asteroids have orbits that intersect Earth’s and if these come within 0.05 AU (7.5 million kilometres) of Earth, they are classified as Potentially Hazardous Asteroids (PHA). For reference, the average distance of the Moon from Earth is about 0.0026 AU (384,402 kilometres). Figure 2.2 displays the orbits of a typical NEA and PHA³ along with the orbits of the inner solar system planets. Most NEAs emigrated from the main asteroid belt between Mars and Jupiter either through collisions or as the

³https://www.nasa.gov/mission_pages/WISE/multimedia/gallery/neowise/pia15628.html

result of passing too close to the planets.

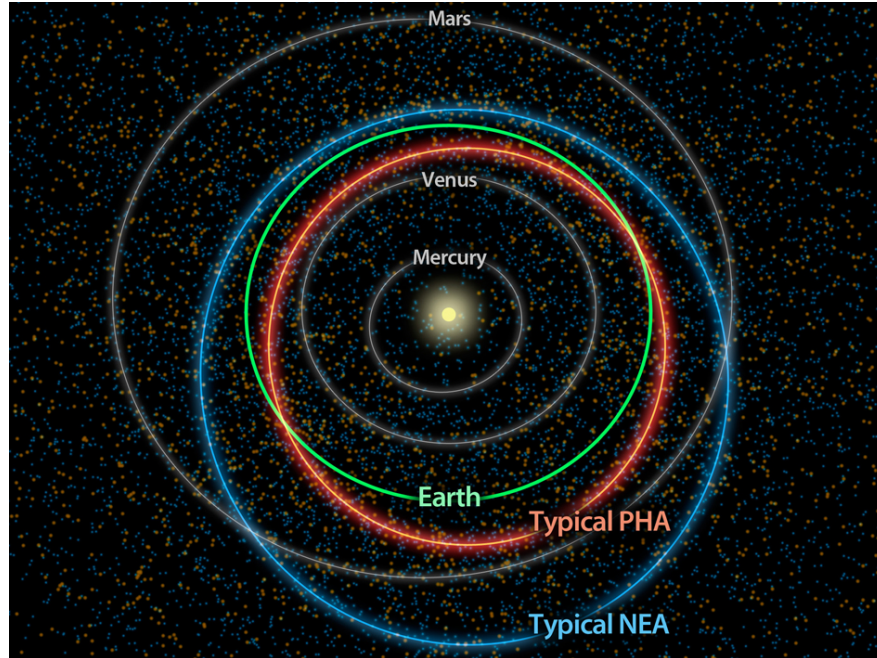


Figure 2.2: Comparing the orbits of a typical NEA and PHA. *Credit: NASA/JPL-Caltech*

There is a further vast store of minor planets beyond the orbit of Neptune in the Kuiper Belt called the Kuiper Belt Objects (KBOs). The Kuiper belt is a circumstellar disc that extends from the orbit of Neptune at 30AU to a distance of approximately 55AU. It is similar to, though much larger than, the main asteroid belt and its most famous denizen is the dwarf planet Pluto. We are still very much in the discovery phase with this far flung region of our solar system, which is thought to contain millions of small icy objects and the theorised origin of short-period comets (orbital periods of less than 200 years). Figure 2.3 show Pluto's orbit (inclined with respect to the plane of the solar system) within the Kuiper Belt⁴ along with an illustration of the hundreds of thousands of icy bodies and comets that are thought to reside within it.

⁴<https://solarsystem.nasa.gov/solar-system/kuiper-belt/overview/>

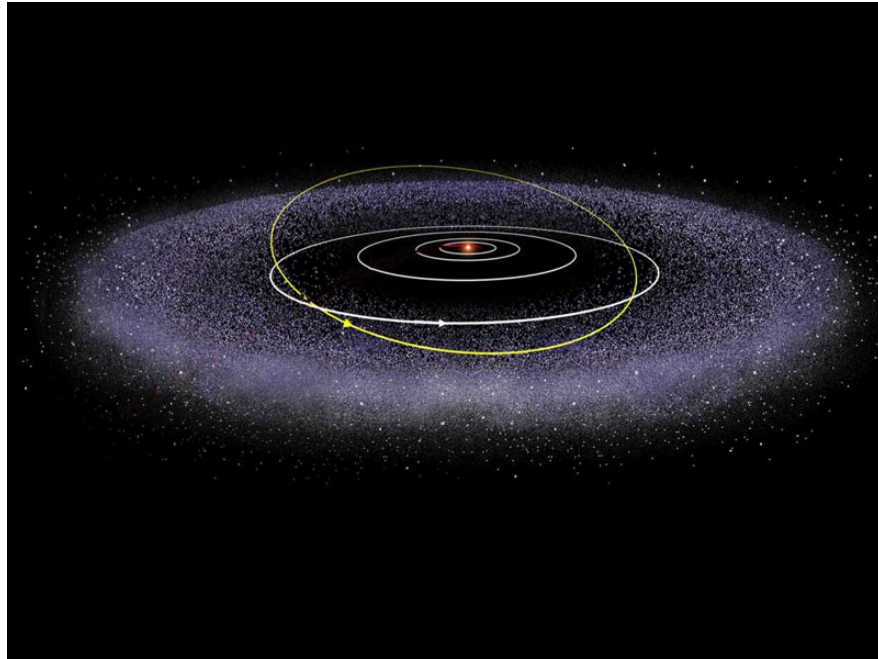


Figure 2.3: The torus (doughnut-shaped) Kuiper belt, which extends beyond the orbit of Neptune and is home to Pluto. *Credit: NASA/JPL-Caltech*

2.2 From Early Detectors to MOPS: An Overview

Early astronomers primarily used a combination of mirrors, lenses, and prisms to make their visual observations and largely serendipitous discoveries. In the late 19th century, the advent of photographic plates enabled these observations to be recorded and studied further. The asteroid 323 Brucia was discovered by Max Wolf in 1891, heralding a new era in finding solar system objects. The plates could be used in a device called a blink comparator, which facilitated astronomers finding differences between two photographs by rapidly switching from viewing one to the other. This “blinking” back and forth made it easier to spot objects that had changed position. The most notable discovery using photographic plates in a blink comparator was of Pluto in 1930 by Clyde Tombaugh.

The Palomar Planet-Crossing Asteroid Survey (PCAS) (Helin and Shoemaker (1979)) was the first astronomical survey dedicated to finding NEAs and applied a blink comparator to photographic plates to discover new minor planets. Its successor was the Palomar Asteroid and Comet Survey (PACS) (Shoemaker et al.

(1988)), which further utilized a stereoscopic microscope to scan photographic plates for minor planets. The NEA's image would appear to "rise" above the background stars when two different and slightly offset images were viewed with a special stereo viewing microscope. PACS's most ground-breaking discovery was of the comet Shoemaker-Levy 9, which was captured by and crashed into Jupiter in 1994.

The late 80s saw a digital revolution in observational astronomy with the introduction of the "charge-coupled device" or CCD. CCDs are small and light-sensitive microchips on to which the light that a telescope collects is focused. Each microchip is a grid of several tiny pixels (photodiodes), each of which record the amount of light falling on it as a build up of charge. They are a powerful tool for observational astronomy as they enable taking long exposures images, which lead to collecting more light from an observed region, which in turn allows faint, far away objects to be photographed. The Hubble Space Telescope has perhaps the most famous CCD camera and it has given us many awe-inspiring images of celestial objects in the last 30 years. Of the Earth-based observatories, between the mid 80s and 90s, Spacewatch, pioneered the first formal survey of near-Earth asteroids using CCDs and automated detection software (Rabinowitz (1991)), revolutionizing the way asteroids are discovered and tracked. The detection algorithm required a set of three consecutive exposures of 30 minutes each. The length of the exposure meant that asteroids would appear as streaks and these were identified in the first two exposures and their displacement from each other was calculated with respect to the third. The detections thus reported were then verified by the observer.

In the decades since, the advances in computer processors, data storage capabilities, and ever-improving detection algorithms has meant the current asteroid discovery rate is nearly 2000 new NEAs per year. One popular approach was using the shift-and-stack technique as applied by Cochran et al. (1995) to detect moving objects in the then-new field of discovering Kuiper Belt Objects (KBOs). They obtained 34 exposures over a 2-day period of one field near the ecliptic with the Hubble Space Telescope (HST). Each exposure was between 500-600 seconds long. To maximise the signal to noise ratio, they took the median sum of all 34 exposures and were left with the "static" objects like the distant stars and galaxies, but none of the moving objects or radiation noise. This median

sum image was then subtracted from each exposure, leaving behind the moving objects and noise. Next, the images were shifted and combined so that each KBO was highlighted and combined into the same pixel/blob. Median sum was performed again to remove all other noise/radiation, leaving behind the KBO. They discovered 29 KBOs, ranging between 5-10km in size, marking the first detection of short-period comets in their theorized reservoir of the Kuiper Belt.

Arguably, the true behemoth in the field is the Pan-STARRS⁵ Moving Object Processing System or MOPS (Denneau et al. (2013)). MOPS is a well-developed and influential software suite for automated asteroid discovery, trained with a simulated but realistic asteroid distribution for the Pan-STARRS telescopes. MOPS receives a set of transients not associated with any known source from the Pan-STARRS Image Processing Pipeline (IPP, Magnier (2006)) and applies a sophisticated tree-based spatial linking algorithm (Kubica et al. (2007)) to further parse and form associations between these point sources. The potential orbital track is first estimated for each set of new detections received, followed by scanning a kd-tree structure consisting of previous detections to find the closest association. The detection estimates are then updated accordingly, as is the kd-tree. If no associations are found, a new track is tentatively proposed. Associations are maintained both nightly as well as on subsequent nights. This allows the software to make accurate predictions about the orbital path of the objects and provides the data points required to confirm whether the objects are solar-bound. One of the strengths of the MOPS software suite is that it can be ported to other surveys. Its success over the years means that it will also be used by the next generation transient survey at the Rubin Observatory (Jones et al. (2016)), which is currently under construction.

The aforementioned is a snapshot of asteroid detection processes and offers a glimpse at the procedures involved. These days, all surveys that aim to discover asteroids employ automated techniques specific to their telescopes and have well structured pipelines in place to support the process. The aim of this work, however, is to investigate the application of deep learning for the asteroid discovery and focuses next on the current research undertaken on that front.

⁵<https://www.ifa.hawaii.edu/research/Pan-STARRS.shtml>

2.3 Deep Learning for Discovering Small Bodies in our Solar System

A brief primer about neural networks is included in Appendix A.

2.3.1 Meteor Detection

An early and successful application of deep learning for discovering moving astronomical objects was for the detection of meteors. Zoghbi et al. (2017) applied both LSTM (Long Short Term Memory) and CNN (Convolutional Neural Network) to look for debris from long-period comets. These objects are potentially hazardous, but the impact trajectory would likely only be discovered 6-12 months before the impact, when the object becomes visible. To provide earlier warning, the orbits of the comet's debris could be used to guide the search for the comets themselves while they are still far out. Determining the orbit requires the night sky to be monitored for an extended period of time (about 60 years) from locations around the globe and this data (low-light video) is provided by the Cameras for Allsky Meteor Surveillance (CAMS). The detections are typically made by astronomers who, on an average night, receive around 500 detections per camera consisting of images and light curves. That's a total of 8,000 observations with 16 cameras per site. Sorting through these every night is not scalable, hence the need to automate the process. Note that their goal was not to achieving human-level performance, they simply wanted an automation that was "good enough".

The advantage of both CNNs and LSTMs is that they can work with raw data, which was appealing to the researchers. To that end, they trained a single layer LSTM with 1000 units to distinguish between light curves of meteors and non-meteors. Of the 200,000 light curves input, only 3% were of meteors. They also set up a CNN with 5 convolutional layers and 2 fully connected layers with a softmax classifier to distinguish between images of meteors and non-meteors. The inputs were 480 x 640 greyscale images, 23% of which were meteors. The data was augmented by rotating and flipping the images. Both the CNN and LSTM had a F1 Score of about 89.5%, which is particularly remarkable given the imbalance between positives and negatives in the data. It's unclear if there is any

overlap in their misclassification; it's also possible that it did not matter because of the redundancy inherent in the data. Regardless, the researchers were happy with the result and deployed both networks, presumably to work in tandem. The paper does not contain any further details. Overall, this endeavour demonstrates the very real benefit of applying deep learning to astronomy research.

2.3.2 Euclid

Lieu et al. (2018) applied CNNs to the task of detecting small solar system objects (SSO) in data simulated for the ESA's Euclid⁶ space telescope, which is scheduled to launch in 2023. Euclid is an optical and near-infrared telescope that will scan about a third of the extragalactic sky. Extensive data simulations were undertaken for the express purpose of preparing the information pipeline and system support architectures for the mission.

The researchers use the Euclid Visible InStrument Python Package (VIS-PP)⁷ to create the simulated images to mimic what Euclid would see as closely as possible. The total field of view covers 4096 x 4136 pixels and is composed of four quadrants, each being 2048 x 2066 pixels. Postage stamp cut-outs centred on the objects of interest - SSOs, cosmic rays, stars, galaxies - were then taken from the simulated images, creating a dataset of 3756 (single channel) images. These are resized as per the requirements of the network architecture applied. Note that the asteroids are expected to have a "trailed" or streak-like profile for bright objects and a more diffused, oblong profile for faint objects. It was hypothesised that as the profiles of cosmic rays, stars, and galaxies were the most common cause of false positives, teaching the network to recognise them would prove useful. Next, the researchers added four different sorts of dithering manoeuvres to the images, three of which were combined to create 3756 additional 3-channel images. They chose to use three network implementations offered in Google's Tensorflow; Inception-v4 (Szegedy et al. (2016)), NASNetLarge (Zoph et al. (2017)), and MobileNets-v1 (Howard et al. (2017)).

All three models are initialised with pre-trained weights (ImageNet) before

⁶<https://sci.esa.int/web/euclid/>

⁷<http://www.mssl.ucl.ac.uk/~smn2/>

being fine-tuned with the Euclid simulations. The training/validation/test split ratios for the dataset were 70%, 20%, and 10% respectively. It is important to note that their sample dataset is not representative of the true proportions of each object seen during observation. Rather, they focused on having a balanced class distribution to focus the training on objects of interest. First, the networks were trained with the single-channel images to perform binary classification, with cosmic rays, stars, and galaxies grouped into one class called “non-SSOs”. They then did the same with the 3-channel images. The results with the 3-channel images were better and MobileNet performed best with an accuracy of 95.6% and recall of 93.5%. Next, they split both the single and multi-channel data into four classes - SSO, cosmic ray, star, and galaxy and retrained the networks. This time, MobileNet with the single channel images performed better with an accuracy and recall of 83%. While the researches also made adjustments to these results based on the expected abundance of the objects in the Euclid images, the true test will be how well the networks perform with real data. However, their investigation does give them a good set of pre-trained weights, which can be used when fine-tuning the network with real data.

2.3.3 DeepStreaks

Moving back to ground-based observatories, the Zwicky Transient Facility (ZTF)⁸ has been scanning the entire northern sky for astronomical transients every three nights since 2018. Among the objects they monitor are near Earth objects (NEO). Of particular interest were the objects under 140m as only around 30% of the estimated population had been found.

Duev et al. (2019b) proposed a CNN-based deep learning model, called DeepStreaks, to aid in the discovery of these objects. A Random-Forest based classifier called ZStreak (Ye et al. (2019)) was already part of the data pipeline at ZTF, however it still produced around $10^4 - 10^5$ candidate streak detections, which resulted in several human hours spent filtering out false positives. DeepStreaks was proposed to cut down the number of false positives but still deliver accurate detections. The candidate streaks are referred to as “fast moving objects” (FMOs) and could have either natural or man-made origins.

⁸<https://www.ztf.caltech.edu/>

The researchers experimented with two approaches to achieve their goal. The first was to split the task into three binary classification sub-tasks: identifying all streak-like object (“rb”), identifying all short streaks (“sl”), and identifying streaks made by real FMOs (“kd”). Each sub-task was solved by a group of classifiers and, for an object to be considered a candidate, at least one classifier from each group would need to give it a probability score that passed a pre-defined threshold. The second approach was to attempt accurate binary classification in a single step (“os”) where at least one classifier would need to output a score greater than a given threshold. In both cases, three CNN architectures were chosen to be used as an ensemble; a small 6-layer CNN, ResNet50 (He et al. (2016)), and DenseNet101 (Huang et al. (2016)). The rationale was that with their diversity, the classifiers would complement each other. The input image was 144 x 144 pixels, with the streaked object generally centred. The initial dataset consisted of 1000 real difference images from the ZTF pipeline, 8270 simulated images, and 6000 non-candidate streaks from satellites, airplanes, cosmic rays, bad subtractions, and other artefacts commonly seen in astronomical data. This data was first used to train the classifiers to detect all streak-like object (“rb”). This ensemble was evaluated on a month of ZTF data, which was then labelled and used to retrain the classifiers. This process was undertaken several times, though it is unclear for how long, with how much data, and what the results were at each step. Eventually, the same technique was employed to train the “sl”, “kd”, and “os” tasks as well, with “sl” benefiting from the results of “rb” and “kd” benefiting from the results of “rb” and “sl”.

The ROC curves indicate that, in each case, each individual classifier performed well, with a similar distribution, and had an area-under-the-curve of over 98%. However, tests with raw ZTF data indicated that the “rb”/“sl”/“kd” ensemble was superior while also significantly reducing the number of candidate detections flagged for follow up; DeepStreaks found 33,000, while ZStreaks found 1.7 million over the same period. With this, DeepStreaks achieved its stated goal and reduced the time taken by a human to review the candidates from several hours to several minutes, without sacrificing detection sensitivity. The crowning glory was the discovery of 15 NEOs at the time the paper was published.

2.3.4 ATLAS

Rabeendran and Denneau (2021) developed a deep learning solution for NEO candidate vetting for the robotic, ground-based astronomical survey and early warning system, Asteroid Terrestrial-impact Last Alert System (ATLAS)⁹. ATLAS aims to identify small solar system objects within a 0.01 AU distance from Earth before they impact. As with most astronomical surveys, candidate detections must be vetted by human observers before being submitted to the Minor Planet Center (MPC)¹⁰ for further evaluation. At the time the paper was published, ATLAS had discovered more than 500 NEOs since it became operational in 2016. While ATLAS can identify thousands of known asteroids nightly, it also has to contend with hundreds of false positives caused by various spurious artefacts that are an expected part of astronomical data. The stated goal with deep learning was to greatly reduce the number of false detections and thus speed up the process of follow up observations of new objects. ATLAS uses the MOPS pipeline, which groups source detections that are consistent with linear motion through the field of view. Four such detections form a “tracklet”, which ATLAS then uses to report asteroid observations.

ATLASs’ image processing pipeline generates reference-subtracted images, which highlight any transient phenomena observed, such as supernovae, variable stars, and moving objects. Small 100 x 100 pixel cut-out images of the source detections from the subtracted images form the basis of the input to the CNN. Note that the objects of interest are once again centred in the images. Tracklets are classified as *real* or *bogus* depending on whether or not they correspond to a small solar system object. The data is split into eight classes, three of which are *real* objects and the rest *bogus*. Objects classed as *real* are astronomical sources (rounded shape - asteroid, variable star, small comets), streaks (trailed appearance - NEO or satellite), and comets (larger profile with a possible trail). Objects classed as *bogus* are noise, cosmic rays, burns (readout artefact caused by bright sources), scars (result of imperfect subtraction), and spikes (diffraction spike caused by bright stars). A curated dataset of 3500 images was created with 500 images in each class except tracklet - specific. From each of the seven classes, 470 images were used in the training set and the rest in the validation set. Each

⁹<https://atlas.fallingstar.com/home.php>

¹⁰<https://www.minorplanetcenter.net/iau/mpc.html>

image was only associated with one class, though the authors note that some classes may be indistinguishable from others. This dataset was augmented by horizontal and vertical flipping, rotation, and cropping. A second dataset (referred to as the evaluation set) that consisted of 250,000 images of real known, real unknown, and bogus tracklets from two lunations of observations was constructed in tandem. The data from the first lunation was used for the training and the second for validation. The images in the evaluation set could contain objects associated with multiple classes.

The researchers designed a two-stage detector, the first of which was the Image Classification Network (ICN) to classify images and the second was the Tracklet Classification Network (TCN) to classify tracklets. The networks were modelled in the PyTorch¹¹ framework. The ICN was a pre-trained ResNet-18 (He et al. (2016)), fine-tuned to classify the curated dataset into one of eight classes. It must be noted that while eight classes are mentioned originally, the curated dataset only mentions seven. It is unclear if 500 more images were added to the curated set. The TCN was a multi-layer perceptron that is initially trained with the evaluation set but is designed to eventually take the output of the ICN as input. To keep the false negatives to a minimum, the loss function was weighted in favour of the real tracklet. The ICN performed well with the disparate classes (scar vs spike) but had a harder time distinguishing between astronomical sources and streaks, which was expected as the two can be very similar. The TCN found 99.6% of the real tracklets and 90.8% of the bogus ones. This is expected both because of the weighted loss and as well as the greater variation seen in the bogus tracklet images. However, catching nearly 91% of bogus tracklets does reduce the human workload when vetting tracklet detections, which was one of the stated goals of the research. This two-stage model is deployed on ATLAS and used for filtering out bogus tracklets before submission to the MPC and has proven successful at reducing the NEO candidate list by about 95% without sacrificing any real tracklets. The model continues to be evaluated and retrained to identify other artefacts that lead to false detections.

¹¹<https://pytorch.org/>

2.3.5 Tails

The team at ZTF undertook another investigation, this time applying deep learning to automate the search for comets that are detectable solely by their morphology (Duev et al. (2021)). As comets get close to the sun, they start to display a more pronounced coma and tails, which gives them a distinctive appearance in survey images. Further, we can get accurate positioning data about known comets, which would effectively identify the centroid the comet. This data is leveraged to train an object detection model, EfficientDet (Tan et al. (2020)), to localize comets in ZTF imaging data. The project was named Tails.

A labelled dataset was built by first identifying all known comet observations in the ZTF data from March 5, 2018 - March 4, 2020 based on their predicted position and brightness. The initial ephemerides were obtained from the MPC, followed by more accurate positional data from JPL Horizons¹². It was decided that, to provide more contextual information, three 256 x 256 pixel images would be used as the input simultaneously: the science/observation image, the reference image, and the difference image. Sixty thousand individual observations with comet magnitude between 10 - 23 were selected, 20,000 of which were chosen for manual annotation. This gave them 3,000 images of comets with identifiable morphology, to which they added 20,000 images with various other astronomical artefacts, but no comets. They then trained a ResNet (no further details given) for binary classification with this dataset and used its results to supplement their dataset with a further 2000 comet and 2000 no-comet images. Their final dataset consisted of 5000 comet images and 22,000 no-comet images. Each triplet input set was given a label $[p_c, x, y]$, where p_c (1 for comets) indicates the presence of a comet in the image and (x, y) the relative position of the comet's centroid as reported by JPL Horizon. Negative examples were labelled $[0, ?, ?]$. The data was augmented with random horizontal and vertical flipping.

Google's EfficientDet with its EfficientNet (Tan and Le (2019)) backbone feature extractor was the chosen object detection architecture. It was customised to accept and output centroid data similar to the label rather than the usual bounding box. The network weights were randomly initialized, and they used a training/validation/test split of 81%/9%/10%. The metric monitored was root

¹²<https://ssd.jpl.nasa.gov/horizons/app.html/>

mean squared error (RMSE). When evaluated on the test set with a class probability threshold of 0.5, the detector had a false positive and false negative rate of 1.7% and the RMSE was only 1-2 median pixel RSME from the JPL Horizon’s data. An expected aspect of object detection models is that the predicted location will never exactly match the ground truth location. Given this, it is unclear why object detection was chosen over the more usual classification model here. The work does not mention if the centroid predicted by Tails is a data point shared with the MPC. Nonetheless, Tails was deemed to fulfill its brief and was deployed soon after. As with most astronomical detections, the candidate detections provided by Tails were vetted by an astronomer. The model is also regularly retrained with new data to improve its performance. In experimenting with the inputs, the researchers established that the network performs just as well with the science and reference image alone and the network could potentially be updated to reflect this. Tails’ notable achievement was the discovery of the long-period comet C/2020 T2 in October 2020.

2.3.6 Hubble Asteroid Hunter

In early 2022, Kruk et al. (2022) shared their research about identifying asteroid trails in archival data from the ESA Hubble Space Telescope¹³ with deep learning. Space telescopes can capture fainter objects than can be seen by ground-based observatories and can thus help with constraining their population. In 30 years of operation, Hubble has taken many memorable images of galaxies, nebulae, and deep fields. Solar system objects such as asteroids also occasionally move across the field of view of the telescope, leaving a trailed footprint on the image. While these were highlighted by Evans et al. (1998), no further work was undertaken on that front until Kruk et al. (2022)’s research. Finding these trails is a time consuming process as they are infrequent occurrences. This added to the appeal of using deep learning as, once trained, the trails could be rapidly discovered.

The images used in the research were from the HST Advanced Camera for Surveys Wide Field Camera (ACS/WFC) and the Wide Field Camera 3 Ultraviolet and Visible Channel (WFC3/UVIS) and spanned from 30 April 2002 - 14 March 2021 for the former and 24 June 2009 - 14 March 2021 for the latter.

¹³<https://hubblesite.org/>

Composite HST images were used as they made the asteroid trails appear longer and thus easier to detect. The trails appear as curved “C” or “S” shaped streaks because of the motion of both the spacecraft and the asteroid. Cosmic rays, strong gravitational lenses, and satellites present in the images could be mistaken for asteroids. Their final dataset contained a total of 149,292 1050 x 1050 pixel images. The citizen science project, Hubble Asteroid Hunter¹⁴, was undertaken between 2019 - 2020 to aid with the search for asteroid trails in 144,559 of these images. After a short training tutorial, volunteers were asked to indicate whether or not an image had an asteroid trail and, if it did, to mark its beginning and end position. The volunteers also had access to static images with examples of asteroid trails as well as the objects that could be mistaken as such. They could also tag these objects that could be confused as asteroids using hashtags (satellite, cosmic_ray and gravitational_lens) on the project’s forum. Each image was vetted by 10 volunteers. The project attracted 11,482 volunteers and resulted in nearly 1.8 million classifications. A total of 1488 asteroid trails were discovered.

The classifications resulting from the citizen science project were used to train Google Cloud’s AutoML Vision¹⁵, a black box object detection algorithm based on CNNs that makes deep learning more accessible. Training the model with only asteroid trails resulted in a large number of false positives, so it was instead trained with four class labels: asteroid, satellite, cosmic ray, and gravitational lens arc with 1488, 1673, 1343, 698 images each, respectively. The model outputs both class probabilities and bounding box coordinates. At a confidence threshold and IOU of 50% it achieves a precision of 78.3% and a recall of 61.1%. The AutoML classification returned 2041 asteroid trails, 997 of which were new candidate detections. The authors visually scanned all the candidate asteroid trails identified by both volunteers and AutoML and were left with 1701 real asteroid trails after discounting false detections, duplicates, and known observation targets. Further research was undertaken to determine which of these trails belonged to known asteroids and this left the authors with 1031 unidentified/unknown trails, most of them from faint objects. The authors are continuing their work with investigating these unknown objects. The study highlights the usefulness of deep learning together with effective citizen science for mining archival data from any astronomical survey spanning decades.

¹⁴<https://www.zooniverse.org/projects/sandorkruk/hubble-asteroid-hunter>

¹⁵<https://cloud.google.com/vision/automl/object-detection/docs>

2.4 Summary

We have discussed asteroid populations in our solar system and their discovery through the ages. The large amounts of data produced by astronomical surveys necessitates the automation of as much of the processing pipeline as possible. Astronomical data contains many expected man-made and astronomical sources of noise that add to the complexity of discovering moving objects. It is also expected that candidate detections will include false positives and that human observers will be needed to screen these before earmarking a candidate as an SSO. The application of deep learning for finding SSOs is still in the early stages with one obstacle being the lack of large labelled datasets. Nonetheless, the appeal of deep learning is the speed of inference once a network is trained and the potential to refine and improve networks as more labelled data is obtained. Early research has demonstrated that deep learning models produce fewer false positives without adversely affecting the true positives. Some of the early research has even led to the discovery of new SSOs. With vast archival repositories and planned future surveys, the potential for future discovery and knowledge expansion abound.

Chapter 3

MOA-II Telescope: Building The Dataset

The MOA project is a Japan/NZ collaboration that has been operating the 1.8m MOA-II optical research telescope¹(Figure 3.1) at Mt. John since 2004. It has a wide-field mosaic CCD camera called the MOA-cam3 (Sako et al. (2008)), which consists of 10 CCD chips arranged as shown in Figure 3.2. Each CCD chip is 3cm x 6cm and has 2048 x 4096 pixels, recording over 800 megapixels of data with each exposure. The telescope has a total field of view of 1.6 x 1.3 degrees. MOA-II surveys the Galactic Bulge (GB) and the Large Magellanic Cloud (LMC), both of which are regions of the sky that are densely packed with stars. It operates at a high sampling rate, which means that the fields are surveyed several times each night, which in turn makes it particularly good for observing short duration events. Figure 3.3 is a single exposure of one of the fields surveyed and Figure 3.4 shows us all of the fields surveyed by MOA-II towards the Galactic Bulge.

¹<http://www.phys.canterbury.ac.nz/moa/index.html>



Figure 3.1: The MOA-II telescope at the Mt John Observatory



Figure 3.2: CCD arrangement for MOA-cam3

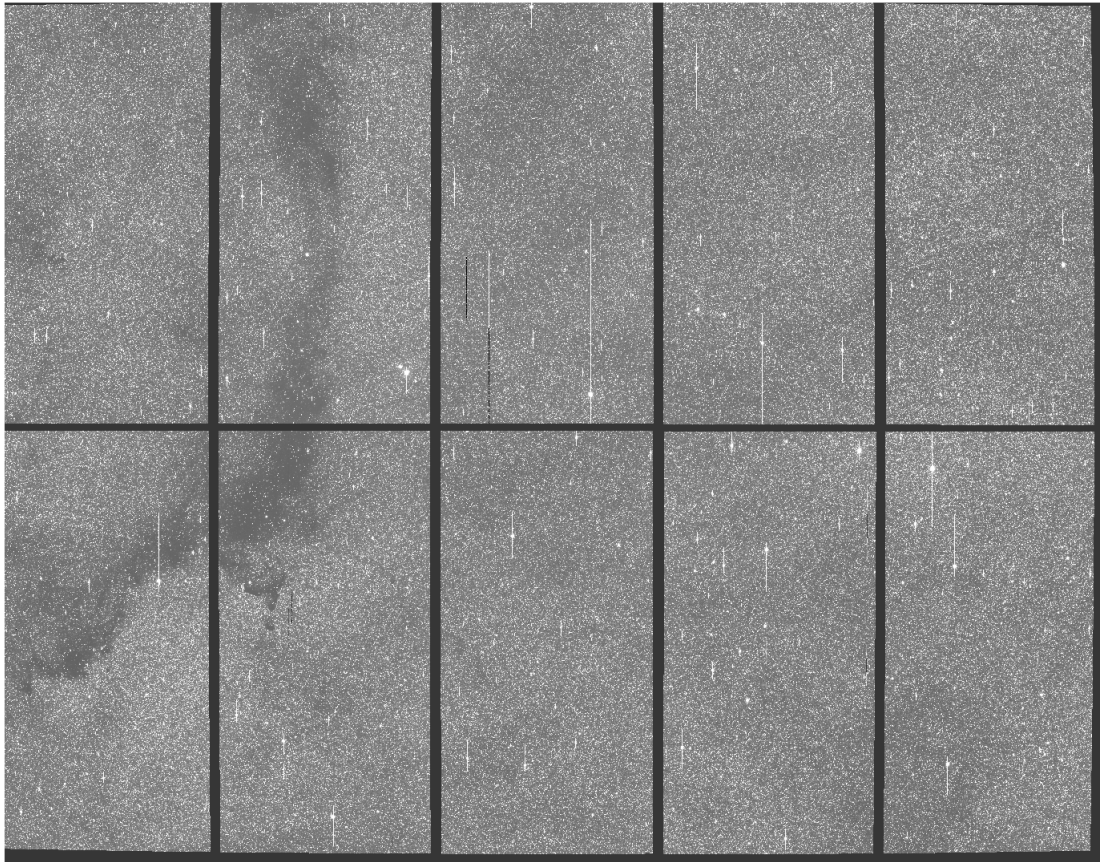


Figure 3.3: Single exposure of a field surveyed by MOA-II

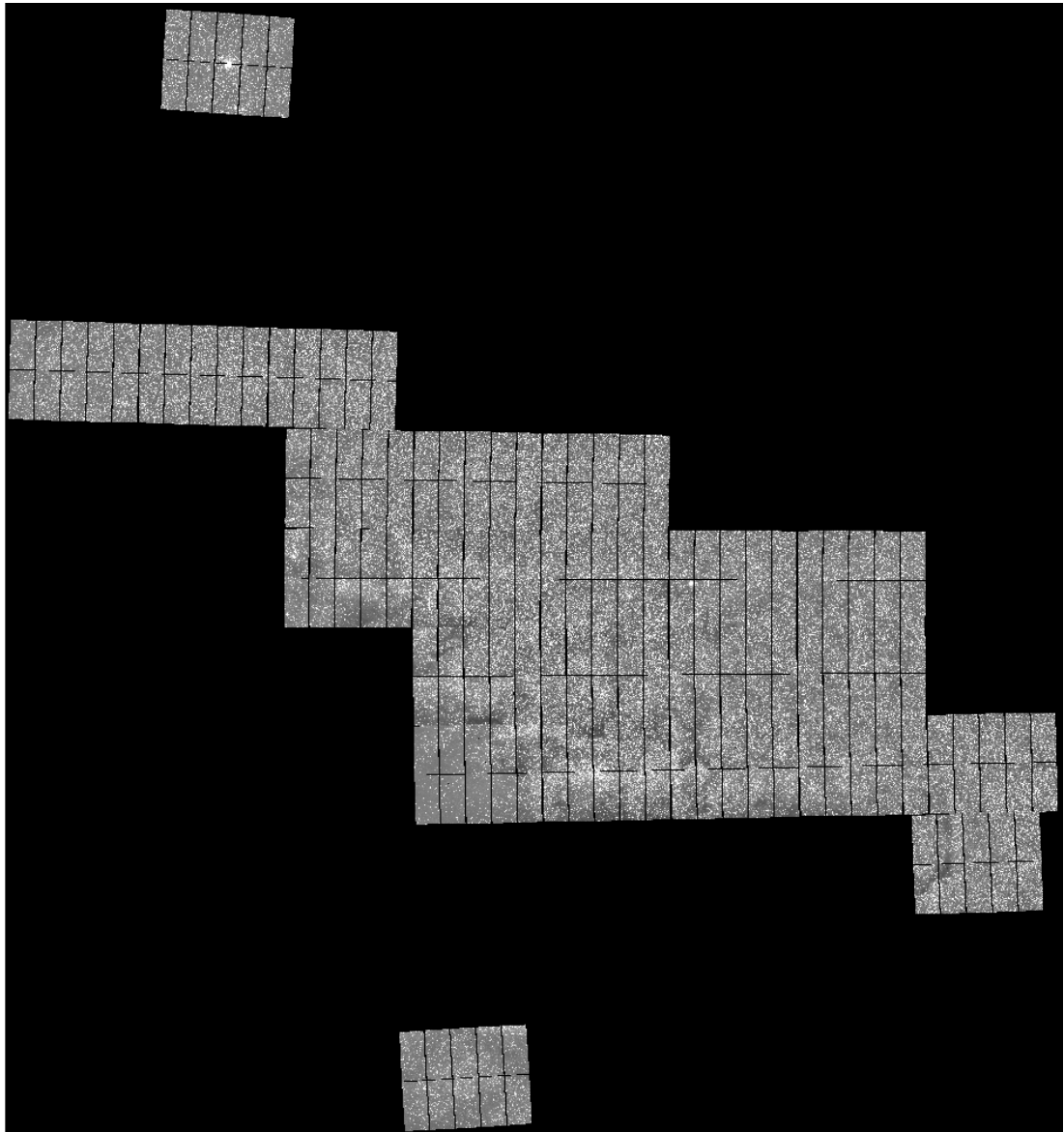


Figure 3.4: Fields surveyed by the MOA-II telescope towards the Galactic Bulge

MOA stands for Microlensing Observations in Astrophysics (Sumi et al. (2003)) and the project's primary research has been to observe gravitational microlensing events to find exoplanets, dark matter, and to study stellar atmospheres. Gravitational microlensing is a phenomenon where the light of a foreground star acts as a lens to a background star, causing the background star to momentarily shine a bit brighter (Einstein (1936); Liebes (1964)). The presence of a planet or planetary system around the lens star causes a brief spike in the brightness of the source (Mao et al. (1991); Gould and Loeb (1992); Bennett and Rhie (1996)). Since the phenomenon requires two stars to line up along our line of sight, the chances of observing such events is greatly improved by focusing on a dense star field, such as the Galactic Bulge (Paczynski (1986); Griest et al. (1991)). The MACHO survey (Alcock et al. (1993)) pioneered scanning for microlensing events and was joined by EROS (Aubourg et al. (1993)) and OGLE (Udalski et al. (1993)). MOA joined the collaboration in 1998, though the MOA-II telescope was not active until 2004. KMT (Kim et al. (2016)) is the latest survey to join in the search for microlensing events. MOA-II led the discovery of the first microlensing planet (Bond et al. (2004); Bond (2012)) and has played a key role in the discovery of all 129 confirmed exoplanets discovered via microlensing to date².

Congruently, Gould and Yee (2013) posit that microlensing surveys are particularly good for determining the rotation period and orbital trajectory of asteroids because they survey a given region of space several times each night. This means that asteroids could spend several nights travelling through the survey field of the telescope, giving us the opportunity to observe their path and study the light emitted from them.

²<https://exoplanetarchive.ipac.caltech.edu/docs/microlensing.html>

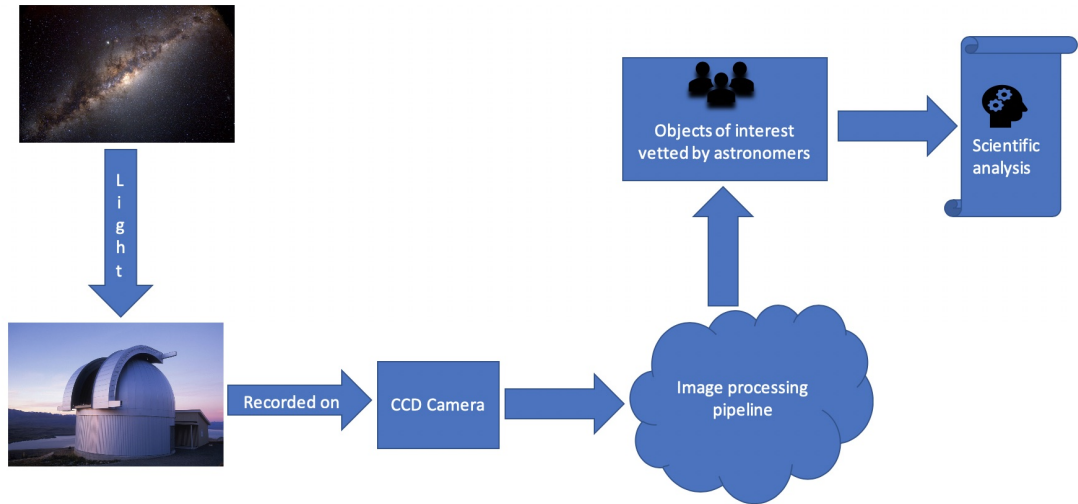


Figure 3.5: Dataflow for the MOA-II telescope

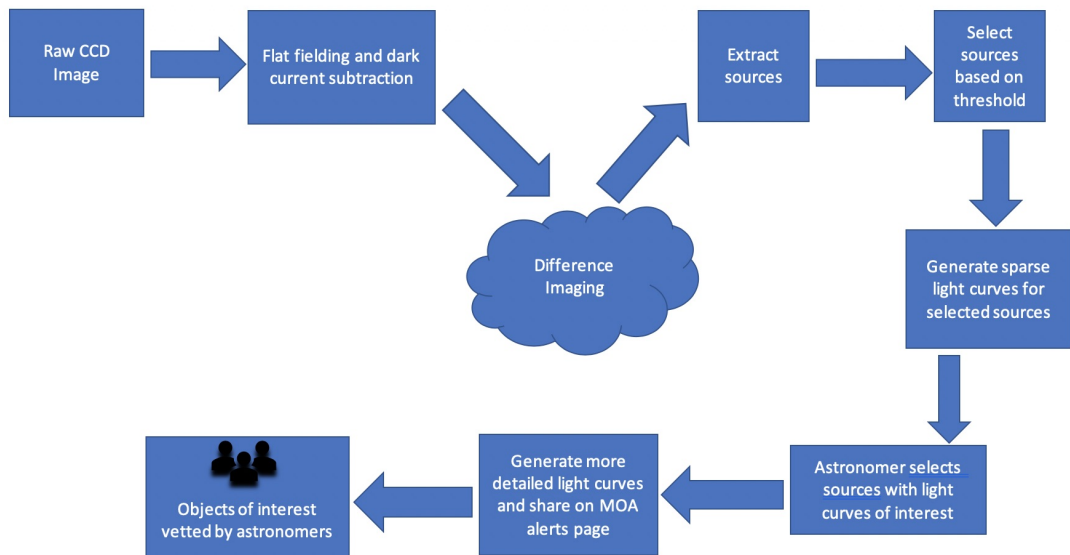


Figure 3.6: Image processing pipeline for the MOA-II telescope

3.1 Difference Imaging Analysis for MOA-II

Microensing is a transient astronomical phenomenon and detection relies on a robust image processing pipeline. Figure 3.5 illustrates the typical dataflow from data collection by the telescope to meaningful scientific analysis.

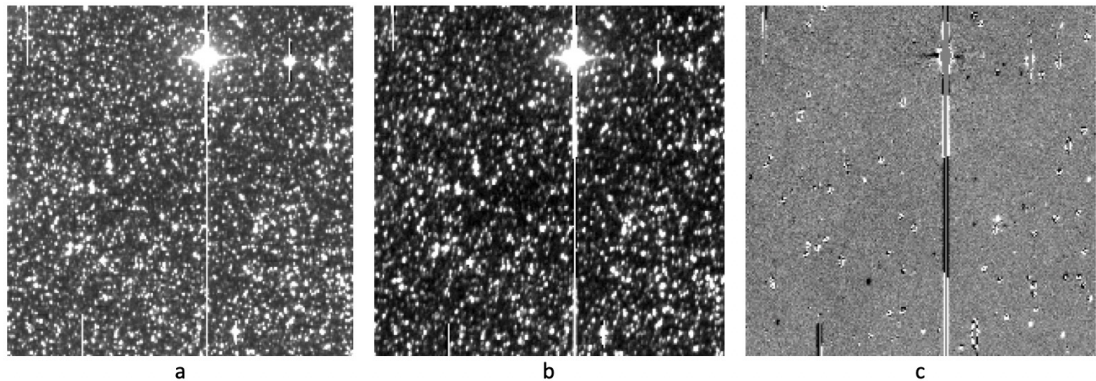


Figure 3.7: Difference imaging analysis for generating an image (c) that represents the changes seen in a new observation image (a) since the reference image (b) was taken (Bond et al. (2001)).

Figure 3.6 breaks down the image processing pipeline further. Briefly, the raw image from the CCD is first processed to adjust for pixel-to-pixel variations and image noise by applying flat-field correction and dark-frame subtraction. Then the difference image (Tomaney and Crotts (1996); Alard and Lupton (1998); Alard (2000); Bramich (2008)) is generated. The technique used by MOA (Bond et al. (2001)) involves calculating a convolutional kernel to extrapolate the differences in the seeing (essentially the atmospheric conditions, ambient light, etc.) between the reference image and the new observation image, convolving the reference image with this kernel, and then subtracting it from the new observation image. We are left with an image that records the changes in intensity of the light received (Figure 3.7). This process is essential as it would otherwise be impossible to find transient phenomena in the observations. Each difference image is then run through a source extraction algorithm, which divides the image into several patches based on the intensity of the pixels. These patches are further filtered based on a threshold of negative pixel values. This leaves us with a set of

possible transient candidates and a sparse light curve is generated for each. False positives are an expected part of this process. An astronomer reviews the light curves and selects the ones that should be further analysed. A more detailed light curve is generated for these, which is then shared on the MOA Alerts page where an astronomer vets each as a possible microlensing event.

The MOA-II difference images are a key component of this research and where the search for asteroids begins. Ideally, a difference image would only contain transient astronomical phenomena but, in reality, there are several spurious artefacts to contend with. Figure 3.8 highlights some of these, which are too-bright saturated stars, imperfect subtractions, satellite trails, and any residue on the chip or mirror assembly, most often seen at the edges. Additionally, noise could also be due to atmospheric dust, differential refraction, instrumentation error, or proximity to a bright astronomical object (like the moon). Consecutive observations on the same night can also be markedly different, as seen in Figure 3.9. Part of the challenge is finding objects of interest amongst these spurious artefacts.

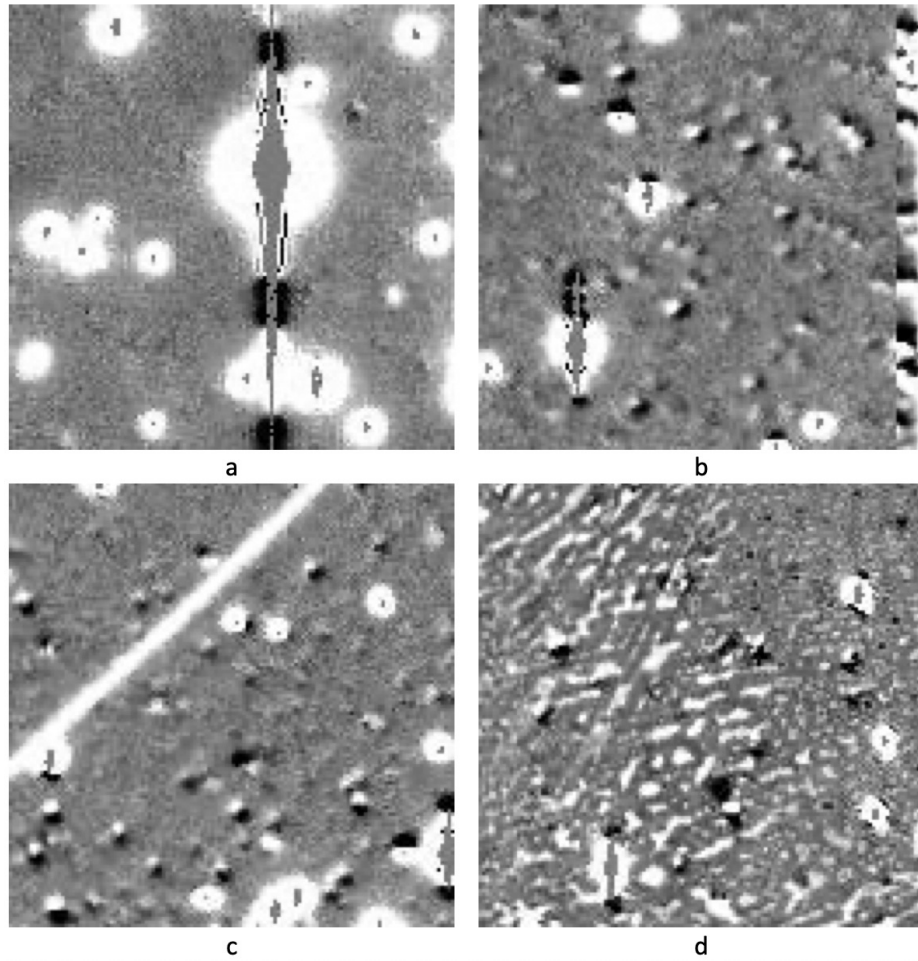


Figure 3.8: Various artefacts present in difference images: a) and b) saturated stars and imperfect subtractions, c) satellite streak, d) possible imperfections on the camera

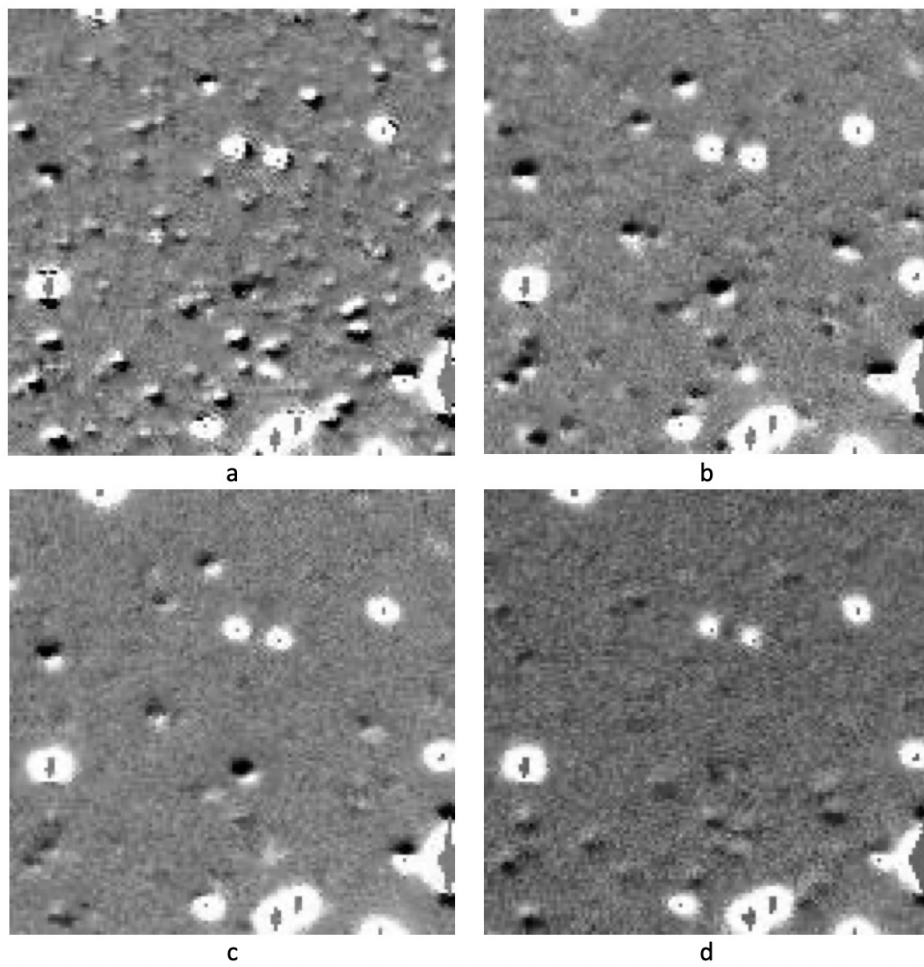


Figure 3.9: Consecutive observations of the same region on the night of 9 June 2007.

3.2 Asteroids in MOA-II

The Galactic Bulge is visible in the Southern Hemisphere between February and October. A combination of long nights and Bulge elevation in the mid-winter months make them the best time for observation. On clear nights with good seeing conditions, several of the MOA-II fields are surveyed frequently, at times as often as every 10 minutes. This provides ideal conditions for observing asteroids in our solar system.

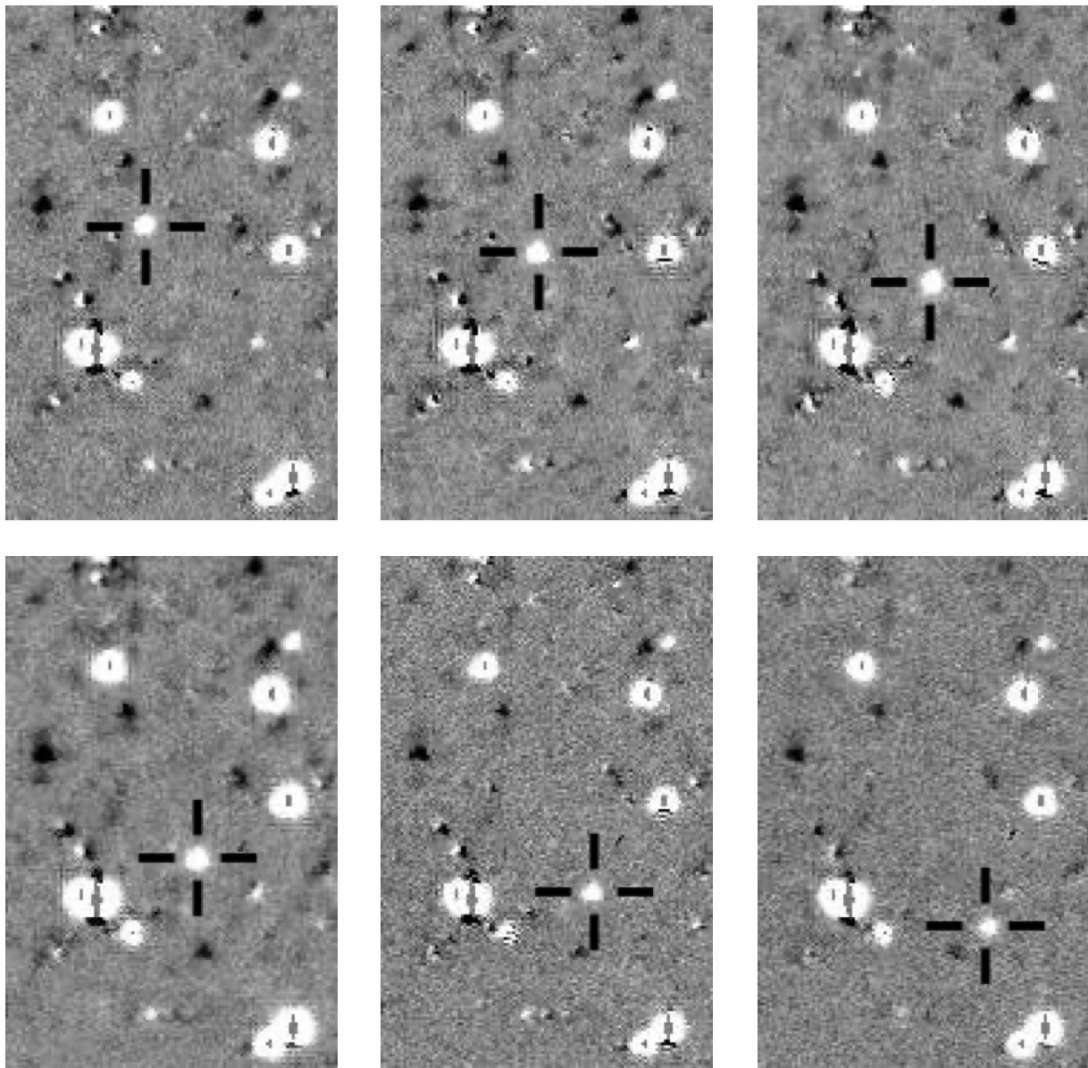


Figure 3.10: Six consecutive observations of the asteroid (78153) 2002 NX24 on 23-June-2006

Figure 3.10 demonstrates this with six observations of a bright main-belt asteroid (78153) 2002 NX24, with a limiting magnitude (V) of 17.5, taken at 10-12 minute intervals on the 23rd of June, 2006. If we stack these images together by brightest pixel, we can clearly see the tracklet for this asteroid (Figure 3.11(a)). A tracklet is part of the orbital arc of the asteroid as it travels through the field of view of the telescope. The profile or point spread function of the object changes slightly in each exposure, which is expected due to mild fluctuations in atmospheric conditions. Image (b) in Figure 3.11 is the stack of all the observations of NX24 on the same night. The asteroid is visible in twelve of the fifteen observations recorded and the gap between the 5th and 6th observation is due to an interval of 26 minutes between those two observations.

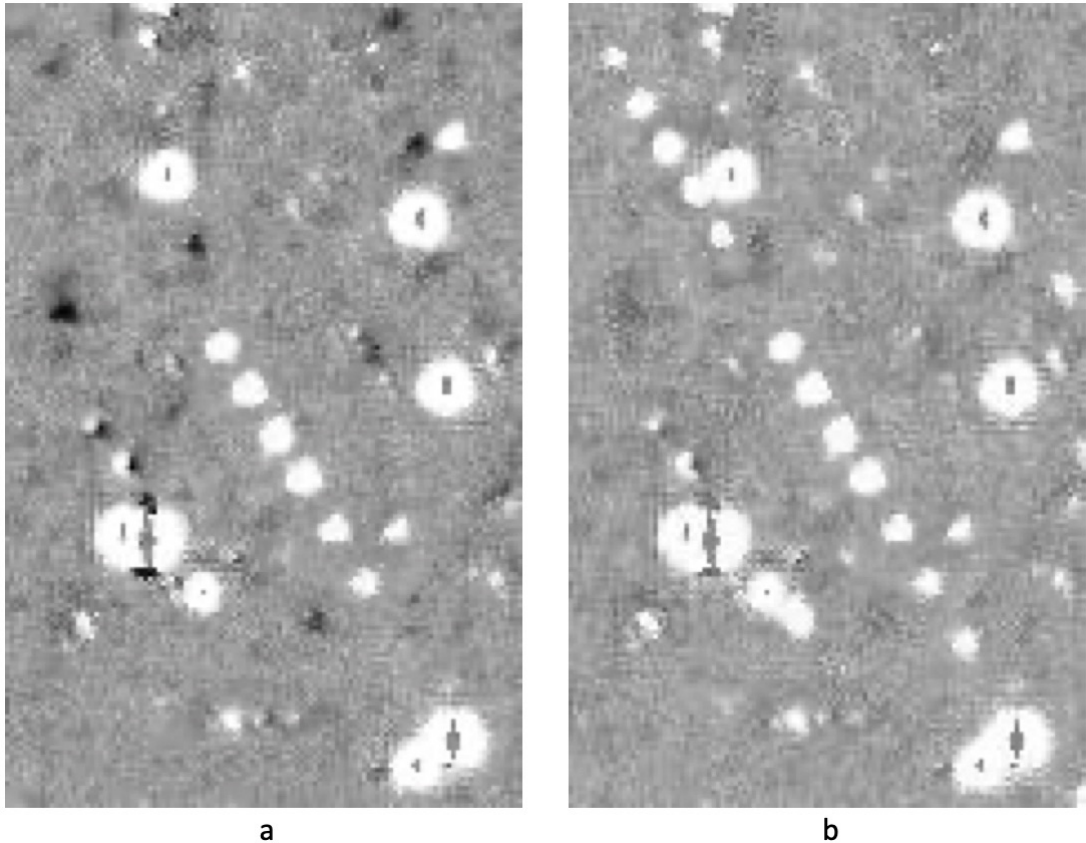


Figure 3.11: Tracklet for the main-belt asteroid (78153) 2002 NX24 on 23-June-2006 as seen in the MOA-II data. a) is the tracklet based on the observations in Figure 3.10 and b) is the tracklet from all the observations on the same night. Most of the observations are 10-12 minutes apart but there is an interval of 26 minutes between observation 5 and 6.

3.2.1 MOA-II Archival Data

All of the astronomical data is stored in the Flexible Image Transport System or FITS³ format that was designed for and has evolved with astronomical data. Data is stored as an N-dimensional array along with its associated metadata, which is in a human-readable header. The format is non-lossy, high resolution, and capable of storing data from multiple passbands in the same file.

Two main sets of archival data from MOA-II were used in this research.

1. GB5-R5: 49,901 difference images in FITS format consisting of all observations from 2006 - 2019 from one chip (chip 5) in the CCD array for the field GB5. The reference image used for generating these difference images was chosen from the night of 23-April-2012, which makes this set different from the default MOA-II difference images. This amounts to 1.1 TB of data. The entirety of this dataset was used.
2. GB-All: 905,302 difference images in FITS format consisting of all observations from all 23 of MOA-II's fields from 2013 - 2015. The reference images used for generating these are the default ones used by the MOA project and were chosen at the start of operations for each chip in each field on a night with excellent seeing conditions. This amounts to 11 TB of data. The significant size of this dataset means that only a small portion of it has been labelled thus far. To start with, one night (28-June-2013) of observations was extracted to serve as an additional test set for networks trained with the GB5-R5 data.

Each CCD chip is 2048 x 4096 pixels, with a resolution of 0.01 arc minutes (0.01') per pixel. Each exposure is 60 seconds long. Of the 23 fields, some get surveyed more often than others; GB5 is one such frequently observed field.

³https://fits.gsfc.nasa.gov/fits_documentation.html

3.2.2 Known Asteroids

The Minor Planet Center⁴ (MPC) is the official repository for all known minor planets, which are all the small bodies including asteroids, in our solar system. They collect observational data, calculate orbits, and provide various free web-based services for observers to access this information. One such service is the Minor Planet Checker (MPChecker)⁵. It can be used to generate a list of all the known minor planets that are visible at a given location, within a given radius, at a specific point in time. Figure 3.12 shows the results of one such query, which includes the asteroid NX24 (Figure 3.11).

To build a repository of all known asteroids in the MOA-II dataset, a screen scraper was written in Python to submit a query for each observation in the dataset and the results were saved. For the GB5-R5 data, the right ascension (RA) and declination (Dec) of the central pixel of the reference image was used, along with a search radius of 22.89 arc minutes. For the GB-All data, the RA and Dec of the central pixel of CCD assembly (midpoint of the field being surveyed) was used, along with a search radius of 100 arc minutes. Both of queries resulted in asteroids that were potentially outside the CCDs and these were filtered out in the next step. At the start of this research, the limiting magnitude for minor planets visible in the data was not known, so a conservative value of 24 was used. Note that the magnitude is a logarithmic scale and the higher the value, the fainter the object.

Once all of the celestial coordinates (i.e. the RA and Dec) for the known minor planets were obtained, astrometric calibration was performed to translate these into x,y positions on the CCDs for each field. MOA-II's custom built gnomonic projection procedure was utilized for this.

For ease of access, all of the minor planet data was stored in a custom-built database as described in Appendix B.

⁴<https://minorplanetcenter.net/>

⁵<https://minorplanetcenter.net/cgi-bin/checkmp.cgi>

CHAPTER 3. MOA-II TELESCOPE: BUILDING THE DATASET

The following objects, brighter than $V=20$, were found in the 22.0-arcminute region around R.A. = 17 52 28.11, Decl. = -29 56 38.96 (J2000.0) on 2006 06 23.48 UT:

Object designation	R.A.			Decl.			V	Offsets		Motion/hr		Orbit	Further observations? Comment (Elong/Decl/V at date 1)
	h	m	s	°	'	"		R.A.	Decl.	R.A.	Decl.		
(41668) 2000 TP25	17	51	53.7	-29	57	06	18.6	7.4W	0.5S	38-	3-	15o	None needed at this time.
(201846) 2003 YN87	17	51	56.3	-30	02	47	19.9	6.9W	6.1S	29-	4+	13o	None needed at this time.
2004 XL196	17	53	21.9	-29	55	48	19.7	11.6E	0.9N	35-	2-	11o	Desirable between 2021 Aug. 18-Sept. 17. At the f:
(97948) 2000 QF124	17	53	29.3	-30	06	15	19.5	13.3E	9.6S	39-	2+	13o	None needed at this time.
(78153) 2002 NX24	17	51	16.8	-30	03	18	17.5	15.5W	6.6S	33-	20+	16o	None needed at this time.
(52627) 1997 WU2	17	51	32.3	-29	43	54	18.1	12.1W	12.7N	32-	12-	18o	None needed at this time.
(18566) 1997 RS3	17	53	32.7	-30	08	34	18.5	14.0E	11.9S	30-	0-	20o	None needed at this time.

Figure 3.12: Results from an MP Checker query to find all known minor planets visible from the Mt John Observatory within the search radius of one CCD chip at a specific date-time.

3.2.3 Building The Dataset

Image Stacks

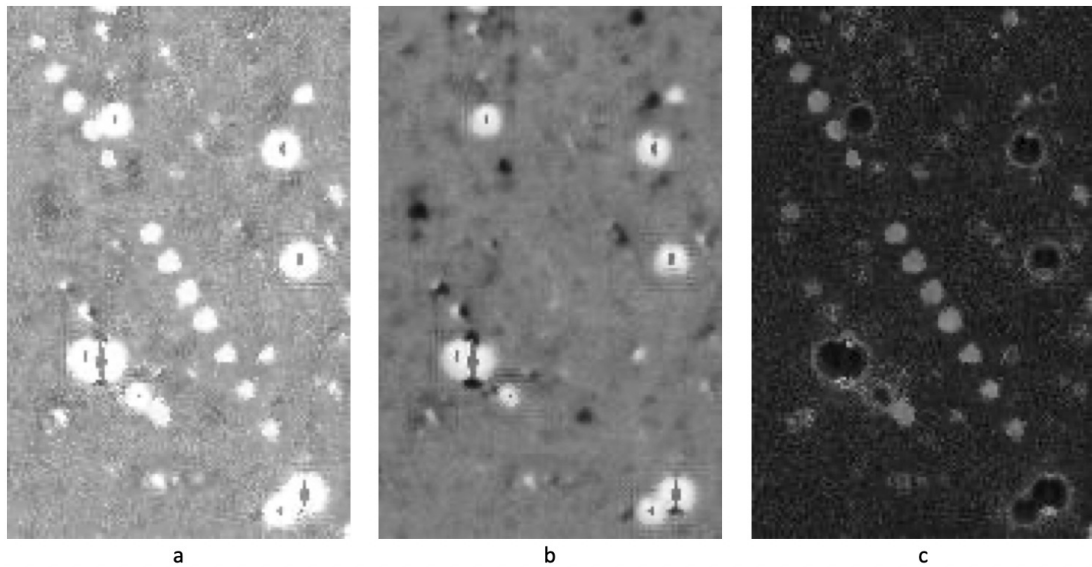


Figure 3.13: Stacking all of a night's observations on one chip in one field gives us a clear tracklet for asteroid (78153) 2002 NX24. In (a) the observations are stacked by brightest pixel; in (b) they are stacked by the median pixel; and in (c) we see the result of subtracting (b) from (a).

As demonstrated in Figure 3.11, stacking images from the same night by the brightest pixel will reveal the tracklet of an asteroid moving through the field of view. Stacking the observations by median pixel results in an image that is devoid of any random noise or moving objects. Thus, the stack image can be further simplified by subtracting the brightest pixel stack from the median pixel stack. This gives us an image without the bright background and saturated stars,

which highlights the tracklet better (Figure 3.13).

The following process was followed in code to produce the subtracted image stacks:

1. Convert all of the FITS data to JPGs. This leads to a significant reduction in storage requirements (1.1 TB to 116 GB for the GB5-R5 data) without compromising the image composition. The JPG images were encoded to look resemble the SAOImage DS9's⁶ “zscale stretch” format, which applies a pixel stretching algorithm and is commonly used to analyze difference images.
2. Group all observations by field/chip/night. This list is saved and referred to several times.
3. Stack the nightly observations for each chip in each field by brightest pixel (Figure 3.14).
4. Stack the nightly observations for each chip in each field by median pixel (Figure 3.14).
5. Subtract the median stack from the corresponding brightest stack; the resulting image (subtracted image) contains noise plus any object moving in the field of view (Figure 3.15).

Only nights with 3 or more observations were considered when building the image stacks. The GB5-R5 dataset resulted in 2252 subtracted images with which the search for asteroid tracklets could begin. Additionally, on the night of 28-June-2013, 12 of the fields had more than 10 observations, which resulted in 120 subtracted images from the GB-All dataset.

⁶<https://ds9.si.edu>

```
1: load dictionary of observations grouped by field/chip/day
2: for each field/chip/day, stack_key do
3:   load a list of all the observation images, flist
4:   create array of zeros that has the same dimensions as an
   image, stack_array
5:   for each image in flist do
6:     open the image
7:     convert the image into an array, img_array
8:     if brightest stack then
9:       compare stack_array and img_array element-wise and
       save the maximum in stack_array
10:    if median stack then
11:      calculate img_array divided by the length flist
       (element-wise), median
12:      add median to stack_array and save in stack_array
13:    convert stack_array into an image, stack_img
14:    save stack_img as stack_key.jpg
```

Figure 3.14: Pseudocode for creating brightest and median pixel stacks.

```
1: load dictionary of observations grouped by field/chip/day
2: for each field/chip/day, stack_key do
3:   open the brightest pixel stack image, bright
4:   open the median pixel stack image, median
5:   element-wise, calculate the absolute difference between
   bright and median and save in subtract
6:   save subtract as stack_key.jpg
```

Figure 3.15: Pseudocode for creating the subtracted stack image.

Localising Asteroid Tracklets

The minor planet ephemeris data from the MPC, together with the astrometric calibrations specific to MOA-II, were used to extrapolate the minimum and maximum (x, y) positions for asteroid tracklets in stacked images. This was used to crop sub-regions expected to contain tracklets from each stack. Asteroids that fell beyond the boundaries of a CCD chip were ignored.

Further, by visual inspection of the tracklets, it was established that only asteroids with a limiting magnitude of 20.5 or brighter are visible in the MOA-II exposures. The objects with magnitude between 19.5 and 20.5 are often at the very edge of visibility, requiring excellent seeing condition and a high signal to noise ratio to be visible in the observations. Figure 3.16 was one such night where tracklets of asteroids with a limiting magnitude between 19.5 and 20.4 were clearly visible.

Altogether, 7840 tracklets, from 3580 distinct asteroids, with a limiting magnitude of 20.5 or brighter were expected to be found in the GB5-R5 data. Careful examination of this data was undertaken to verify which of these tracklets were actually visible. This resulted in 2073 tracklets from 1178 distinct asteroids (some asteroids are visible over multiple nights) over 1078 distinct nights. It was from these 1078 nights that a single night with good observational data (28-06-2013) was selected to extract observations from the GB-All dataset.

Figures 3.17 and 3.18 display some of these tracklets. The size of the tracklet is indicative of how many exposures were taken on the night as well as at what point the asteroid entered the field of view. Gaps in a tracklet are either caused by a longer interval between observations or poor seeing conditions. The tracklet might also not be visible if the object moves to a very noisy part of the CCD like at the edges as seen in the last frame (c) of Figure 3.17. In certain cases, the PSF of the asteroid might be so pronounced that the tracklet appears like a solid line with wavy edges (Figure 3.18b). Tracklets can also be very small, appearing in only about a 50px by 50px area or smaller (Figure 3.18c).

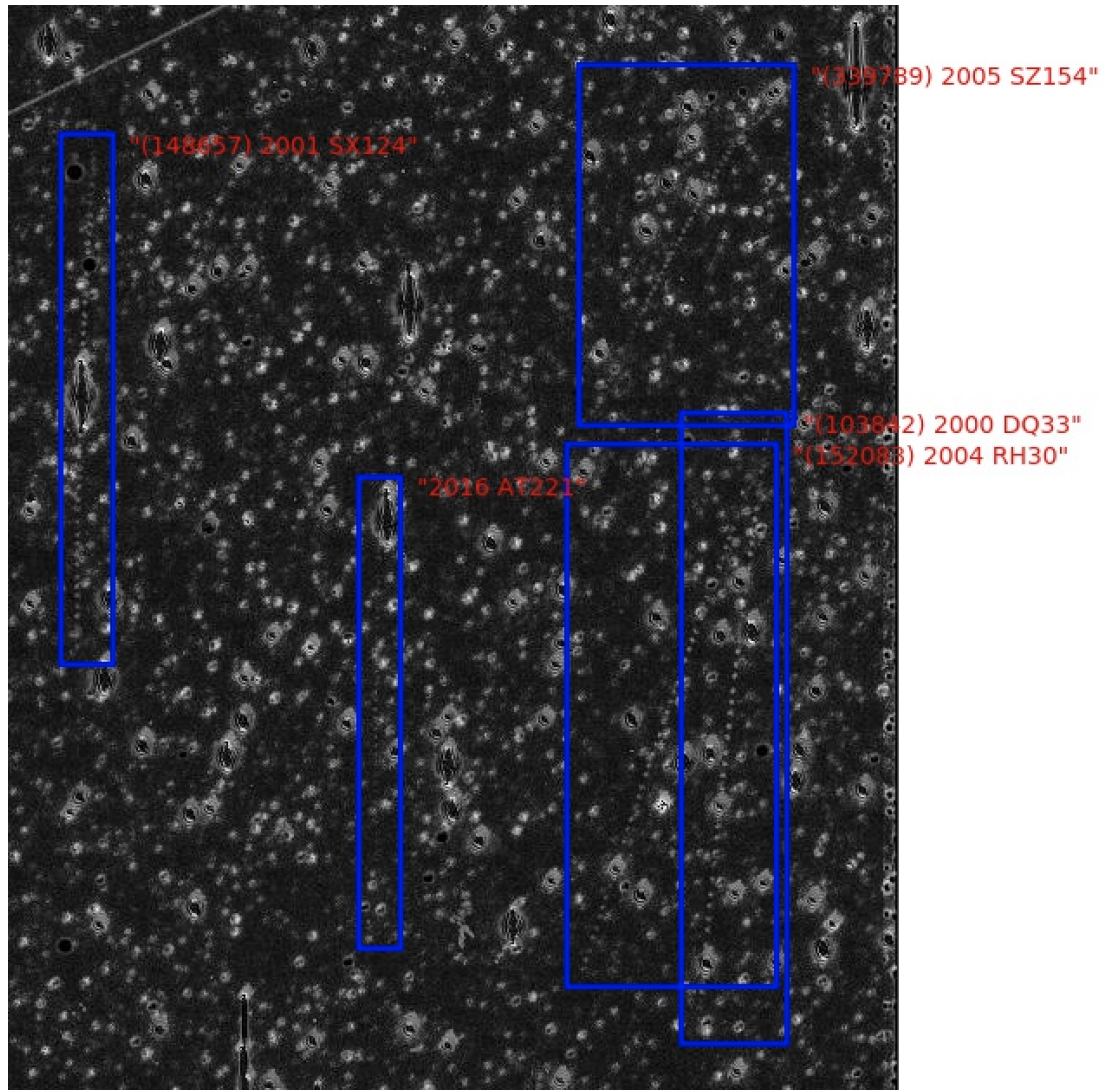


Figure 3.16: A stack of all 51 observations from the night of 15-May-2008 reveals 5 asteroid tracklets clearly visible in a sub-region of the stack image. Clockwise from left to right, these are: (148657) 2001 SX124 ($V = 19.6$); (338789) 2005 SZ154 ($V = 20.4$); (103842) 2000 DQ33 ($V = 19.5$); (152083) 2004 RH30 ($V = 19.9$); 2016 AT221 ($V = 20.4$). The line/streak on the top left is from a satellite.

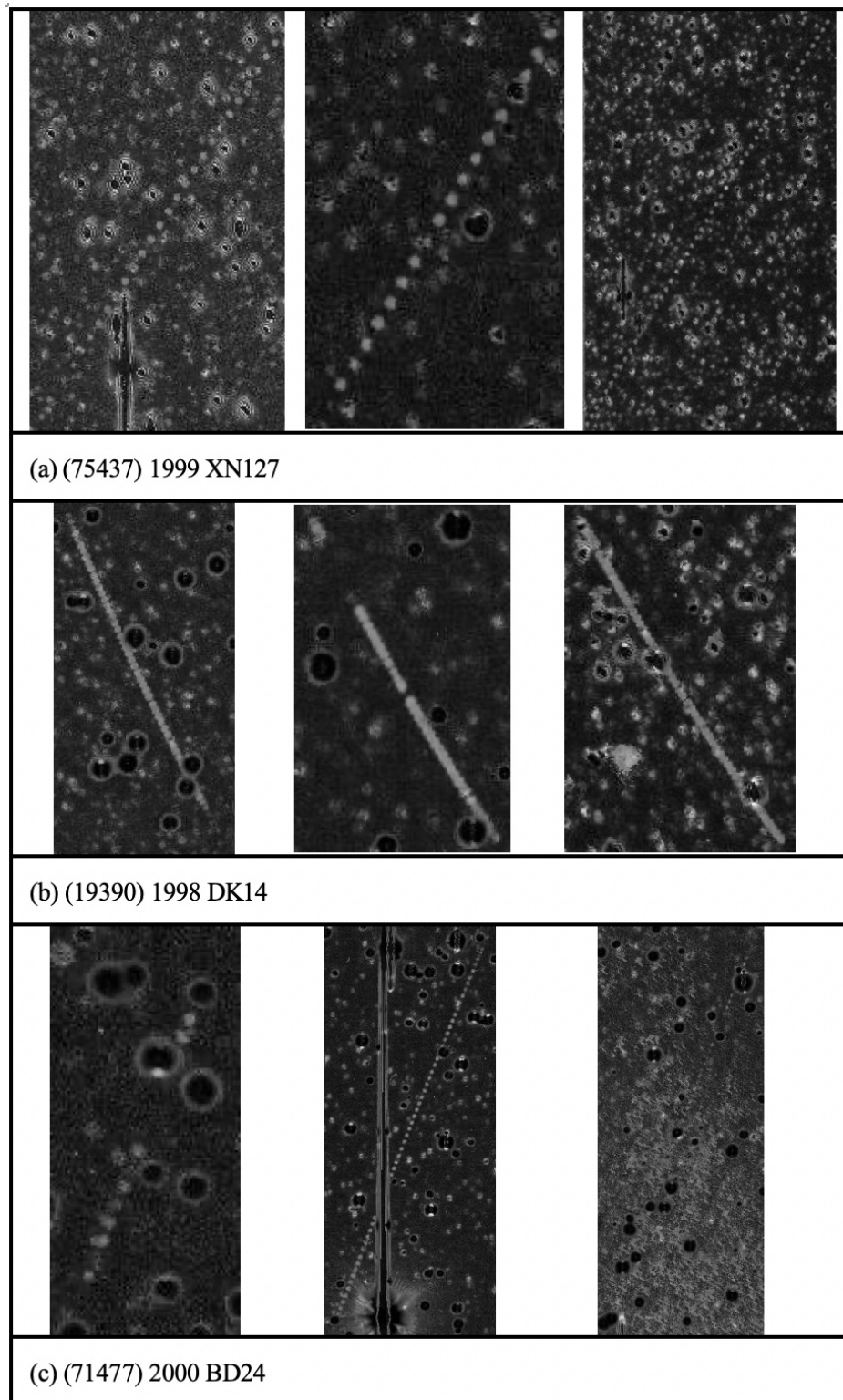


Figure 3.17: Tracklets seen for 3 different asteroids over 3 consecutive nights.

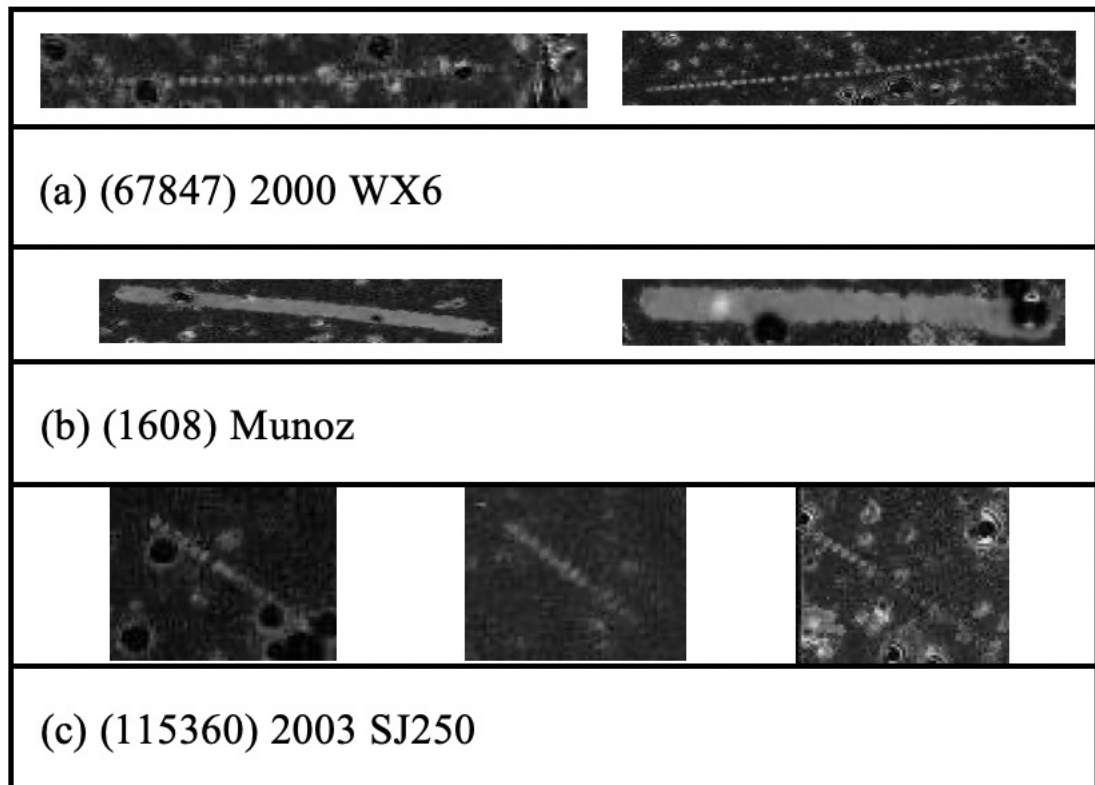


Figure 3.18: In addition to the tracklets seen in Figure 3.17, asteroid tracklets can also appear to move left to right or vice versa across the field of view.

Uniform Tiles and Line Clipping

Having obtained and verified visible tracklets in the data, the next step was to assemble the images in a format that could be used for training neural networks. We have seen that the tracklets come in a variety of sizes, but we need images of a uniform size going forward. The original size of the observations - 2048 x 4096 - is too big to use as is because of the computational cost as well as the potential for the tracklets to be lost amongst the other artefacts in the image. Therefore, a decision was made to split each image into 128 x 128 tiles, giving us 512 tiles per image. These dimensions were chosen to limit the noise and other artefacts present with the smaller tracklets. For GB5-R5, this gave us a total of 551,936 tiles from the 1078 nightly stacks that contained visible asteroid tracklets. One day of data from GB-All gave us 61,440 tiles from 120 nightly stacks.

The next task was to find the tiles with known asteroid tracklets. If we consider each tracklet to be a line, we can use the first and last observation of each asteroid each night to determine the line's direction. If we further consider each image tile to be a box that the line intersects or is fully contained in, we can apply a line clipping algorithm to determine which tiles have asteroid tracklets.

The Cohen-Sutherland line clipping algorithm (W. M. Newman and R. F. Sproull (1973)), which is a well-known computer graphics algorithm, was used for this purpose because it is efficient at accepting or rejecting lines that are completely inside or outside the area of interest. The algorithm involves dividing a rectangular space (in this case, a tile image) into 9 regions - 8 "outside" and 1 "inside" - as seen in Figure 3.19 and determining which lines (tracklets) are fully or partially inside of the space. Each of the 9 regions have associated outcodes (4-bit numbers) that are calculated by performing a bitwise OR operation after comparing the start and end points of the line/tracklet with the coordinates of the tile. There are three possible solutions for any line:

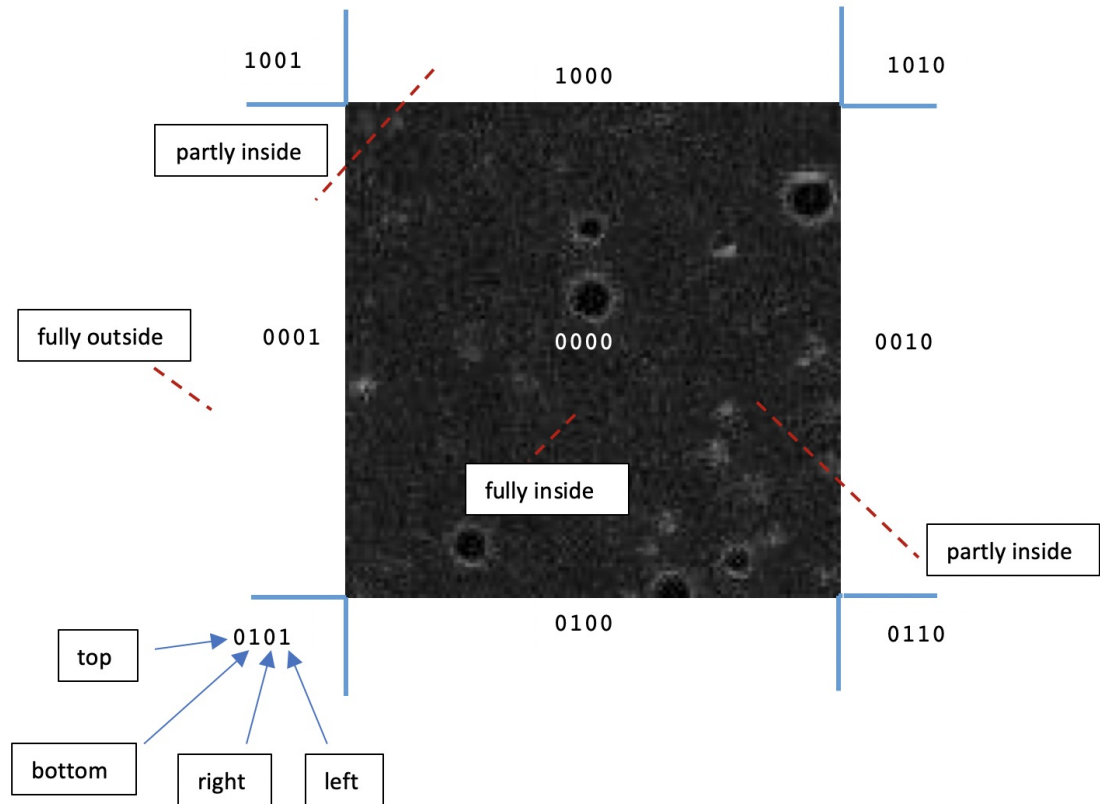


Figure 3.19: The Cohen-Sutherland line clipping algorithm was used to reject tracklets outside the area of interest and determine intersection points of the ones partially inside.

- If both endpoints of the line are inside the region, the bitwise OR computation returns 0 (trivial accept).
- If both endpoints of the line are outside the region, they will share at least one outside region and the bitwise OR computation returns a non 0 value (trivial reject).
- If both endpoints are in different regions, at least one endpoint will be outside the image tile. In this case, the intersection point of the tracklet's outside point and image tile boundary becomes the new endpoint for the tracklet and the algorithm repeats until the bitwise OR returns a trivial accept or reject.

The application of this algorithm resulted in 8341 tiles from GB-R5 and 4037 tiles from GB-All (28-06-2013) that potentially had visible tracklets. Once again these were meticulously scanned to ensure that a tracklet could clearly be seen. Any tile with less than 3 points of a tracklet were rejected, along with tiles that did not contain a visible portion of a tracklet. Additionally, it was noted that tracklets from observations that were on average 10 to 20 minutes apart produced clear and distinctive tracklets and were thus best suited to this research. Figure 3.20 shows a tracklet where the cadence is 15 minutes versus a tracklet where the cadence is 40 minutes.

At the conclusion of this process, there were 4153 128 x 128 tiles with visible tracklets and 543,595 without known visible tracklets from GB5-R5. Of the GB-All (28-06-2013) data, there were 508 visible tracklets along with 57,403 tiles without visible tracklets. However, only 6 of these GB-ALL fields (GB3, GB4, GB5, GB9, GB10, and GB14) had a cadence of 15-20 minutes; these gave us 394 images with visible tracklets. Figure 3.21 displays some of these 128 x 128 images with tracklets. It is of note that while a small portion of this GB-All data covers GB5-R5, the reference image used is different, thus resulting in both different noise patterns as well as a shift in the positioning of each feature. An example of this can be seen in Figure 3.22. The image numbers are summarised in Table 3.1.

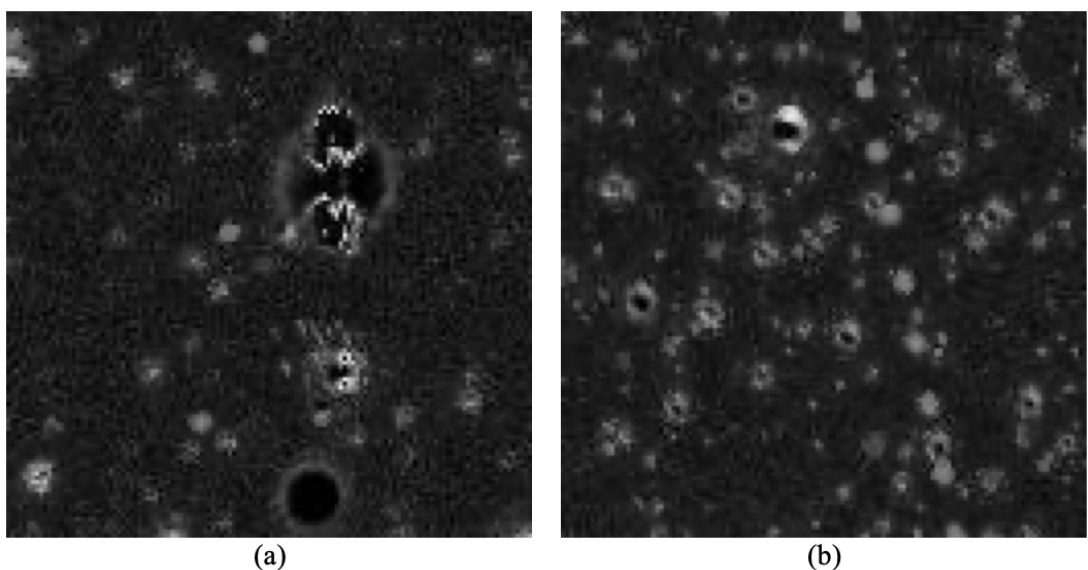


Figure 3.20: Tracklets from observations with low (a) and high (b) cadence.

	GB5-R5 (2006-2009)	GB-ALL (28-06-2013)
Observations	49,901	306
Subtracted Stacks	2252	120
Stacks with Visible Tracklets	1078	120
128x128 images	551,936	61,440
Images with Tracklets per Line Clipping	8341	4037
Images without Tracklets	543,595	57,403
Actual Images with Known Tracklets	4153	508

Table 3.1: Imaging data numbers

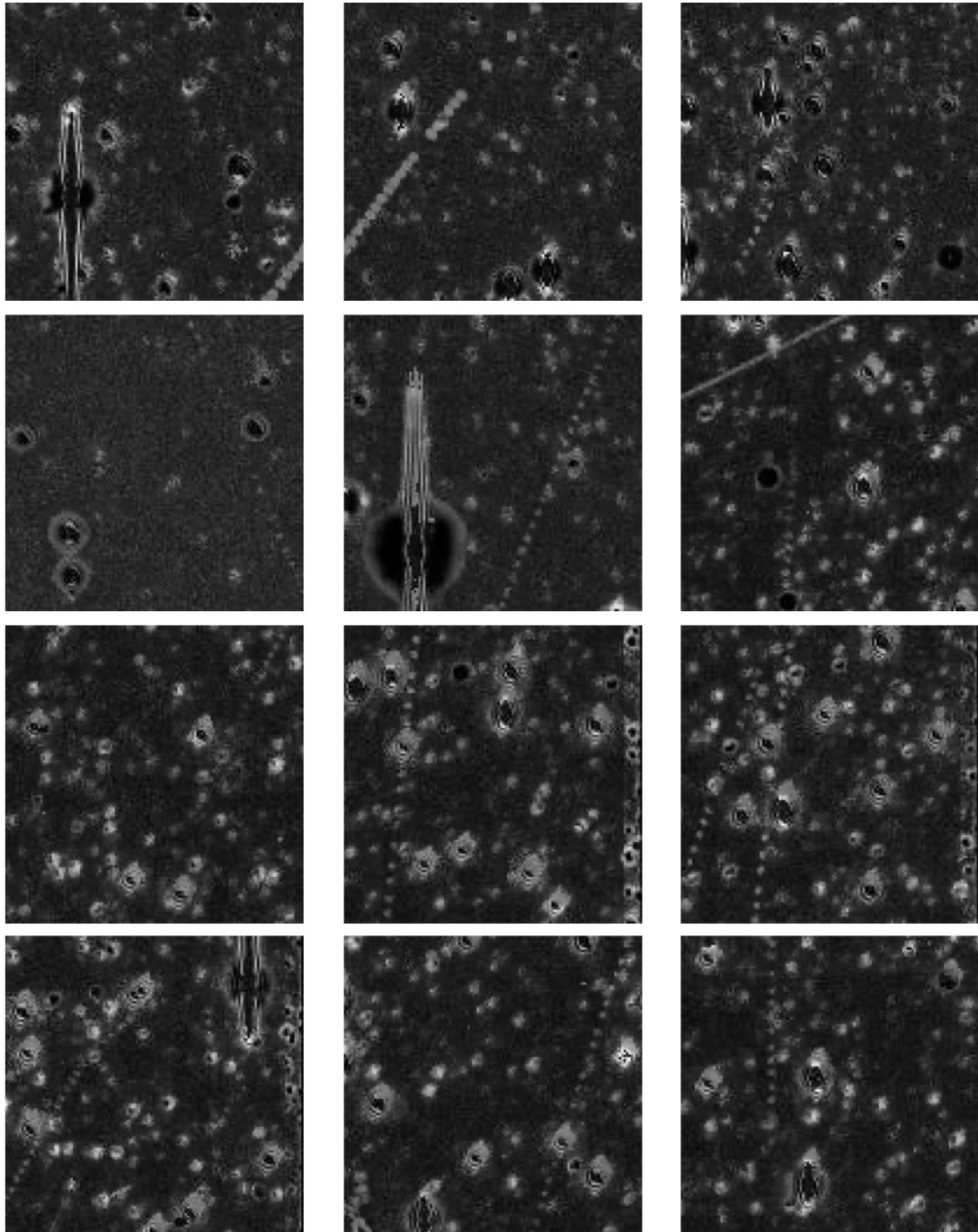


Figure 3.21: A selection of 128 x 128 tiles with tracklets of known asteroids.

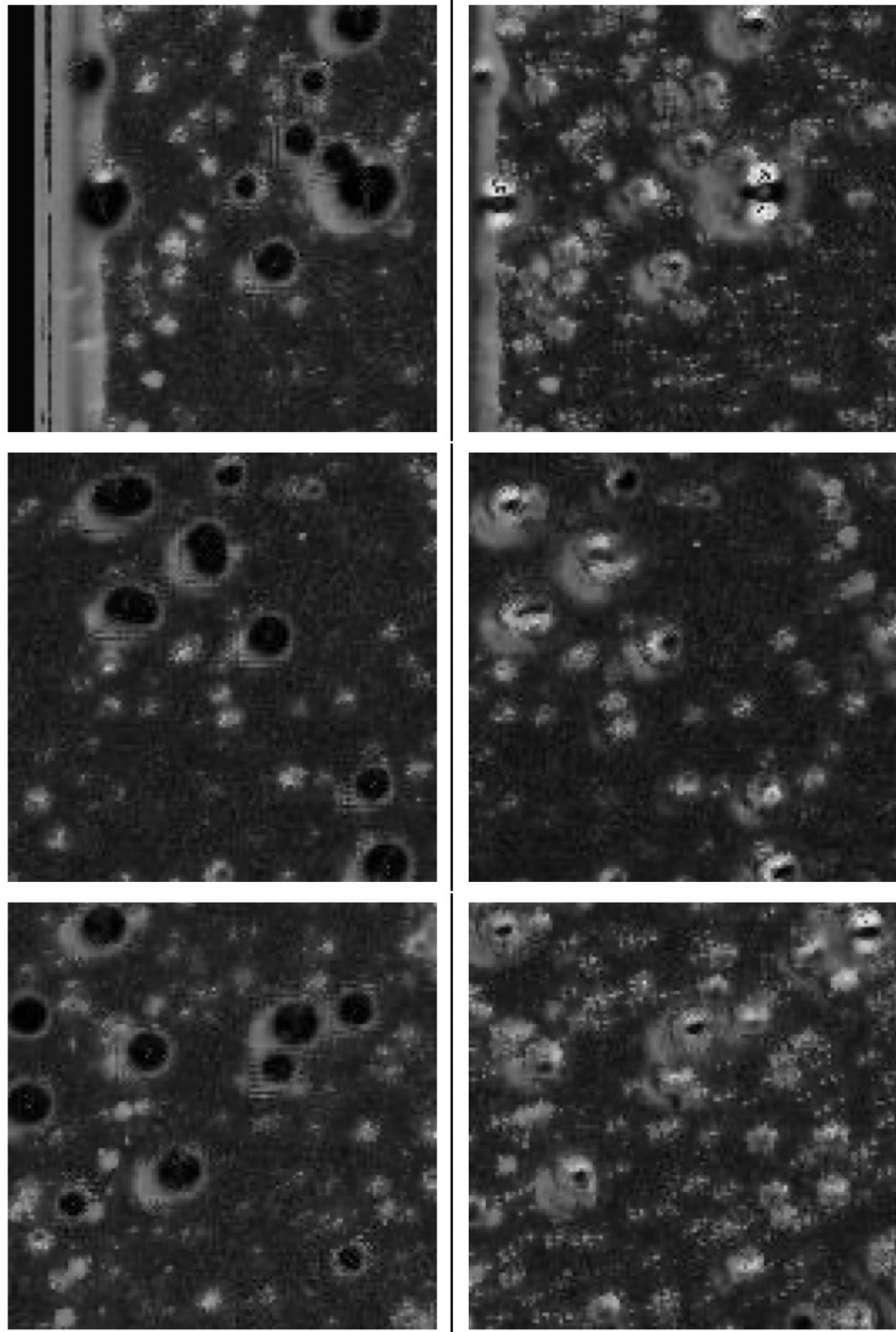


Figure 3.22: What a difference the reference image makes. On the left are images from the GB5-R5 dataset and on the right are the corresponding sub-regions from the same night, field (GB5) and, chip (R5) from the GB-All dataset. Differences are due to a different reference image being used.

3.3 Summary

The MOA project at the Mt John Observatory has thus far produced over 100 TB of data since start of operations in 2004. The 1.8m MOA-II telescope surveys 23 fields towards the Galactic Bulge and uses a camera fitted with 10 CCD chips, each of which record 2048 x 4086 pixels of data. There were 2 data sets: 14 years of observational data from a single chip (R5) of the field GB5 (GB5-R5) and 2 years of data from all of 23 fields surveyed by MOA-II (GB-All). From the GB-All data, one night (28-06-2013) of observations was extracted with the intention of using this data as an additional test set for the deep learning networks. The survey's high cadence observations make it ideal for finding asteroids in our solar system. These are clearly visible upon stacking all of a night's observations by brightest pixel for each CCD chip in each of the fields surveyed. To further highlight moving objects, the nightly brightest pixel stack was subtracted from its corresponding median pixel stack to get a subtracted stack image that only contained any moving objects and noise. The MPC was queried to get all of the known asteroids that may have been captured by MOA-II and careful examination of the data lead to 1178 distinct main-belt asteroids from GB5-R5 with limiting magnitude of 20.5 and lower. Each of the 1078 nightly stacks from GB5-R5 and 120 stacks from GB-All (28-06-2013) that contained visible tracklets were split into 512 128 x 128 tiles. The Cohen-Sutherland line clipping algorithm was used to discover that 8341 of these tiles from GB5-R5 and 4037 GB-All (28-06-2013) were expected to have tracklets. Further visual inspection of these tiles resulted in 4547 tiles with visible tracklets that could be used as inputs for a neural network, as we will see in the following chapter.

Chapter 4

Tracklet Classification With CNNs

The appeal of neural networks and deep learning is in their capacity to make predictions about complex data in real time after they have been trained with a representative dataset. In the last decade, deep learning has matured significantly and has proven to be highly effective for classification tasks. As we have seen in Chapter 2, researchers have been steadily gearing up to leverage this ability for analysing astronomical data. Here, the efficacy of several classification architectures for identifying images that contain asteroid tracklets is tested. The tracklets are in essence a sequence of fuzzy dots that could describe a line. The challenge is to find these in noisy, unevenly sampled data.

4.1 Data

Supervised learning with any neural network requires labelled data to train the network. In this case, both images with and without tracklets are required to effectively train the networks and achieve the desired result. The focus is on tracklets observed when fields are surveyed at 10-20 minute intervals as these produce well-defined tracklets as seen in the figures from the previous chapter. Note that the following subsections (4.1.1 and 4.1.2) detail how the GB5-R5 data was structured for the training and validation sets required for supervised

learning. The GB-All data was kept aside to be exclusively used as an additional test set. Table 4.1 summaries the data used for classification. Table 4.2 in Section 4.3 summarises how this data is distributed for supervised learning.

4.1.1 Images with Tracklets

Chapter 3 described the creation of a dataset with 4153 128 x 128 subtracted stack images from GB5-R5 that contained visible tracklets of main belt asteroids. We have seen examples of these tracklets in Figures 3.21. These were further refined by removing the images where the tracklet was too obscured by noise or where there were less than 3 clearly visible adjacent point sources related to a tracklet. Following this, there were 4072 images with tracklets.

Labelling this data is a time-intensive process, but having plentiful labelled data is crucial to training neural networks. To make up for the current dearth of labelled data, several image enhancement procedures were applied to augment the data. It was decided that each image with a tracklet would be supplemented by being rotated by 180 degrees, flipped horizontally and vertically, brightened, darkened, blurred, and with the contrast both increased and decreased. This gives us 8 augmentations per tracklet image and these can be seen in Figure 4.1.

Creating and using simulated tracklet data was briefly considered and rejected as it is impossible for such data to accurately represent the nuances of real astronomical observations.

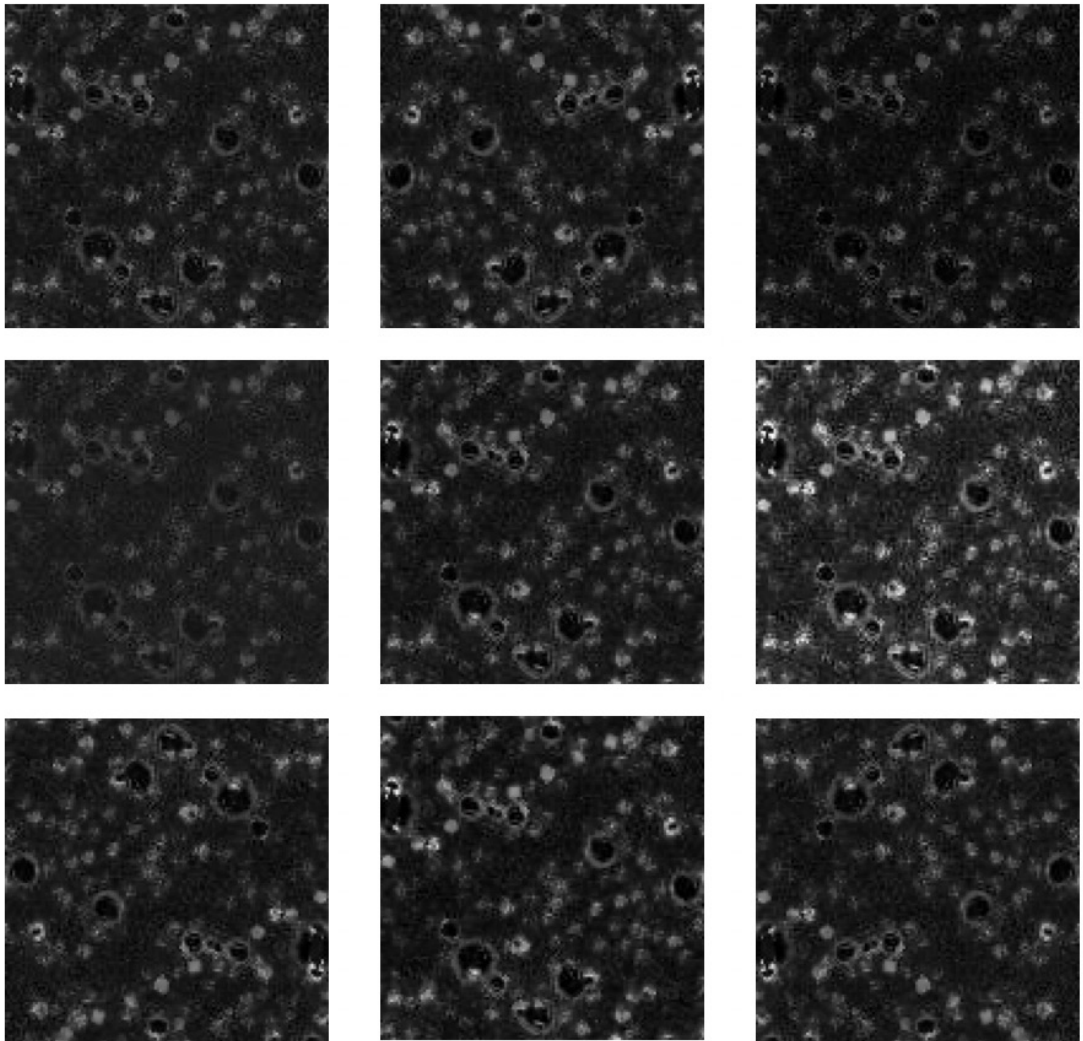


Figure 4.1: A tracklet image (centre) with its augments. Clockwise from top left: low contrast, horizontal flip, darkened, brightened, vertical flip, high contrast, rotated, blurred.

4.1.2 Images without Tracklets

Image with tracklets form only a small part of the sum total of the imaging data. For example, from the GB5-R5 data, there were 543,595 (128 x 128) images that were deemed to have no known asteroid tracklets. Given this, it is essential that the network be capable of differentiating between images with and without tracklets. Again, using generated data was considered. An attempt was made to stack randomly chosen observations from randomly chosen days and generate the subtracted stack images. However, the generated data did not contain the variations seen in the real data and thus was discarded.

Since each of the 2048 x 4096 is divided into 128 x 128 tiles, this gives us 512 images from each stack. Of the ones that did not have any known tracklets, 40 images were randomly chosen from each of the 512 sub-regions, giving us 20,480 images that had no known tracklets. This collection was visually inspected and any images that could potentially have tracklets were removed. This process resulted in 19,682 images that could be used as a representative sample of no tracklet data. Figure 4.2 shows a some of these images. Augmenting this data is discussed in Section 4.3.

	With Tracklets	Without Tracklets
Total	4153	543,595
Used for Classification	4072	19,682

Table 4.1: Summary of data from GB5-R5 used for classification

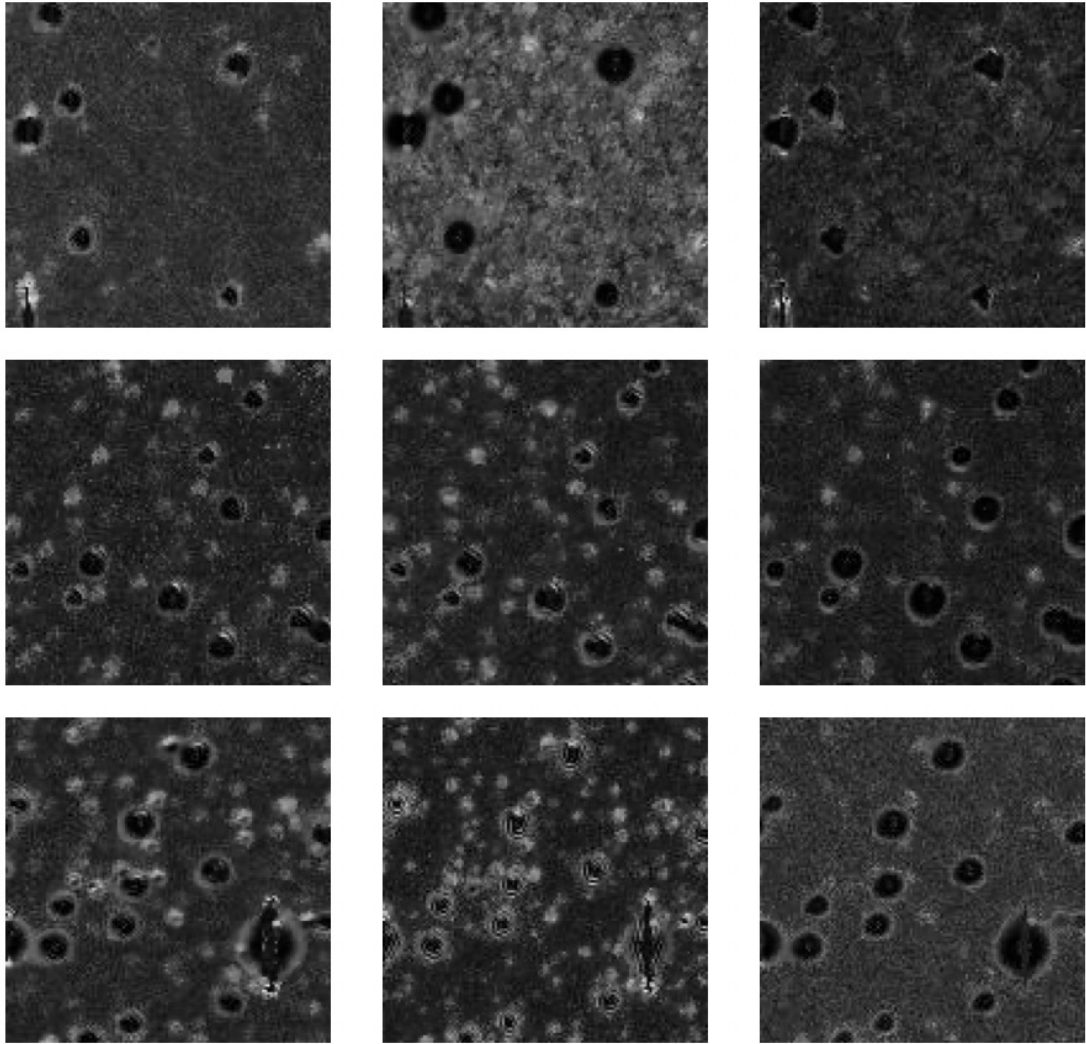


Figure 4.2: Images with no tracklets. Each row is from the same sub-region, but different nights.

4.2 Network Architecture

Appendix A contains a brief description of neural networks in general, with a short introduction to convolutional neural networks (CNN). To recap, a CNN takes an image as an input and, through a series of layers that apply weights or importance to certain features in the images, gives us the desired output. The desired output in this case is the correct classification of a subtracted stack image as either having a tracklet or not. Here, CNN-based networks are discussed in more detail and we take a look at the different architectures applied to the task of classifying asteroid tracklets.

While several different architectures as well as slight variations of each were tested during this research, only the ones that produced useful results will be discussed. All networks were built using TensorFlow and Keras. Further details about the training environment are in Section 4.3.

4.2.1 Stacked Architectures

The traditional form of a CNN consists of a number of consecutive CNN layers placed one after the other, as typified by LeNet-5 (Lecun et al. (1998)) and later AlexNet (Krizhevsky et al. (2012)). In the case of these stacked network architectures, the input for each layer is the output of the preceding layer. The number of filters in a layer increases with the depth of the network. The purpose of the filters is to capture the patterns in the input. These patterns are combined in each subsequent layer and get more complex further down the network. It is postulated that the more layers a network has, the better it is at extracting higher level features from an image, and thus learning from complex data. The limiting factor here, however, is the size of the input and how localised the features we are interested in are. After a given number of layers, a network could start overfitting to the data by focusing on the irregularities in the images. So, while increasing the number of convolutional layers improves the performance of the network, it is not the case that the deeper network is always the best option. Part of the challenge is finding the ideal depth for a network to optimally predict the output.

The research started with AlexNet, which proved to be the seminal work with CNNs and popularised the use of GPUs for training neural networks. It consists of 5 convolutional layers and 3 fully connected layers, the last one being the output. The ReLU (Glorot et al. (2011); Nair and Hinton (2010)) non-linearity is used as the activation function, with batch normalization (Ioffe and Szegedy (2015)) after each convolution and dropout (Srivastava et al. (2014)) between each fully connected layer. Additionally, the first convolution has a receptive field (filter size) of 11x11 with stride of 4 and each maxpool layer has a receptive field of (3x3) with a stride of 2. This architecture was adapted to the research question by first updating the final output to have a sigmoid output of 1 (binary classification). A second variation was then implemented which replaced the first convolutional layer with a 7x7 filters with a stride of 2. This gives us a significant performance boost, as seen in 4.4. The receptive field for the maxpool layer was also changed to cover 2x2 pixels in an effort to preserve more information as we traverse through the network. Both the original and the custom architecture are represented in Figure 4.3.

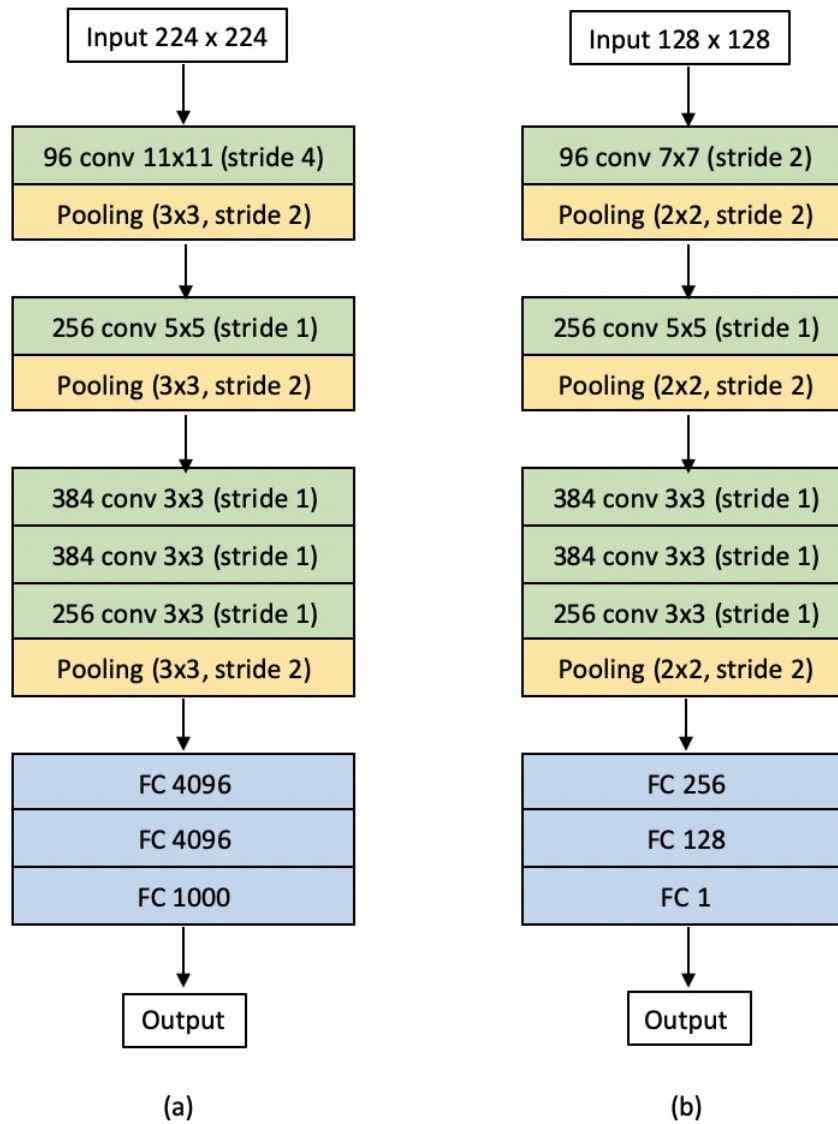


Figure 4.3: Architecture of the original AlexNet (a) and the custom version (b)

After starting with the AlexNet architecture, the number of convolutional (conv) layers were gradually increased with performance improving appreciably from 12 convolutional layers. This brings us to VGG16 and VGG19 (Simonyan and Zisserman (2015)).

Simonyan and Zisserman (2015) proposed an effective way to increase the number of convolutional layers to significantly bolster the accuracy of models thus structured and demonstrate the performance improvement with a greater number of layers. The crux of the architecture is the convolutional block, which consists of two or more convolutional layers followed by a pooling layer (usually maxpool, stride of 1). Figure 4.4(a) displays the general structure of VGG16. VGG19 has one additional 256 conv layers and two additional 512 conv layers. These architectures also popularised the use of small receptive fields for filters and the use of the 3x3 filters is now ubiquitous. These networks have a large number of parameters, with 14 million for VGG16 and 20 million for VGG19.

Several custom stacked architectures that followed the conventions of the VGG networks were also created and they are each referred to by the prefix “MOA” followed by the number of convolutional layers included. Figure 4.4(b) and (c) are two of the custom architectures implemented: MOA-12 and MOA-14 with 12 and 14 convolutional layers, respectively. Each has three fully connected layers, the last of which is the sigmoid output. The pooling employed is maxpool with 2x2 filters and a stride of 1 and dropout is used after the two fully connected layers before the output. The default stride in each case is 1x1 pixels. MOA-13 has one more 512 conv layer than MOA-12 and MOA-15 has one more 32 conv layer than MOA-14. These custom architectures have far fewer filters, and thus fewer parameters, than the VGG networks. MOA-12 and MOA-13 have around 5.2 million parameters, MOA-14 has around 10 million parameters, and MOA-15 has 12 million parameters. The number of parameters and the time taken to train each network architecture is noted in Table 4.3.

One drawback of this architecture, with adding ever more layers and filters to a model in a bid to improve performance, is that the networks are prone to overfitting. The architecture we will look at next attempts to address this problem as well as allow data to be analyzed at different scales.

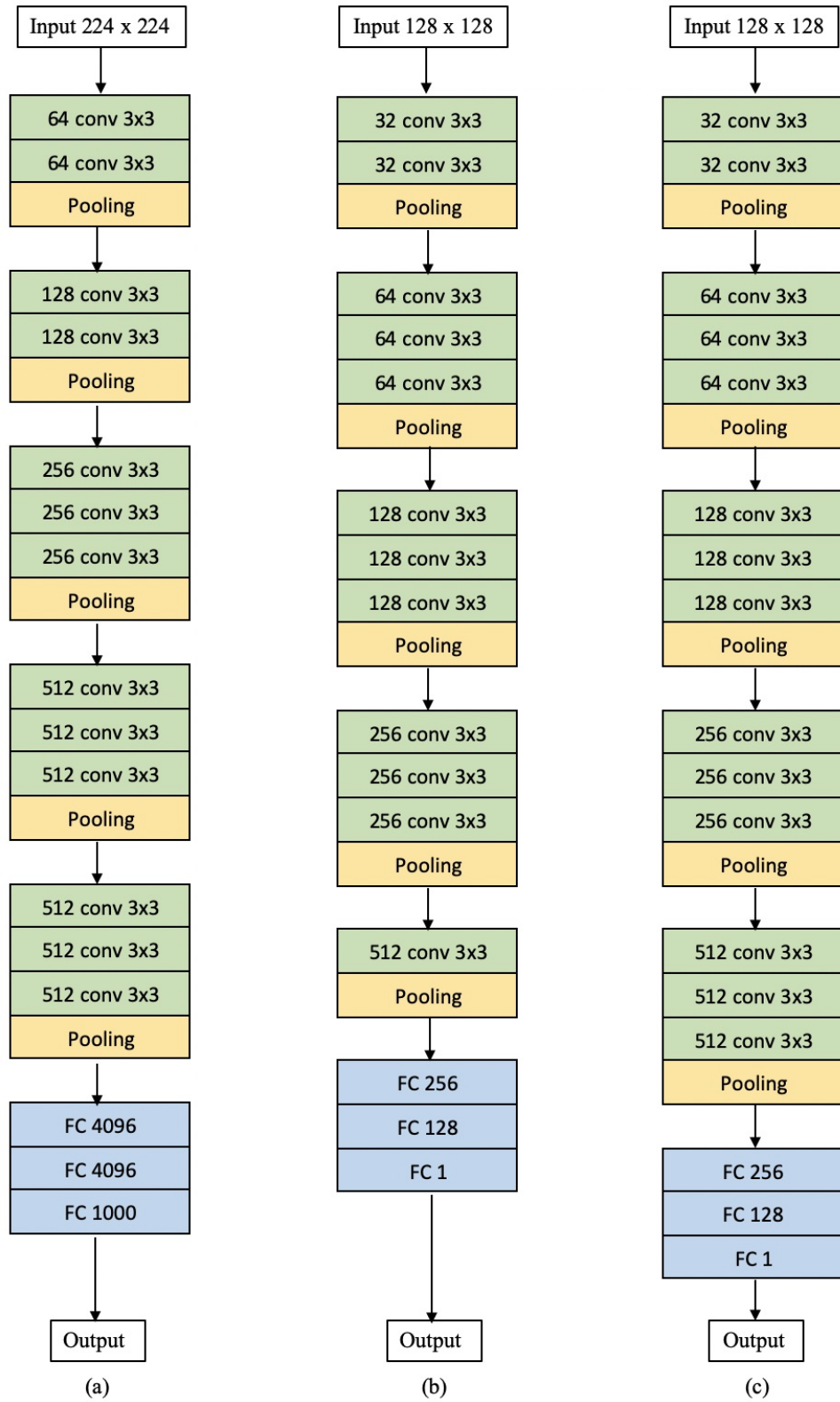


Figure 4.4: Architecture of VGG16(a) and the custom architectures MOA-12(b) and MOA-14(c). VGG19 has one more 256 conv layer and two more 512 conv layers than VGG16. MOA-13 has one more 512 conv layer than MOA-12 and MOA-15 has one more 32 conv layer than MOA-14.

4.2.2 Inception-ResNet

So far we have focused on networks that follow a stacked, linear structure. With the raising popularity of using CNNs for computer vision tasks came new architectures which suggested ways of improving the representational power of a network without relying solely on increasing the depth and number of filters.

Inception

GoogLeNet (Szegedy et al. (2015a)), more popularly known as Inception, introduced the idea of modifying the width of a network as well as employing filters of a variety of sizes to better capture multi-scale data from images. Given the nature of tracklets - some spanning the whole image, others only a few pixels - utilizing an architecture capable of analysing relevant data at multiple scales was deemed prudent.

Central to this architecture is the “inception module”, which is a block of parallel convolutional layers with 1x1, 3x3, and 5x5 filters, and a 3x3 maxpooling layer. These are concatenated and the result passed to the next node in the model. Additionally, 1x1 convolutional layers are added before the 3x3 and 5x5 convolutions and after the maxpool layer to reduce the depth dimension (channels) and thus the computational cost of the module. The ReLU activation, used in all of the convolutional layers, is used on the 1x1 convolutional layer as well, adding desirable complexity to the output. The structure of the inception module can be seen in Figure 4.5.

While Szegedy et al. (2015a) do recommend a 22-layer architecture for a network model employing the inception module, it is quite complex and so a simpler 15-layer architecture (with 12 convolutional layers) was implemented for this research. The architecture for the custom model can be seen in Figure 4.7. The number of filters in each of the layers in an inception module is set by the original paper (Szegedy et al. (2015a)) and can be seen in Figure 4.6. GlobalAveragePooling, which pools the data across each channel to select an average, is used

instead of the more tradition “flatten”, further reducing the number of parameters. Dropout is once again included between the fully connected layers. Additionally, Keras Applications has an implementation of Inception V3 (Szegedy et al. (2015b)), which was also applied to this problem. The results from both are in Section 4.4. Note that when counting number of layers in the Inception architecture, we only consider the fully connected layers and all convolution layers except the 1x1 convolutions.

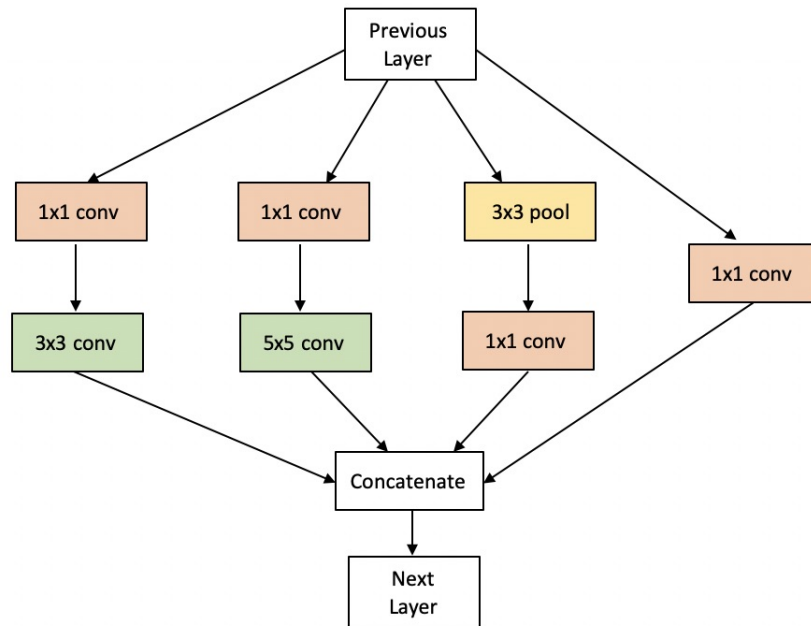


Figure 4.5: Composition of an Inception module

	1x1	3x3 reduce	3x3	5x5 reduce	5x5	pool project
Module 1	64	96	128	16	32	32
Module 2	128	128	192	32	96	64
Module 3	192	96	208	16	48	64
Module 4	160	112	224	24	64	64
Module 5	128	128	256	24	64	64

Figure 4.6: Number of filters in each layer of five Inception modules. The “reduce” columns refer to the 1x1 filters before the 3x3 and 5x5 convolutions.

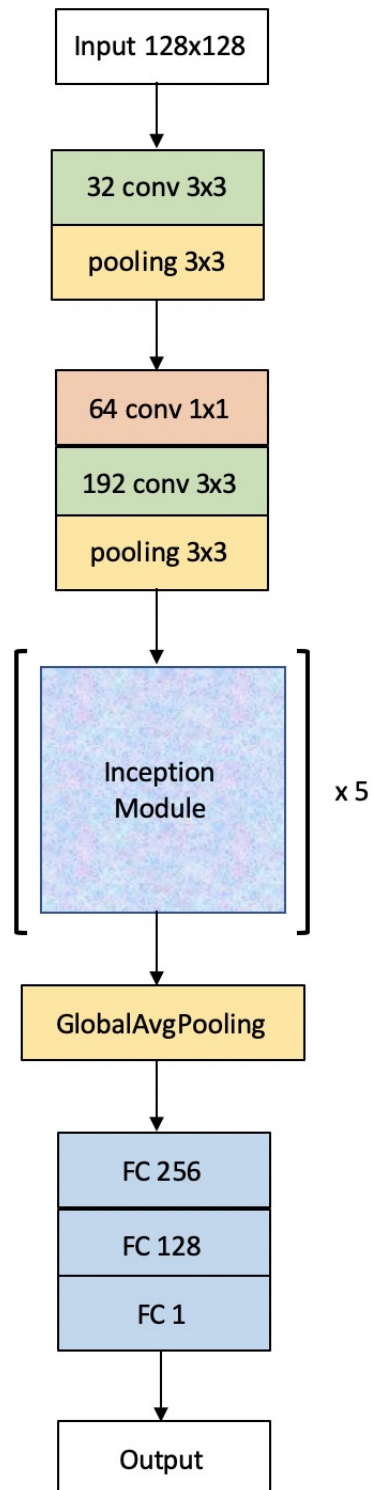


Figure 4.7: Custom 15 layer Inception architecture using the Inception module seen in Figure 4.5

ResNet

While the key implementation goals for Inception are both accuracy for multi-scale data and lower computational cost, it still suffers from the vanishing/exploding gradient problem that has plagued deep neural networks (Bengio et al. (1994); Glorot and Bengio (2010)). Backpropagation (Rumelhart et al. (1986)) is at the heart of neural networks and works by calculating the error gradient while traversing the network from output to input. These gradients can get smaller and smaller, approaching zero, leaving the weights in the earlier layers unchanged, and so hindering optimization. Conversely, these gradients can also get very large, once again stopping gradient descent from converging. Very deep networks also suffered from a degradation in accuracy in the later layers, with the error increasing with depth.

Residual networks (He et al. (2016)), or ResNets, were introduced to address this degradation of training accuracy by presenting an alternative pathway for the algorithm to follow called the “skip connection”. The central element in this architecture is the “residual block”, which consists of two convolutional layers with 3x3 filters. The input to this block is added to the output of the second convolution, thus creating a “shortcut” connection. The added output is then passed through a ReLU activation function before following on to the next layer. This structure can be seen in Figure 4.8. Note that while the inception module concatenates its layers, the residual block adds them. This means that the same sized filters must be used by all the layers in a residual block. Figure 4.8(a) is the model used if the previous and current layer have the same number of filters. If they do not, then a 1x1 filter is applied to the input to adjust the number of filters before the add function (Figure 4.8(b)).

ResNets proved to be very effective and paved the way for very deep networks with over 50 layers, some over a 100. Since a network with fewer layers is more appropriate for this research, a custom 18-layer architecture (Figure 4.9) was implemented to test its ability to classify asteroid tracklets. “Residual” in the figure refers to a residual block, with the associated number being the number of filters in a given block. Maxpooling is used with a 2x2 receptive field and stride of 1. ReLU activation is the non-linearity applied throughout except in the final layer where sigmoid activation is used instead.

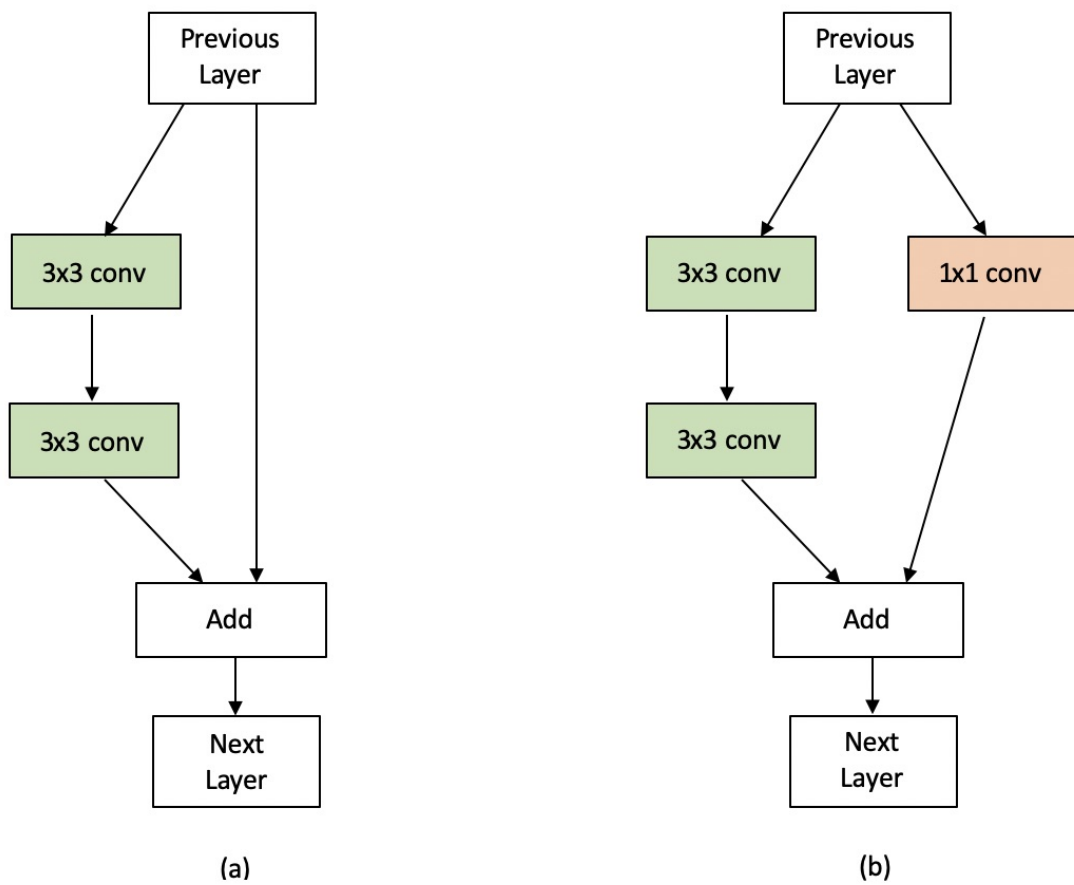


Figure 4.8: Residual block when the previous and current layer have the same (a) and different (b) filters

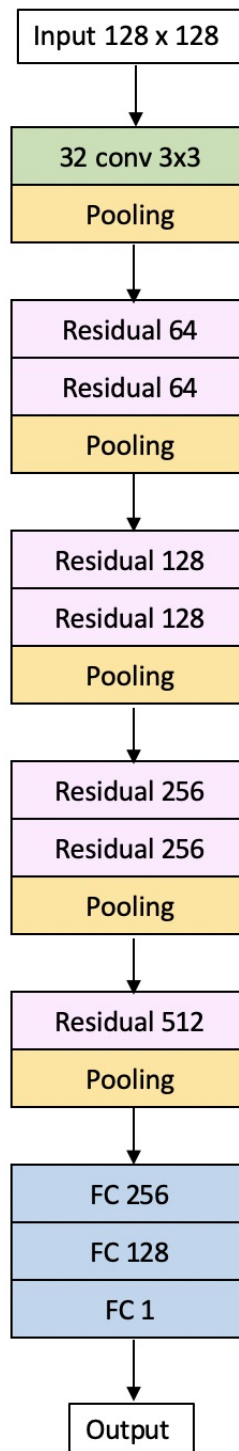


Figure 4.9: Custom 18 layer ResNet

Inception-ResNet Hybrid

With the success of the ResNets, Szegedy et al. (2016) proposed combining the inception module and the residual block. The proposed architecture - as the previous ones - was designed to suit the requirements and challenges posed by the ImageNet database and add support for ever deeper networks for analysing similarly structured data. While this has proven very successful at solving that given set of complex tasks, astronomical data is significantly different from the images that these networks are optimised to handle.

A novel architecture is proposed to suit the goal of finding asteroid tracklets in the subtracted stack images. A hybrid module was created that combined the Inception multi-scale architecture with the skip connections of ResNets. A variety of combinations of the inception module and residual block were considered before the hybrid module seen in Figure 4.10 was selected.

The hybrid module consists of four branches that are combined by the 'add' function. The four branches use the same number of filters and are:

- Branch 1 is a 1x1 convolution with a ReLU activation, followed by a 3x3 convolution.
- Branch 2 is a 1x1 convolution with a ReLU activation followed by two 3x3 convolutions. The two 3x3 convolutions serially linked have the effective receptive field of a 5x5 convolution, but with fewer parameters and thus calculations.
- Branch 3 is a 1x1 convolution with a ReLU activation.
- Branch 4 is the input. If the number of filters in the previous layer are not equal to the current layer's filters, a 1x1 convolution is applied to the input so the filters can match (a requirement for the add function). Activation function ReLU works best here as well.

The add function is followed by a ReLU activation before being passed to the next layer. The many 1x1 convolutions in the module may seem superfluous but removing even one negatively effects the performance of the network. This

may be because of the beneficial complexity introduced by the associated non-linearity. Additionally, a weight constraint was added to the convolutional layers in the module, limiting the weights to have a magnitude of 3 or less.

Three custom architectures that uses this hybrid module can be seen in Figure 4.11. Each take the same image as input, but the first convolutional layers applied to each is slightly different. Each of the architectures start with a 3x3 convolutional layer, but while (a) has a receptive field of 3x3, (b) has an effective receptive field of 5x5, and (c) has the effective receptive field of 7x7. The numbers of filters passed to the hybrid module follows the conventional structure in (b), whereas the increase in filter size is much more gradual in (a) and (c). There is no performance loss with having fewer filters. Once again, maxpool with a receptive field of 2x2 and stride of 1 was used along with an aggressive dropout of 0.4 between the fully connected layers. Networks (a) and (c) have around 6.4 million parameters, (b) has around 5.2 million parameters.

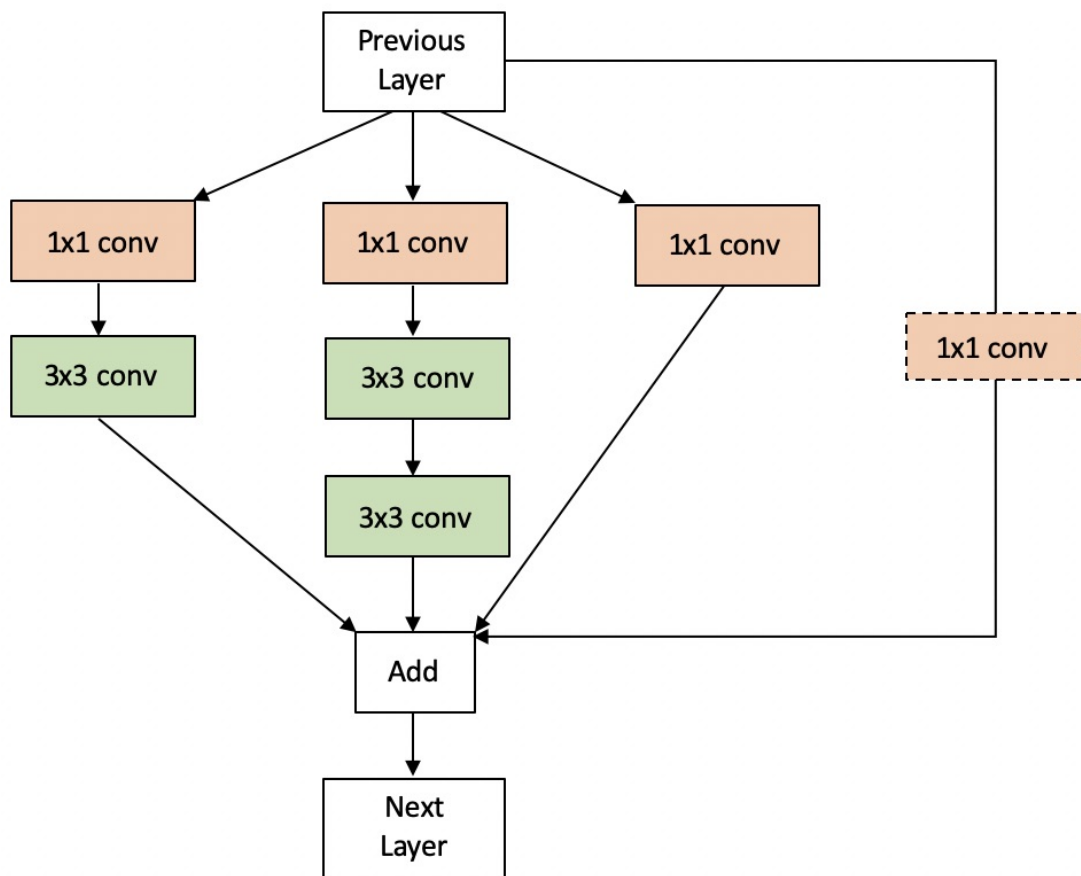


Figure 4.10: Hybrid module combining salient features of a ResNet block and an Inception module

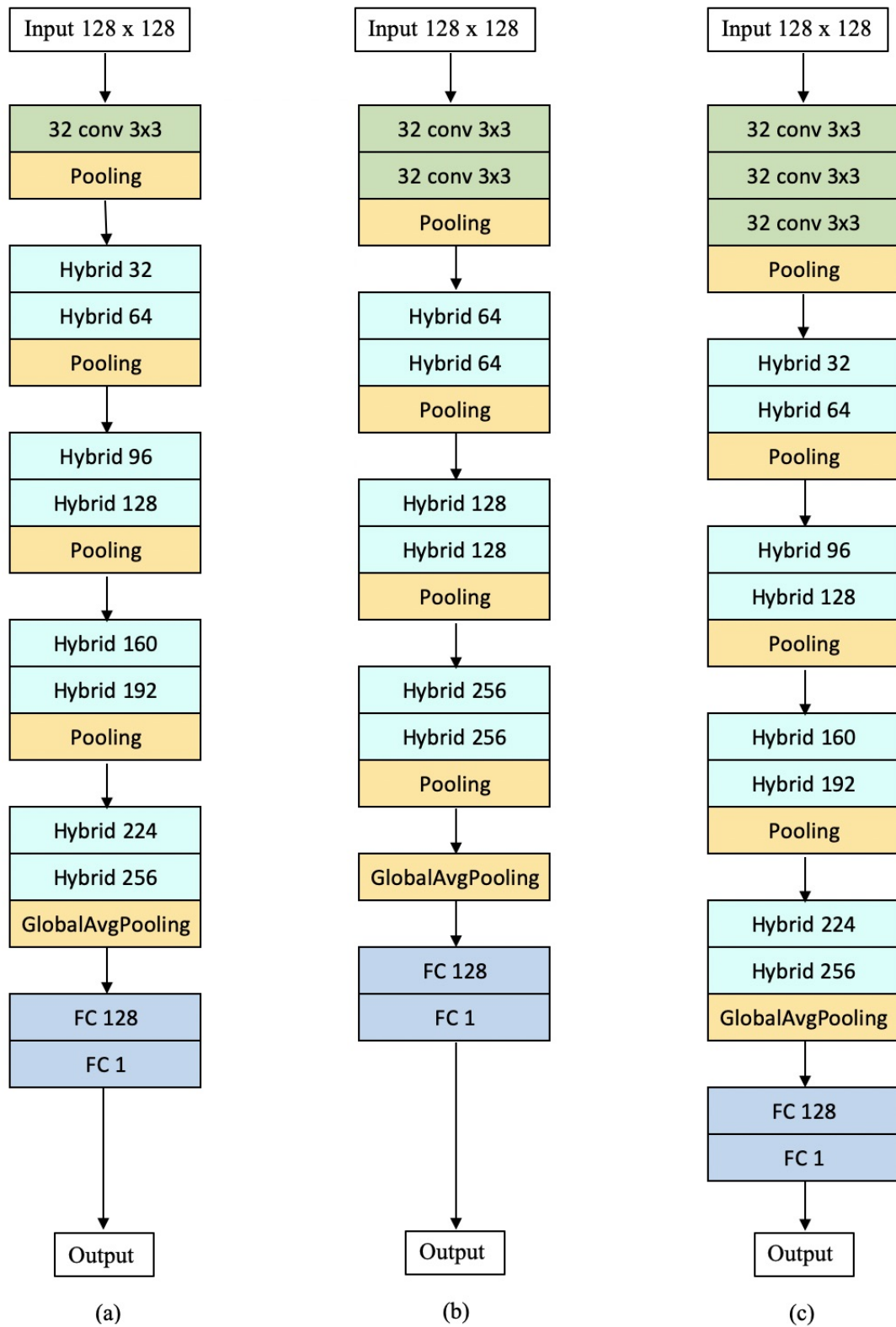


Figure 4.11: Custom architectures using the hybrid module.

4.3 Training

All of the algorithms were trained on a Linux machine running Ubuntu 18.04 with a NVIDIA Quadro M4000 GPU (8 GB, 2.5 TFLOPS). The code was written in Python 3.6 in the Jupyter Notebook environment. TensorFlow GPU 2.4.1 (CUDA 11.0), along with the Keras deep learning API, was used for creating the CNN models tested.

The training, validation, and test data sets were created from the 4072 tracklet and 19,682 no tracklet images of dimension 128x128. Both of these sets of images were first split into the groups before augments were included. Figure 4.2(a) shows the distribution of the data before it was augmented. “Yes” and “No” refers to images that either have tracklets (yes) or do not (no). As discussed in Section 4.1.1, the tracklet data was augmented with 8 image. These were generated only for the training and validation “Yes” sets. The same set of augments was generated for the no-tracklet training and validation data, but since there was already significantly more of that than the “Yes” data, only a random 35% of these augments were added to the “No” sets. None of the data in the test set was augmented. The final numbers in the train, validation, and test set can be seen in Figure 4.2(b). There is no overlap between the 3 data sets. An additional test set composed entirely of images from the GB-All (28-06-2013) subset was also constructed and had 300 tracklet and 2000 no tracklet images.

A batch size of 32 was used for training each network model. Training was set to run for 50 epochs, with a callback for early stopping if the validation loss failed to minimise after a set number of epochs. This number was either 25, or 30 depending on evaluation following the initial two runs for a model. The Adam (Kingma and Welling (2014)) optimiser was used along with accuracy as the metric monitored during training. The loss function to minimise was binary cross entropy. The metric of recall was also tested but performed poorly in comparison. The learning rate was initialised at 0.0001 for all networks except Hybrid(a), which started with a learning rate of 0.001. In each case, the learning rate was reduced after 15 epochs. Some networks performed better with the learning rate reduced by a factor of two every even numbered epoch after 15 and others performed better with a blanket reduction by a factor of 10 after 15 epochs. Callbacks were included to save the best weights for both validation and training

accuracy. The Keras models performed better when pre-loaded with ImageNet weights before fine-tuning with the MOA-II data. The inputs for these were also pre-processed to best suit the Keras models' requirements. Table 4.3 details the number of trainable parameters in each model as well as the time taken to train them in the environment described here.

	Train	Validation	Test
Yes	3322	415	335
No	15,595	2039	2048

(a)

	Train	Validation	Test
Yes	29,898	3735	335
No	59,261	7748	2048

(b)

Table 4.2: The Train/ Validation/ Test set before (a) and after (b) augmentation.

	Parameters (in millions)	Time to Train	Model Reference
AlexNet	4.8	1hr 25mins	Figure 4.3 (a)
Custom AlexNet	8	1hr 40 mins	Figure 4.3 (b)
MOA-12	5.3	3hrs	Figure 4.4 (b)
MOA-13	5.3	3hr 10mins	Figure 4.4
MOA-14	10	3hrs	Figure 4.4 (c)
MOA-15	12.2	4hrs 5mins	Figure 4.4
Keras-VGG16	14	4hrs	Figure 4.4 (a)
Keras-VGG19	20	16hrs 15mins	Figure 4.4
Custom Inception	2.2	2hrs 55mins	Figure 4.7
Keras-Inception	20	2hrs 25mins	Szegedy et al. (2015a)
Custom ResNet	6.6	3hrs 40mins	Figure 4.9
Keras-ResNet50	21	2hrs 50mins	He et al. (2016)
Hybrid (a)	6.4	6hrs 50mins	Figure 4.11 (a)
Hybrid (b)	5.2	7hrs 30mins	Figure 4.11 (b)
Hybrid (c)	6.4	7hrs 45mins	Figure 4.11 (c)

Table 4.3: The number of trainable parameters in each network and the time taken to train them.

4.4 Results

In order to identify the network(s) most suitable for the task of classifying asteroid tracklets, the performance of each is evaluated using the GB5-R5 test set mentioned previously in Section 4.3 as well as with the GB-All (28-06-2013) data.

To recap, there are 335 images with tracklets and 2048 images without tracklets from the GB5-R5 the test set, none of which are augmented. In addition, 300 tracklet images and 2000 no tracklet images were also randomly chosen from GB-All (28-06-2013). The purpose was to evaluate how networks trained only on data from GB5-R5 would perform with data from the other MOA-II fields, including the other chips in GB5.

Note that there is no overlap between the test sets and either the training or validation sets. Each network will make predictions about both sets of test data at the confidence threshold of 0.5, which will be compared with the ground truth labels.

The goal is to have a model capable of finding asteroid tracklets in 128x128 subtracted stack images. As there will always be more images without tracklets, the accuracy metric is a poor measure as a high score is possible simply by classifying everything in the majority class. Here, the focus will be on other more robust measures based on the metrics derived from a confusion matrix. A confusion matrix breaks down the predictions made by a classifier into 4 outcomes: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). An example of this can be seen in Figure 4.12. The “positive” cases are where an image was classified as containing a tracklet and the “negative” cases are where an image was classified as without a tracklet. In this case, the ideal scenarios would be to have as few false negatives as possible together with a manageable number of false positives.

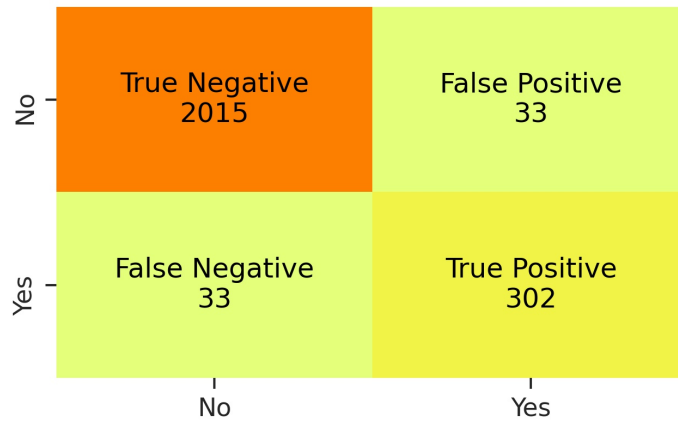


Figure 4.12: Confusion matrix for the MOA-15 custom stacked network

The evaluations metrics applied derive from the confusion matrix and are:

- Recall: also known as the true positive rate, it summarizes the number of correct positive predictions (i.e. images with tracklets) made by the classifier. This is the metric that tells us how well a classifier minimizes false negatives and is the most significant one for this research. It is calculated as follows:

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

- Precision: summarizes the number of positive predictions that are classified correctly. A large number of false positives leads to a lower precision score. It is calculated as follows:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

- F1 Score: a measure that is the balanced harmonic mean of precision and recall, giving equal weight to both metrics. The F1 score is only high if both the precision and recall are high. For binary classifications, this is a better single evaluation measure than accuracy. It is calculated as follows:

$$\text{F1} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

- F2 Score: the weighted mean of precision and recall, which assigns greater significance to recall via a parameter β with a value of two. This is another of the metrics that accurately reflects the ideal result for the classifier in a single metric: a measure that reflects a balance between precision and recall while seeking to minimising the false negatives. Thus, this value is only high when the recall high. It is calculated as:

$$F2 = \frac{(1 + \beta^2) * (\text{Precision} * \text{Recall})}{(\beta^2 * \text{Precision}) + \text{Recall}}$$

- Precision-Recall Area Under the Curve (PR AUC): area under the precision-recall curve that indicates the trade-off between precision and recall for a given classifier and, here, it indicates the cost of false positives. The precision-recall curve is obtained by plotting precision (y-axis) and recall (x-axis) at different probability thresholds.

The tables and figures over the next few pages detail the recall, precision, F1 score, F2 scores, and PR AUC for the various CNN architectures, along with the false negative and false positive numbers, when they each were tasked with making predictions. All of the metrics are taken at the 0.5 confidence threshold, with values over 0.5 indicating the presence of an asteroid tracklet in the image. The evaluation metrics for each classifier for the GB5-R5 test set are in Table 4.4 and for the GB-All (28-06-2013) test set are in Table 4.5. On both tables, the networks that perform well on both test sets are highlighted. Additionally, the confusion matrices for the top six networks with either test set are in Figure 4.13 and Figure 4.14. Confusion matrices for all other networks are included in Appendix C, along with the PR and ROC curves (GB5-R5 test set only).

The networks with the best recall value with the GB5-R5 and GB-All (28-06-2013) test sets were MOA-15 and Hybrid(b), respectively. Overall, the custom, moderately sized networks with fewer parameters generalized better than the larger Keras models. The reasons for this are likely two-fold. First, the pattern the classifiers must learn is likely best represented in the earlier layers and the custom networks are at the optimal depth to parse and benefit from this. Second, the dataset is relatively small, which might cause the networks with a large number of parameters to overfit to the training set and thus perform poorly with the test set.

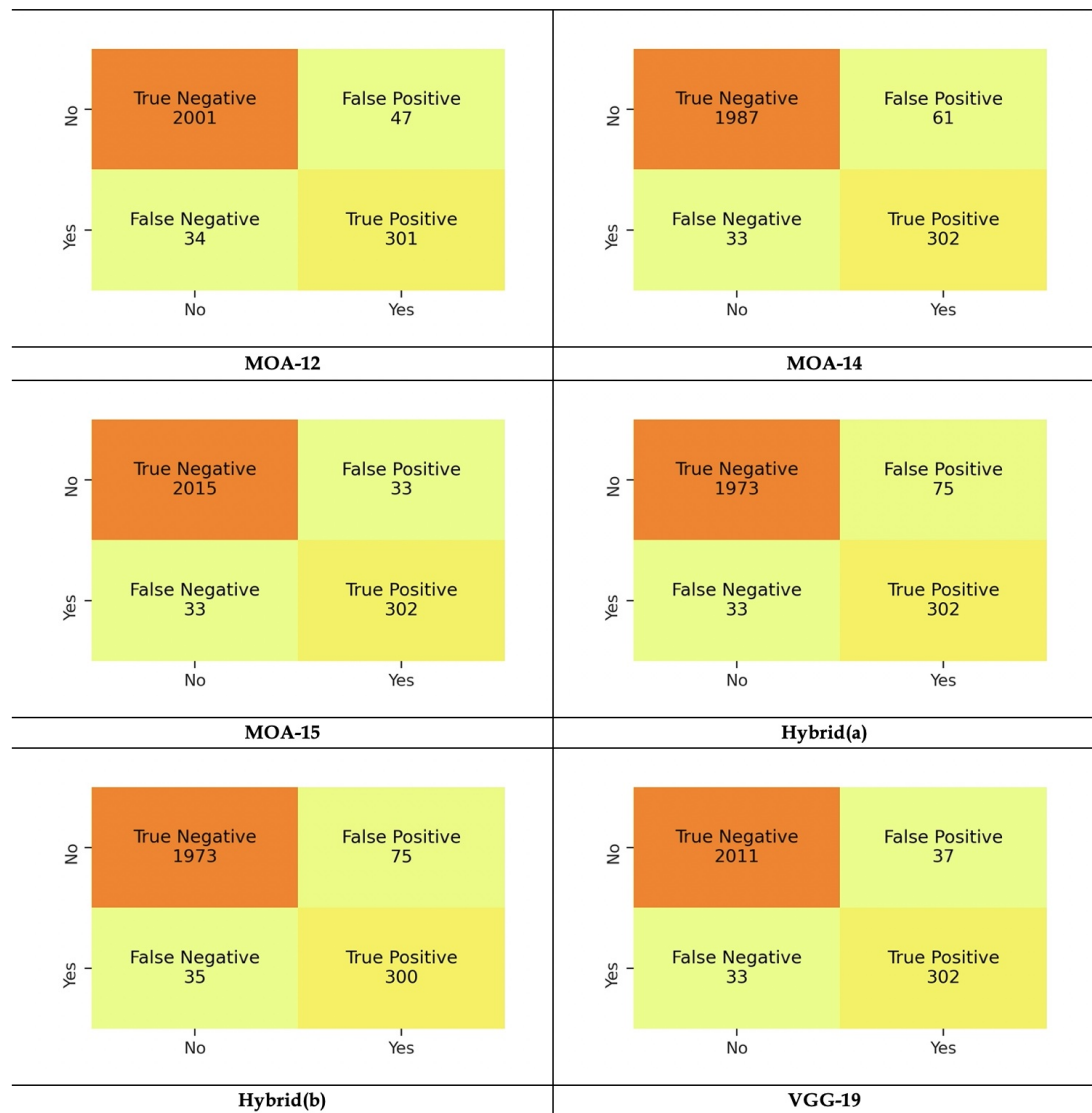


Figure 4.13: Confusion matrix of the predictions made by the top six CNN architectures on the GB5-R5 test set at the 0.5 threshold

	Recall	Precision	F1 Score	F2 Score	PR AUC	FN/FP
AlexNet	57.91%	82.55%	68.07%	61.59%	75.67%	141/41
Custom AlexNet	81.79%	88.96%	85.23%	83.13%	89.76%	61/34
MOA-12	89.85%	86.49%	88.14%	89.16%	93.28%	34/47
MOA-13	88.36%	87.32%	87.83%	88.15%	93.34%	39/43
MOA-14	90.15%	83.20%	86.53%	88.67%	93.26%	33/61
MOA-15	90.15%	90.15%	90.15%	90.15%	94.35%	33/33
Keras-VGG16	83.88%	94.61%	88.92%	85.83%	92.98%	56/16
Keras-VGG19	90.15%	89.09%	89.61%	89.93%	95.09%	33/37
Custom Inception	85.37%	86.93%	86.14%	85.68%	92.84%	49/43
Keras-Inception	66.27%	67.27%	66.77%	66.47%	75.80%	113/108
Custom ResNet	88.06%	90.21%	89.12%	88.48%	93.61%	40/32
Keras-ResNet50	19.10%	54.24%	28.26%	21.95%	38.96%	271/54
Hybrid (a)	90.15%	80.11%	84.83%	87.94%	90.62%	33/75
Hybrid (b)	89.55%	80.00%	84.51%	87.46%	90.54%	35/75
Hybrid (c)	87.16%	84.64%	85.88%	86.65%	90.56%	43/53

Table 4.4: Evaluating the effectiveness of the various CNNs at finding images with tracklets in the GB5-R5 test set at the 0.5 threshold

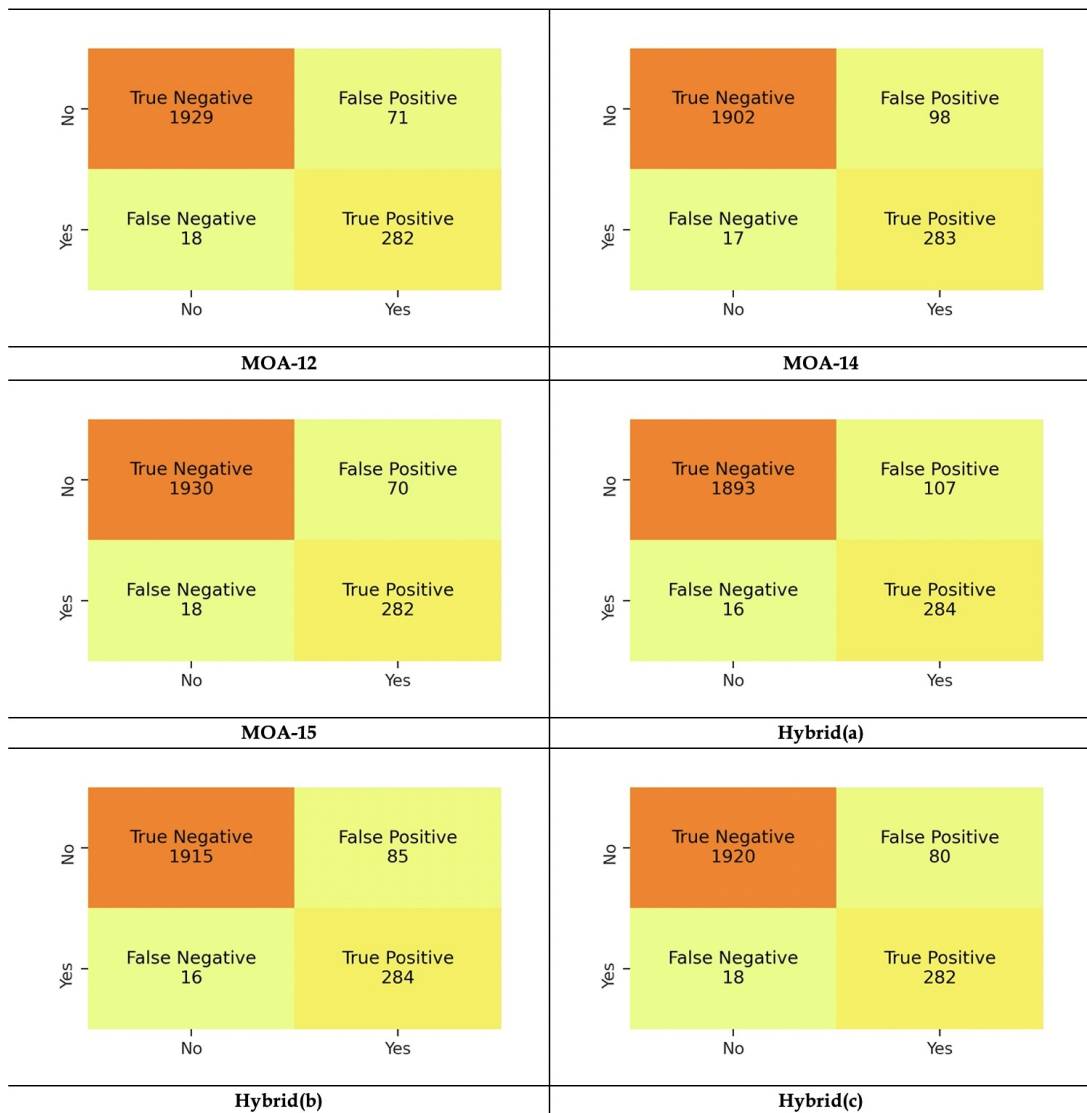


Figure 4.14: Confusion matrix of the predictions made by the top six CNN architectures on the GB-All (28-06-2013) test set at the 0.5 threshold

	Recall	Precision	F1 Score	F2 Score	PR AUC	FN/FP
AlexNet	62.33%	57.72%	59.94%	61.35%	67.82%	113/137
Custom AlexNet	84.00%	82.08%	83.03%	83.61%	90.18%	48/55
MOA-12	94.00%	79.89%	86.37%	90.79%	95.77%	18/71
MOA-13	94.00%	78.33%	85.45%	90.38%	95.55%	18/78
MOA-14	94.33%	74.28%	83.11%	89.50%	94.95%	17/98
MOA-15	94.00%	80.11%	86.50%	90.85%	95.74%	18/70
Keras-VGG16	82.33%	93.21%	87.43%	84.30%	90.69%	53/18
Keras-VGG19	90.33%	83.90%	87.00%	88.97%	93.90%	29/52
Custom Inception	89.00%	81.65%	85.17%	87.43%	93.51%	33/60
Keras-Inception	67.33%	42.80%	52.33%	60.41%	59.39%	98/270
Custom ResNet	90.00%	84.64%	87.24%	88.87%	93.81%	30/49
Keras-ResNet50	37.67%	26.40%	31.04%	34.71%	28.03%	187/315
Hybrid (a)	94.67%	72.63%	82.20%	89.25%	92.04%	16/107
Hybrid (b)	94.67%	76.96%	84.90%	90.50%	93.79%	16/85
Hybrid (c)	94.00%	77.90%	85.20%	90.27%	94.86%	18/80

Table 4.5: Evaluating the effectiveness of the various CNNs at finding images with tracklets in the GB-All (28-06-2013) test set at the 0.5 threshold

While the ideal number of false negatives will always be zero, the results are encouraging given the small amount of training data. Several of the models report competitive results and so the metrics were sorted by PR AUC, F2 Score, and recall (Figure C.7) to get the top five models that perform well on both test sets. Recall is given higher precedence is general, but where recall values are similar, networks with a higher PR AUC are chosen. These models - MOA-12, MOA-14, MOA-15, Hybrid(a) and Hybrid(b) - are highlighted in Tables 4.4 and 4.5. While VGG-19 performed well with the GB5-R5 set, its performance fell with the GB-ALL test set. The average number of false negatives from these five models is 33 with the GB5-R5 test set and 17 with GB-All. It is notable that following an analysis of the false negatives and further filtering/reducing them to the ones that are shared by each model shows that there are 19 shared false negatives from GB5-R5 and 7 from GB-All. These can be seen in Figures 4.15 and 4.16.

The false negatives largely seem to be cases where the tracklet appears too much like a satellite streak, or where the point sources are slightly further apart and potentially obscured by noise. In some of these cases - for example with the final image in Figure 4.16 - parts of the tracklet are on other images that have been correctly classified as containing a tracklet. The issue of hard-to-spot tracklets will be addressed in Chapter 5.

As several of the classifiers performed equally well, it was clear that choosing one model as a “winner” was not the best approach for discovering tracklets. It is proposed that assembling an ensemble structure incorporating predictions from all five of these models would lead to better predictions.

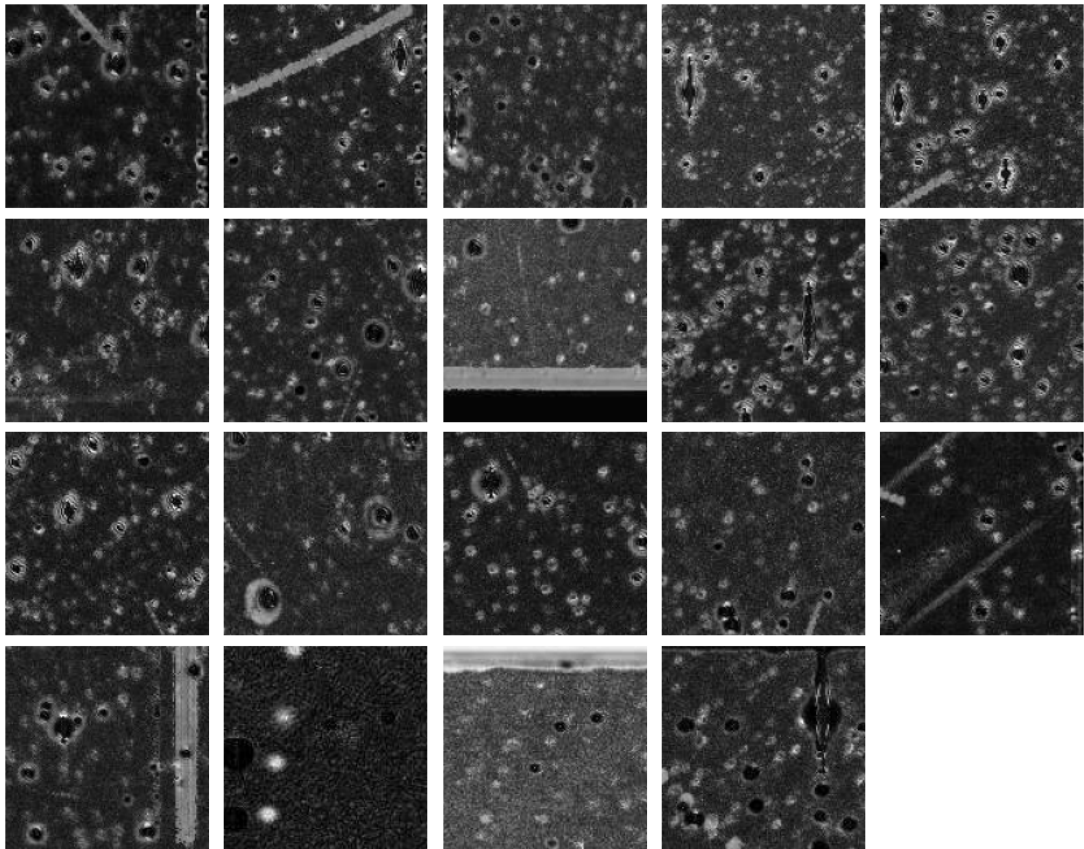


Figure 4.15: False negatives shared by the models MOA-12, MOA-14, MOA-15, Hybrid(a), and Hybrid(b) in the GB5-R5 test set

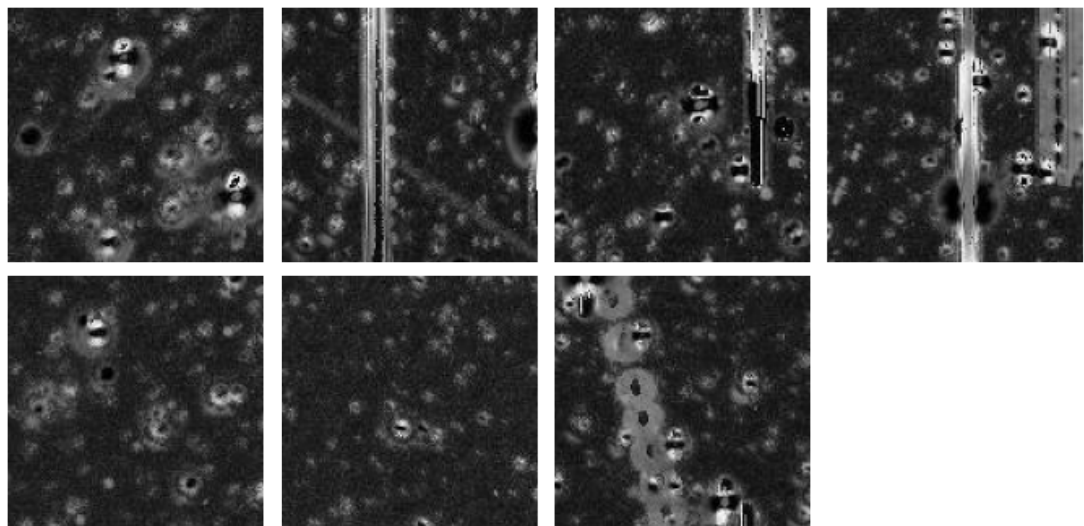


Figure 4.16: False negatives shared by the models MOA-12, MOA-14, MOA-15, Hybrid(a), and Hybrid(b) in the GB-All test set.

4.4.1 Ensemble

The lifeblood of neural networks is the complex nonlinear mathematical representation of data, which is instrumental for mapping the input to the desired output. The probability distribution learnt by each network is unique, which means that each model learns a different mapping to traverse from input to output. Chollet (2021) states that these differences in representation offer new ways of looking at the data, which may in turn boost performance by capturing and highlighting aspects missed by other representations. This potential is harnessed by creating an ensemble of classifiers where the predictions from each are pooled in order to make better predictions.

The ensemble consists of predictions from MOA-12, MOA-14, MOA-15, Hybrid(a), and Hybrid(b). All of these models perform at a similar level with an average recall of 90% with the GB5-R5 test set and 94% with the GB-All test set. Each network makes predictions about an input image and two approaches to choosing the winning prediction were trialled: averaging the predictions from all five classifiers or selecting the highest (max) predicted value.

	Recall	Precision	F1 Score	F2 Score	PR AUC	FN/FP
Ensemble-Max-GB5-R5	94.33%	63.71%	76.05%	86.06%	91.64%	19/180
Ensemble-Avg-GB5-R5	90.75%	91.29%	91.02%	90.85%	95.46%	31/29
Ensemble-Max-GB-All	97.67%	50.78%	66.82%	82.44%	91.52%	7/284
Ensemble-Avg-GB-All	96.33%	86.27%	91.02%	94.14%	97.67%	14/27

Table 4.6: Evaluating the predictions made by the ensemble of classifiers at the 0.5 threshold

Table 4.6 and Figure 4.17 show the results of both approaches with the GB5-R5 and the GB-All test set. While the averaging ensemble only offers a slight reduction in false negatives, a clear improvement is seen with the max-value ensemble. This comes at the cost of a higher number of false positives, which translated to lowered precision, F2 Score, and PR AUC. However, since the aim is to have as few false negatives as possible, choosing the max-value prediction is the

preferable approach. While this leaves us with a higher number of false positives, these are easily rejected when the images are verified. The chosen ensemble achieves a recall of 94.33% with GB5-R5 and 97.67% with GB-All, once again generalizing well when applied to unseen data. A further look at the PR curve (Figure 4.18) shows us how the max-value ensemble leans into prioritising the least number of false negatives at the expense of the false positives. For completeness, the ROC curve is also included (Figure 4.19), highlighting the distribution of values favouring a true high positive rate (recall) for the max-value ensemble. The ROC curve is generated by plotting the true positive rate (recall, y-axis) against the false positive rate ($FP/(TN + FP)$) at difference thresholds.

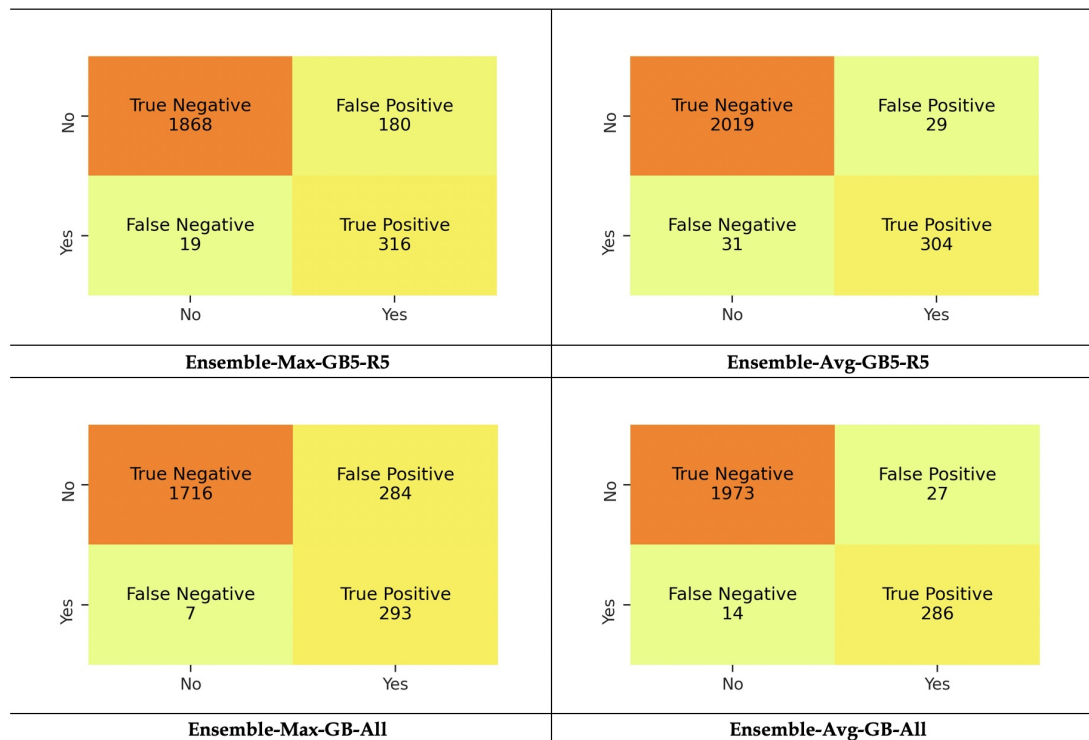


Figure 4.17: Confusion matrix of the predictions made by the ensemble of classifiers at the 0.5 threshold

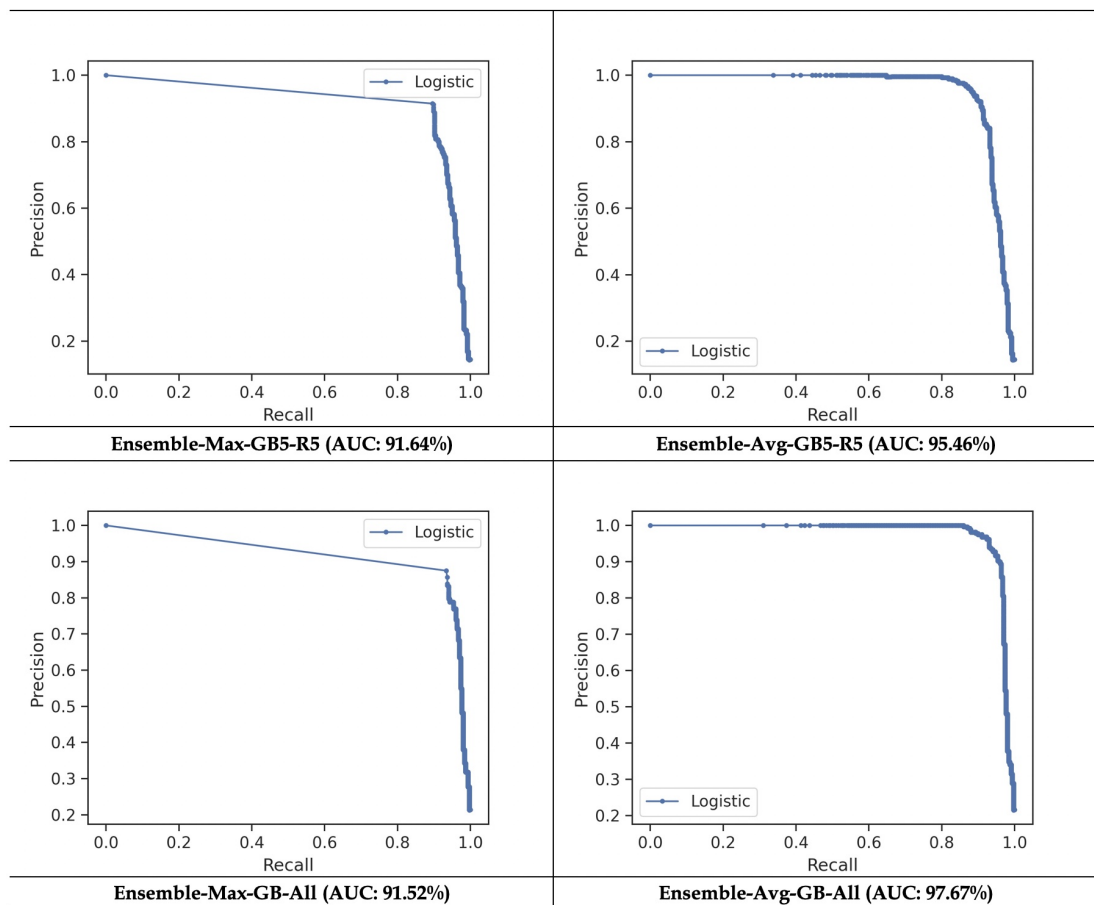


Figure 4.18: PR curve for the ensemble of classifiers

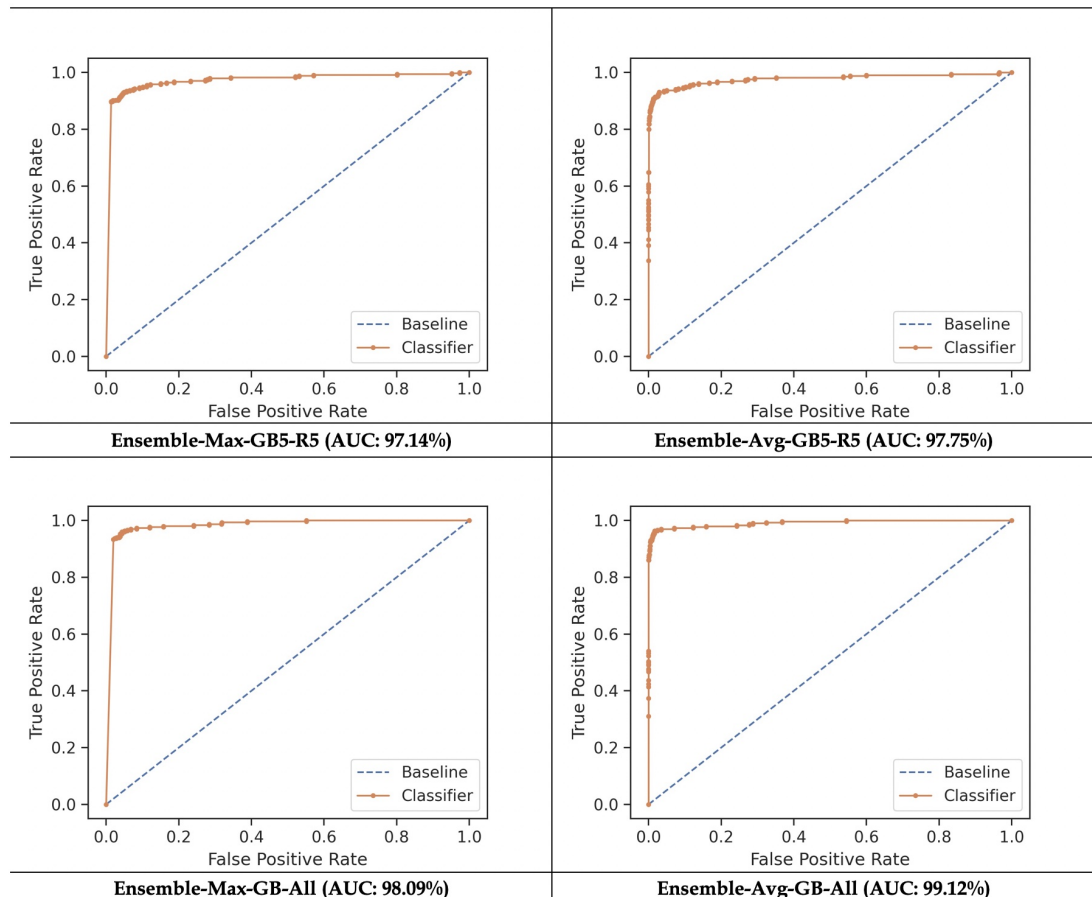


Figure 4.19: ROC curve for the ensemble of classifiers

4.5 Discussion

The promise of deep learning is to automate those tasks that require a degree of intuition. Here, it was demonstrated that it is possible to train deep convolutional neural networks with a relatively small number of parameters to discover tracklet-like patterns - essentially a series of uniformly spaced blobs - in a set of noisy images. The ensemble of five CNN-based classifiers, trained with a relatively small amount of labelled data from just one chip in one field, has a recall of 97.67% when classifying a set of images from other fields and chips, highlighting that it generalizes well to unseen data. The next steps for this research are to use the classifier ensemble to find tracklets in the remaining vast store of MOA-II difference images.

This research is the first work of its kind in the field and as such provides both a launch pad for future work as well as a template upon which to improve. One avenue for further research would be to denoise the images without losing the integrity of the tracklets themselves. This could either mean effectively denoising the difference images or to focus the denoising effort on the stacked images. Any reduction in the noise would lead to an immediate improvement in the predictions. Rejecting poor observations when creating the stack would also improve image quality.

Experimenting with different ways of generating the image tiles may also lead to better results, especially for tracklets on the edges. At the moment, tracklets at the edges could potentially have part of the pixel blob representing an asteroid's position end up in an adjacent tile. There are only a handful of instances of this in the current dataset, but that is only a small fraction of the total available data.

Training the network with more than one input image may also boost performance by giving the network an alternative view to consider. The secondary image could be the reference image or the corresponding median pixel stack. Both will be free of the distinctive pattern of blobs that make up a tracklet, offering a good counterpoint.

Further investigation with other network architectures like DenseNet (Huang et al. (2016)) and the newer scalable architectures like EfficientNet (Tan and Le

(2019)) may also lead to interesting results, especially when more labelled data is available. Adding CSPNet (Wang et al. (2020)) like connections to existing networks could further boost performance.

Gathering more training data will also be necessary to make the network more robust. For this research, using the trained classifier together with the known asteroid data is the preferred approach to generating more labelled data. To incorporate data from a different survey, the recommended approach is hosting a citizen science project on Zooniverse¹ to separate the line clipping images into those that have tracklets and those that do not. The added benefit of crowdsourcing is the outreach opportunity and raising the profile of the research. The same techniques could also potentially be used to discover the small worlds in the farther reaches of our solar system.

¹<https://www.zooniverse.org/>

Chapter 5

Tracklet localization with YOLO

Stacking a night’s observations gives us distinctive tracklets for asteroids moving in the field of view of the telescope. These stacked images can, however, be very noisy, causing the tracklet to be obscured. Here, we investigate the application of the YOLO object detection architecture for localising asteroid tracklets in the MOA difference images.

5.1 Background

In recent years, CNNs have been taken beyond classifying images and have been applied to object detection and localization in images. While there are several models proposed and used for this, they are broadly split into two categories: region proposal based detectors and single shot detectors.

The general idea behind region proposal based detectors is that the image is divided into several “regions of interest” or ROIs. Each pixel starts of in its own group, then the texture of each group is calculated, and the two closest are combined into the same group. This process continues until all pixels are assigned to groups and several (predefined number) of regions are recognised. Each region is treated as a separate image and passed to a CNN to classify into various classes. Once each region has an associated class score, the image is put back together with the detected objects highlighted, together with their

class probabilities. Examples of this type of detector are R-CNN (Girshick et al. (2014)), Faster R-CNN (Ren et al. (2017)), and R-FCN (Dai et al. (2016)).

While region proposal networks have been very successful, the models are complex and computationally expensive, which is why the more streamlined approach of single shot detectors was considered. These detectors question the need to have region proposals and make their predictions in a single step by using the feature maps produced by the CNN. Several convolutional filters are applied to the output feature map to predict bounding boxes and class probabilities. These detectors consider the whole image but can tend towards trading accuracy for speed, especially with small objects. But the gap between the two types of detectors is narrowing thanks to the single-shot detectors adopting more complex CNNs for feature extraction. Examples of this type of detector are SSD (Liu et al. (2016)), YOLO (Redmon et al. (2016)), and RetinaNet (Lin et al. (2017b)). YOLO is the focus of this research and we will discuss the earlier versions of the YOLO family of models next.

5.1.1 YOLOv1

YOLO (You Only Look Once) is an object detection algorithm that extends a convolutional neural network (CNN) to both locate and classify the desired object in an image. The algorithm re-frames object detection as a regression problem and presents a single network that, given an image, will output bounding box coordinates and class probability (Redmon et al. (2016)). In the first version of YOLO (YOLOv1), the CNN at its heart has 24 convolutional layers for feature extraction, followed by two fully connected layers for refining predictions. The network model is based on GoogLeNet (Szegedy et al. (2015a)), though it uses 1x1 dimension reduction convolutions together with 3x3 convolutions rather than the inception module. The network divides each image into N $S \times S$ dimension grids and each grid cell predicts B bounding boxes along with a confidence score for the boxes. The confidence score represents the likelihood that the cell contains the object as well as how confident the model is about its predictions. Broadly, confidence is defined as $Pr(Object) * IOU_{pred}^{truth}$. We expect a confidence of zero from all cells that do not contain the object and the IOU in all other cases.

Intersection over union (IOU) is a fundamental evaluation metric for object detectors. To determine this metric, we need both the ground truth and the predicted bounding box (Figure 5.1). We now calculate IOU as,

$$IOU = \frac{\text{Area of overlap} \quad \square}{\text{Area of union} \quad \blacksquare}$$

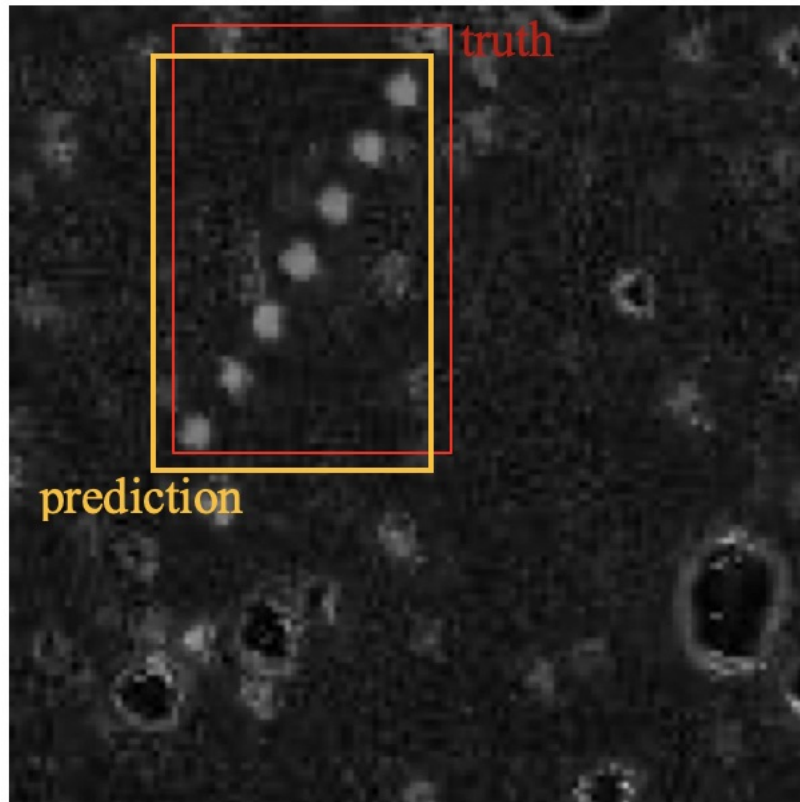


Figure 5.1: Tracklet image with both the ground truth and the predicted bounding box

Each predicted bounding box consists of five elements: centre-x, centre-y, width, height, and confidence. The $(centre-x, centre-y)$ coordinates are relative to the dimensions of the grid cell; the width and height are relative to the whole image. Additionally, each grid cell also predicts C conditional class probabilities. The cell that the midpoint of the object of interest falls in is responsible for correctly predicting its presence. Hence, each cell attempts to predict the centre of one object, limiting each cell to one class prediction. As most object detection

tasks involve finding multiple objects that might be in the same cell, this is a drawback of YOLOv1. The output tensor is of shape $(S \times S) \times (B * 5 + C)$. If $S = 7$, $B = 1$, and $C = 1$, the final prediction will be of shape $7 \times 7 \times 6$. Since multiple bounding boxes are expected, only the ones above a set confidence and IOU threshold are selected.

The relative simplicity of YOLO's network architecture and prediction model is in direct contrast to the loss function (Figure 5.2) developed to anchor the framework. The loss function minimises the sum-squared error between the ground truth and predicted boxes and is composed of three parts:

- Localization loss - calculates the errors in the predicted bounding box. $\mathbb{1}_{ij}^{obj}$ is 1 if the j th bounding box in the i th cell is responsible for making the prediction; 0 otherwise. λ_{coord} is a weighting parameter that increases the loss on the box prediction, thus focusing the training here.
- Confidence loss - as there are several box predictions that do not contain any objects, it is desirable to ensure that they do not overwhelm the gradient calculations. Hence, the loss on low confidence predictions is down weighted (λ_{noobj}). \hat{C}_i is the box confidence score. $\mathbb{1}_{ij}^{noobj}$ is the opposite of $\mathbb{1}_{ij}^{obj}$.
- Classification loss - if a box is present, sum-squared error is used to calculate the error in the conditional class probabilities predicted. $\mathbb{1}_i^{obj}$ is 1 if an object is present in a cell i ; 0 otherwise. $\hat{p}_i(c)$ is the conditional probability of class c in cell i .

The loss function is the key to learning good representations and YOLO's complex and carefully crafted loss function allows the application of a simpler network model that boosts the speed of the network.

YOLOv1 demonstrated object detection in real-time at 45 frames per second (fps) with a mean average precision (mAP) of 63.5% on the PASCAL VOC 2007 dataset (Everingham et al. (2009)), which has 20 classes. The paper also introduced a much smaller version of the detector with only 9 convolutional layers dubbed Fast YOLO because of its impressive 155 fps processing speed, though the mAP was lower at 52.5%, with the same dataset. mAP will be further discussed in Section 5.5.

$$\begin{aligned}
& \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
& + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \\
& \quad + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} (C_i - \hat{C}_i)^2 \\
& \quad + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} (C_i - \hat{C}_i)^2 \\
& \quad + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}$$

Localization loss
Confidence loss
Classification loss

Figure 5.2: YOLO’s loss function (Redmon et al. (2016))

5.1.2 YOLOv2

While YOLOv1 was an excellent start for one-stage detectors, its limitations with generalization, finding small objects, or multiple objects per cell were ripe for further work. YOLOv2 (Redmon and Farhadi (2017)) introduced a raft of enhancements, the most significant of which was the introduction of anchor boxes. Anchor boxes are a set of predefined bounding boxes that are representative of the dimensions of bounding boxes in the dataset. Thus, during training, these anchor boxes are adjusted and refined to match the objects in the image. While the use of anchor boxes reduces the mAP slightly to 62.2%, the recall jumps from 81% to 88%. Rather than use hand-engineered anchor boxes, the researchers applied the K-means clustering algorithm to the training set to choose the top K most common boxes. Since it is desirable to have anchors that results in good IOU scores, the distance metric used to compute the clusters is: $d(\text{box}, \text{centroid}) = 1 - \text{IOU}(\text{box}, \text{centroid})$. Each grid cell now predicts five bounding boxes each and the predictions are scaled with respect to the anchor boxes based on the location (centroid) of prediction. YOLOv2 also updated the network architecture from having 24 convolutional layers to 19 and the fully connected layers were replaced with global average pooling. These enhancements, together with the addition of batch normalization as well as training at a higher resolution, result in YOLOv2 achieving an mAP of 76.8% at 67 FPS on the PASCAL VOC 2007 data.

5.1.3 YOLOv3

The advent of Residual Networks (ResNets)(He et al. (2016)) and the Feature Pyramid Network (FPN)(Lin et al. (2017a)) led to increased accuracy for object detection networks. Thus, YOLO evolved further to incorporate these improvements and the result was YOLOv3 (Redmon and Farhadi (2018)). YOLOv3's backbone network was reworked to be significantly larger with 53 convolutional layers and ResNet's skip connections, makes bounding box predictions at 3 different scales similar to FPN. These changes led to increased accuracy with detecting smaller objects, with an mAP of 57.9% with MS COCO dataset (Lin et al. (2014)), which has 80 categories.

The trademarks for YOLOv2 and YOLOv3 are currently owned by Apple¹. In Section 5.3 we will discuss YOLOv4, the current installment that will be used for detecting asteroid tracklets. Ahead of that, we discuss how a suitable dataset was created.

¹<https://bit.ly/3yyQudr>

5.2 Data

Chapter 3 describes the process of creating a dataset of stacked differences images with and without tracklets. This resulted in 4153 128 x 128 images with visible tracklets and 547,783 without known visible tracklets from chip 5 of the field GB5. Thus far, all the tracklet images we have seen have been for well defined, clearly visible tracklets. However, there are also several images where the tracklet is faint and/or in a noisy background. Some of these images can be seen in Figure 5.3.

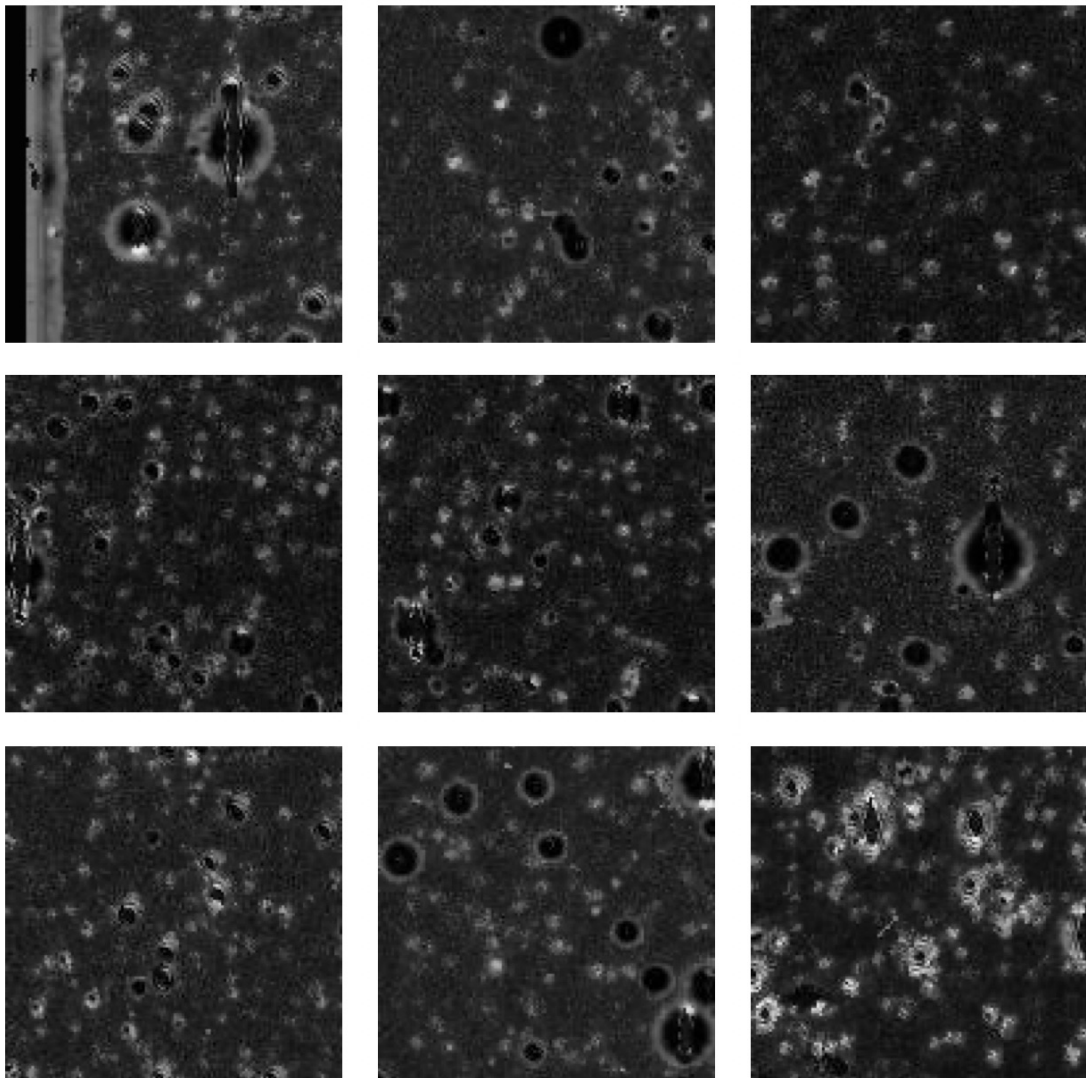


Figure 5.3: Images with faint tracklets in a noisy field

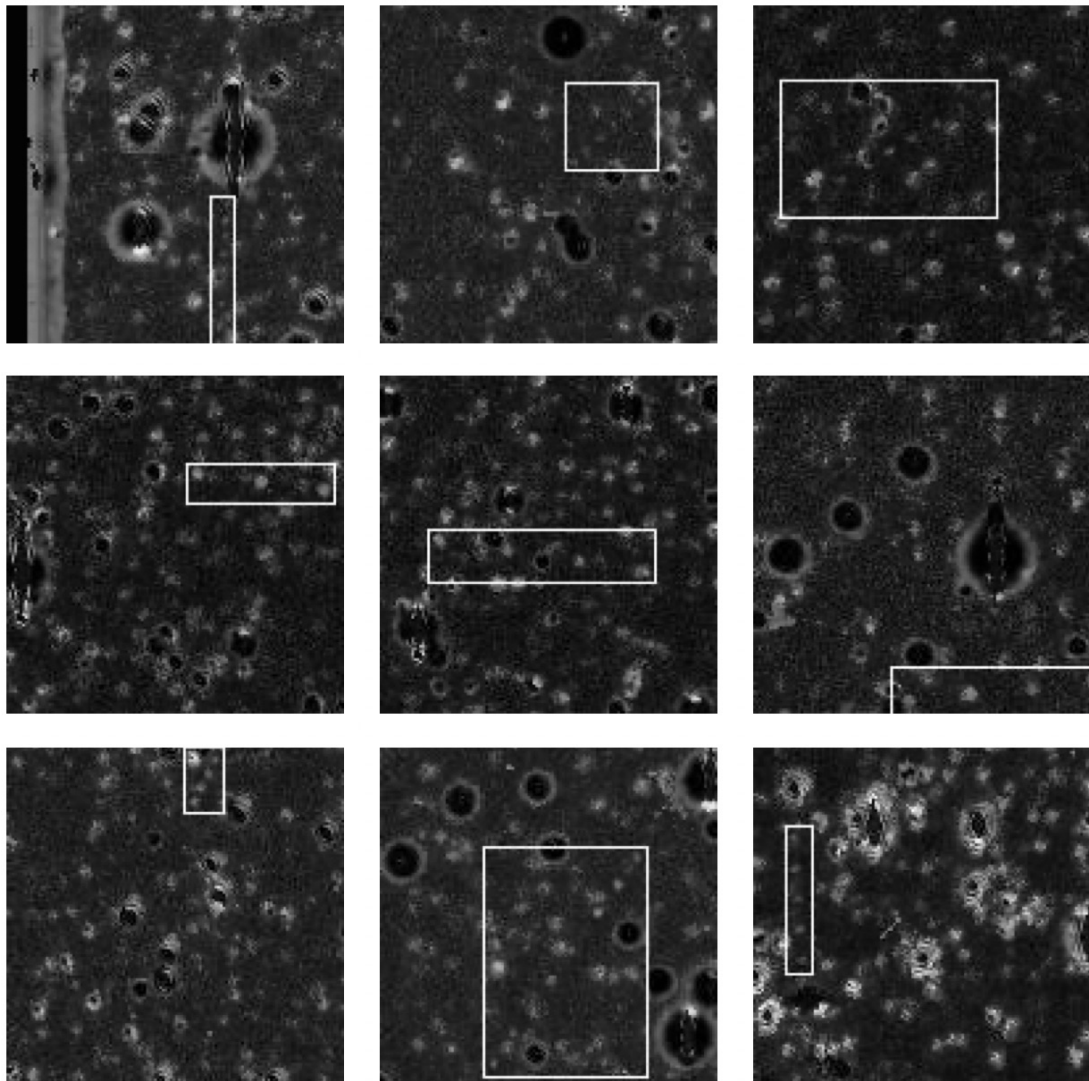


Figure 5.4: Images from Figure 5.3 with tracklets of known asteroids highlighted with bounding boxes

The standard size of a difference image stack is 2048 x 4096 pixels and the full tracklets of known asteroids come in a variety of sizes, either due to the seeing conditions, the number of nightly observations, or when an asteroid enters the field of view of the telescope. In order to better suit using this data as input for a neural network, each 2048 x 4096 image was split into 512 128 x 128 images. The next step was to find all the images with tracklets of known asteroids. Section 3.2.3 describes applying the Cohen-Sutherland line clipping algorithm to find the relevant images along with the probable intersection/end points of the tracklets within a 128 x 128 image. These intersection points were used to extrapolate bounding boxes for tracklets, thus highlighting them in the images. Further visual inspection was undertaken to ensure that bounding boxes encapsulated the tracklets correctly without any superfluous background included. In Figure 5.4 we can see these bounding boxes in action as they highlight the faint tracklets from Figure 5.3. The bounding box coordinates for each tracklet in each images is saved in a corresponding file, with one data file per image.

Next, the bounding box data was converted to the format required by the YOLO algorithm. Rather than absolute values for the coordinates, YOLO expects the data to be floating point values representing the relative position of the box with respect to the width and height of the image. One such file with extension .txt is expected for each image with a tracklet, with multiple tracklets in the same image on separate lines. The YOLO format is as follows:

$$\langle \textit{object - class} \rangle \langle \textit{x - center} \rangle \langle \textit{y - center} \rangle \langle \textit{width} \rangle \langle \textit{height} \rangle$$

Where:

- $\langle \textit{object - class} \rangle$ is an integer representing the object class. As we only have one class - minor planet - this value is always 0. A separate obj.names file is maintained that contains a single class “mp”, which stands for “minor planet”.
- $\langle \textit{x - centre} \rangle \langle \textit{y - centre} \rangle$ are the centre of the bounding box with respect to the height and width of the image. Values are expected to be between 0.0 and 1.0.

- $\langle width \rangle \langle height \rangle$ are the width and height of the bounding box with respect to the width and height of the image. Values are expected to be between 0.0 and 1.0.

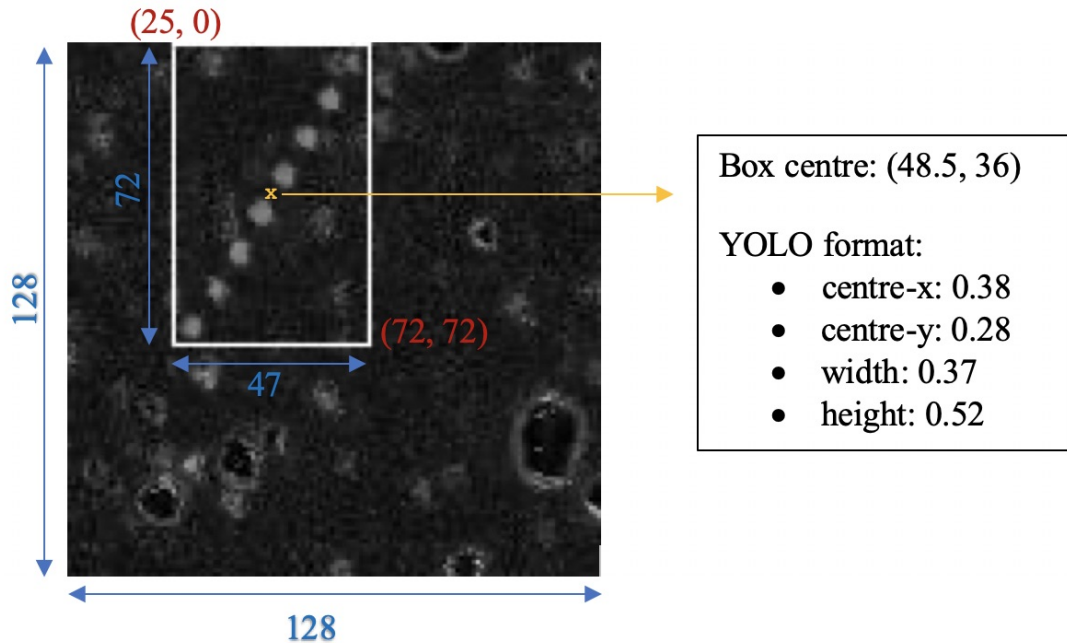


Figure 5.5: Breaking down the YOLO data format

While the bounding box data is enough for training YOLO, including images with no tracklets could serve to make the algorithm better at identifying tracklets. Thus, 8306 images without tracklets - twice the amount of tracklets images - were added to the YOLO training dataset. As they have no tracklets, there is an empty .txt file included with the no-tracklet images.

Additionally, we also included horizontally flipped, vertically flipped, and rotated the augments for the tracklet images and updated the bounding box data accordingly.

Thus, we have 16612 images with tracklets with a further 16612 .txt files with location details of the tracklets there in as well as 8306 images with no tracklets that will be used to train the YOLO algorithm to localise tracklets. Further training details are in Section 5.4. Next we look at the architecture of the YOLO object detection model that was trained for localizing asteroid tracklets with this dataset.

5.3 YOLOv4

The foundational premise of YOLO was for object detection to be fast - both to train and for inference. With ever evolving research about the best methodologies to apply, YOLO underwent another evolutionary change to adapt state-of-the-art best practices for improved performance. YOLOv4 (Bochkovskiy et al. (2020)) is the result of testing a variety of enhancements and choosing the best fit for object detection.

Figure 5.6 illustrates the architecture of YOLOv4. The convolutional backbone feature extractor for the architecture is composed of a 53 layer DenseNet (Huang et al. (2016)) with the cross-stage-partial (CSP) connections of CSPNet (Wang et al. (2020)). DenseNets extend ResNet’s concept of skip connections by adding connections between all the layers in the network in a feed-forwards fashion. Feature maps from all preceding layers are concatenated and form the input for any given layer, ensuring that low-level features are propagated through the network. A DenseNet consists of multiple dense blocks (Figure 5.7) with a transitional convolutional (1x1) and pooling (2x2) layer between each block. CSP connections involve splitting the input feature map into two parts, one of which goes through the dense block and the other goes straight through to the next transitional step (Figure 5.8). Additionally, the network includes spatial pyramid pooling (SPP) (He et al. (2014)) after the last convolutional layer. This has the effect of separating out the most important features and increasing the receptive field. The final feature map is divided into $m \times m$ bins, following which maxpool is applied to each bin. The resulting feature maps are concatenated and represent the output of the feature extractor.

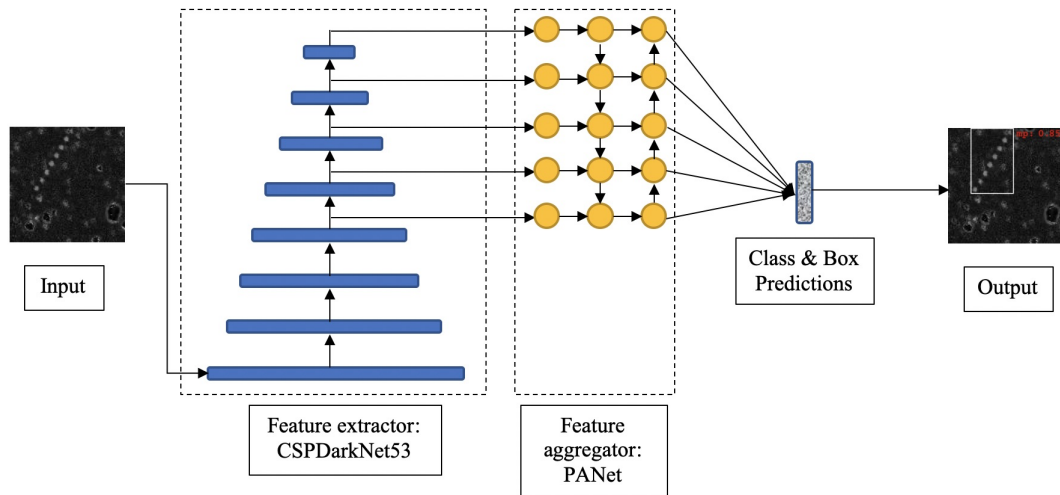


Figure 5.6: Architecture of YOLOv4

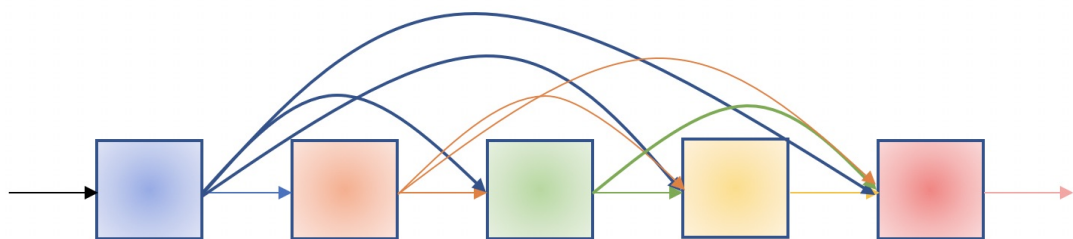


Figure 5.7: A single block in a DenseNet where the input for a layer is the feature maps of all preceding layers.

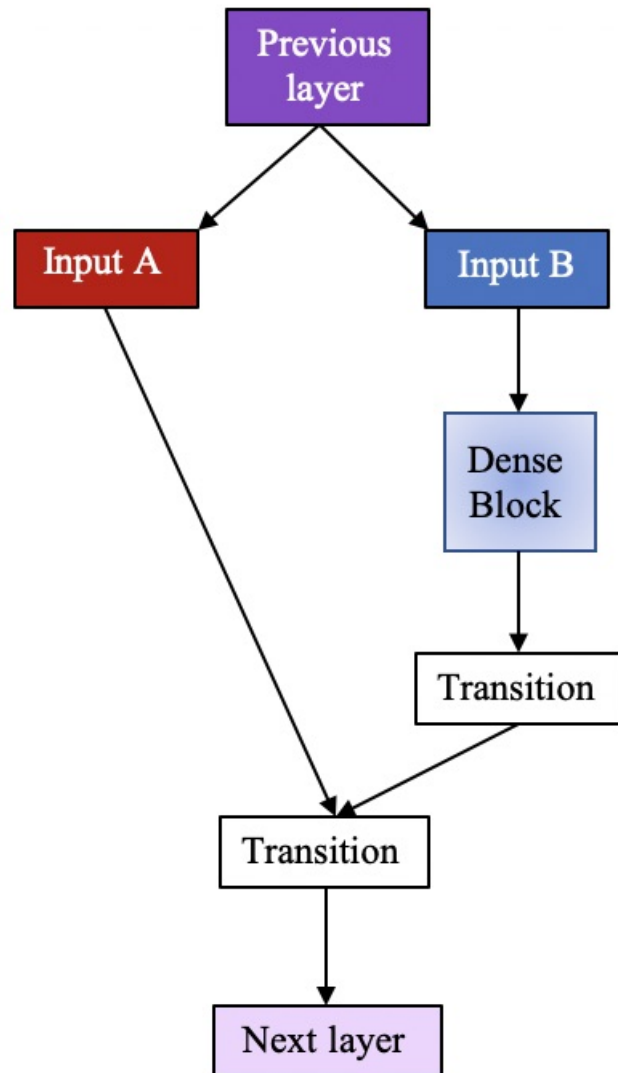


Figure 5.8: Cross stage partial connection

CNNs naturally attain a pyramid-like structure with each layer as the image goes from high to low resolution. As we get deeper in a CNN, we lose the fine-grained details of the input, which usually makes it harder to detect small objects. As the resolution lowers, however, the filters learn ever more complex abstractions about the image, making the feature maps more semantically rich. Thus, is it desirable to combine the feature maps from the higher resolution layer with the more semantically rich ones to facilitate detecting objects at multiple scales. This task falls to a feature aggregator and YOLOv4 uses the approach suggested by PANet (Liu et al. (2018)). The feature maps are concatenated from both the top-down and the bottom-up path, ensuring the propagation of semantically rich localization information through to the final part of the network where the class and bounding box predictions are made. Predictions are made and output in the same form as seen in preceding models of YOLO, by predicting the centroid and location information with respect to the grid cell.

Finally, YOLOv4 also updates the loss function used to train the network. While it is still composed of three parts, each has been updated to improve performance. The confidence and classification loss now calculate binary cross entropy rather than sun-squared error. The localization loss had been overhauled to implement Complete Intersection over Union (CIoU) loss (Zheng et al. (2020)). In previous versions of YOLO, the loss function was ineffective at determining the direction in which to shift the weights when there was no overlap between the predicted and the ground truth box. Similarly, when there is more than one predicted box, the loss calculation should be able to determine which one is closer to the ground truth. CIoU loss addresses both these scenarios and the localization loss is now calculated as:

$$Loss_{CIoU} = 1 - IOU + \frac{p^2(b, b^{gt})}{c^2} + \alpha v,$$

where:

- $p^2(b, b^{gt})$ is the Euclidean distance between the centroid of the predicted and ground truth box;
- c is the diagonal distance in the smallest closed-space representing the union of the predicted and ground truth boxes;

- α is a positive trade-off parameter that facilitates increasing the overlapping area for non-overlapping boxes and is calculated as:

$$\alpha = \frac{v}{(1 - IOU) + v}$$

- v maintains the consistency of the aspect ratio of the boxes and is calculated as:

$$v = \frac{4}{\pi^2} \left(\arctan \frac{w^{gt}}{h^{gt}} - \arctan \frac{w}{h} \right)^2$$

The model also includes a raft of new additions: a new activation function, MISH (Misra (2020)) that provides smoother gradients; updates to the spatial attention module (Woo et al. (2018)) and multi-input weighted residual connections (Tan et al. (2020)) to better suit the architecture; new data augmentation techniques, Mosaic and self-adversarial training. The updated architecture lead to YOLOv4 achieving state-of-the-art results with 65.7% mAP for the MS COCO dataset at a real-time speed of 65 FPS on Tesla V100.

In addition, a compressed model architecture, YOLOv4-tiny (Wang et al. (2021)) with only 29 convolutional layers, was proposed to support object detection on low-end GPU machines and mobile devices. While not as accurate as the larger model (42.0% mAP), it is nonetheless very fast (443 FPS on RTX2080Ti) and potentially makes object detection a viable option for more research projects.

YOLOv4 is currently being used by the Taiwanese Government for traffic management², BMW³, and Amazon⁴. Next, we train both YOLOv4 and YOLOv4-tiny to detect asteroid tracklets in the MOA dataset.

²<https://www.taiwannews.com.tw/en/news/3957400>

³<https://github.com/BMW-InnovationLab/BMW-YOLOv4-Training-Automation>

⁴<https://github.com/amzn/distance-assistant>

5.4 Training

The YOLO family of models are supported by Darknet⁵, a custom framework written in C and CUDA and designed for fast object detection. We can access and train the Darknet implementations⁶ of YOLOv4 and YOLOv4-tiny with the MOA-II dataset via the Google Colaboratory with a hosted GPU runtime environment. YOLOv4 and YOLOv4-tiny were trained with both unaugmented tracklets-only data (SetA) as well as data containing images with and without tracklets, including augments for the tracklet images (SetB). The complete object detection dataset contains: 4153 unaugmented images with tracklets; 11211 augmented (rotated, flipped horizontally, and flipped vertically) tracklet images and; 8306 images with no tracklets.

The data is split into a training (90%) and test (10%) dataset. SetA contains 3737 and 416 of the unaugmented tracklet-only images in the training and test set, respectively. SetB contains 22422 and 2494 original and augmented images with and without tracklets in the training and test set, respectively. The images were resized to 416 x 416 by the detector to facilitate better object detection.

The model was first trained with the default anchor boxes, before k-means clustering was used to discover anchor boxes that were a better fit for the MOA-II dataset. After several variations were tested, the best combination of anchor boxes was discovered by hand-engineering the various clusters. The batch size was set to 32, with 8 subdivisions, 6000 max_batches, and with steps set to 4800,5400. The learning rate of 0.001 was used for YOLOv4 and 0.00261 for YOLOv4-tiny.

Increasing the size of the training dataset had negligible impact on the time it took to train the model. Training YOLOv4-tiny with both SetA and SetB took around 1.5 hours to train on a Tesla K80 (12GB, 4.1TFLOPS) and 45 minutes with a Tesla T4 (16GB, 8.1 TFLOPS) in the Google Colab environment. Training YOLOv4 with SetA took around 12 hours to train on a Tesla K80 and 6 hours with a Tesla T4 in the Google Colab environment. Training YOLOv4 with SetB was unsuccessful as it exceeds the memory capacity of the GPU (Tesla T4).

⁵<https://github.com/pjreddie/darknet>

⁶<https://github.com/AlexeyAB/darknet>

5.5 Results

The preferred metric for quantifying the performance of object detection models is mean Average Precision or mAP. The purpose of an object detector is to localize and classify a predefined set of objects in input images. The expected output is bounding boxes coordinates, class predictions, and the model's confidence in the predictions for the objects present therein. To measure the efficacy of a model, it is desirable to have a metric to quantify how the model performs across all of the images in a test set, which is what mAP gives us.

The evaluation metric *precision* measures how many of the predictions are correct and the metric *recall* measures how good the model is at finding the positive predictions. The classification models, for example, aim to minimise false negatives and thus high recall is given preference. Object detection models aim to have a high overlap between the predicted and the ground truth bounding boxes (IoU). It is expected that the predicted box will not exactly match the ground truth box. In general terms, predictions are grouped as follows:

- True Positive: $\text{IoU} > 0.5$
- False Positive: $\text{IoU} < 0.5$ (or a duplicate)
- False Negative: box not detected or the $\text{IoU} > 0.5$ but the object is classified wrong

Precision and recall are calculated as:

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

The average precision (AP) measures the trade-off between precision and recall and is calculated by integrating the area that falls under the precision-recall curve for each unique values of recall where the precision value decreases. Calculated as:

$$AP = \Sigma(r_{(n+1)} - r_n)\tilde{p}(r_{(n+1)})$$
$$\tilde{p}(r_{(n+1)}) = \max_{\tilde{r} \geq r_{(n+1)}} (p(\tilde{r})),$$

where r is the recall value, n represents the locations where the precision decreases, and $\tilde{p}(r_{(n+1)})$ is the maximum precision where the r value changes. The mAP is the mean of the AP across all object classes that the model can detect, which is identical to the AP for this dataset.

Table 5.1 details the mAP achieved by using both SetA and SetB to train YOLOv4 and YOLOv4-tiny with default as well as custom anchor boxes. The confidence threshold of 25% is used for determining the mAP. The default anchor boxes proved resilient to the task of detecting asteroid tracklets and performed fractionally better than the custom boxes. The addition of augmented data as well as no-tracklet data had the effect of lowering the overall performance of the model. When only no-tracklet data was added, the mAP fell by 10% even with the default anchor boxes. The best performing model was the YOLOv4 trained with SetA and with default anchor boxes with an mAP of 90.96%. YOLOv4 with SetA and custom anchor boxes was a very close second with an mAP of 90.95%.

ID	Model – Dataset - Anchors	mAP @ 0.5
(a)	YOLOv4-tiny - SetA - defaults	64.53%
(b)	YOLOv4-tiny - SetA - custom	65%
(c)	YOLOv4-tiny - SetB - defaults	60.33%
(d)	YOLOv4-tiny - SetB - custom	52.01%
(e)	YOLOv4 - SetA - defaults	90.96%
(f)	YOLOv4 - SetA - custom	90.95%

Table 5.1: mAP achieved when training YOLOv4 and YOLOv4-tiny to detect asteroid tracklets with both SetA and SetB while applying both default and custom anchor boxes

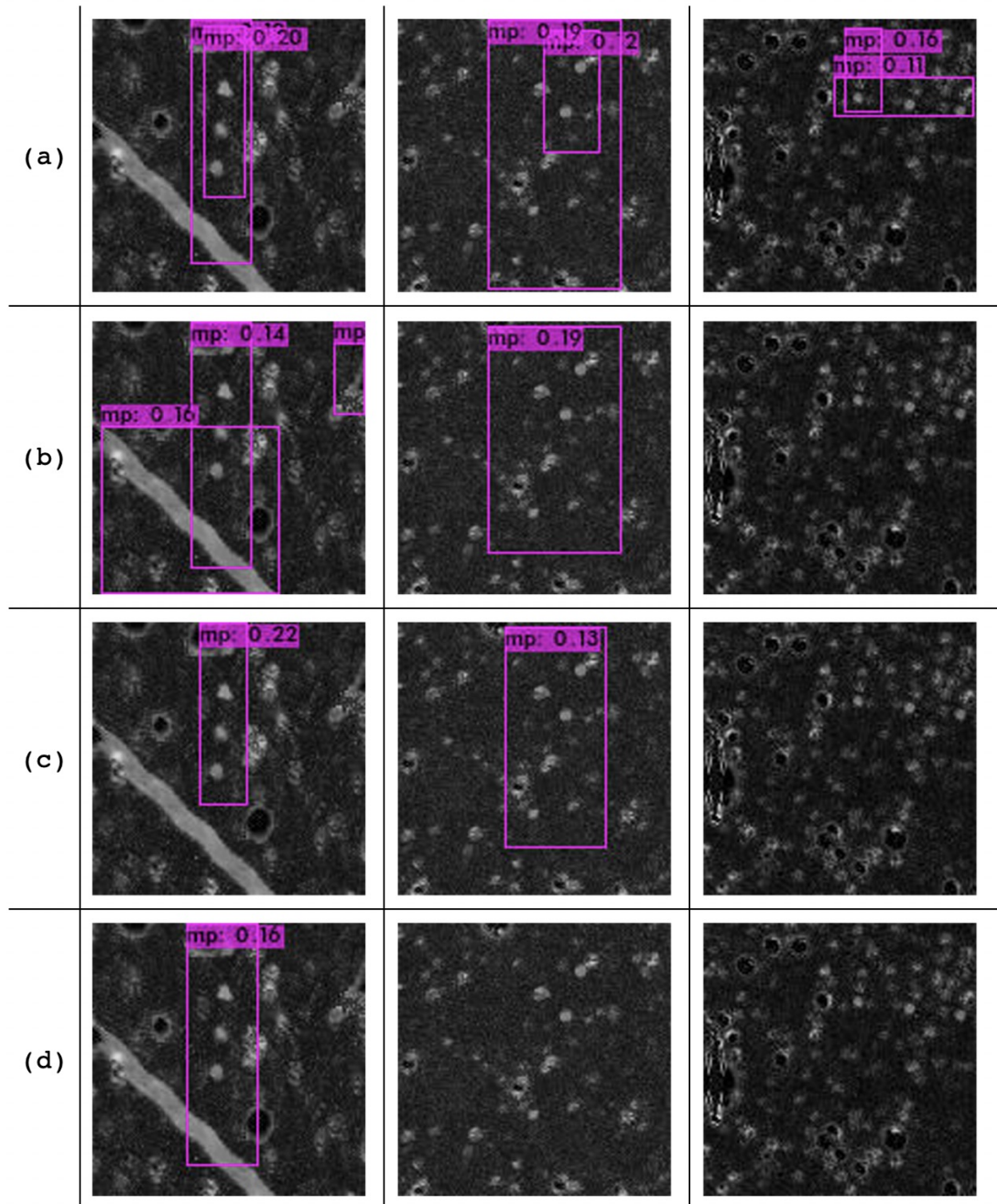


Figure 5.9: Predictions made with the trained YOLOv4-tiny

Some of the predictions made by the trained YOLOv4-tiny can be seen in Figure 5.9 and those by trained YOLOv4 can be seen in Figure 5.10. The confidence threshold for the predictions was set to 10%. YOLOv4-tiny had an inference speed of 15 milliseconds and YOLOv4 had an inference speed of 33 milliseconds. In general, while YOLOv4-tiny was good at finding the tracklets that were clearly defined, it had lower confidence with predictions and, in some case, could not discover the hard-to-find tracklets. Nevertheless, YOLOv4-tiny with the default anchor boxes is a good option for training on low-end GPUs with a small amount of labelled data. More predictions can be seen in Appendix D. The best option for discovering asteroid tracklets with a high confidence, including for hard-to-find tracklets, is YOLOv4 trained with SetA.

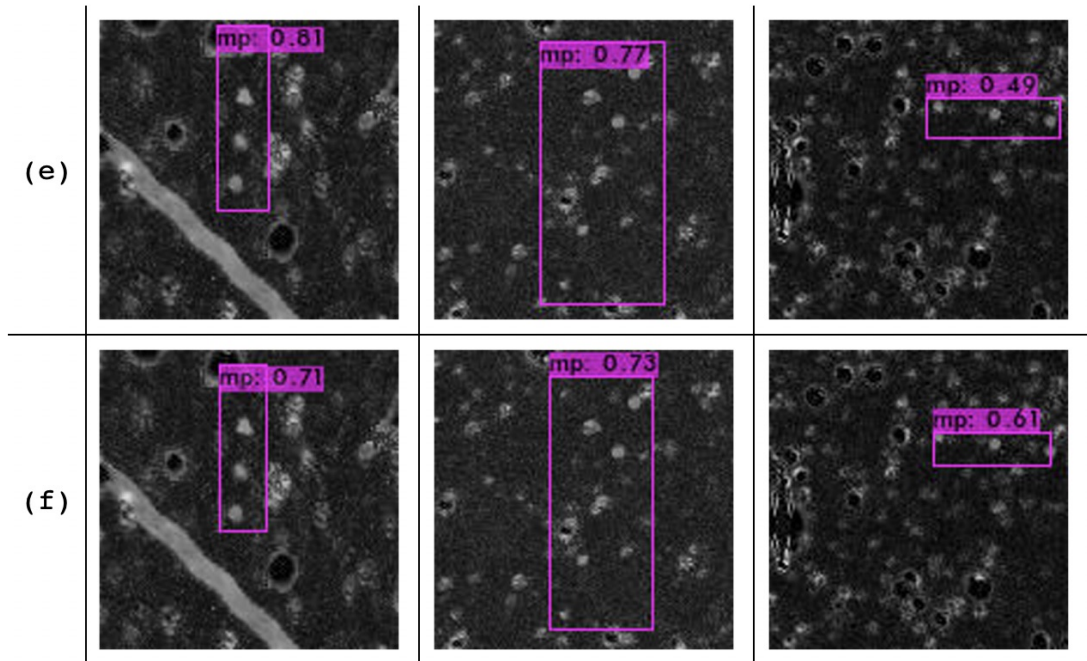


Figure 5.10: Predictions made with the trained YOLOv4

5.6 Discussion

We have shown that with a relatively small amount of data, it is possible to train a YOLOv4 model to localise asteroid tracklets in the GB5-R5 dataset. The best performing model had an mAP of 90.96% and a high level of confidence when predicting tracklets. It will be used in conjunction with the classification ensemble (4.4.1) to localize potential tracklets in predictions made with the larger, unexplored MOA-II dataset, further streamlining the detection process. Additionally, we now have a labelled dataset that can be used for future research.

As the first work of its kind in the field, the opportunity for further development abound, starting with training the backbone feature extractor, CSPDarknet53, with the classification data first. Further, it will be interesting to investigate if increasing the number of anchor boxes for both YOLOv4 and YOLOv4-tiny could boost performance. Preliminary tests suggest that the process of choosing anchor boxes is more complex than merely choosing the top k clusters. The custom anchor boxes have achieved mAP (90.95%), which is very closely matched to the performance of the default anchor boxes. Additionally, establishing the performance benchmarks by scaling the model uniformly by resolution, width, and depth (EfficientDet(Tan and Le (2019)) or Scaled-YOLOv4(Wang et al. (2021))) could also provide useful insight into the ideal model architecture for localizing asteroid tracklets.

The biggest challenge going forward is acquiring labelled data with which to train the object detector. Ensuring accurate bounding boxes around tracklets is a labour-intensive process, requiring many hours to build even a small dataset like the one used here. While this research will continue to use the manual approach to create more bounding box data, the best option for a different survey would be to harness crowd-sourcing via a platform like Zooniverse, which allows volunteers to mark the start and end position of a tracklet as we saw in Kruk et al. (2022). This approach has the added benefit of allowing for a predefined number of people to label the same image, which would serve to improve the accuracy of the bounding box. Since this task can also be combined with classifying whether or not an image has a tracklet, it would result in two useful research datasets.

The central tenet of the YOLO family of object detection models is that the networks should be both fast and accurate, thus better at approximating innate human ability. A human expert would take at least a second or two to identify a hard-to-find asteroid tracklet in a stacked and subtracted difference image; this model takes a fraction of a second to identify tracklets. Utilizing YOLOv4 for screening archival and future astronomical data to discover asteroid tracklets is both practicable and beneficial to furthering our knowledge about these denizens of our solar system.

Chapter 6

Tracklet Classification with CNN-LSTM

We have looked at both classification and object localization with static stacked images with and without tracklets. Here, we harness the time dimension of the observational data to see if it can provide the necessary boost to finding tracklets in nightly observations. Rather than using a stack image as the input, we will be using the sequence of images that go into making the stack. In doing so we will also be doing away with the additive noise in the stacked images that could be misclassified as a tracklet. Asteroids can clearly be seen moving through a field from one exposure to the next in observational data - in our case difference images. Our next architecture will attempt to learn this movement in a sequence of observations and apply it to finding asteroids in the MOA-II data.

6.1 Background

A critical facet of learning is understanding the relationship between multiple elements in a sequence and using this to augment our knowledge base. While CNNs are excellent at predictions based on spatial relationships in fixed-length data, they are not designed to deal with sequential input connected by a temporal relationship. The group of networks that specialise in analysing such sequential data are called recurrent neural networks or (RNNs). Though described in the

seminal Rumelhart et al. (1986) paper, it was not until the confluence of processing power with the availability of labelled data in the mid 2000s that they began to raise in popularity and have since been very successfully applied to fields such as natural language processing and action recognition.

In spartan terms, RNNs consist of feedback loops that enable the network to remember previous inputs in a sequence and incorporate these at each step of processing the sequence before presenting its prediction. This can be seen in Figure 6.1. On the left, we see the input x_t passed to an RNN module (N), which gives us the output \hat{y}_t . The loop represents how the input is shared at each step in the sequence, which can be seen in the unrolled network on the right. The t in this case refers to the number of elements in the sequence, corresponding to a single time step for the network. Each time step could be conceptualised as a copy of the network, which receives an element in the sequence, the output of the previous time step, and passes its output on to the next time step. The composition of the repeating module in a standard RNN can be seen in Figure 6.2. The input is both the output of the previous module \hat{y}_{t-1} as well as the next time step x_t . These two inputs are concatenated, passed through a \tanh activation, after which one copy forms the output of the module and another the input to the next module. Note that \tanh forces the values to lie between -1 and 1.

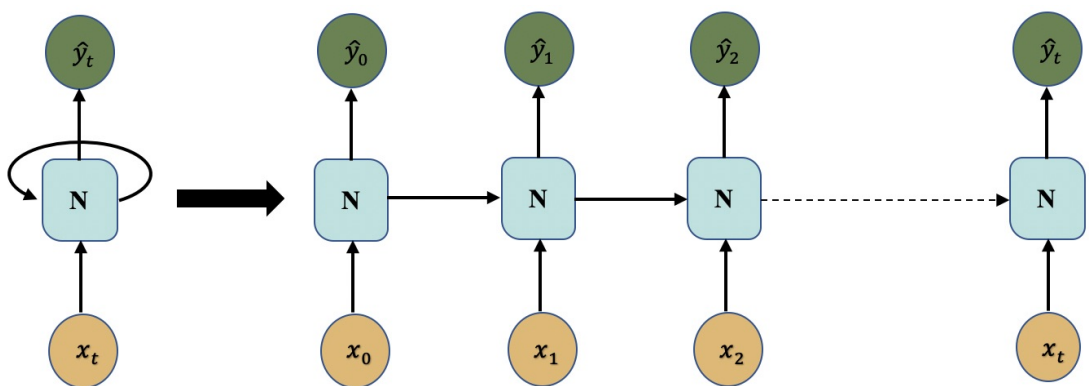


Figure 6.1: A simple uni-directional RNN

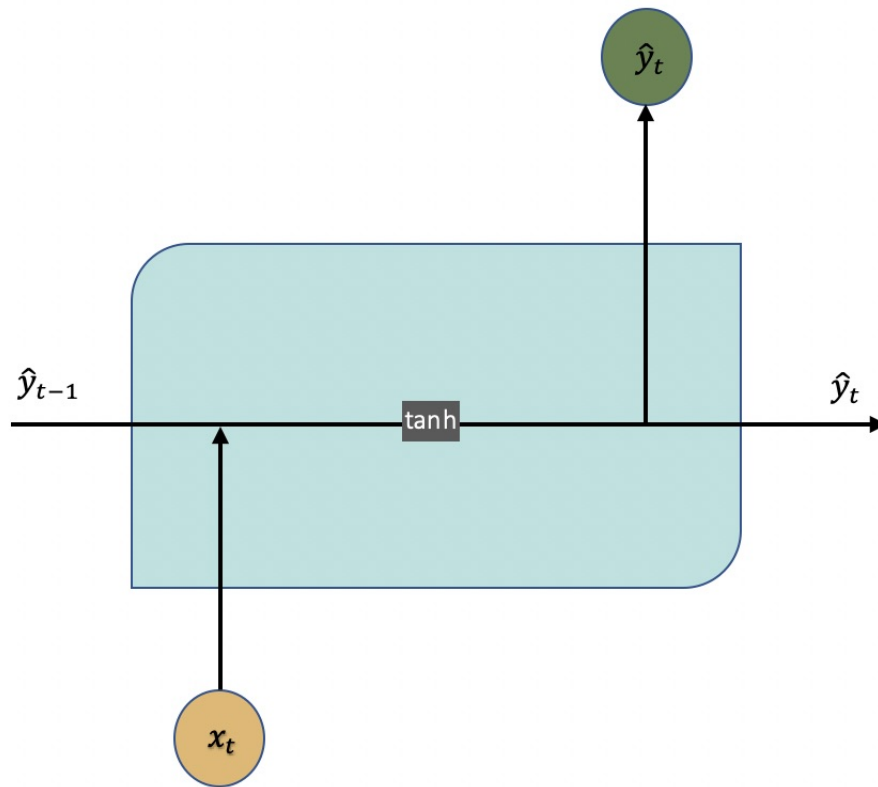


Figure 6.2: Composition of a standard RNN module

Backpropagation is re-framed as Backpropagation Through Time (BPTT) (Williams and Peng (1990)) where errors are calculated at each time step and the collated value of these is used at the end of the sequence to update the weights for all of the time steps. The more time steps in the sequences, the more computationally expensive BPTT is. Much like with its cousin the stacked CNN, the RNN is prone to the vanishing or exploding gradients as it gets deeper in the network, which in turn means it cannot effectively learn long-term dependencies (Bengio et al. (1994)). This is particularly problematic for this research because the network could miss asteroids that only appear early in a long sequence.

The Long Short Term Memory network or LSTM (Hochreiter and Schmidhuber (1997)) is a widely used and highly successful solution to the vanishing/-exploding gradients problem in RNNs. In contrast to the original RNN, they specialise in learning long-term dependencies. The key difference is in the composition of the repeating module. While the standard RNN is composed of a

single activation function layer, LSTMs comprise of much more complex structures, which can be seen in Figure 6.3.

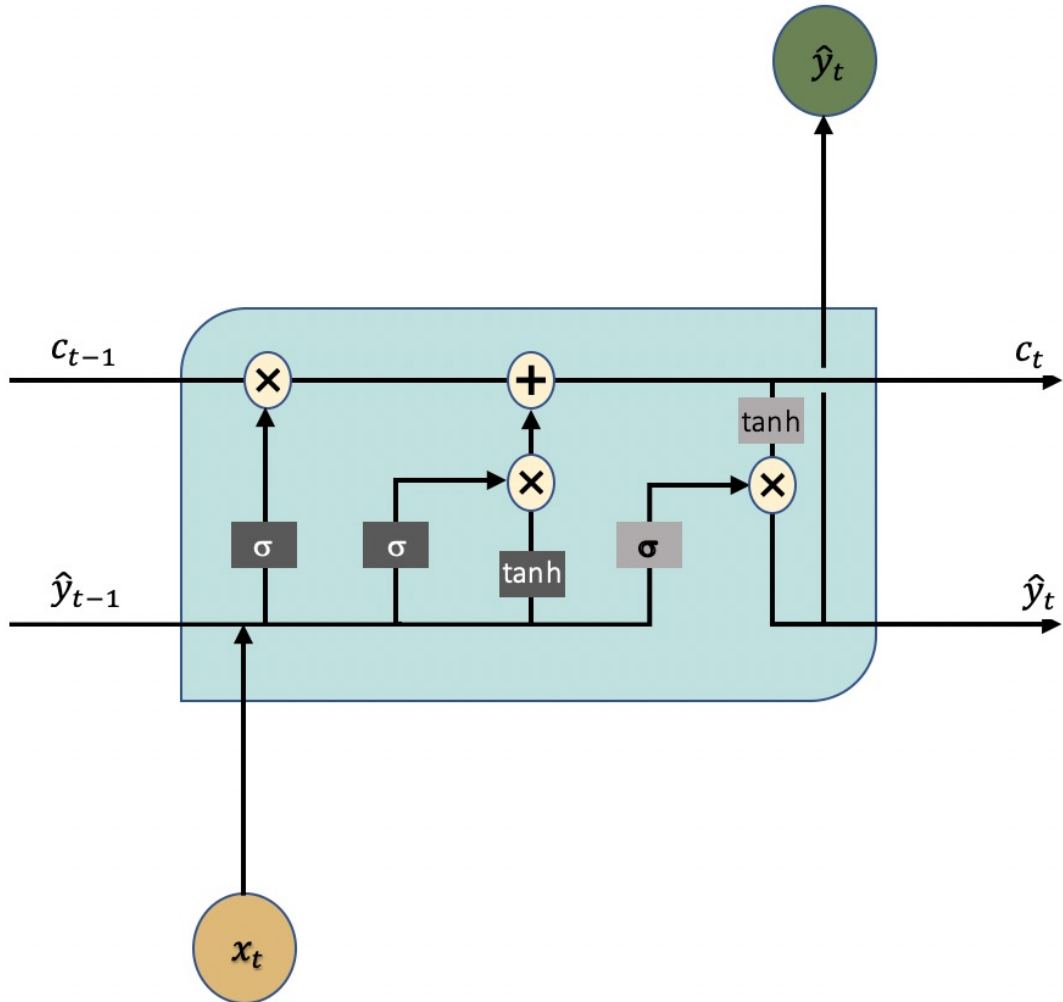


Figure 6.3: Composition of a LSTM module

The LSTM module has three inputs: a sequence value (x_t), the output of the previous module (\hat{y}_{t-1}), and the cell state (c_{t-1}). The values of x_t and \hat{y}_{t-1} are concatenated at the start and will be referred to as $x\hat{y}$. The cell state acts as an information highway through all of the time steps and the information there can be modified by three gates in any given time step. These gates are represented by the dark grey rectangular operations seen in Figure 6.3, which are on the feeder pathways to the cell state from the other two input vectors. The first of these is called the “forget gate”. It involves passing $x\hat{y}$ through a *sigmoid* layer, which forces the values to be between 0 and 1. This is then pointwise multiplied with the

cell state, in effect deciding which values to keep and which to throw away. The next two gates determine the new information to be stored in the cell state. The second gate is a *sigmoid* layer to determine what to save and the third gate is a *tanh* layer that determines what new values to store in the cell state. The results of these two gates are combined (pointwise multiplication) and added (pointwise addition) to the cell state pathway. Finally, the output \hat{y}_t is a filtered version of the cell state c_t , with a *sigmoid* layer once again determining what must be kept.

We have discussed neural networks for handling sequential data and looked at the LSTM structure for effectively learning long range connections. We will now discuss how these are applied to the task of discovering asteroids in the MOA-II data, starting with the dataset.

6.2 Data

Chapter 4 Section 4.1 describes the process of creating the dataset for image classification and Figure 4.2 breaks down how many images are in the training, validation, and test set. To summarise, we have 4072 128x 128 subtracted stack images with tracklets and 19,682 images without tracklets. These are divided into the training, validation, and test set with just over 80% going to the training set. The tracklet images in the training and validation set are further augmented with 8 image enhancements per image. These enhancements to the original images are, rotating by 180 degrees, flipping horizontally and vertically, blurring, brightening, darkening, and lowering and increasing the contrast. Thus, we have 29,898 tracklet images in the training set and 3735 tracklet images in the validation set. The same augments are generated for the images without tracklets as well but only a randomly chosen 35% of these are included in the training and validation set. We have a total of 59,261 no tracklet images in the training set and 7748 no tracklet images in the validation set. The data in the test set is not augmented and there is no overlap between the 3 subsets. There are a total 103,025 images in the dataset.

The dataset for the CNN-LSTM model is built with this classification dataset as its foundation. It is modified to suit the needs of the sequential architecture by replacing each stack image with a directory of the same name, containing

the sequence of observations that comprise the stack image. The augments are also included in this decomposition, with the enhancement begin applied to each observation image in the stack. Figure 6.4 illustrates the directory structure of the modified dataset. Note that there can be between 10 to 150 observations on any given night, which translates to a variable number of observation images in stack directories from different nights. Figure 6.5 displays the decomposed stack images together with the corresponding subtracted stack image. This dataset is significantly larger than its progenitor with 3.1 million images requiring 25.5 GB of storage.

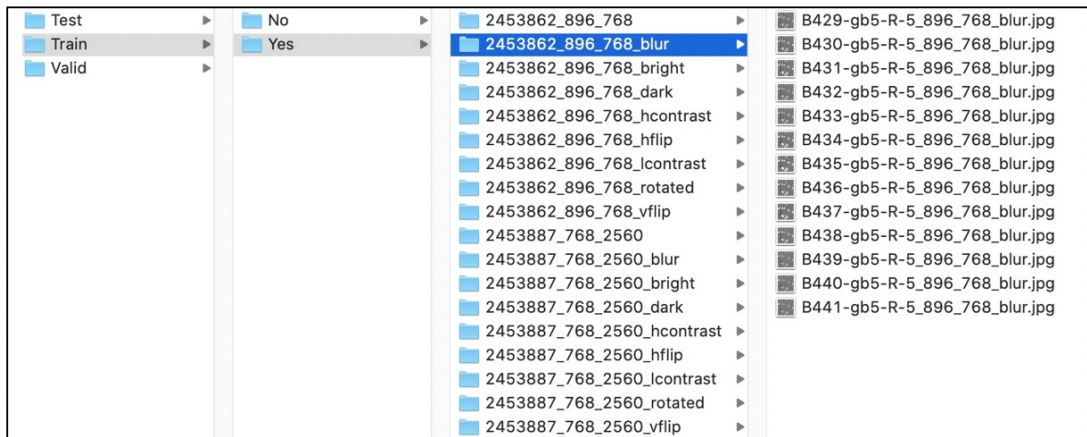


Figure 6.4: Structure of the dataset for the CNN-LSTM

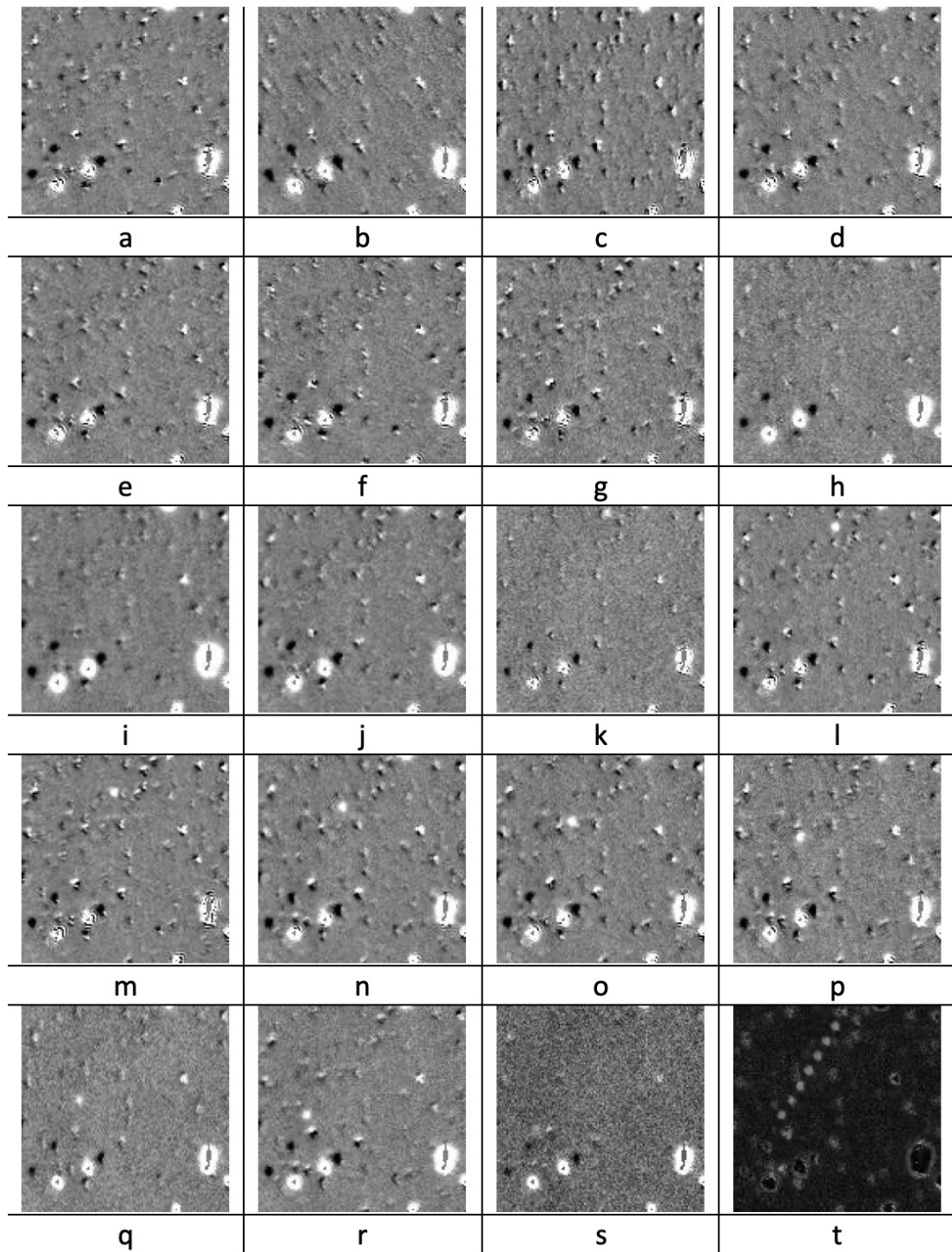


Figure 6.5: All observations for a sub-region on one night (a to s) followed by the corresponding subtracted stack image. The asteroid is visible from frames k to r.

6.3 CNN-LSTM Model

To harness the capabilities of LSTMs for asteroid detection, we once again turn to CNNs to first extract meaningful spatial information from the images before temporal relationships can be established. The CNN-LSTM model (Donahue et al. (2014)) was proposed to tackle image-based time series data, with applications in image captioning and activity recognition. In our case, we want a network that takes a sequence of observation from one night and determines whether a moving object is present. Figure 6.6 offers an overview of the structure, highlighting how each item in the sequence will essentially have its own copy of the network. The ideal CNN feature extractor is one that has already been rigorously trained with similar data. Here, we re-purpose the classification CNNs, MOA-12, Hybrid(a), and Hybrid (b) from Chapter 4, with their pre-trained weights and with the final sigmoid classification layer removed. While the data they are trained with is not identical, they still offer a significant advantage because of the general structural similarities. A simple CNN-LSTM was also constructed as a test case. Figure 6.7 shows the network structure with the pre-trained CNN as well as of the simpler model.

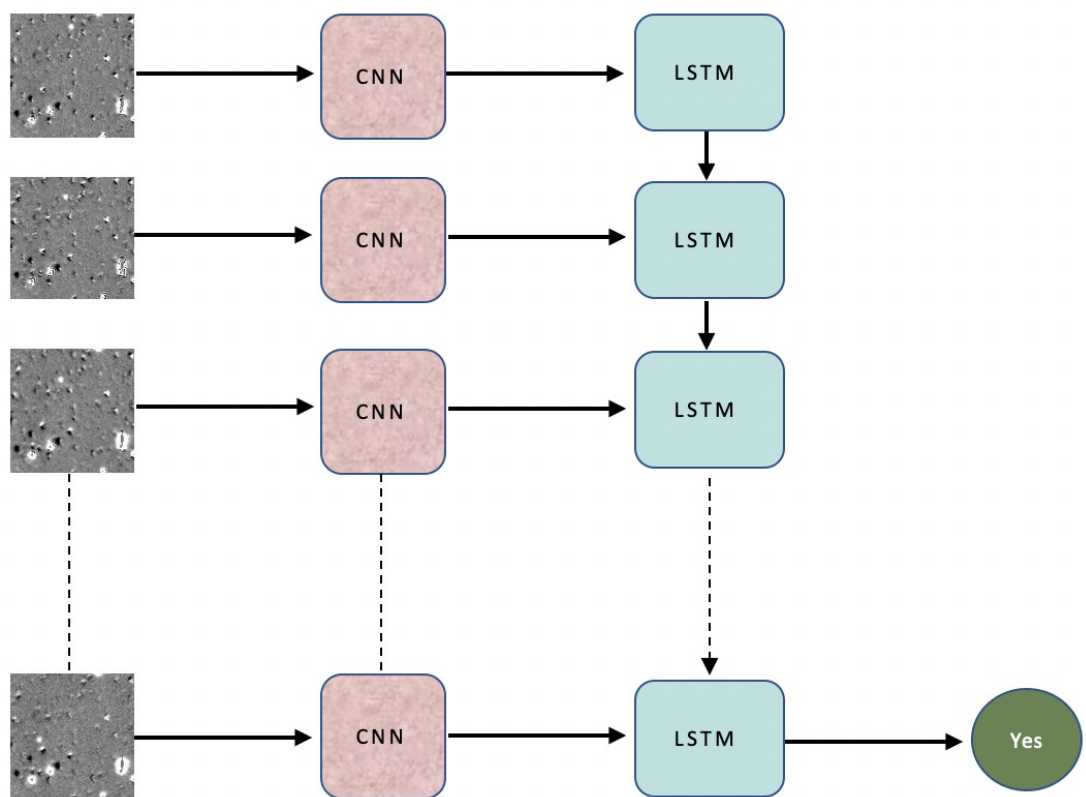


Figure 6.6: An overview of the CNN-LSTM architecture for asteroid detection

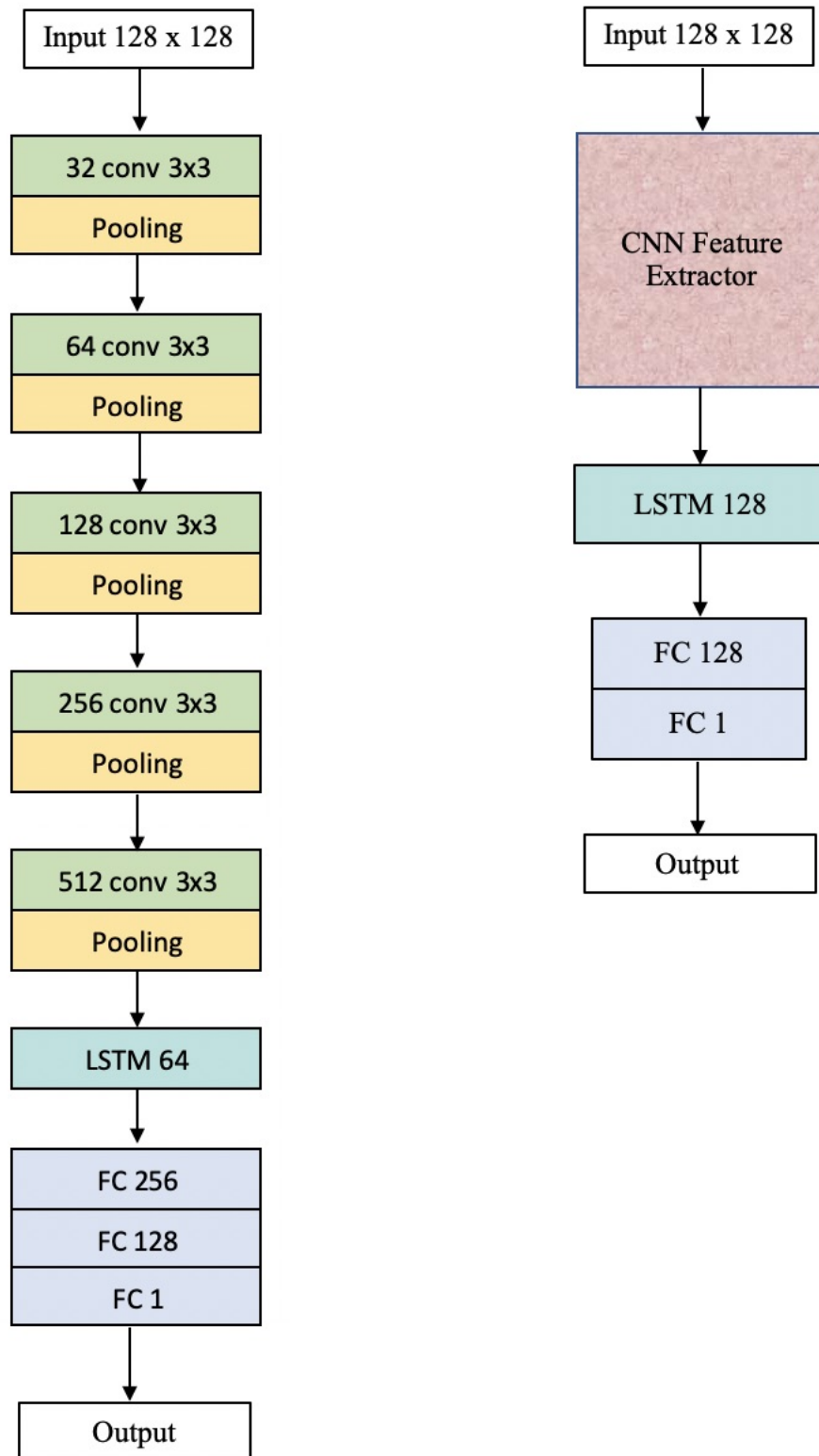


Figure 6.7: Simple CNN-LSTM model (left) and CNN-LSTM model with a pre-trained CNN (right)

6.4 Training

All of the model architectures were trained on a Linux machine running Ubuntu 18.04 with a NVIDIA Quadro M4000 GPU (8 GB, 2.5 TFLOPS). The code was written in Python 3.6 in the Jupyter Notebook environment. TensorFlow GPU 2.4.1 (CUDA 11.0), along with the Keras deep learning API, were used for creating the CNN-LSTM models tested.

The training/validation/test split for the dataset can be seen in Table 6.1. A new Keras data generator object was created to load only one batch of data at a time. GPU memory constraints along with the fact that several sequences have over 50 time steps (some have over 100) meant that each batch could only contain one sequence. The Keras TimeDistributed wrapper was used for applying a convolutional layer, as well as the entire CNN model, to every temporal slice. The learning rate in most cases was 0.0001, though learning was started at 0.001 in some cases before moving to 0.0001 after two epochs. In each case, the learning rate was reduced by a factor of two every five epochs. The optimiser Adam was used along with the binary cross-entropy loss. Each network was initially trained for 50 epochs, the results of which indicated the number of epochs for further training (25, 50, or 80).

After some initial tests, it was decided that the networks would first be trained on the validation data and followed by fine-tuning with the training data. The smaller dataset is faster to train, which made it easier to see the impact of various hyper-parameters. LSTMs have a reputation of being notoriously hard to train and that was certainly borne out here, as we will see next. The unfortunate consequence of having to use a batch size of one is that the larger the network the longer it has to run for learning to start. This has significantly impacted our results. Time constraints did not allow for comprehensive experimentation and testing, and our results only represent the first steps in eventually successfully training a CNN-LSTM for asteroid discovery.

6.5 Results

Unfortunately, our architectures with a pre-trained CNN failed to learn, never moving past the accuracy of 68%, which is only 1% more than the baseline for a no-skill network. The next step was to investigate whether training the CNN along with the LSTM would yield better results. Our smallest CNN, MOA-12, was chosen for this task and but once again failed at learning.

The simple network with five convolutional layers was then trained to gauge if a CNN-LSTM model could learn to discover moving objects in the MOA images. The network started learning at the 36th epoch and reached an accuracy of 96% on the 98th epoch. Evaluating the model with the test set gave us an accuracy of 76%. However, further evaluation (Figure 6.8) revealed that this was likely because the dataset is imbalanced. The precision and recall remain very poor at around 20% and as we can see from the ROC plot, the classifier is only marginally better than a baseline no-skill model. The poor result, while discouraging, does indicate that with the right model structure and time to train the model for longer could lead to better results. The batch size is a significant stumbling block, which will need to be addressed in future implementations.

	Train	Validation	Test
Yes	29,898	3735	335
No	59,261	7748	2048

Table 6.1: Train/ Validation/ Test set for the CNN-LSTM

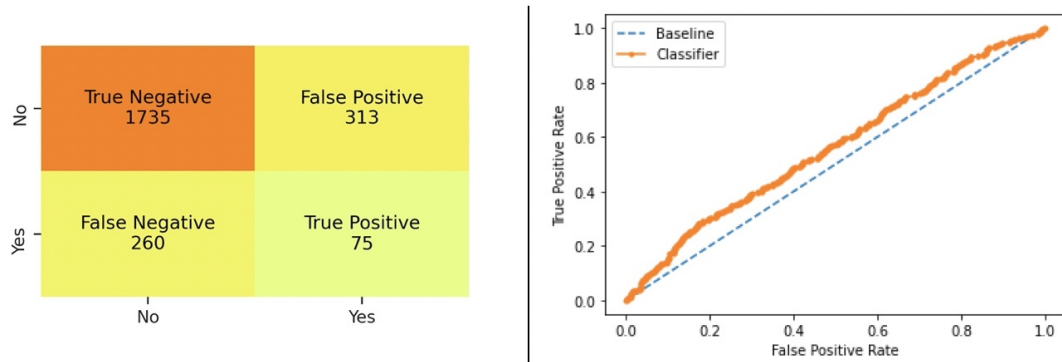


Figure 6.8: Confusion matrix (left) and ROC plot (right) for a simple CNN-LSTM trained to identify asteroids

6.6 Discussion

The CNN-LSTMs perhaps come closest to how asteroids were traditionally discovered - by serendipitously identifying movement from one image to the next. But while evolution has ensured that humans are superlative at this task, that intuitive leap is harder for an artificial neural network. We investigated the application of the CNN-LSTM architecture for discovering an asteroid moving in a night's observations. Though our efforts at training proved unsuccessful, it marks only the first attempts at learning. The existence of a suitable dataset alone creates several opportunities for future work.

It might be prudent to attempt training without data augmentation to create a set of baseline weights that can then be fine-tuned with the augments, or indeed more labelled data. The time saved from training a large dataset could be used to train a smaller one for longer, with an initial training run of at least 100 epochs. This may also allow for any problem areas to be identified sooner. Training initially with only the tracklet data could also be an option. It could then be further fine-tuned with both tracklet and no-tracklet data.

The sequence of observations used as input are of variable length and can be anywhere from 10 to 150 time steps long, adding to the complexity. While it is tempting to convert these into fixed-length sequences by breaking up long ones and padding shorter ones, care must be taken to ensure that the data is still labelled correctly as one or more asteroids may appear and disappear at any point

during an observation cycle. For example, if there are 108 observations on a night, an asteroid may appear in steps 35 - 51 and another from 73 - 95. If we break up the sequence into groups of 20, we will have asteroids visible from 20-40 and 40-60 as well as 60-80 and 80-100 but not in any of the other groups. This simple scenario can largely be automated if we could assume that the seeing conditions remain excellent all through the night. However, there are several cases where the seeing deteriorates for part of the night, which leads to a gap in the observed tracklet for the asteroid. Hence, the data will need to be relabelled to reflect this as well as the scenario where only one or two observations are visible in a group. Breaking up the sequences would allow the use of batches when training, which could reduce the time taken for learning to start.

A possible reason for the inability of the network to learn could be that the composition of the images does not convey enough useful motion information for the LSTM to “remember”. The change we want the CNN-LSTM to recognize is a pixel blob being in a slightly different location in a consecutive image. Given that there can be a lot of variability in each successive observation in one night, this movement might not register as significant. Further, the point source representing a faint asteroid may be indistinguishable from the noise. We lose the low-level features as we get deeper in the network, which likely impacts the final feature maps. A solution could be to use a DenseNet (Huang et al. (2016)) as the feature extractor, which will see the fine-grained features propagate through the network. Equally, it will be interesting to experiment with model scaling by applying the EfficientNet (Tan and Le (2019)) architecture to the feature extractor. It is possible that a small feature extractor will prove better at distilling the salient features of a single observation image.

The poor result obtained here by the CNN-LSTM represents a single step in the path to the eventual successful application of this architecture for discovering asteroids. The method requires the least amount of image pre-processing and is therefore absent the noise seen in the stacked images. Finding moving objects in survey data is a challenging problem and offers a rich avenue for further research.

Chapter 7

Conclusions

Asteroids are remnants from the creation of our solar system 4.6 billion years ago. They are small, rocky worlds that number in the millions with a significant population in the region called the main asteroid belt between Mars and Jupiter. The composition of asteroids informs our knowledge about the building blocks of planets as well as, potentially, life on Earth. They form part of the puzzle for understanding the evolution of our solar system. While there are several surveys dedicated to the detection of asteroids, particularly near-Earth ones, archival data from other surveys can also be a useful tool for finding asteroids. The MOA-II telescope at the Mt John Observatory has been part of the MOA project since 2004 with the aim of observing microlensing events in the Galactic Bulge. It has amassed over 100TB of observational data, which has only been used for its stated purpose. Thus, its full scientific potential has not yet been realised. As permanent fixtures in our night skies, asteroids are present in all survey data, including MOA's. The aim of this research was to develop a dataset suitable for supervised learning and to develop deep learning solutions for discovering asteroids in the archival MOA data.

Modern astronomical surveys typically generate more data than can be analyzed without purpose-built software suites. The application of deep learning in the field is still in the early stages, with the lack of available labelled data deemed a significant challenge. Deep learning is even more sparsely applied for the purposes of asteroid detection. Of the six published works, two involve comets,

two involve space telescopes, and two involve near-Earth asteroids. The research with space telescopes comes closest to the aim of this research, however data from ground-based observatories pose a different set of challenges and the profile of asteroids observed is significantly different. Therefore, this work is the first of its kind for asteroid detection.

7.1 Dataset

The first step was to build a dataset from the archival MOA data. The 1.8m MOA-II telescope consists of a wide-field mosaic CCD camera with 10 chips, each of which are 2048 x 4096 pixels with a resolution of 0.01 arc minutes per pixel. Each chip is considered separate from its neighbours. The telescope surveys 23 fields towards the Galactic Bulge, the densest star field in our galaxy. It operates at a very high sampling rate with each exposure lasting 60 seconds. This means that we can clearly see asteroid tracklets - part of the arc of an asteroid's orbit - in a series of observations from the same night. MOA researchers work with difference images, which are the result of subtracting the observation image from a reference image. The resultant image highlights the changes captured in the observation image, including any asteroids that cross the telescope's field of view. They also contain any noise in the form of imperfect subtractions, satellite trails, differences in seeing conditions, and other spurious artefacts. Stacking a night's observations sequentially reveals tracklets of asteroid moving through the field at the time, with the faintest object observed being of limiting magnitude 20.5.

The first (GB5-R5) of two datasets was built from 49,901 difference images, consisting of 14 years of observations from one chip (chip 5) in the CCD assembly for the field GB5. The majority of this dataset was used for training neural networks. The second dataset (GB-All (28-06-2013)) consisted of one night's observations from all fields (and all 10 chips) surveyed on the night of 28-06-2013. This dataset was used exclusively for testing how the classification networks performed with data they had never seen.

To convert the difference images into a suitable format for neural networks, nightly observations from the same chip and field were first stacked by brightest pixel. This brought the tracklets into sharp relief. A corresponding median pixel

stack was also created, which was subtracted from the brightest stack, leaving behind the subtracted stack image containing only the asteroid tracklets and noise. A list of all minor planets that were expected to be in MOA's field of view at the time was then obtained from the MPC. Further careful visual inspection of the data resulted in locating 1178 distinct asteroid tracklets. Next, each of the 2048 x 4096 stack images was split into 512 128 x 128 pixel images. The Cohen-Sutherland line clipping algorithm was used to determine which of these 128 x 128 images were expected to contain tracklets. Visual inspection was once again undertaken, resulting in a total of 4547 (GB5-R5: 4153, GB-All: 394) 128 x 128 images with visible tracklets.

7.2 Classification CNN

With a suitable dataset in hand, the next step was investigating various CNN-based architectures for binary classification. Given the brevity of the data, it was hypothesised that architectures with fewer parameters that were shallower than their famous counterparts would do better. Fifteen architectures were chosen, some the result of prior computer vision research, some custom built. The ones based on prior research were: AlexNet, VGG-16, VGG-19, ResNet50, and Inception. The custom built architectures were: modified AlexNet, shallower ResNet, shallower Inception model, four VGG-like architectures with significantly fewer parameters (MOA-12, MOA-13, MOA-14, MOA-15), and three architecture with hybrid Inception-ResNet modules (Hybrid(a), Hybrid(b), and Hybrid(c)).

The GB5-R5 dataset was used for training and initial testing of the architectures, with the tracklet images reduced to 4072 after further visual inspection. The images were divided into the training/validation/test sets with 3322/415/335, respectively. Each image in the training and validation data was augmented by being horizontally flipped, vertically flipped, rotated by 180 degrees, darkened, blurred, brightened, and with the contrast both increased and decreased. This resulted in 29,898 and 3735 images with tracklets in the training and validation set. Since the classifiers also needed to learn how to differentiate between images with and without tracklets, the dataset needed images without tracklets. These were obtained from the 550K 128x 128 images without known

tracklets. Around a third of the no-tracklet data in the training and validation set consisted of augments. The no-tracklet data was also split into the training/-validation/test sets with 59,262/7748/2048 images, respectively.

The main evaluation metric considered was recall or true positive rate, which quantifies how many of the tracklet images were correctly classified. Favours this metric does mean accepting a larger number of false positives, but false positives are an accepted part of analysing astronomical data. Six of the networks had a recall of 89% or above with the GB5-R5 test set. The true test was to see how the networks performed with the GB-All test data and here, seven of the networks - the four MOA architecture and three hybrid architectures - outperformed the other networks with a recall of 94% or above.

An ensemble of classifiers was constructed with MOA-12, MOA-14, MOA-15, Hybrid(a), and Hybrid(b). It was expected that the probability distribution learnt by each would serve to bolster the final predictions. In each case, the class with the highest predicted class value was chosen, which led to achieving a recall of 97.67% with the GB-All test set.

The success of the custom CNN architectures with fewer training parameters is perhaps down to two aspects. First, the networks might be at the optimal depth for learning to identify the pattern of fuzzy blobs that represent a tracklet. Second, the architectures are trained with a small dataset, which perhaps leads to more over-fitting in the larger models. Further, rather than adding more layers, the hybrid architectures lean into adding complexity with several 1x1 convolutions. It is possible this caused the networks to learn representations that ultimately boosted the performance of the ensemble.

7.3 YOLO

The impetus behind seeking a tracklet localization solution was that some of the tracklets are not obvious at a glance, either because the object is too faint, too widely spaced, too obscured by noise, or some combination thereof. Additionally, as a result of the line clipping algorithm, we had the starting coordinates

for bounding boxes to enclose the tracklets. Further visual inspection and manual adjustments to these baseline bounding boxes resulted in a dataset of 4153 tracklet images that could be used for object detection.

The aim of the original YOLO object detection architecture was to make object detection both fast and accessible. As a single-shot architecture, it was capable of making class and bounding box predictions with the feature maps produced by a single CNN network. The simplicity of the feature extraction CNN at the heart of YOLO is in direct contrast to its complex loss function, which computes the classification loss, localization loss, as well as the loss quantifying the network's confidence in the prediction. Later versions of the network introduced default anchor boxes that would be adjusted to fit the model's bounding box predictions and a more complex feature extractor to make predictions on multiple scales. This research applied YOLOv4, which is the latest evolution of the architecture with a major overhaul to include several new techniques. Among the changes are a new feature extractor based on DenseNet with CSP connections, a new feature aggregator based on PANet, and an improved loss function that quantifies the IoU loss in the place of the old localization loss. YOLOv4 also includes a raft of updates like a new activation function (Mish), Mosaic augmentation, self-adversarial training, spatial attention module, and multi-input weighted residual connections. Altogether, these changes make YOLOv4 state-of-the-art while still being very fast. Additionally, a smaller, scaled back version called YOLOv4-tiny was also developed to make YOLO more accessible to everyone.

Both YOLOv4 and YOLOv4-tiny were trained to localize asteroid tracklets. Along with using the default anchor boxes, custom anchor boxes were also trialled. The networks were trained with just the tracklet data as well as a combination of tracklet data, augmented (rotated, horizontally and vertically flipped) tracklet data, and data with no tracklets. Surprisingly, the networks trained with only the tracklet data performed far better by an appreciable percentage. While the results with the custom anchor boxes were good, the default anchor boxes proved to be robust enough to adapt to the task. The mAP (mean average precision) for each network was calculated at the confidence threshold of 25%. The best mAP with YOLOv4-tiny was 65% and the best mAP with YOLOv4 was 90.96%.

7.4 CNN-LSTM

Having successfully used static images for both classification and object detection, the next step was to determine whether the temporal aspect of observational data could be harnessed for discovering moving objects. An advantage of this approach is that we do not have to contend with the additive noise in the stack images. LSTMs are part of a family of neural networks that were specially created to analyze sequential data. Here, we use a CNN to extract meaningful features from an image, which are then used by the LSTM to remember important features that will take us to the desired output. The classification dataset was re-purposed for the CNN-LSTM by breaking each stack down to its component images. Rather than a stack image, the input for the CNN-LSTM was this sequence of images.

While the attempts at training the CNN-LSTM were unsuccessful, it does not conclusively demonstrate that the architecture cannot be trained for this task. Rather, what does not work has been eliminated and a dataset exists with which to continue the research. It is likely that the CNN feature extractors used were not suitable for distilling the right information from the observation images. Each successive observation on a given night can have a high degree of variability and in some cases the asteroid may be indistinguishable from the noise. Further research to find the best CNN architecture to accurately represent this will be crucial to the success of the network. Ultimately, the CNN-LSTM architecture comes closest to modelling the intuitive human process of recognising movement and should be further investigated.

7.5 Future Work

As each of the networks and the datasets created here are the first of their kind in the field, there are plenty of opportunities to extend the research. The biggest challenge for other surveys looking to adapt the methodologies developed here is gathering the required labelled data. This could be mitigated by harnessing the power of crowd-sourcing via the Zooniverse platform. Itself born from a successful crowd-sourcing exercise for labelling galaxy data, the platform could be used for both sifting through and identifying images with tracklets. It could

also be extended to prompting the citizen scientists to mark the start and end points of these tracklets, which could be used to refine the results from the line clipping. An immediate advantage of crowd-sourcing is that multiple people will look at and classify an image, thus reducing errors. It also offers an outreach opportunity and would increase the profile of the research.

Investigating effective denoising techniques for the stacked images or the difference images would lead to an immediate performance boost for both the classification and object detection networks. The classification network may benefit from having two inputs - the subtracted stack image along with, perhaps, the median stack image. This would provide the model with additional information it could use to distinguish between images with or without tracklets. The object detection network would benefit from training with tracklet images from other MOA fields and chips. Additionally, training the YOLO backbone feature extractor with the classification data first might also lead to better results. The CNN-LSTM is primed for further research into whether the architecture can learn to detect small fuzzy blobs moving appreciably from one image to the next. Converting the large variable length sequences to multiple fixed length sequences will allow for batch processing and perhaps aid the network in remembering the movement. Furthermore, since much of the salient information is contained in the first layer, a small CNN based on DenseNet could be a more effective feature extractor.

The next steps for this research will be to use the classification and object detection networks developed to look for asteroid tracklets first in the 550k images from GB5-R5 without known asteroids, and then in the larger, as-yet-untapped, MOA-II archival data. Making predictions with the 550K images first will help with establishing some benchmark expectations and also highlight any potential problem areas. It will also provide an indication of how many undiscovered asteroids could potentially be in the data.

Discovering tracklets in the remaining archival data will start with first assembling the dataset in the required format. The list of known asteroids traversing these observations has already been obtained. Given the sheer amount of data, it would be prudent to break it up into more manageable groups with two years of data from all of the fields and chips in each. The full GB-All dataset mentioned

in this work has data from all the fields from 2013 - 2015 and is a good starting point for exploring the possibilities with the archival data. To recap, this dataset contains 905,302 difference images, amounting to 11TB of data. One night (28-06-2013) of observational data from this set was used to test the classification models. YOLOv4 will also be retrained with the known tracklets from this night. Subtracted image stacks will be created from the remaining observations, following which they will each be split into 128 x 128 image tiles. The line clipping algorithm will then be applied to find all the images tiles expected to have known asteroid tracklets. These potentials will then be given to the ensemble to filter out the images with visible tracklets. Previously, this task had to be done by visually scanning all the potential tracklet images, which could take over a month. With the trained networks, this task can be performed in a matter of hours. However, at this stage, it is anticipated that some visual inspection will also be crucial to gauge the effectiveness of the networks. The most beneficial next step will be to use all of the tracklets of known asteroids discovered in the GB-All set to fine-tune the classifier ensemble and object detector. Once retrained, all of the remaining 128 x 128 images from the GB-All set will be given to the networks to classify, with special note made of any unknown/newly discovered tracklets.

It would be judicious to repeat the same course undertaken with the initial GB-All set with the next two groups of archival data (four more years). After the networks have been fine-tuned, they will be used to make predictions on the remaining data, perhaps one field worth at a time. Since we have the list of expected known asteroids in the data, we can check if a tracklet detected is from a known object. Unknown tracklets can be further verified by checking the original difference images to ascertain if it does indeed match the profile of an object moving in our solar system. Code has already been written to convert a stack image into a “movie” composed of the individual observations that form the basis of the stack with the file names included. The inclusion of the file name makes it easier to isolate the observations in which an object appears.

Given the sheer amount of data gathered by MOA-II, along with the ideal observation interval for capturing moving solar system objects, there is optimism about discovering new asteroids in the archival data. Moreover, the ephemeris data that can be gathered from the observed tracklets will be useful to the MPC for refining the available orbital information. A light curve could be generated

as well, providing further information about the composition and rotation period of the asteroid. Looking beyond objects in our solar system, there is also the potential of finding interstellar objects that may have traversed our solar system undetected in years past.

7.6 Conclusion

The appeal of deep learning is in its ability to learn complex representations directly from the data and provide inference in real-time. The classification networks achieve a recall of 97.67% with a test data from fields not seen by the network. The YOLOv4 object detector achieves an mAP of 90.96% and is very good at locating tracklets in the GB5-R5 data. The networks can be easily fine-tuned as more labelled data is available. With larger datasets, we can also work towards refining the current networks with newer techniques and train more complex architectures. Eventually, this could lead to large labelled datasets for astronomy research and end the dependence on using pre-trained weights from datasets like ImageNet that share little similarity with astronomical data.

The datasets and deep learning architectures applied here are the first documented instances of the application of deep learning for discovering asteroids tracklets in ground-based survey data and they offer a foundation for continuing research in the area. Our methodology and networks can be applied to discover asteroids in other archival survey data as well as to strengthen the analysis pipeline for current and future surveys. Perhaps they could even aid in the discovery of trans-Neptunian and interstellar objects. With deep learning in our toolkit and next generation surveys like LSST on the horizon, we look set to leap ahead on our civilization-spanning quest to understand the universe and our place in it.

Appendix A

Neural Networks Primer

In its simplest form, a multi-layer neural network consists of a one dimensional input vector, which is transformed into the desired output by a series of hidden layers (Figure A.1). Each hidden layer consists of independent neurons, each of which are fully connected to every node in the previous layer. The connections between each node or neuron are meant to mimic synapses and each have a **weight** vector associated with them. The objective of a network is to learn the best combination of weights and biases - known as parameters - to take us from the input to the output. The final layer is the output. If we were classifying objects, the output would be the class scores. We evaluate how well a network models the data by using a loss function to calculate the error between the expected and computed output.

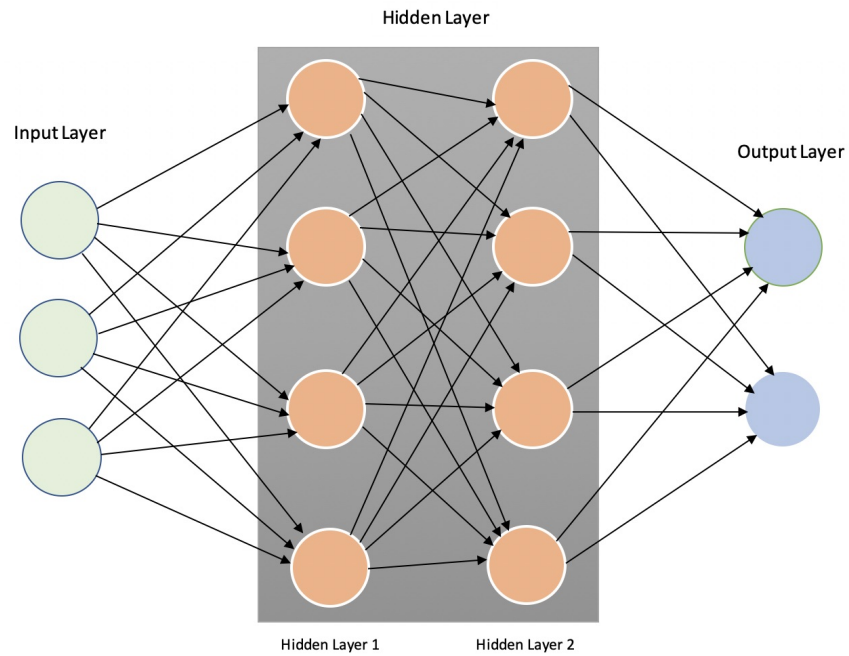


Figure A.1: A fully connected artificial neural network with two hidden layers and two output nodes.

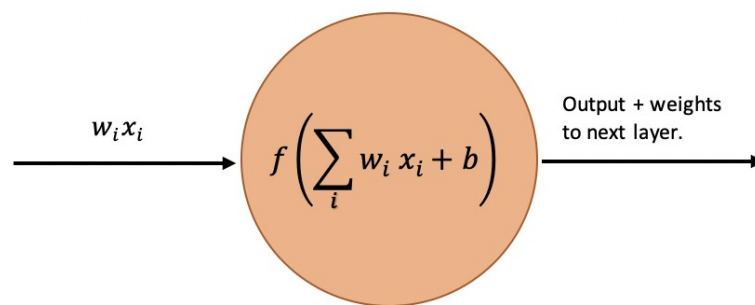


Figure A.2: A fully connected artificial neural network with two hidden layers and two output nodes.

Figure A.2 shows the basic structure of a neuron in a hidden layer. x_i represents input features to the neuron and w_i represents the weights attached to the input. A neuron sums up the dot product of each input with its corresponding weights and adds a bias element. If we look at it in terms of linear algebra, the weights change the slope of the line and the bias changes where the line intersects the y-axis. An activation function, f , is applied to this linear computation to add complexity. The activation function is a non-linearity such as sigmoid, softmax, or a rectified linear unit. The result of the activation is the input to the next layer, which is again relayed along with a bias term and weights vector. The process continues until we get to the output layer. This is the **forward pass** or **propagation** of a neural network.

The next important component is the **loss function**. The purpose of the loss function is to evaluate the error or how far the algorithm's computed output is from the desired output. The goal of a neural network is to minimize this loss. The commonly used loss function for binary classification problems is binary cross-entropy or log loss. The equation for the j^{th} sample is as follows:

$$L(y_j, \hat{y}_j) = -(y_j \log \hat{y}_j + (1 - y_j) \log(1 - \hat{y}_j))$$

where y is the expected output and \hat{y} is the computed output. The expected values of \hat{y} are between 0 and 1.

Neural networks next employ an algorithm called **backpropagation** (Rumelhart et al. (1986)) to compute the weights and bias until we reach a combination that can best map the input to the desired output. This is done by propagating the error calculated after forward propagation back through the network, that is from output to the first hidden layer. At each layer, for each neuron we calculate the gradient of the error with respect to the weights (w) and bias (b). In essence, we are attempting to compute how the total error will change if we update the parameters of the network by a small value. The weights and bias are updated with an optimisation method; the most commonly used is **gradient descent**. The equations for updating (w) and (b) in layer (l) are as follows:

$$w_l \longrightarrow w'_l = w_l - \alpha \frac{\delta L}{\delta w_l}$$

$$b_l \longrightarrow b'_l = b_l - \alpha \frac{\delta L}{\delta b_l}$$

While the calculated gradients set the direction the training should move in, we use an additional hyperparameter, **learning rate** α , to specify the size of the step we take with gradient descent. A network thus trained and optimised can be used to make predictions. To summarize, the essential steps in setting up and training a neural network are:

1. Define model architecture;
2. Initialize parameters;
3. Iterate a predefined number times to:
 - (a) Forward propagate;
 - (b) Calculate loss;
 - (c) Calculate gradients (backpropagation);
 - (d) Update parameters (gradient descent).
4. Use trained model to make predictions.

The network model described above is the simplest form of an artificial neural network (ANN). While the essential steps for training neural networks remain similar, the model architecture can get quite sophisticated. The idea of **deep learning** is to build complex models with multiple layers and multiple nonlinearities to better represent the complexity of the data (Bengio et al. (2006)). The layers form a hierarchy, with each subsequent layer extracting a more abstract representation of the input. In essence, this hierarchy serves as a self-contained knowledge base for the algorithm: it learns a complex concept by breaking it down to simpler concepts and learning from how these are linked together. The deeper the hierarchy, we hypothesize, the better the knowledge base that the algorithm learns from.

Historically, deep networks have been difficult to train because gradient information would vanish as we backpropagated to the lower layers. Unsupervised pre-training to initialize the parameters of a deep neural network was proposed as a solution to the vanishing gradients and was a promising solution (Hinton et al. (2006)). Later, the **rectified linear unit** (ReLU) was proposed as an alternative to the logistic **sigmoid** activation function in deep networks (Glorot et al. (2011); Nair and Hinton (2010)). This significantly reduced the possibility of vanishing gradients and made the unsupervised pre-training unnecessary. Two other important additions to improving the performance of deep networks are **batch normalization** (Ioffe and Szegedy (2015)) and **dropout** (Srivastava et al. (2014)). Batch normalization involves pushing the inputs of each layer into a gaussian distribution. While the input data is usually normalised at the start, the computation at each layer means that the distribution of values changes at each layer, which can slow down the training. Introducing batch normalization improves the performance of the network. Dropout is a regularization technique that prevents overfitting by randomly dropping the output values of a neuron based on some probability p , which is a hyperparameter.

CNN: Convolutional Neural Networks

Convolutional neural networks (CNN) are a type of neural network that deal with images. In Figure A.3 we see the architecture of a simple CNN. Instead of multiple input nodes (input vector), the input of a CNN is an image. If we used the ANN structure to classify an image, we would have to unroll the image into a single input vector. So, a small 100 x 100 RGB image would give us 30,000 input features. If we had 100 neurons in the hidden layer, we'd end up with 300,000 parameters, which means that even a simple network could get very large and be computationally expensive. Thus, the rationale behind a CNN is to make deep networks a plausible avenue for analysing 3D data, such as images, for classification or detection tasks. Accepting an image as the input also means that features don't have to be hand engineered by an expert as a CNN can learn directly from the data. Moreover, parameters are shared across a single feature map, thus reducing the total number of parameters in the network.

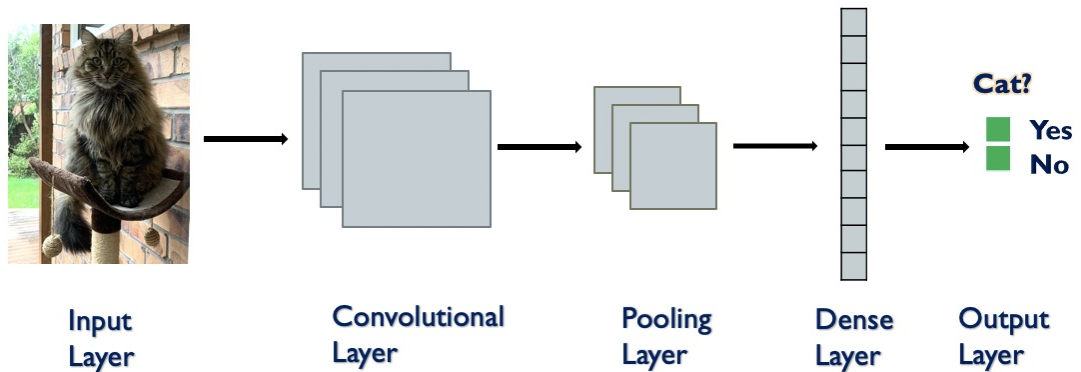


Figure A.3: A simple CNN with one convolutional layer

In a CNN, each neuron in a hidden layer is connected to a subset of features in the previous layer. This subset is called the convolutional filter and it acts like a sliding window over the image. At each location, a matrix multiplication is performed between the pixel value and the filter, which is then summed together, passed through a non-linearity, and added to a feature map. The filter parameters represent the weights that best transform the input to the output. A convolutional network is trained to learn the best combination of these filters/weights for the task. Several of these filters could be applied to an input. If we used 10 different filters we would have 10 feature maps. We'd then stack them along the depth dimension to get the output of the convolution layer. CNNs also often contain pooling layers. The most common form of pooling is a max pool, where the highest pixel value in say a 2x2 region is chosen. Their purpose is to reduce the dimensions of the previous layer. Reducing the dimensions reduces the number of parameters and makes the training faster. Different architectures use different combinations of convolutional and pooling layers, with modern convolutional networks having over 150 layers.

Appendix B

Minor Planet Database Schema

Here we see the schema of the database created to store relevant data retrieved from the minor planet center together with the observational metadata. This format makes it each to retrieve grouped of data in particular, for example a list of all days with more then 40 observations.

APPENDIX B. MINOR PLANET DATABASE SCHEMA

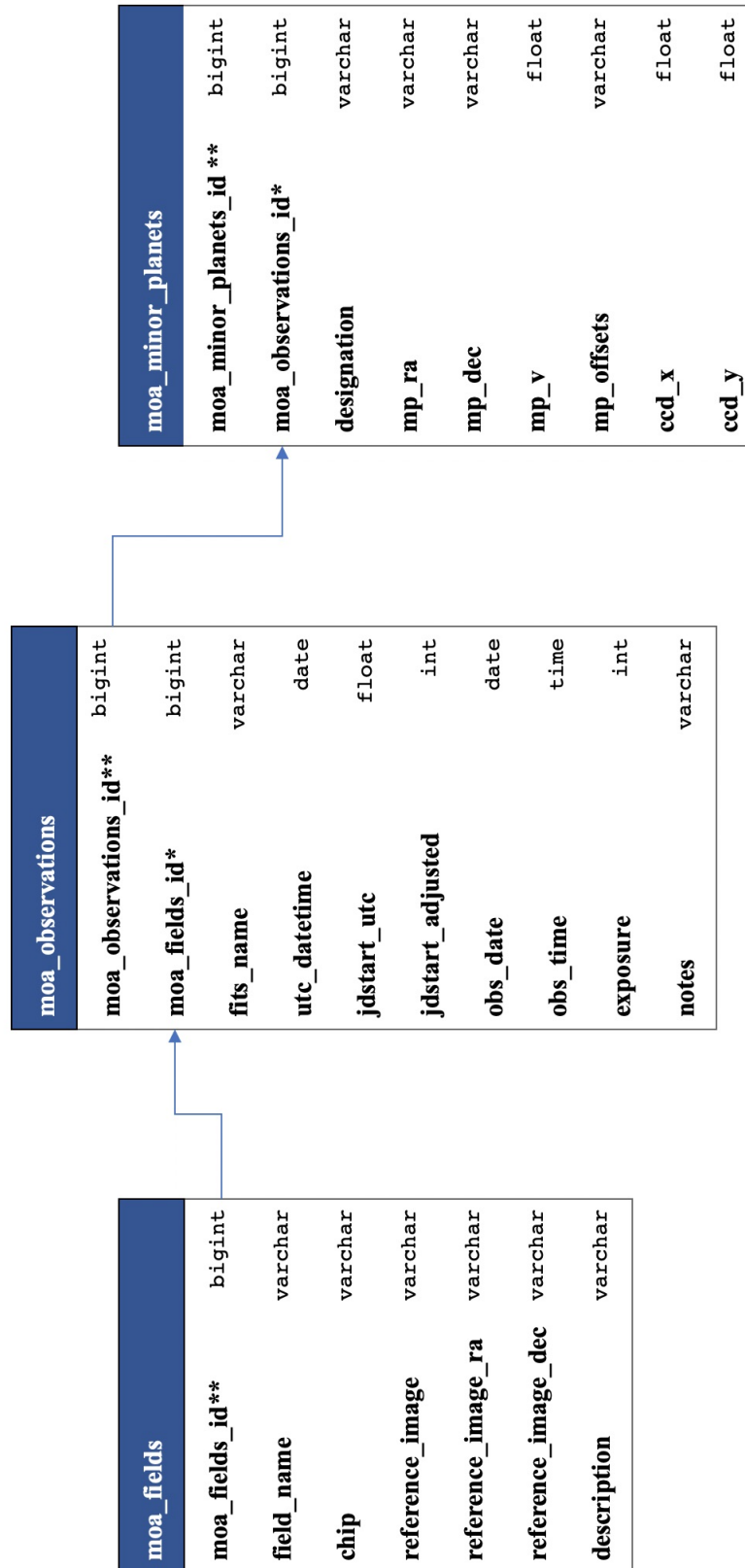


Figure B.1: Database for storing metadata about minor planets in MOA fields

Appendix C

More Metrics for the Classification CNNs

Here we include the plots representing the trade-off between precision and recall (PR Plot) as well as between true positive rate and false positive rate. In each case, we look at how the two types of errors - false negative and false positive effect the performance of the classifier. The benefit of these plots is that they give us a snapshot of the performance at different thresholds and gives us a more generalised picture of a model's performance.

Precision is an evaluation metric that quantifies how many of the positive predictions are actually positive (how many of the images predicted to have tracklets actually have tracklets) and is calculated as: $\text{Precision} = \frac{\text{True Positives}}{(\text{True Positives} + \text{False Positives})}$

Recall (also known the the true positive rate) is an evaluation metric that quantifies how many of the positive predictions were correctly predicted. This is, it tells us how many of the images with tracklets were miss-classified. It is calculated as: $\text{Recall or True Positive Rate} = \frac{\text{True Positives}}{(\text{True Positives} + \text{False Negatives})}$

We can see that the precision value is influenced by the false positive and the recall value by the false negatives. In our case, we want to minimise the false negatives and thus favour a high recall value or true positive rate. A PR

Plot plots the precision (y-axis) and the recall (x-axis) for different probability thresholds and the plots for the various classifiers with the GB5-R5 dataset can be seen in Figure C.1. The best plots are the ones that tend towards high recall values.

The false positive rate is a measure of the number of the positive predictions that are wrong. In other words, how many of the images without tracklets were predicted to have a tracklet. It is calculated as: $\text{False Positive Rate} = \text{False Positive} / (\text{True Negative} + \text{False Positive})$

The ROC plot (C.2) then is essentially the proportion of the positive (tracklet) predictions (y-axis) versus the proportion of incorrect predictions for the negative (no tracklet) class (x-axis) at different thresholds. Here, we favour the true positive rate. The perfect plot would have values clustered around the top left and one that is good for our classifiers would have most values at the top (that is, a good true positive rate).

Additionally, the confusion matrices for the networks not included in Chapter 4 Section 4.4 are in Figures C.3, C.4, C.5, and C.6

Lastly, we see some of the evaluation metrics for the GB5-R5 and GB-All test sets sorted by recall, F2 score, and PR AUC in Figure C.7.

APPENDIX C. MORE METRICS FOR THE CLASSIFICATION CNNs

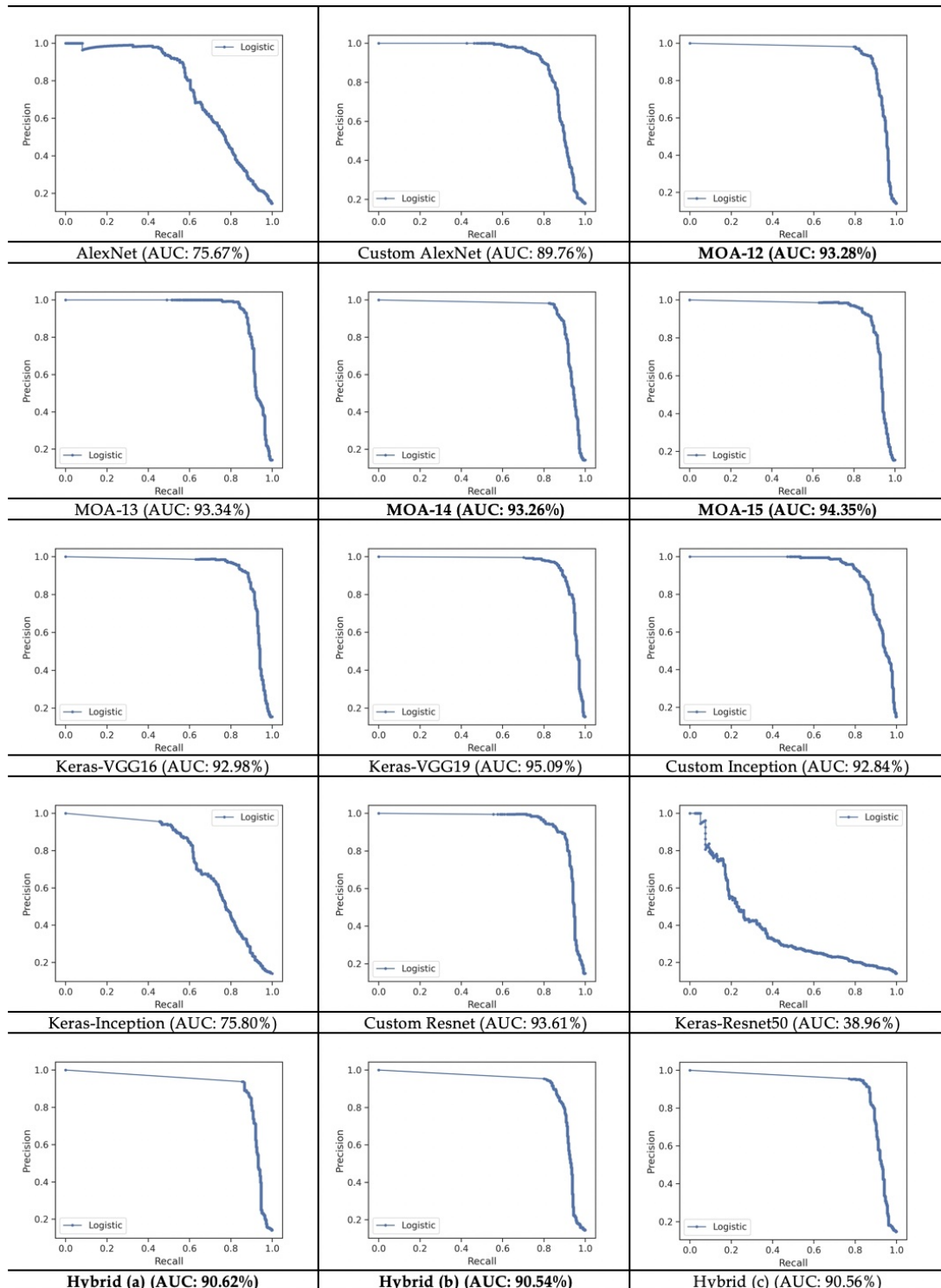


Figure C.1: PR plot for classification models with GB5-R5 test set

APPENDIX C. MORE METRICS FOR THE CLASSIFICATION CNNs

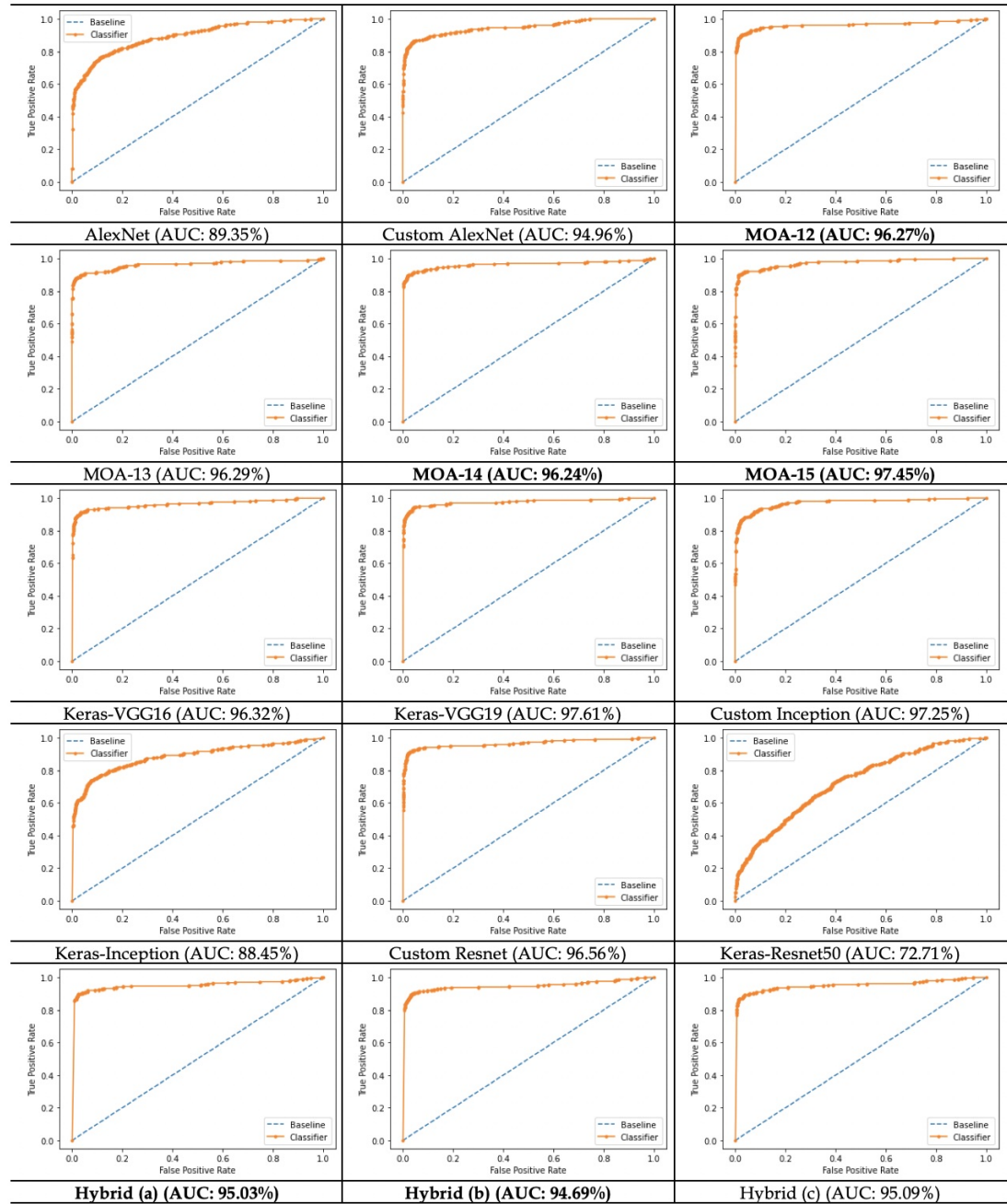


Figure C.2: ROC plots for classification models with GB5-R5 test set

APPENDIX C. MORE METRICS FOR THE CLASSIFICATION CNNs



Figure C.3: Confusion matrix for some classification models with GB5-R5 test set

APPENDIX C. MORE METRICS FOR THE CLASSIFICATION CNNs

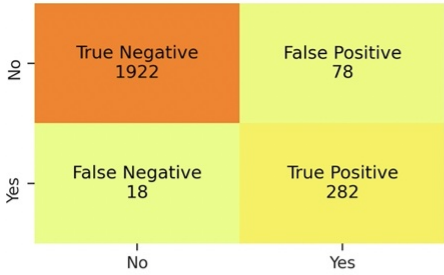
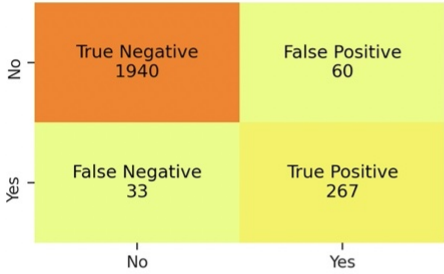
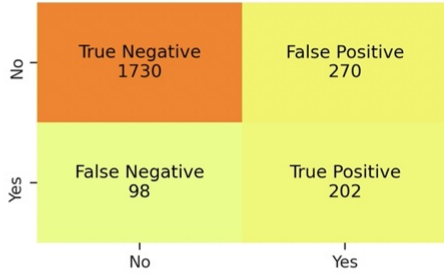
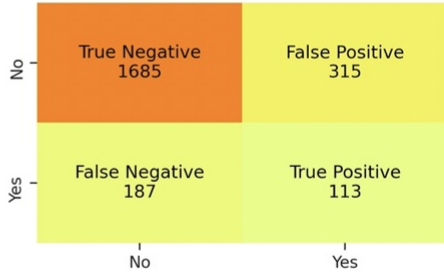
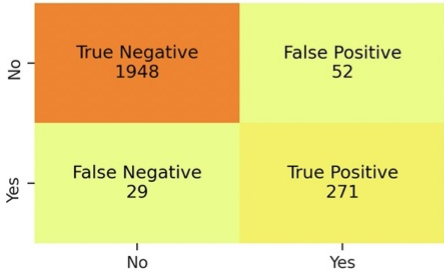
 <p>Confusion matrix for MOA-13 model:</p> <table border="1"> <tr> <td>No</td> <td>True Negative 1922</td> <td>False Positive 78</td> </tr> <tr> <td>Yes</td> <td>False Negative 18</td> <td>True Positive 282</td> </tr> <tr> <td></td> <td>No</td> <td>Yes</td> </tr> </table>	No	True Negative 1922	False Positive 78	Yes	False Negative 18	True Positive 282		No	Yes	 <p>Confusion matrix for VGG-16 model:</p> <table border="1"> <tr> <td>No</td> <td>True Negative 1982</td> <td>False Positive 18</td> </tr> <tr> <td>Yes</td> <td>False Negative 53</td> <td>True Positive 247</td> </tr> <tr> <td></td> <td>No</td> <td>Yes</td> </tr> </table>	No	True Negative 1982	False Positive 18	Yes	False Negative 53	True Positive 247		No	Yes
No	True Negative 1922	False Positive 78																	
Yes	False Negative 18	True Positive 282																	
	No	Yes																	
No	True Negative 1982	False Positive 18																	
Yes	False Negative 53	True Positive 247																	
	No	Yes																	
MOA-13	VGG-16																		
 <p>Confusion matrix for Custom Inception model:</p> <table border="1"> <tr> <td>No</td> <td>True Negative 1940</td> <td>False Positive 60</td> </tr> <tr> <td>Yes</td> <td>False Negative 33</td> <td>True Positive 267</td> </tr> <tr> <td></td> <td>No</td> <td>Yes</td> </tr> </table>	No	True Negative 1940	False Positive 60	Yes	False Negative 33	True Positive 267		No	Yes	 <p>Confusion matrix for Keras-Inception model:</p> <table border="1"> <tr> <td>No</td> <td>True Negative 1730</td> <td>False Positive 270</td> </tr> <tr> <td>Yes</td> <td>False Negative 98</td> <td>True Positive 202</td> </tr> <tr> <td></td> <td>No</td> <td>Yes</td> </tr> </table>	No	True Negative 1730	False Positive 270	Yes	False Negative 98	True Positive 202		No	Yes
No	True Negative 1940	False Positive 60																	
Yes	False Negative 33	True Positive 267																	
	No	Yes																	
No	True Negative 1730	False Positive 270																	
Yes	False Negative 98	True Positive 202																	
	No	Yes																	
Custom Inception	Keras-Inception																		
 <p>Confusion matrix for Custom ResNet model:</p> <table border="1"> <tr> <td>No</td> <td>True Negative 1951</td> <td>False Positive 49</td> </tr> <tr> <td>Yes</td> <td>False Negative 30</td> <td>True Positive 270</td> </tr> <tr> <td></td> <td>No</td> <td>Yes</td> </tr> </table>	No	True Negative 1951	False Positive 49	Yes	False Negative 30	True Positive 270		No	Yes	 <p>Confusion matrix for Keras ResNet50 model:</p> <table border="1"> <tr> <td>No</td> <td>True Negative 1685</td> <td>False Positive 315</td> </tr> <tr> <td>Yes</td> <td>False Negative 187</td> <td>True Positive 113</td> </tr> <tr> <td></td> <td>No</td> <td>Yes</td> </tr> </table>	No	True Negative 1685	False Positive 315	Yes	False Negative 187	True Positive 113		No	Yes
No	True Negative 1951	False Positive 49																	
Yes	False Negative 30	True Positive 270																	
	No	Yes																	
No	True Negative 1685	False Positive 315																	
Yes	False Negative 187	True Positive 113																	
	No	Yes																	
Custom ResNet	Keras ResNet50																		
 <p>Confusion matrix for VGG-19 model:</p> <table border="1"> <tr> <td>No</td> <td>True Negative 1948</td> <td>False Positive 52</td> </tr> <tr> <td>Yes</td> <td>False Negative 29</td> <td>True Positive 271</td> </tr> <tr> <td></td> <td>No</td> <td>Yes</td> </tr> </table>	No	True Negative 1948	False Positive 52	Yes	False Negative 29	True Positive 271		No	Yes										
No	True Negative 1948	False Positive 52																	
Yes	False Negative 29	True Positive 271																	
	No	Yes																	
VGG-19																			

Figure C.4: Confusion matrix for some classification models with GB-All (28-06-2013) test set

APPENDIX C. MORE METRICS FOR THE CLASSIFICATION CNNs

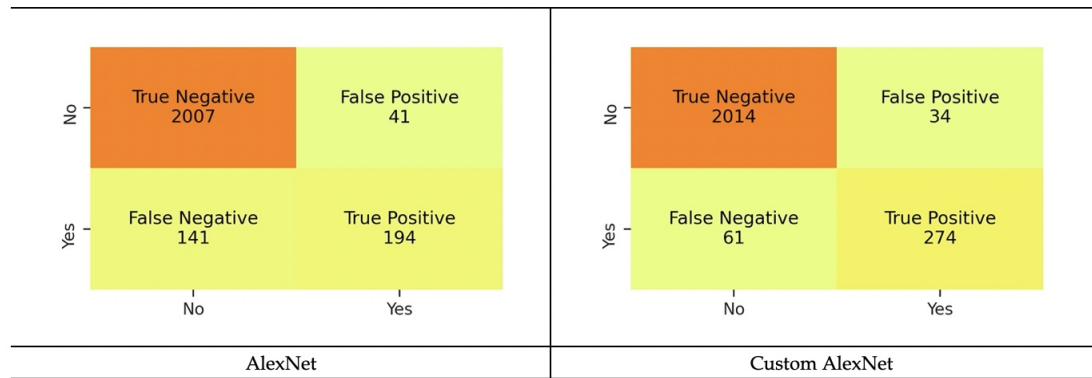


Figure C.5: Confusion matrix for AlexNet and Custom AlexNet with GB5-R5 test set

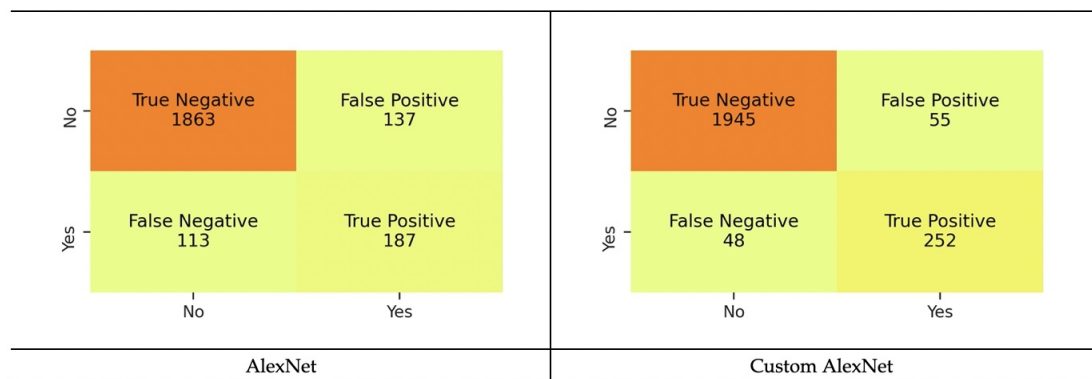


Figure C.6: Confusion matrix for AlexNet and Custom AlexNet with GB-All (28-06-2013) test set

APPENDIX C. MORE METRICS FOR THE CLASSIFICATION CNNs

	Recall	F2 Score	PR AUC		Recall	F2 Score	PR AUC
MOA-15	90.15%	90.15%	94.35%	Hybrid (b)	94.67%	90.50%	93.79%
Keras-VGG19	90.15%	89.93%	95.09%	Hybrid (a)	94.67%	89.25%	92.04%
MOA-14	90.15%	88.67%	93.26%	MOA-14	94.33%	89.50%	94.95%
Hybrid (a)	90.15%	87.94%	90.62%	MOA-15	94.00%	90.85%	95.74%
MOA-12	89.85%	89.16%	93.28%	MOA-12	94.00%	90.79%	95.77%
Hybrid (b)	89.55%	87.46%	90.54%	MOA-13	94.00%	90.38%	95.55%
MOA-13	88.36%	88.15%	93.34%	Hybrid (c)	94.00%	90.27%	94.86%
Custom ResNet	88.06%	88.48%	93.61%	Keras-VGG19	90.33%	88.97%	93.90%
Hybrid (c)	87.16%	86.65%	90.56%	Custom ResNet	90.00%	88.87%	93.81%
Custom Inception	85.37%	85.68%	92.84%	Custom Inception	89.00%	87.43%	93.51%
Keras-VGG16	83.88%	85.83%	92.98%	Custom AlexNet	84.00%	83.61%	90.18%
Custom AlexNet	81.79%	83.13%	89.76%	Keras-VGG16	82.33%	84.30%	90.69%
Keras-Inception	66.27%	66.47%	75.80%	Keras-Inception	67.33%	60.41%	59.39%
AlexNet	57.91%	61.59%	75.67%	AlexNet	62.33%	61.35%	67.82%
Keras-ResNet50	19.10%	21.95%	38.96%	Keras-ResNet50	37.67%	34.71%	28.03%
GB5-R5				GB-ALL			

Figure C.7: Evaluation metrics sorted by PR AUC, F2 Score, and Recall, highlighting the top five architectures

Appendix D

More Detections from YOLOv4 and YOLOv4-tiny

More detections by YOLOv4 are included in the figures in the next few pages.

Even though the mAP of YOLOv4 with default anchor boxes (Figure D.1) was 0.01% better than YOLOv4 with custom anchor boxes (Figure D.2), the latter found both faint tracklets in the leftmost image on the last row. Overall, both versions of YOLOv4 make detections with high confidence with both clear and hard-to-find tracklets. Consequently, both version of YOLOv4 will be used for localizing tracklets in predictions.

YOLOv4-tiny, while very fast, made lower confidence predictions and failed to find the tracklets in some images (Figures D.3 and D.4).

APPENDIX D. MORE DETECTIONS FROM YOLOV4 AND YOLOV4-TINY

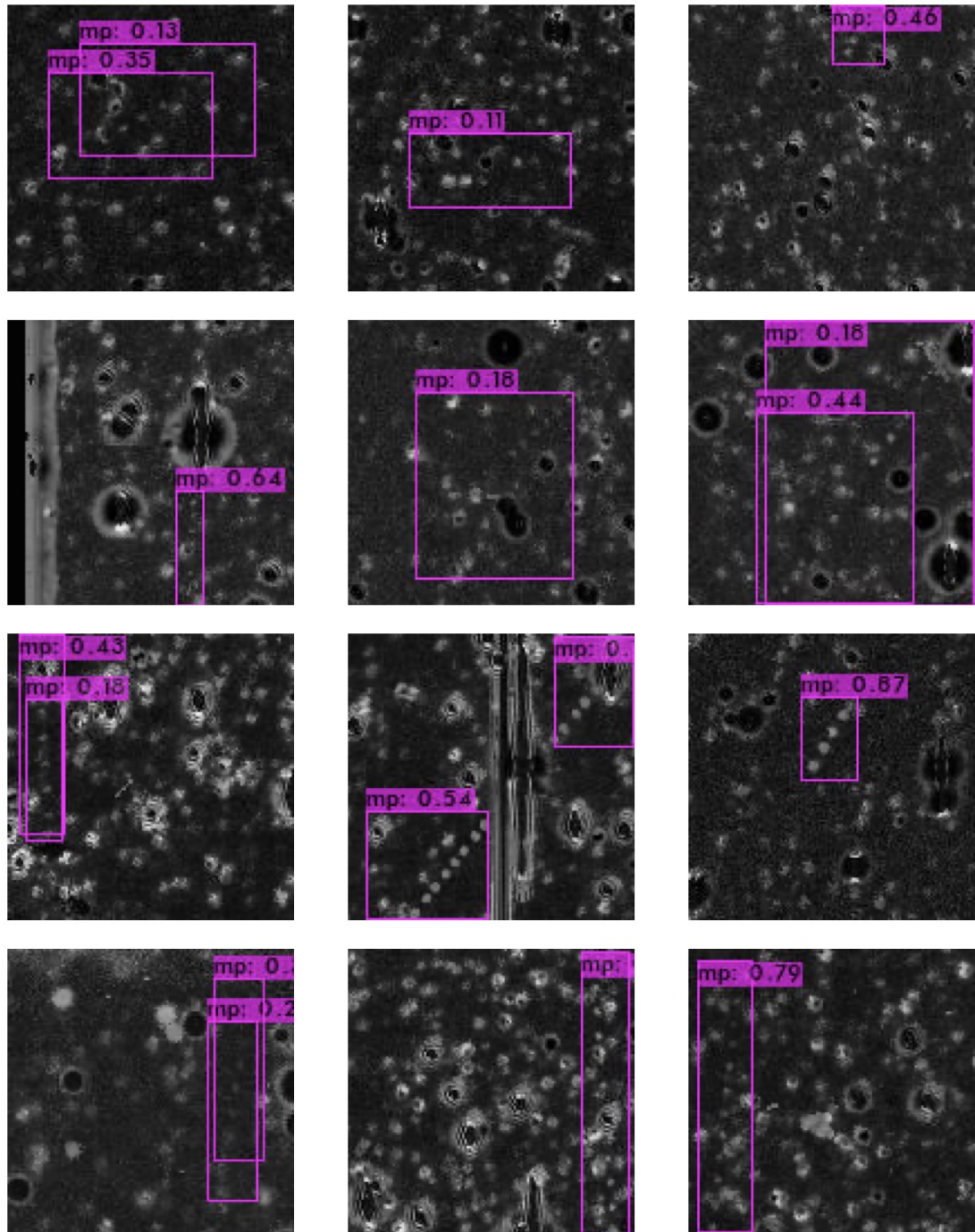


Figure D.1: Detections from YOLOv4 with default anchor boxes

APPENDIX D. MORE DETECTIONS FROM YOLOV4 AND YOLOV4-TINY

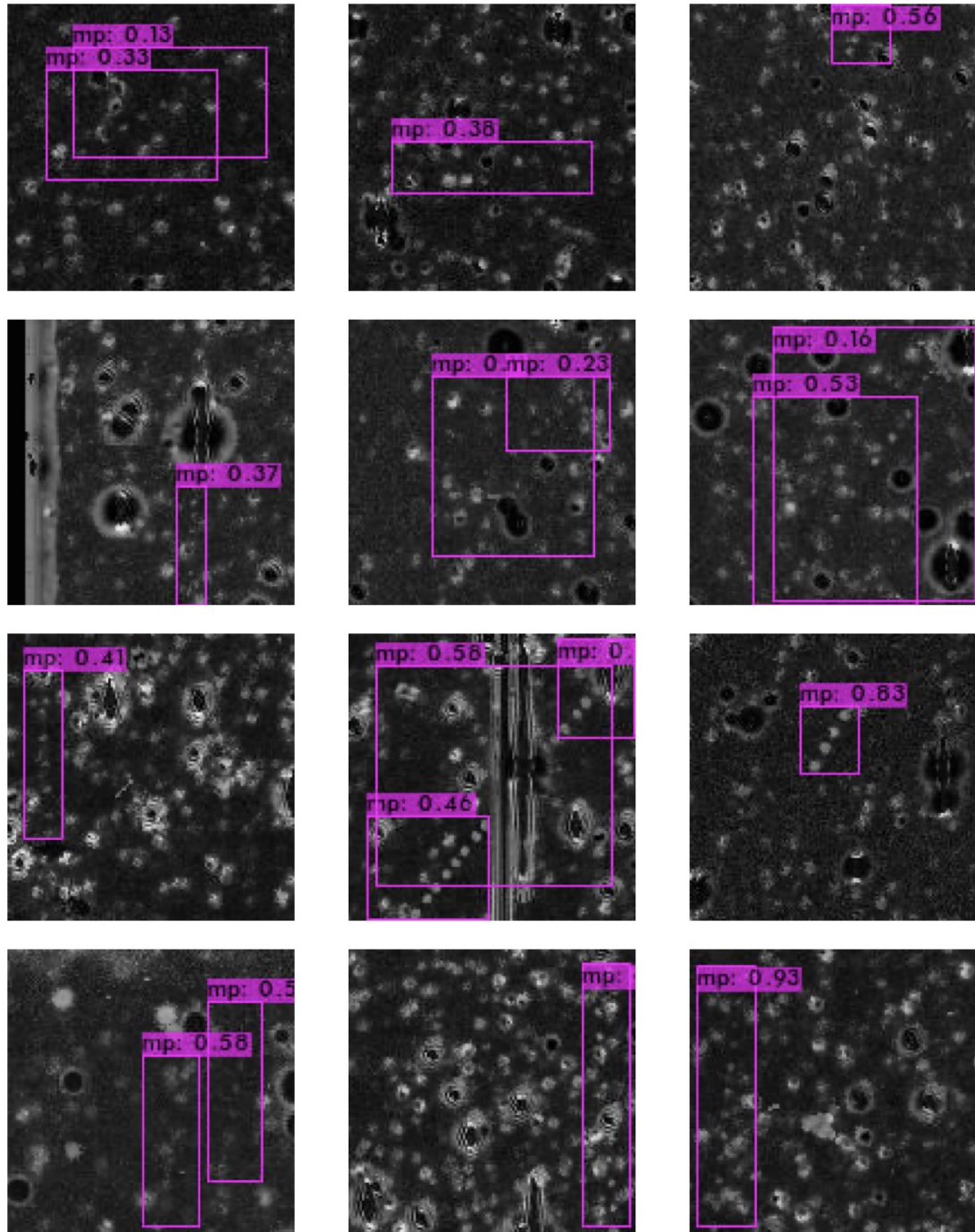


Figure D.2: Detections from YOLOv4 with custom anchor boxes; only one to find both faint tracklets in the leftmost image on the last row.

APPENDIX D. MORE DETECTIONS FROM YOLOV4 AND YOLOV4-TINY

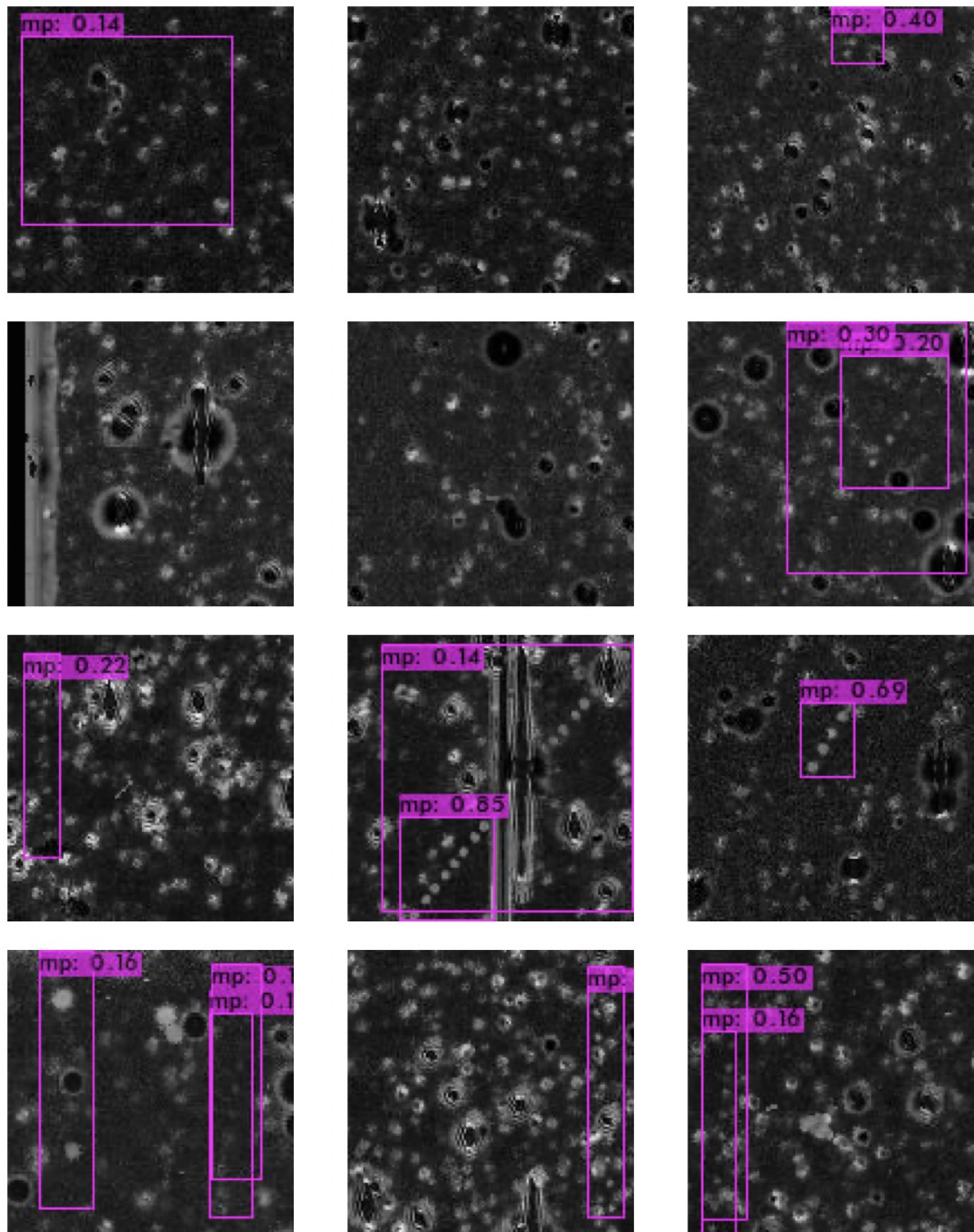


Figure D.3: Detections from YOLOv4-tiny with default anchor boxes

APPENDIX D. MORE DETECTIONS FROM YOLOV4 AND YOLOV4-TINY

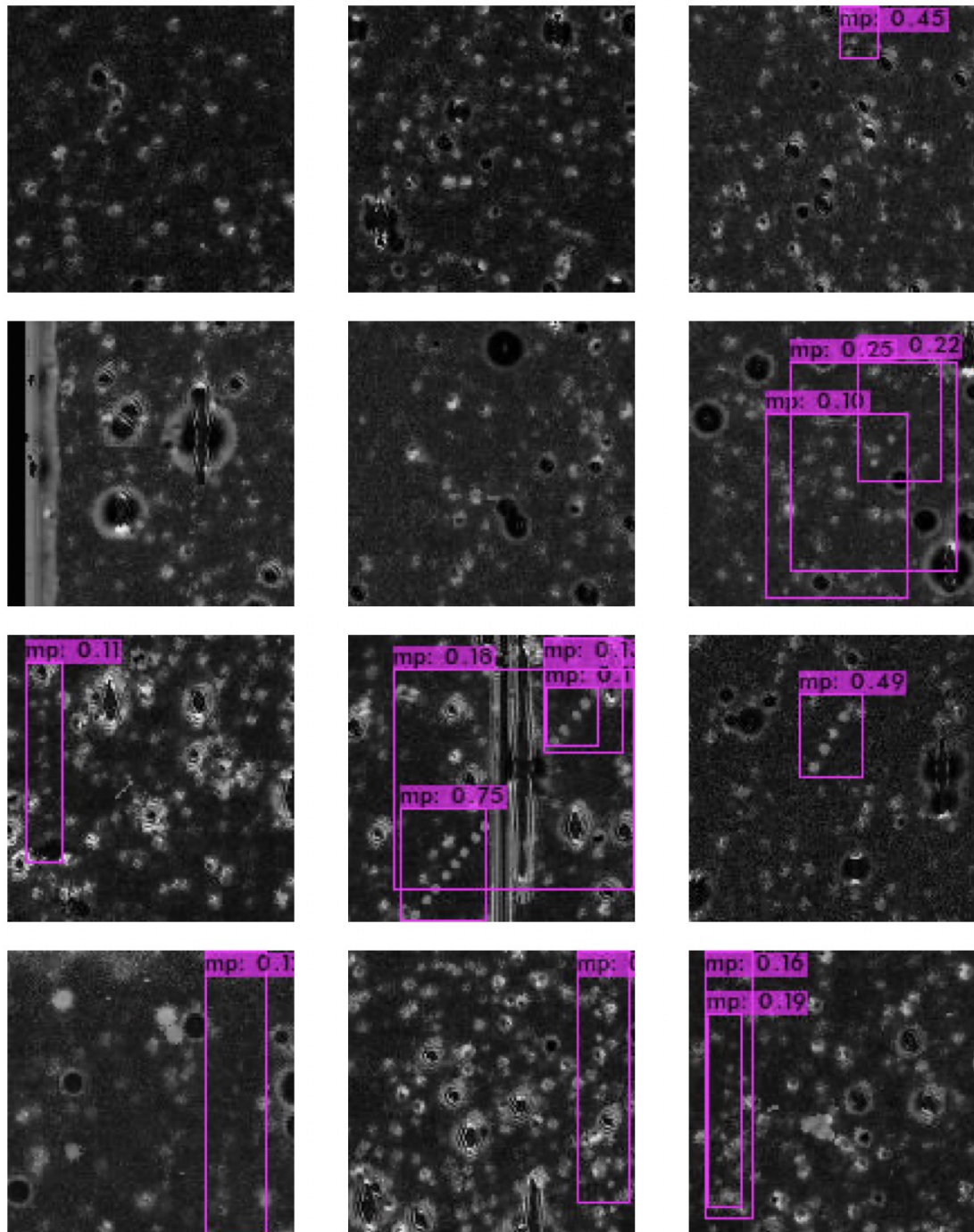


Figure D.4: Detections from YOLOv4-tiny with custom anchor boxes

Bibliography

- Alard, C. (2000). Image subtraction using a space-varying kernel. *Astronomy and Astrophysics Supplement*, 144:363–370.
- Alard, C. and Lupton, R. H. (1998). A Method for Optimal Image Subtraction. *The Astrophysical Journal*, 503(1):325–331.
- Alcock, C., Akerlof, C., Allsman, R., Axelrod, T., Bennett, D., Chan, S., Cook, K., Freeman, K., Griest, K., Marshall, S., Park, H. S., Perlmutter, S., Peterson, B., Pratt, M., Quinn, P., Rodgers, A., Stubbs, C., and Sutherland, W. (1993). Possible gravitational microlensing of a star in the Large Magellanic Cloud. *Nature*, 365(6447):621–623.
- Alvarez, L. W., Alvarez, W., Asaro, F., and Michel, H. V. (1980). Extraterrestrial cause for the Cretaceous-Tertiary extinction. *Science*, 208(4448):1095–1108.
- Armstrong, D. J., Pollacco, D., and Santerne, A. (2017). Transit shapes and self-organizing maps as a tool for ranking planetary candidates: Application to Kepler and K2. *Monthly Notices of the Royal Astronomical Society*, 465(3):2634–2642.
- Aubourg, E., Bareyre, P., Bréhin, S., Gros, M., Lachièze-Rey, M., Laurent, B., Lesquoy, E., Magneville, C., Milsztajn, A., Moscoso, L., Queinnec, F., Rich, J., Spiro, M., Vigroux, L., Zylberajch, S., Ansari, R., Cavalier, F., Moniez, M., Beaulieu, J. P., Ferlet, R., Grison, P., Vidal-Madjar, A., Guibert, J., Moreau, O., Tajahmady, F., Maurice, E., Prévôt, L., and Gry, C. (1993). Evidence for gravitational microlensing by dark objects in the Galactic halo. *Nature*, 365(6447):623–625.
- Ball, N. M. (2001). Morphological Classification of Galaxies Using Artificial Neural Networks. *ArXiv*, astro-ph/0110492.

BIBLIOGRAPHY

- Bazell, D. and Peng, Y. (1998). A Comparison of Neural Network Algorithms and Preprocessing Methods for Star-Galaxy Discrimination. *The Astrophysical Journal Supplement Series*, 116(1):47–55.
- Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2006). Greedy Layer-Wise Training of Deep Networks. *NIPS*.
- Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.
- Bennett, D. P. and Rhie, S. H. (1996). Detecting Earth-Mass Planets with Gravitational Microlensing. *The Astrophysical Journal*, 472(2):660–664.
- Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *ArXiv*, abs/2004.1.
- Bond, I., Abe, F., Dodd, R., Hearnshaw, J., Honda, M., Jugaku, J., Kilmartin, P., Marles, A., Masuda, K., Matsubara, Y., Muraki, Y., Nakamura, T., Nankivell, G., Noda, S., Noguchi, C., Ohnishi, K., Rattenbury, N., Reid, M., Saito, T., Sato, H., Sekiguchi, M., Skuljan, J., Sullivan, D., Sumi, T., Takeuti, M., Watase, Y., Wilkinson, S., Yamada, R., Yanagisawa, T., and Yock, P. (2001). Real-time difference imaging analysis of MOA Galactic bulge observations during 2000. *Monthly Notices of the Royal Astronomical Society*, 327(3):868–880.
- Bond, I. A. (2012). The first extrasolar planet detected via gravitational microlensing. *New Astronomy Reviews*, 56(1):25–32.
- Bond, I. A., Udalski, A., Jaroszynski, M., Rattenbury, N. J., Paczynski, B., Soszynski, I., Wyrzykowski, L., Szymanski, M. K., Kubiak, M., Szewczyk, O., Zebun, K., Pietrzynski, G., Abe, F., Bennett, D. P., Eguchi, S., Furuta, Y., Hearnshaw, J. B., Kamiya, K., Kilmartin, P. M., Kurata, Y., Masuda, K., Matsubara, Y., Muraki, Y., Noda, S., Okajima, K., Sako, T., Sekiguchi, T., Sullivan, D. J., Sumi, T., Tristram, P. J., Yanagisawa, T., Yock, P. C. M., and Collaboration, O. (2004). OGLE 2003-BLG-235/MOA 2003-BLG-53: A planetary microlensing event. *The Astrophysical Journal, Volume 606, Issue 2, pp. L155-L158.*, 606:L155–L158.
- Bramich, D. (2008). A new algorithm for difference image analysis. *Monthly Notices of the Royal Astronomical Society*, 386(1):L77–L81.

BIBLIOGRAPHY

- Cabrera-Vives, G., Reyes, I., Förster, F., Estévez, P. A., and Maureira, J.-C. (2017). Deep-HiTS: Rotation Invariant Convolutional Neural Network for Transient Detection. *The Astrophysical Journal*, 836(1):97.
- Cavanagh, M. K., Bekki, K., and Groves, B. A. (2021). Morphological classification of galaxies with deep learning: comparing 3-way and 4-way CNNs. *MNRAS*, 000:1–18.
- Charnock, T. and Moss, A. (2016). Deep Recurrent Neural Networks for Supernovae Classification. *The Astrophysical Journal*, 837.
- Chollet, F. (2021). *Deep Learning with Python, Second Edition*. Manning Publications.
- Cochran, A. L., Levison, H. F., Stern, S. A., and Duncan, M. J. (1995). The Discovery of Halley-sized Kuiper Belt Objects Using HST. *The Astrophysical Journal*, 455:342.
- Dai, J., Li, Y., He, K., and Sun, J. (2016). R-FCN: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*, pages 379–387.
- Denneau, L., Jedicke, R., Grav, T., Granvik, M., Kubica, J., Milani, A., Vereš, P., Wainscoat, R., Chang, D., Pierfederici, F., Kaiser, N., Chambers, K. C., Heasley, J. N., Magnier, E. A., Price, P. A., Myers, J., Kleyna, J., Hsieh, H., Farnocchia, D., Waters, C., Sweeney, W. H., Green, D., Bolin, B., Burgett, W. S., Morgan, J. S., Tonry, J. L., Hodapp, K. W., Chastel, S., Chesley, S., Fitzsimmons, A., Holman, M., Spahr, T., Tholen, D., Williams, G. V., Abe, S., Armstrong, J. D., Bressi, T. H., Holmes, R., Lister, T., McMillan, R. S., Micheli, M., Ryan, E. V., Ryan, W. H., and Scotti, J. V. (2013). The Pan-STARRS Moving Object Processing System. *Publications of the Astronomical Society of the Pacific*, 125(926):357–395.
- Dieleman, S., Willett, K. W., and Dambre, J. (2015). Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly Notices of the Royal Astronomical Society*, 450(2):1441–1459.
- Djorgovski, S. G., Graham, M. J., Donalek, C., Mahabal, A. A., Drake, A. J., Turmon, M., and Fuchs, T. (2016). Real-time data mining of massive data

BIBLIOGRAPHY

- streams from synoptic sky surveys. *Future Generation Computer Systems*, 59:95–104.
- Donahue, J., Hendricks, L. A., Rohrbach, M., Venugopalan, S., Guadarrama, S., Saenko, K., and Darrell, T. (2014). Long-term Recurrent Convolutional Networks for Visual Recognition and Description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):677–691.
- Duev, D. A., Bolin, B. T., Graham, M. J., Kelley, M. S. P., Mahabal, A., Bellm, E. C., Coughlin, M. W., Dekany, R., Helou, G., Kulkarni, S. R., Masci, F. J., Prince, T. A., Riddle, R., Soumagnac, M. T., and van der Walt, S. J. (2021). Tails: Chasing Comets with the Zwicky Transient Facility and Deep Learning. *The Astronomical Journal*, 161(5):218.
- Duev, D. A., Mahabal, A., Masci, F. J., Graham, M. J., Rusholme, B., Walters, R., Karmarkar, I., Frederick, S., Kasliwal, M. M., Rebbapragada, U., and Ward, C. (2019a). Real-bogus classification for the Zwicky Transient Facility using deep learning. *Monthly Notices of the Royal Astronomical Society*, 489(3):3582–3590.
- Duev, D. A., Mahabal, A., Ye, Q., Tirumala, K., Belicki, J., Dekany, R., Frederick, S., Graham, M. J., Laher, R. R., Masci, F. J., Prince, T. A., Riddle, R., Rosnet, P., and Soumagnac, M. T. (2019b). DeepStreaks: Identifying fast-moving objects in the Zwicky Transient Facility data with deep learning. *Monthly Notices of the Royal Astronomical Society*, 486(3):4158–4165.
- Einstein, A. (1936). Lens-Like Action of a Star by the Deviation of Light in the Gravitational Field. *Science*, 84(2188):506–507.
- Evans, R. W., Stapelfeldt, K. R., Peters, D. P., Trauger, J. T., Padgett, D. L., Ballester, G. E., Burrows, C. J., Clarke, J. T., Crisp, D., Gallagher, J. S., Griffiths, R. E., Grillmair, C., Hester, J. J., Hoessel, J. G., Holtzmann, J., Krist, J., McMaster, M., Meadows, V., Mould, J. R., Ostrander, E., Sahai, R., Scowen, P. A., Watson, A. M., and Westphal, J. (1998). Asteroid Trails in Hubble Space Telescope WFPC2 Images: First Results. *Icarus*, 131(2):261–282.
- Everingham, M., Van Gool, L., Williams, C. K., Winn, J., and Zisserman, A. (2009). The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision 2009 88:2*, 88(2):303–338.

BIBLIOGRAPHY

- George, D. and Huerta, E. A. (2018). Deep Learning for real-time gravitational wave detection and parameter estimation: Results with Advanced LIGO data. *Physics Letters, Section B: Nuclear, Elementary Particle and High-Energy Physics*, 778:64–70.
- Gieseke, F., Bloemen, S., van den Bogaard, C., Heskes, T., Kindler, J., Scalzo, R. A., Ribeiro, V. A. R. M., van Roestel, J., Groot, P. J., Yuan, F., Möller, A., and Tucker, B. E. (2017). Convolutional neural networks for transient candidate vetting in large-scale surveys. *Monthly Notices of the Royal Astronomical Society*, 472(3):3101–3114.
- Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 580–587.
- Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Journal of Machine Learning Research*, volume 9, pages 249–256.
- Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep Sparse Rectifier Neural Networks. *AISTATS*.
- Gould, A. and Yee, J. C. (2013). Microlens surveys are a powerful probe of asteroids. *Astrophysical Journal*, 767(1).
- Gould, A. P. and Loeb (1992). Discovering planetary systems through gravitational microlenses. *Astrophysical Journal*, 396:104.
- Griest, K., Alcock, C., Axelrod, T. S., Bennett, D. P., Cook, K. H., Freeman, K. C., Park, H.-S., Perlmutter, S., Peterson, B. A., Quinn, P. J., Rodgers, A. W., Stubbs, C. W., Collaboration, M., Griest, K., Alcock, C., Axelrod, T. S., Bennett, D. P., Cook, K. H., Freeman, K. C., Park, H.-S., Perlmutter, S., Peterson, B. A., Quinn, P. J., Rodgers, A. W., Stubbs, C. W., and Collaboration, M. (1991). Gravitational Microlensing as a Method of Detecting Disk Dark Matter and Faint Disk Stars. *Astrophysical Journal Letters*, 372:L79.
- He, K., Zhang, X., Ren, S., and Sun, J. (2014). Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *Lecture Notes in Computer*

BIBLIOGRAPHY

- Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8691 LNCS(PART 3):346–361.
- He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.
- Helin, E. F. and Shoemaker, E. M. (1979). The Palomar planet-crossing asteroid survey, 1973-1978. *Icarus*, 40(3):321–328.
- Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A Fast Learning Algorithm for Deep Belief Nets. *Neural Computation*, 18(7):1527–1554.
- Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.
- Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *ArXiv*, abs/1704.0.
- Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2016). Densely Connected Convolutional Networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-Janua*:2261–2269.
- Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *32nd International Conference on Machine Learning, ICML 2015*, volume 1, pages 448–456.
- Jones, R. L., Jurić, M., and Ivezić, Ž. (2016). Asteroid Discovery and Characterization with the Large Synoptic Survey Telescope. *Proceedings of the International Astronomical Union*, 10:282–292.
- Kim, E. J. and Brunner, R. J. (2017). Star-galaxy classification using deep convolutional neural networks. *Monthly Notices of the Royal Astronomical Society*, 464(4):4463–4475.
- Kim, S.-L., Lee, C.-U., Park, B.-G., Kim, D.-J., Cha, S.-M., Lee, Y., Han, C., Chun, M.-Y., and Yuk, I. (2016). KMTNET: A Network of 1.6 m Wide-Field Optical Telescopes Installed at Three Southern Observatories. *Journal of Korean Astronomical Society*, 49(1):37–44.

BIBLIOGRAPHY

- Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR.
- Krizhevsky, A., Sutskever, I., and Geoffrey E., H. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems 25 (NIPS2012)*.
- Kruk, S., García Martín, P., Popescu, M., Merín, B., Mahlke, M., Carry, B., Thomson, R., Karadağ, S., Durán, J., Racero, E., Giordano, F., Baines, D., de Marchi, G., and Laureijs, R. (2022). Hubble Asteroid Hunter. *Astronomy & Astrophysics*, 661:A85.
- Kubica, J., Denneau Jr, L., Moore, A., Jedicke, Robert, and Connolly, A. (2007). Efficient Algorithms for Large-Scale Asteroid Discovery. *Astronomical Data Analysis Software and Systems XVI ASP Conference Series*, 376:395–404.
- Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- Liebess, S. (1964). Gravitational Lenses. *Physical Review*, 133(3B):835–844.
- Lieu, M., Conversi, L., Altieri, B., and Carry, B. (2018). Detecting solar system objects with convolutional neural networks. *Monthly Notices of the Royal Astronomical Society*, 485(4):5831–5842.
- Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017a). Feature pyramid networks for object detection. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 936–944.
- Lin, T.-y., Goyal, P., Girshick, R., He, K., and Dollar, P. (2017b). Focal Loss for Dense Object Detection. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007. IEEE.
- Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. In

BIBLIOGRAPHY

- Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8693 LNCS, pages 740–755. Springer Verlag.
- Liu, S., Qi, L., Qin, H., Shi, J., and Jia, J. (2018). Path Aggregation Network for Instance Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8759–8768.
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., and Berg, A. C. (2016). SSD: Single shot multibox detector. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 9905 LNCS, pages 21–37.
- Magnier, E. (2006). The Pan-STARRS PS1 Image Processing Pipeline. In *The Advanced Maui Optical and Space Surveillance Technologies Conference*.
- Mahabal, A., Sheth, K., Gieseke, F., Pai, A., Djorgovski, S. G., Drake, A. J., and Graham, M. J. (2017). Deep-learnt classification of light curves. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, volume 2018-Jan, pages 1–8. IEEE.
- Mao, S., Paczynski, B., Mao, S., and Paczynski, B. (1991). Gravitational Microlensing by Double Stars and Planetary Systems. *Astrophysical Journal Letters*, 374:L37.
- Misra, D. (2020). Mish: A Self Regularized Non-Monotonic Activation Function. In *BMVC*.
- Naim, A. (1995). Morphological Classification of Galaxies By Artificial Neural Networks. *Proceedings of the RGO-ESA Workshop Future Possibilities for Astrometry in Space*.
- Nair, V. and Hinton, G. E. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. *ICML*.
- Naul, B., Bloom, J. S., Pérez, F., and van der Walt, S. (2018). A recurrent neural network for classification of unevenly sampled variable stars. *Nature Astronomy*, 2(2):151–155.

BIBLIOGRAPHY

- Odehahn, S. C., Stockwell, E. B., Pennington, R. L., Humphreys, R. M., and Zumach, W. A. (1992). Automated star/galaxy discrimination with neural networks. *Astronomical Journal*, 103:318–331.
- Paczynski, B. (1986). Gravitational Microlensing by the Galactic Halo. *Astrophysical Journal*, 304:1.
- Pasquet-Itam, J. and Pasquet, J. (2018). Deep learning approach for classifying, detecting and predicting photometric redshifts of quasars in the Sloan Digital Sky Survey stripe 82. *Astronomy and Astrophysics*, 611:A97.
- Rabeendran, A. C. and Denneau, L. (2021). A Two-Stage Deep Learning Detection Classifier for the ATLAS Asteroid Survey. *Publications of the Astronomical Society of the Pacific*, 133(1021).
- Rabinowitz, D. L. (1991). Detection of earth-approaching asteroids in near real time. *The Astronomical Journal*, 101(4):1518.
- Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788. IEEE.
- Redmon, J. and Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525. IEEE.
- Redmon, J. and Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *ArXiv*, abs/1804.0.
- Ren, S., He, K., Girshick, R., and Sun, J. (2017). Faster R-CNN: Towards Real-Time Object Detection with. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149.
- Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.
- Sako, T., Sekiguchi, T., Sasaki, M., Okajima, K., Abe, F., Bond, I. A., Hearnshaw, J. B., Itow, Y., Kamiya, K., Kilmartin, P. M., Masuda, K., Matsubara, Y., Muraki, Y., Rattenbury, N. J., Sullivan, D. J., Sumi, T., Tristram, P., Yanagisawa, T., and Yock, P. C. (2008). MOA-cam3: A wide-field mosaic

BIBLIOGRAPHY

- CCD camera for a gravitational microlensing survey in New Zealand. *Experimental Astronomy*, 22(1-2):51–66.
- Sedaghat, N. and Mahabal, A. (2018). Effective image differencing with convolutional neural networks for real-time transient hunting. *Monthly Notices of the Royal Astronomical Society*, 476(4):5365–5376.
- Shallue, C. J. and Vanderburg, A. (2018). Identifying Exoplanets with Deep Learning: A Five-planet Resonant Chain around Kepler-80 and an Eighth Planet around Kepler-90. *The Astronomical Journal*, 155(2):94.
- Shoemaker, C. S., Shoemaker, E. M., Nasa, L., and Planetary Institute Tx, United States,, H. (1988). The Palomar Asteroid and Comet Survey (PACS), 1982-1987. *Lunar and planetary science XIX; abstracts of papers submitted to the Nineteenth lunar and planetary science conference.*, 19 Part 3:1077–1078.
- Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.
- Sumi, T., Abe, F., Bond, I., Dodd, R., Hearnshaw, J., Honda, M., Honma, M., Kan-ya, Y., Kilmartin, P., Masuda, K., Matsubara, Y., Muraki, Y., Nakamura, T., Nishi, R., Noda, S., Ohnishi, K., Petterson, O., Rattenbury, N., Reid, M., Saito, T., Saito, Y., Sato, H., Sekiguchi, M., Skuljan, J., Sullivan, D., Takeuti, M., Tristram, P., Wilkinson, S., Yanagisawa, T., and Yock, P. (2003). Microlensing Optical Depth toward the Galactic Bulge from Microlensing Observations in Astrophysics Group Observations during 2000 with Difference Image Analysis. *The Astrophysical Journal*, 591(1):204–227.
- Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pages 4278–4284.
- Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015a). Going deeper with convolutions.

BIBLIOGRAPHY

- In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June.
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2015b). Rethinking the Inception Architecture for Computer Vision. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-Decem:2818–2826.
- Tan, M. and Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *36th International Conference on Machine Learning, ICML 2019*, 2019-June:10691–10700.
- Tan, M., Pang, R., and Le, Q. V. (2020). EfficientDet: Scalable and efficient object detection. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 10778–10787.
- Tomaney, A. B. and Crofts, A. P. (1996). Expanding the Realm of Microlensing Surveys with Difference Image Photometry. *Astronomical Journal*, 112:2872.
- Turing, A. M. (1950). Computing Machinery and Intelligence. *Mind*, 49(236):433–460.
- Udalski, A., Szymanski, M., Kaluzny, J., Kubiak, M., Krzeminski, W., Mateo, M., Preston, G., and Paczynski, B. (1993). The Optical Gravitational Lensing Experiment. Discovery of the First Candidate Microlensing Event in the Direction of the Galactic Bulge. *Acta Astronomica*, 43:289–294.
- W. M. Newman and R. F. Sproull (1973). Cohen-Sutherland Algorithm. In *Principles of Interactive Computer Graphics*, pages 124, 252. McGraw-Hill Education, internatio edition.
- Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2021). Scaled-YOLOv4: Scaling Cross Stage Partial Network. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13024–13033. IEEE.
- Wang, C. Y., Mark Liao, H. Y., Wu, Y. H., Chen, P. Y., Hsieh, J. W., and Yeh, I. H. (2020). CSPNet: A new backbone that can enhance learning capability of CNN. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, volume 2020-June, pages 1571–1580.

BIBLIOGRAPHY

- Williams, R. J. and Peng, J. (1990). An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories. *Neural Computation*, 2(4):490–501.
- Woo, S., Park, J., Lee, J. Y., and Kweon, I. S. (2018). CBAM: Convolutional Block Attention Module. *Lecture Notes in Computer Science (including sub-series Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11211 LNCS:3–19.
- Ye, Q., Masci, F. J., Lin, H. W., Bolin, B., Chang, C. K., Duev, D. A., Helou, G., Ip, W. H., Kaplan, D. L., Kramer, E., Mahabal, A., Ngeow, C. C., Nielsen, A. J., Prince, T. A., Tan, H., Yeh, T. S., Bellm, E. C., Dekany, R., Giomi, M., Graham, M. J., Kulkarni, S. R., Kupfer, T., Laher, R. R., Rusholme, B., Shupe, D. L., and Ward, C. (2019). Toward efficient detection of small Near-Earth asteroids using the Zwicky Transient Facility (ZTF). *Publications of the Astronomical Society of the Pacific*, 131(1001).
- Zheng, Z., Wang, P., Ren, D., Liu, W., Ye, R., Hu, Q., and Zuo, W. (2020). Enhancing Geometric Factors in Model Learning and Inference for Object Detection and Instance Segmentation. *IEEE Transactions on Cybernetics*, XX(1).
- Zoghbi, S., Cicco, M. D., Ordonez, A. J., Stapper, A. P., Collison, J., Gural, P. S., Ganju, S., Galache, J.-l., and Jenniskens, P. (2017). Searching for Long-Period Comets with Deep Learning Tools. In *Workshop on Deep Learning for Physical Sciences*.
- Zoph, B., Vasudevan, V., Shlens, J., and Le, Q. V. (2017). Learning Transferable Architectures for Scalable Image Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8697–8710.