National Technical University of Athens
School of Electrical and Computer Engineering
Machine Learning and Data Science
Division of Signals, Control and Robotics

Neuro-Symbolic Visual Question Answering for Physical Interaction Understanding and Dynamic Scene Interpretation

Master Thesis

Myron Sampsakis-Bakopoulos

Supervisor: Dr. Haris Papageorgiou, Research Director, ATHENA RC
Co-Supervisor: Alexandros Potamianos, Associate Professor, NTUA

March 2021, Athens

National Technical University of Athens
School of Electrical and Computer Engineering
Machine Learning and Data Science
Division of Signals, Control and Robotics

Neuro-Symbolic Visual Question Answering for Physical Interaction Understanding and Dynamic Scene Interpretation

Νευρο-Συμβολική Απάντηση Οπτικών Ερωτήσεων για την Κατανόηση Αλληλεπιδράσεων και Επεξήγηση Δυναμικών Σκηνών

Master Thesis
Μεταπτυχιακή Διπλωματική Εργασία

Myron Sampsakis-Bakopoulos
Μύρων Σαμψάκης-Μπακόπουλος

Supervisor: Dr. Haris Papageorgiou, Research Director, ATHENA RC
Co-Supervisor: Alexandros Potamianos, Associate Professor, NTUA
Επιβλέπων: Δρ. Χάρης Παπαγεωργίου, Διευθυντής Ερευνών, ΕΚ ΑΘΗΝΑ
Συνεπιβλέπων: Αλέξανδρος Ποταμιάνος, Αναπληρωτής Καθηγητής, ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 29η Μαρτίου 2021.
Approved by the examination committee on the 29th of March 2021.

.....	.....	.....
Alexandros Potamianos	Haris Papageorgiou	Andreas-Georgios Stafylopatis
Associate Professor	Research Director	Professor
NTUA	ATHENA R.C.	NTUA

.....
Μύρων Σαμψάκης-Μπακόπουλος
Διπλωματούχος Μηχανολόγος-Μηχανικός, Ε.Μ.Π.

To all the people that have cared for me...

Extended Summary

Neuro-Symbolic Visual Question Answering for Physical Interaction Understanding and Dynamic Scene Interpretation

Problem Description

Visual Question Answering is a field of Artificial Intelligence that involves numerous of its sectors, like Natural Language Processing and Computer Vision. Teaching a machine to understand a scene, either static (image) or dynamic (video) has improved vastly within the last years. VQA problems are usually given as a set of images and questions, with the algorithm aimed at providing the correct answer. In most cases, this is solved by encoding the image and the question with a CNN, and a Sequential Encoder respectively and "choosing" an answer in the form of a multiple-choice classification task through another neural network. However, the state-space of the answers is strictly defined, and new answers cannot be inferred with the models current knowledge. This can in some cases be replaced by a special type of neural network, a Decoder, that will provide answers in an open-ended format, but at the same time severely complicating the problem.

The images can also be generalized to videos and the questions can ask about events that take place that are shown in these videos, as well as higher level information that can be extracted through the interaction of the events themselves. These answers while not always hard to answer by a human, cannot be answered by a machine that hasn't been trained with thousands upon thousands of samples.

The above models will usually rely on huge amounts of data, as the networks rely on projecting the meanings of the images and the questions in a latent space and -in turn- re-projecting them in an interpretable space in the form of an understandable answer. These statistical models will rely on costly datasets and will require training on powerful machines for extended periods of time.

Intrigued by the above difficulties, we propose a new dataset, accompanied by a new model, that aims to simulate such events, as well as provide solutions to the above problems. The "**Ph**ysics **A**nswers with **Q**uestions" (**PhAQ**) dataset simulates pseudo-3D ball interactions that follow Newtonian physics on a table, while interacting with elements such as obstacles of various shapes, walls, and gravity distortions. At the same time, we developed "**Newto**n with **N**eural **N**etworks" (**NewtoNN**), a Neuro-Symbolic VQA answering

ensemble model, that aims to interpret and understand the interactions inside each simulation and in turn answer the questions given to it with a hybrid approach, a Neuro-Symbolic approach. This ensemble model utilizes both neural techniques that rely on neural networks in order to extract characteristics and labels from the given videos, while simultaneously utilizing symbolic or else rule-based techniques, in order to combine the various modules of the network.

This approach apart from providing fully interpretable intermediate steps in the ensemble model, has the added benefit of requiring *far less* data in order to train and infer upon given samples. This is due to the fact that many rules, which would need to be statistically approached in a purely neural model, are now given in the form of symbols and commands imposed during the intermediate stages of the models development.

Related Work

Neuro-Symbolic approaches are a new hybrid architecture comprised of two old -and at many times opposing- techniques: Neural Networks and Symbolic Reasoning. It aims to combine the best parts of its components, the raw data management capabilities of Neural Networks, and the compositional and causal structure understanding of Symbolic models.

They have been used in VQA tasks were there is limited data and complicated questions to be answered, and especially where object characteristics are necessary. Neuro-Symbolic VQA tasks will utilize aspects from neural networks, mainly their ability to encode images (CNNs) and sequences (RNNs). One of the best known datasets created for Neuro-Symbolic VQA is the CLEVR [25] dataset which contains pairs of images with items of various shapes and sizes, and questions regarding attributional and spatial information. The questions are translated to a series of programs, which are in turn applied on the interpreted video-frame as extracted by MaskRCNN, in order to provide answers in Natural Language. Further expanding this, the NeuroSymbolic Concept Learner [35] is able to learn visual concepts and semantic parsing of a sentence without explicit supervision, simply by looking at the image and create the programs itself. Introducing temporal information, CLEVRER [54] adds simple moving items that collide together. Due to the simplicity of the simulations, it can additionally train a predictor in order to reply to counterfactual and predictive questions. Other attempts have been made to combine physical understanding with neural networks, such as the case were physical concepts are attempted to be discovered with neural networks [24] given their respective simulations.

This work aims to combine elements from all the above cases, as well as expand certain notions that are introduced in each of the abovementioned works.

Dataset and Model Description

PhAQ consists of procedurally generated videos, that are accompanied with questions from 10 categories with many questions per category and many rephrases per question. Along with each question, a set of symbolic commands called "programs" is generated. The simulations, while random (procedural) are broken into eight (8) different difficulties, with

each difficulty having a different combination of number of balls, walls, obstacles and gravity fields. The dataset is aimed to be visually simple. Each sample consists of a maximum of 10 seconds of simulation, during which all events are recorded. These events are *ball-to-ball*, *ball-to-wall* and *ball-to-obstacle* collisions, and *ball exits*. These events are written to a logfile, which is part of the generated files of each simulations. Along the logfile, small data-tables that include information regarding the balls, the walls, the obstacles and the gravity fields are also generated.

The questions in the dataset are procedurally generated as well. They consists of templated expressions, in which attributes like color, size, shape and side are replaced based on the current simulation. A similar technique is applied to the templates of their respective programmes.

The dataset generation can be fully parallelized to an arbitrary number of threads and is carefully RNG managed, in order to be fully reproducible.

NewtoNN is an ensemble model. That is a model comprised of independent submodels (modules), with each module tied to a specific task. There are different modules for the Frame Interpretation, the Dynamics Predictor, the mass and friction estimators, the Neuro-Symbolic Translator and the Symbolic Executor. The total architecture of the ensemble model can be seen in Figure 0.1.

The Frame Interpreter consists of a paired MaskRCNN [21] for spatial and categorical inference (mask and labels), while an SVM is used in order to predict the color of each entity. Each video is broken into its respective frames, with each frame passing through the MaskRCNN. The interpreted sequence of frames forms the input of the next stage of the algorithm, the Dynamics Predictor, which is in practice a modified PropNet [32], that aims to understand the events taking place in each simulation. The Neuro-Symbolic translator is a sequence-to-sequence model, which uses a GRU unit in its encoder and decoder, with the latter utilizing an attention mechanism. It aims to translate a sentence from Natural Language, to a new *Symbolic* Language (a Domain Specific Language), an order of commands that aims to extract the answer from the output of the Dynamics Predictor. Finally, the Symbolic Executor receives as an input the data tables that are generated by the Dynamics Predictor and in turn, by utilizing the sequence of commands -which are in fact various operations in one or more data-tables- can extract and create an answer in Natural Language.

Each module is trained separately. Since the Symbolic Executor is purely symbolic, it doesn't require any training, as its inference is purely rule-based. The modular nature of the model can help develop, optimize and evaluate every model separately, as well as give us interpretable results in every intermediate steps.

Experimental Evaluation

The training of all the models takes place separately. The Frame Interpreter mostly consists of a MaskRCNN. Since MaskRCNNs are hard and very data intensive to train, we fine-tune a pre-trained MaskRCNN with a version of our dataset that contains the masks and labels of randomly sampled frames from the simulations. A total of 900 frames are used as training test, with a validation set that consists of another 180. In total, we achieve a classification accuracy above 99%.

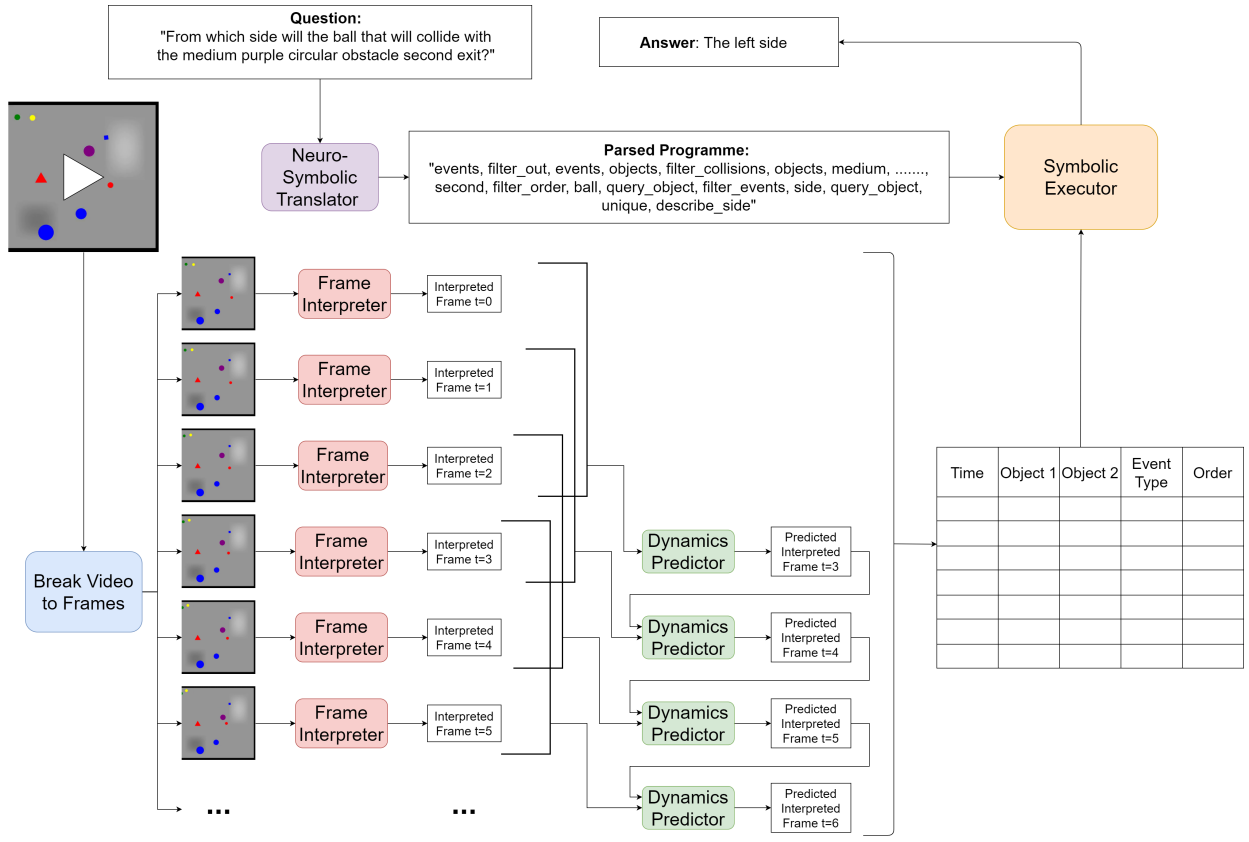

Figure 0.1: End-to-end process for answering questions. Each different element is shown with a different color. *Blue* is the video-to-frame breaker, *red* is the frame interpreter (MaskR-CNN, with color classifier, and all other related code), *green* is the Dynamics Predictor (PropNet), *purple* is the sequence-to-sequence model (Attention-GRU), and *orange* is the Symbolic Executor.

The next model to be trained is the Dynamics Predictor. For this purpose a dataset that consists of 240 samples is generated, with 160:40:40 split for the train, validation and test set respectively. The training takes approximately 2 days in a mid-range GPU (GTX 1650). We can evaluate on a per-event basis. An event is considered detected when it its predicted timestep is the same (within a small margin of error, depending on the sampling rate of PropNet) as the actual event, with the same participants. In the cases of *ball-to-wall* collisions and *ball exits*, the event detection precision was above 90% in all three sets. However, the other events such as *ball-to-ball* and *ball-to-obstacle* detection had a smaller precision, with around 65% and 80% precision respectively, in both the validation and test sets. In all cases, the recall was significantly higher. This reduced precision is more than expected, since the participants are items with variable coordinates, while in the case of *ball-to-wall* and *ball exits* the walls and the sides are stationary and invariant inbetween simulations.

Two different networks have also been utilized in order to predict the masses of the balls within the simulation, simply by looking at their behavior, as well as the type and level of resistance that the balls are subject to. In both these cases, the datasets used were far larger,

and in the case of mass prediction the accuracy could surpass the baseline by about 15-20%, while in the case of friction prediction the accuracy fared off better. However, generalization in the latter cases where gravity fields were included, yielded lower accuracy (as expected) since the resistance estimation relies heavily on the trajectory of the balls which is heavily altered when interacting with a gravity field.

The Neuro-Symbolic Translator can be trained with the already existing datasets. Variable train sets are used in order to test its accuracy. With train datasets bigger than 5k samples, it can generalize extremely well, surpassing 98% accuracy in generating the correct sequence of commands of a new, unseen natural language question.

Question Family	Question ID	Levels							
		1	2	3	4	5	6	7	8
Compare Integer	1.0	95.83	98.33	100.00	97.50	100.00	97.50	100.00	100.00
	1.1	100.00	98.75	100.00	100.00	100.00	100.00	100.00	100.00
	1.2	97.50	97.50	100.00	98.75	100.00	98.75	100.00	100.00
	1.3	-	-	-	100.00	100.00	97.32	100.00	100.00
	1.4	-	-	-	100.00	100.00	97.47	100.00	100.00
	1.5	-	-	-	100.00	100.00	97.50	100.00	100.00
Same Relate	2.0	96.59	98.65	100.00	97.71	100.00	96.33	100.00	100.00
	2.1	95.94	98.50	100.00	99.25	100.00	99.29	100.00	100.00
	2.2	-	-	-	100.00	100.00	100.00	100.00	100.00
	2.3	99.46	98.80	100.00	98.44	100.00	98.80	100.00	100.00
	2.4	95.60	97.93	100.00	98.05	100.00	97.04	100.00	100.00
	2.5	-	-	-	100.00	100.00	100.00	100.00	100.00
	2.6	92.86	98.08	100.00	95.70	100.00	96.70	100.00	100.00
	2.7	87.50	98.25	100.00	97.14	100.00	97.06	100.00	100.00
	2.8	-	-	-	-	100.00	-	100.00	100.00
	2.9	97.09	100.00	100.00	98.56	100.00	98.53	100.00	100.00
	2.10	96.12	97.14	100.00	98.56	100.00	97.12	100.00	100.00
	2.11	-	-	-	100.00	100.00	100.00	100.00	100.00
Comparison	3.0	96.67	95.24	100.00	99.50	100.00	96.50	100.00	97.89
	3.1	96.15	90.62	100.00	98.61	100.00	97.22	100.00	100.00
	3.2	-	-	-	-	100.00	-	100.00	100.00
Single Or	4.0	95.00	96.25	100.00	98.12	100.00	98.12	100.00	99.39
	4.1	94.97	96.25	100.00	98.75	100.00	99.38	100.00	99.39
	4.2	96.25	96.88	100.00	98.12	100.00	99.38	100.00	99.39
Zero Hop	5.0	96.25	96.25	100.00	98.75	100.00	97.50	100.00	100.00
	5.1	99.17	96.67	100.00	100.00	100.00	99.17	100.00	100.00
	5.2	-	-	-	100.00	100.00	100.00	100.00	100.00
	5.3	99.04	98.61	100.00	98.72	100.00	99.38	100.00	100.00
	5.4	98.67	98.49	100.00	100.00	100.00	98.97	100.00	99.50
Domain Exits	6.0	61.88	65.19	57.50	70.25	68.99	67.50	75.32	73.46
	6.1	65.19	67.50	85.62	75.80	82.91	73.08	94.94	92.55
	6.2	70.44	72.33	78.75	68.55	83.54	74.05	89.24	88.89
	6.3	63.75	71.25	60.00	61.25	71.25	62.50	81.25	74.39
	6.4	-	-	-	66.67	-	100.00	60.00	50.00
	6.5	-	-	-	50.00	52.94	54.55	59.38	64.66
Collision Order	8.0	95.00	85.00	51.90	83.75	49.37	76.25	39.24	32.93
	8.1	-	-	-	66.67	71.01	83.33	81.94	51.22
	8.2	92.50	72.50	70.00	80.00	55.00	75.00	53.75	46.34
	8.3	90.00	76.25	86.25	80.00	82.50	83.33	80.00	86.59
Quantitative Answers	10.0	85.00	80.82	66.67	82.19	60.53	82.43	61.54	45.33
	10.1	97.50	92.50	85.00	92.50	75.00	90.00	80.00	68.29
	10.2	97.50	100.00	95.00	95.00	92.50	97.50	95.00	87.80
	10.3	97.50	92.50	67.50	90.00	62.50	75.00	45.00	26.83
	10.4	97.50	92.44	68.07	89.74	63.03	75.63	42.02	21.31
	10.5	87.29	65.55	39.50	65.55	37.29	70.59	40.34	17.21
	10.6	100.00	100.00	100.00	89.74	52.14	89.92	37.82	17.21
Binary Answers	11.0	97.50	98.75	95.00	96.23	92.41	95.51	93.38	93.20
	11.1	100.00	100.00	96.00	100.00	100.00	100.00	93.75	97.30
	11.2	97.50	95.00	100.00	97.50	100.00	95.00	97.50	95.12
	11.3	92.50	100.00	100.00	95.00	95.00	92.50	95.00	90.00
	11.4	100.00	80.00	66.67	60.00	57.89	62.50	61.54	58.82
	11.5	-	80.00	42.86	60.00	63.16	62.50	69.23	64.71
	11.6	-	-	-	100.00	38.46	50.00	61.11	65.62
	11.7	-	-	-	100.00	38.46	50.00	55.56	67.74
	11.8	-	100.00	50.00	-	57.14	33.33	72.73	55.00
	11.9	-	100.00	50.00	-	57.14	33.33	72.73	70.00
	11.10	-	-	-	100.00	93.94	94.74	93.94	97.44
	11.11	-	-	-	100.00	97.30	100.00	89.19	92.31

Table 0.1: End-to-end accuracies per question for all levels. 40 test samples/simulations are used per level for evaluation. The Dynamics Predictor has been trained only on level 8 samples. Physics Understanding questions are explained separately in their own section.

Finally, end-to-end VQA results were given. The training of PropNet was only performed on simulations of level 8, as it had the best balance between events. The static image reasoning tasks performed with accuracies around 99-100%, since they only rely on the input of the MaskRCNN. The dynamic reasoning tasks had lower accuracies, but in general performed well, with the accuracies varying between questions, question categories and levels. A summary of each question of temporal interpretation can be seen in Table 0.1.

Conclusions

For this work, a completely new dataset (**PhAQ**) was created in order to address VideoQA problems regarding simple static image understanding, as well as complicated temporal event interpretation. This dataset in turn was used to train the various components of the **NewtoNN** ensemble network, an architecture that uses various existing, as well as new modules in order to reply to questions regarding physics simulations.

The development of NewtoNN, and its performance on PhAQ has proven that complicated VideoQA tasks can be answered with ensemble Neuro-Symbolic architectures, even with limited training samples. Neuro-Symbolic architectures utilize non-biased priors inserted by the symbolic language itself, allowing for complicated questions to be broken down to their simplest of elements, and build an answer based on them, rather than relying on complicated latent space representations. This opens a path in cases were data is limited or expensive to produce, as is the case of many real-life scenarios.

Keywords

Visual Question Answering, VQA, VideoQA, Neuro-Symbolic, NS, Sequence-to-sequence, Machine Learning, Deep Learning, Computer Vision, Ensemble Model, Event Interpretation, Simulation, Interaction

National Technical University of Athens
School of Electrical and Computer Engineering
Machine Learning and Data Science
Division of Signals, Control and Robotics

Neuro-Symbolic Visual Question Answering for Physical Interaction Understanding and Dynamic Scene Interpretation

Master Thesis

by **Myron Sampsakis-Bakopoulos**

Supervisor: Dr. Haris Papageorgiou, Research Director, ATHENA RC
Co-Supervisor: Alexandros Potamianos, Associate Professor, NTUA

Athens, 2021

Abstract

Visual Question Answering (VQA) is a task of Machine Learning that combines aspects of Natural Language Processing and Computer Vision. VQA tasks are usually given in the form of an image or a video that is accompanied by a question, with the aim of producing Natural Language answers. The answers can be chosen in the form of a classification problem, or generated by another network in an open-ended form.

While the above conventional methods have been proven to work, they will at most times require a lot of data, and their accuracies will surpass baselines only by a small percentage. The goal of this Master Thesis is to follow a different approach that will utilize Neuro-Symbolic architectures, which combine Neural Networks along with Symbolic reasoning in order to train and infer answers with much less data, and far higher accuracy from even complicated, temporal questions.

This Master Thesis is concerned with the proposal of a novel dataset aimed to tackle issues like the ones mentioned above, as well as the development of an ensemble model trained in that dataset, aimed to evaluate the performance of Neuro-Symbolic approaches on VQA tasks. The dataset consists of a procedurally generated pseudo-3D physics simulations of a variable number of balls, that interact with walls, other balls, obstacles and gravity fields, as well as a number of questions regarding said events. The aim of the ensemble model is to be able to understand the interactions that take place within each simulation, and answer these questions. The ensemble model consists of a frame interpreter in the form of a MaskRCNN, a Dynamics Predictor that utilizes a PropNet, a sequence-to-sequence model aimed to translate the Natural Language questions into series of symbolic commands and finally a Symbolic Executor, aimed to extract Natural language responses from the output of the Dynamics Predictor by utilizing the symbolic commands. Additionally, some extensions that further allow the understanding of scene dynamics, such as mass of objects and friction estimation are developed and paired with the above modules.

The proposed model is trained and tested upon the new dataset. This approach eliminates the existence of prior language biases, and provides beyond expected accuracies in the end-to-end VQA tasks, especially when taking into account the limited dataset size used for training, as well as the rather small computational resources used for training and inference. Apart from end-to-end performance, each modules performance is evaluated separately, in order to further test the strengths and weaknesses of each module.

Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών: Επιστήμη Δεδομένων και Μηχανική Μάθηση
Τομέας Σημάτων, Ελέγχου και Ρομποτικής

Νευρο-Συμβολική Απάντηση Οπτικών Ερωτήσεων για την Κατανόηση Αλληλεπιδράσεων και Επεξήγηση Δυναμικών Σκηνών

Μεταπτυχιακή Διπλωματική Εργασία

Μύρων Σαμψάκης-Μπακόπουλος

Επιβλέπων: Δρ. Χάρης Παπαγεωργίου, Διευθυντής Ερευνών, ΕΚ ΑΘΗΝΑ **Συνεπιβλέπων**: Αλέξανδρος Ποταμιάνος, Αναπληρωτής Καθηγητής, ΕΜΠ

Αθήνα, 2021

Σύνοψη

Η αυτόματη απάντηση οπτικών ερωτήσεων (VQA), είναι ένας τομέας της Μηχανικής Μάθησης που συνδυάζει οντότητες όπως η Επεξεργασία Φυσικής Γλώσσας και η Όραση Υπολογιστών. Τα προβλήματα VQA δίνονται συνήθως με τη μορφή μιας εικόνας ή ενός βίντεο, που συνοδεύεται από μια ερώτηση, με σκοπό την αυτοματοποιημένη απάντηση σε μορφή Φυσικής Γλώσσας. Οι απαντήσεις μπορούν είτε να επιλεχθούν με τη μορφή ταξινόμησης κάποιων προεπιλεγμένων απαντήσεων, είτε να δημιουργηθούν σε ανοιχτή μορφή από κάποια αντίστοιχη αρχιτεκτονική.

Παρότι οι παραπάνω αναφερθείσες μέθοδοι έχουν αποδειχθεί λειτουργικές, τις περισσότερες φορές απαιτούν μεγάλη ποσότητα δεδομένων, με τις τελικές ακρίβειές τους να περνάνε κατά λίγες ποσοστιαίες μονάδες την ακρίβεια αναφοράς. Ο σκοπός αυτής της Μεταπτυχιακής Εργασίας είναι να ακολουθήσει μια διαφορετική σκοπιά, που θα αξιοποιεί Νευρο-Συμβολικές αρχιτεκτονικές, οι οποίες συνδυάζουν νευρωνικά δίκτυα και συμβολική συλλογιστική (βάσει κανόνων) με σκοπό να εκπαιδεύσουν, αλλά και να συμπεράνουν απαντήσεις με πολύ λιγότερα δεδομένα και με σημαντικά αυξημένη ακρίβεια, ακόμα και απο περίπλοκες χρονικά εξαρτώμενες ερωτήσεις.

Η Μεταπτυχιακή αυτή Εργασία ασχολείται με την πρόταση ενός νέου σέτ δεδομένων, που στοχεύει στο να αντιμετωπισθούν τα προαναφερθέντα προβλήματα, αλλά και την ανάπτυξη ενός συνολικού μοντέλου εκπαιδευμένου στο σέτ αυτό, με σκοπό την αξιολόγηση Νευρο-Συμβολικών προσεγγίσεων σε προβλήματα VQA. Το σετ δεδομένων αποτελείται από στοχαστικά γεννημένων ψευδο-τρισδιάστατων προσομοιώσεων φυσικής με μεταβλητό αριθμό μπαλών, που αλληλεπιδρούν με τοίχους, άλλες μπάλες, εμπόδια και βαρυτικά πεδία, καθώς και έναν αριθμό ερωτήσεων που σχετίζονται με τα γεγονότα αυτά. Ο σκοπός του συνολικού μοντέλου είναι να μπορεί να αντιληφθεί τις αλληλεπιδράσεις που λαμβάνουν χώρα στην προσομοίωση, και να απαντήσει στις ερωτήσεις αυτές. Το μοντέλο αποτελείται από έναν ερμηνευτή καρέ, ο οποίος υλοποιεί κατά κανόνα ένα MaskRCNN, έναν δυναμικό αντιληπτή γεγονότων που αξιοποιεί ένα PropNet, ένα ακολουθιακό μοντέλο που μεταφράζει τις ερωτήσεις σε φυσική γλώσσα σε μια ακολουθία συμβολικών εντολών και τέλος, έναν συμβολικό εκτελεστή, με σκοπό να εξάγει απαντήσεις σε φυσική γλώσσα από την έξοδο του δυναμικού αντιληπτή αξιοποιώντας τις συμβολικές αυτές εντολές. Επιπροσθέτως, αναπτύσσονται και ορισμένες επεκτάσεις που επιτρέπουν την κατανόηση δυναμικής συμπεριφοράς όπως η πρόβλεψη μάζας και η εκτίμηση επιπέδου και είδους τριβής.

Το προτεινόμενο μοντέλο εκπαιδεύεται και ελέγχεται επάνω στο σετ δεδομένων. Αυτή η προσέγγιση εξαλείφει την χρήση γλωσσικών πρότερων και προσφέρει ακρίβειες πέραν των προσδοκιών στα συνολικά τελικά VQA αποτελέσματα, ειδικά όταν λαμβάνεται υπόψιν και το σχετικά περιορισμένο μέγεθος του σετ δεδομένων, αλλά και οι λίγοι υπολογιστικοί πόροι που χρησιμοποιήθηκαν. Εκτός από την συνολική επίδοση, η επίδοση και κάθε αυτοτελούς μέρους ελέγχεται ξεχωριστά με σκοπό τον περαιτέρω έλεγχο των πλεονεκτημάτων και των αδυναμιών του καθενός.

Acknowledgments

Throughout the writing of this dissertation, I have received a great deal of support and assistance. First and foremost, I would like to thank my supervisors, Dr. Haris Papageorgiou and Prof. Alexandros Potamianos, who offered me the opportunity to work on a subject of my own proposal and provided valuable guidance and feedback at times of need. It was a valuable learning experience. It would also be unjust to not thank both of them for their valuable lectures in Pattern Recognition and Natural Language Processing during my Master's first semester, which undoubtedly shaped my way of thinking and paved the way to the final choice of my Thesis' subject.

I would also like to show my gratitude for Giorgos Paraskevopoulos and Efthymis Georgiou for their weekly input and ideas, as well as our enlightening discussions. They never hesitated to assist me whenever there was need, and proved to be of invaluable assistance throughout the writing of this Thesis.

Finally, I would like to thank my family and all my friends for their support, who encouraged me when things didn't seem as bright as they truly were. But most of all, I would like to thank my grandmother and her eternal stubbornness. Managing it taught me how to handle stressful situations incredibly well, which in turn helped me overcome many obstacles that I faced during the development of this work.

Myron Sampsakis-Bakopoulos,
Athens, March 2021

Contents

1 Introduction

During the last 10 years, Machine Learning and Artificial Intelligence, have advanced with rapid paces. We are in what is know as the "Third wave of Artificial Intelligence". Machines have been able to perform many tasks significantly better than the average -or even a specialized- human. Cancer detection [42] , Computer Vision [22], and language translation [53] are tasks that are done in almost no time, with accuracies sometimes surpassing those of their human counterparts.

These are the well known, and often advertised perks of modern Artificial Intelligence, and most people that have not dwelt within the domains of this science, perceive Artificial Intelligence as an all powerful tool. However that is not the case.

There are also many tasks where AI algorithms fail, or simply cannot be trained to perform. Simple actions, like driving a car, are considered not yet perfected [56], however, even the simplest of human beings can achieve it. Another one of those task -and not uncorrelated to car driving- is the understanding of a moving surrounding. A toddler can understand action and reaction within the first couple of years in its life. If a ball is thrown towards it, it will lift its arms to either catch it or protect itself from the incoming impact. When it plays football it learns to kick the ball in a specific way in order to shoot it towards the nets. When it is running towards a table, it will duck or turn in order to avoid it. If it leaves an item on the very edge of a table, or on top of the other, he has ways of making it balance. All of the above are done without solving any equations, without running Finite Element Method algorithms. A latent model has been created in the child's brain that can incredibly quickly understand what will happen in each case [34, 49]. While intuitive physics are a yet-to-be-solved problem of AI, other correlated areas fare slightly better, with temporal scene and event understanding and Visual Question Answering being some of them [54].

1.1 Visual Question Answering

Visual Question Answering (VQA) is the task of automatically answering questions by reasoning over a given image or a video (VideoQA). It has attracted considerable attention in the research community during the past few years, as the success of image classification, object detection, and Natural Language Processing (NLP) has shifted interest towards more challenging and demanding tasks. These systems are usually ensemble networks, that pair a visual component in order to understand the image and a Natural Language Processing component, that parses the question, allowing the fusion of the two results to lead to an answer. The general idea can be seen in Figure 1.1. Usual VQA systems rely heavily on their visual component to extract information from the image in the form of a latent vector, and by combining it with the context of the question, they produce an answer similarly to a classification problem.

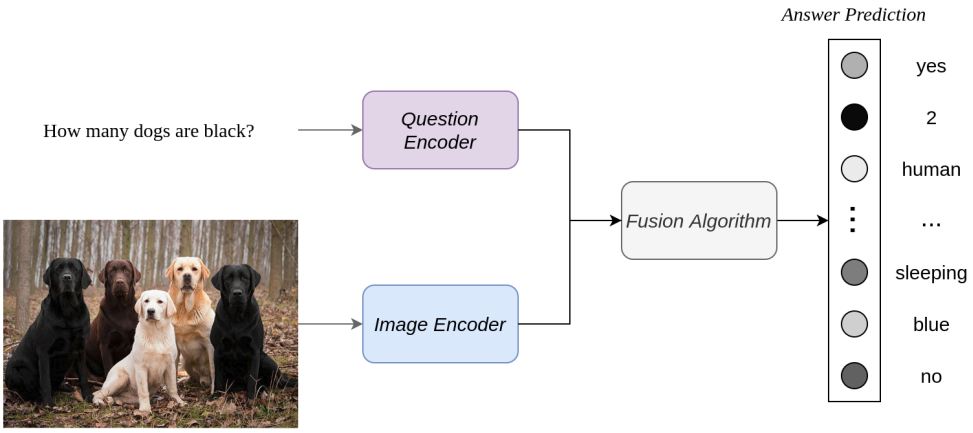

Figure 1.1: Visual Question Answering pipeline.

Since the availability of large-scale datasets [18, 33, 28, 41], multiple approaches have been proposed, with most of them relying heavily on the image encoder to extract information from the image, and the question encoder to produce a good representation of the question in a latent space. But addressing VQA as a classification task poses a limit to current systems, as predicting answers on a phrase-level contradicts our intuition suggesting the meaning of a phrase emerges from the syntax and its semantics, rather than its actual context. In addition to that, it severely limits its generalization properties, since it needs to understand the distribution of inputs behind each answer. Open-answer systems have been developed, with their answers being more realistic, while at the same time requiring a lot more work in evaluating the correctness of an answer [20].

In all the above cases however, the data required to create a working model is at most times huge. Since the ensemble model is facing the VQA task from a statistical perspective, large amounts of data will be required to train it. This, paired with the uninterpretable nature of the total end-to-end structure, leads to problems regarding both training and understanding the internal processes of the ensemble architecture. For the above-mentioned reasons, in recent years a different approach has been developed, pairing two seemingly unrelated branches of computer science. The well known and developed neural networks, and Symbolic Reasoning, the "old-school" rule-based approach to problem solving.

1.2 Neuro-Symbolic Reasoning

Neuro-Symbolic Reasoning, as its name suggests, is an amalgam of architectures utilizing both Neural Networks [15], as well as classic, rule-based Symbolic Reasoning. Despite still being in their early stages of development, these hybrids utilize their parent technologies to improve the total accuracy of algorithms, while simultaneously providing interpretability, and reducing the amount of time and data required to train them. They can recognize properties of objects (color, texture, size) present in an image or video, and reason about them, something which deep networks have had limited success in performing. Neuro-Symbolic AI further demonstrates the ability to reply to questions, an important aspect of human learning. These hybrids require far less data to be able to achieve a certain accuracy, rely on rule based (as opposed to statistical based) logic, and possess interpretable (by a human)

intermediate states, allowing further improvement and supervision during development and failure investigation.

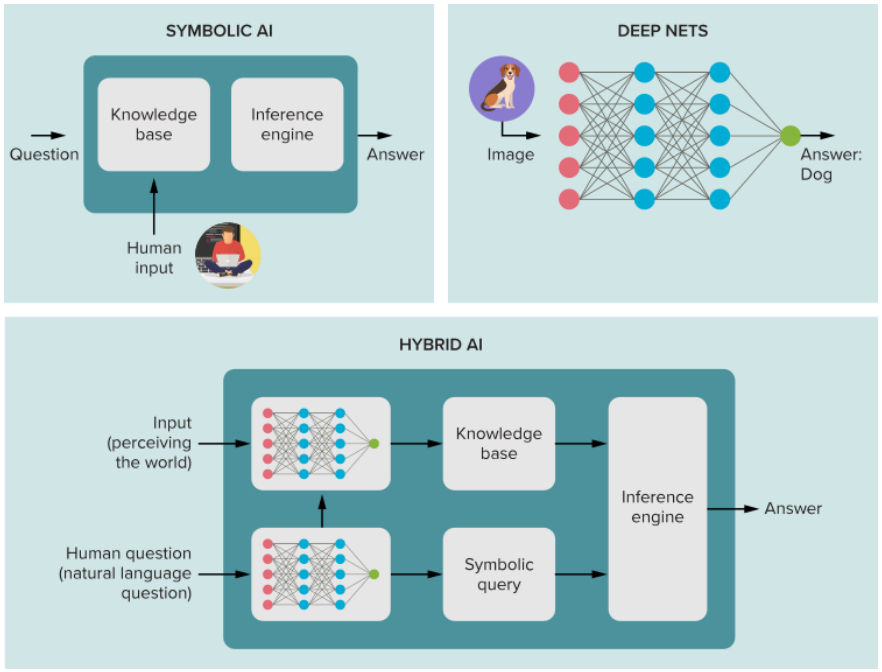

Figure 1.2: Neuro-Symbolic VQA, compared to its "parents", symbolic reasoning, and neural network reasoning [3].

Apart from being able to perform VQA tasks incredibly well, they can also be used as physics engines, in tasks where Artificial Intelligence, and neural networks will at most times fail. One domain, under research that addresses this problem, is Intuitive Physics [36] and Dynamic Scene Reasoning, topics which are hard to model, and even harder to train by conventional means. Neuro Symbolic VQA [55] has emerged as a potential solution. Contrary to popular methods, where a neural network will create a latent representation in the form of a non-interpretable vector and infer based upon its value, Neuro-Symbolic (NS) approaches will instead produce understandable representations by breaking a complicated problem into simpler, smaller and fully interpretable ones. This is -after all- what a human does. He learns how to transfer knowledge from a specific domain to another.

Motivated by this emerging trend, we developed and experimented with the **NewtoNN** (**Newto**n with **N**eural **N**etworks) Neuro-Symbolic VQA system build upon a custom made dataset, called **PhAQ** (**Ph**ysics **A**nswers and **Q**uestions). Apart from answering static image questions, one can query upon temporal events, order and numbering of events, entity of events, as well as physical quantities that can be observed, such as the mass of the participating objects through their interactions, and the type and level of friction that they are facing. This multi-module model has separate models for the visual component, the dynamic event locator, the neuro-symbolic parser, the mass and friction estimators, as well as the Symbolic Executor.

1.3 Goal and structure of this Thesis

This Master Thesis will focus on the aspect of understanding and locating physical events, interactions between items, estimating physical quantities regarding said interactions, and finally reply to questions regarding simple, or more complicated aspects involving the above. As with many previous work, where datasets did not exist for the task that was being researched into, similarly here, a fully customizable and reproducible dataset will be developed and generated. This dataset is composed of small videos that depict the simulated movements, collisions and other events of various balls on a surface that has walls, obstacles, as well as gravity fields. Together with each simulation, a set of questions along with their answers are generated. These questions cover a wide range of topics, from static image understanding, to complicated temporal physical interactions between the various objects participating in the simulation. The aim is to train an ensemble model that will be able to answer those questions successfully, without resorting to language biases (as is the case with pure neural approaches). For this purpose, a mix of conventional methods (neural networks) and symbolic reasoning is developed and implemented. These methods cover a wide range of AI fields, such as that of Computer Vision, Natural Language Processing, as well as symbolic algorithmic techniques. The mix of all these methods that aims to answer questions is called Visual Question Answering (VQA), and is still considered a rather complicated task, even by today's standards.

Within the context of this work, the following were developed:

- **PhAQ**: a new, custom, procedurally generated dataset, involving collisions between objects, event tracking, and question generation. This dataset also includes:
  - Development of a sub-dataset aimed to train a MaskRCNN in order to track the objects above.
  - Development of a sequence-to-sequence dataset aimed to translate a question in Natural Language (English) to its Neuro-Symbolic counterpart.
- Development of **NewtoNN**, an ensemble Neuro-Symbolic VQA model, which includes but is not limited to:
  - The MaskRCNN training procedure, in order to extract the symbolic representations of each scene.
  - The sequence-to-sequence translator, which aims to translate a Natural Language question into it's Symbolic counterpart.
  - The development of a significantly altered Propagation Network, aimed to predict events, as well as dynamics involving the simulations.
  - The development of a Symbolic Executor, aimed to produce Natural Language answers from the outputs of the Propagation Network.
  - The development and implementation of two networks that can understand physical quantities from the simulation, such as the balls' mass, and friction types.

The structure of this Master Thesis is as follows:

- **Chapter 2**: Historic analysis and presentation of existing techniques that are being used in Neural Networks, and AI in general, with focus given on Neuro-Symbolic approaches.
- **Chapter 3**: This chapter explains how the dataset is generated. The physics engine, the question generation techniques, and the way that events are tracked and recorded are analyzed.
- **Chapter 4**: The proposed model to solve the posed problem is analyzed in a macroscopic view, with each model explained separately
- **Chapter 5**: The experimental setup is thoroughly analyzed. The data used, as well as the models for each sub-module are explained, as well as the way that all those modules interoperate and are evaluated.
- **Chapter 6**: The results of each module separately are shown. These include training times, accuracies, and variations that were used during development.
- **Chapter 7**: The end-to-end VQA results are depicted in this chapter. These results are shown in their full form for level 8, and in a more compact form for all the other levels.
- **Chapter 8**: Various issues faced are analyzed in combination with proposals for future work (continuation), and finally a conclusion of all the work done is given.
- **Appendices**:
  - **A**. The questions from all categories, their templates, as well as their respective programmes are presented.
  - **B**. The distribution of answers of the questions for various simulation levels are presented.This shows that despite the imbalanced answers in many cases, biases are non-existent.
  - **C**. Mathematical background on the Dynamics Predictor used is given.

1.4 Material

All the developed software used to simulate, produce the data, train the models, the models themselves, the codes that process the data, as well as the generation of any figures, and some saved models' checkpoints, can be found on this projects GitHub page:

https://github.com/sbmyron/Master_Thesis

All code is written in *Python 3.8*, with libraries being used -but not limited to- *PyTorch*, *torchvision*, *pandas*, *sklean*. Some post-processing software, and the MaskRCNN training is written in *Python Notebooks*, while many handy scripts are written either in *bash*, or in *Python* using the *os* library.

Some samples of the videos that are generated and used to train the models, can be found in the links shown by the QR codes below:

(a) Level 1 (link) (b) Level 2 (link) (c) Level 3 (link) (d) Level 4 (link)

(e) Level 5 (link) (f) Level 6 (link) (g) Level 7 (link) (h) Level 8 (link)

Figure 1.3: QR codes and links for video samples of each simulation difficulty.

2 Related Work

Teaching a machine to understand, reason and act has always been one of the central aims of science. Despite thousands of years of attempts which mostly resulted in myths (such as the legendary robot Talos patrolling the Cretian coast during the Minoan period [40]), most -if not all- progress has happened during the last 50 or so years. Initial aims were focused on building machines that would simply follow rules, as understood by their human designers [51]. These machines would usually perform well on simple tasks, however further complicating things would lead to poor results. As computational power rose, older -computationally expensive- ideas that were previously proposed could now be implemented to surpass this issue. Neural Networks started appearing, and -especially within the last 10 years- they now form the backbone of Machine Learning and AI.

2.1 Symbolic AI

Symbols are things we use to represent and label other things. They can define items and people, represent abstract concepts (bank, transaction) or things that don't exist (web page, blog post). They can describe actions (fishing) and states (active). They can also be organized into hierarchies (a house is made of doors, bricks, plywood, cement, etc.). They can also be used to describe other symbols (a house with a wooden door, a green carpet etc).

Human interpretable symbols were the obvious place to start research in teaching a machine how to think and infer [51]. After all, it was believed by early AI pioneers that if every aspect of learning can be precisely described, a machine can be made to simulate it.

Symbolic AI was the dominant paradigm of AI during the 1950s-1980s. The approach is based on the assumption that many aspects of intelligence are based on the manipulation of symbols, very much like a human does. Intuitively, it was the only way to go when tasked with the development of a machine that can reason, and intended to produce a general, human-like intelligence (unlike the task-oriented approach in today's times).

Symbolic AI is very convenient for settings where rules are very clear cut and input to symbol transformation is easily done. However, introducing the real world to symbolic AI mostly breaks it. Computer vision, for example, could be rule-based in simple artificial scenarios, however being faced with the detection of a cat for example, would need a prohibitively large number of rules, and even then, its success would still be limited since its generalization would most likely not work that well. Even then, computer vision remains one of the "easy" tasks to use in symbolic AI. Tasks such as speech recognition and natural language processing simply cannot be translated to direct rules. Even though efforts to create complicated symbolic AI systems that encompass the multitudes of rules of certain domains exist, these expert systems require massive amounts of human effort by very experienced people and software engineers, and -even then- their limited set of rules poses a serious issue when the problem needs to be generalized, as the number of required rules can very easily explode in number. Even though the success of such ambitions were more limited than initially expected, its echoes can still be heard in modern data structures, and object-oriented languages [8].

2.2 Neural Networks

Neural Networks excel in tasks that purely Symbolic AI fails. Computer vision [22], facial recognition, cancer detection [42] and Natural Language Processing [53] are just some of the tasks that neural networks prove themselves to be excellent in. They are almost as old as Symbolic AI itself. However, due to their perceived inefficiency and huge -for that time- resources needed to train them, they would remain hidden for many decades. With the exponential increase of computing power and the wide availability of Graphics Processing Units (GPUs), they have gained ground with unprecedented rate, pushing past Symbolic AI in both efficiency and effectiveness. Their main advantage lies on their ability to deal with messy and unstructured data. And there is no longer need for rules(!). The neural network *itself* builds a statistical model tailored to its task [15].

To this day, many forms of neural networks have been developed, each initially built to deal with a different task. However, in recent years the borders between what network is used where are rather blurry. The first, and perhaps the most important network is the Multi Layer Perceptron (MLP), or else known as Fully Connected Network (FCN) is usually used to classify an input or predict a number (regression). Every neuron of a layer, is connected to every neuron of the previous as well as the next layer. Initially aimed for image recognition, the Convolutional Neural Network (CNN), "scans" through the input image with different filters and extracts characteristics in the form of a latent vector, which in turn can classify. Finally, for sequential data, a Recurrent Neural Network is used. These types of networks are a repeating architecture of the same element, that receives each timestep of the sequence separately, and outputs a latent vector which is used as input to the next element of the chain. The final latent vector will ideally include a summary of the whole temporal series, and will be classifiable by a FCN.

The three types of networks mentioned above are just the backbone of Machine Learning. Huge numbers of modifications exist for each, from simple meta-parameter tuning, to fundamental changes in their internal structure (e.g. LSTM vs RNN). These networks are also called encoders, since a representation understandable by a human, is *encoded* into a smaller, more dense representation better understandable and more easily manageable by a machine. The opposite of an encoder, is a *decoder*, that transforms a dense, latent representation, into something perceivable by a human. Decoders are used in language translation [53], as well as image creation (De-Convolution Networks).

Neural Networks do not have to work independently. They can operate in ensemble models, that can either be trained end-to-end, or independently. Perhaps the most well known ensemble model is the Generative Adversarial Network (GAN) [16], that aims to create new lifelike examples of its input data that can fool a human, such as new faces, or transforming items from one domain to another (DiscoGAN [26]). Another well known ensemble network that is end-to-end trainable, is the MaskRCNN [21], which is used in image segmentation, in order to classify the observed items, as well as create "masks", that can in turn provide pixel-level segmentation and positional data. This Thesis' model will also be an ensemble model, with each component trained independently. One of its components used, is the Propagation Network (Propnet [32]), which itself is an ensemble network. It receives positional, attributional and mask information from a temporal scene, and understands the underlying physics, elements and interactions, by locating temporal events and propagating

the forces in order to predict future cases.

VQA tasks were first proposed for purely neural architectures, by using ensemble models [18]. The first widely used dataset, was VQA 1.0 [18]. It is a large dataset, with static images from the COCO dataset [33] and questions from numerous categories. Both the visual encoding, as well as the question parser, outputted a latent vector with the extracted meaning, and by passing their fused product through a classifier, an answer (out of all the known answers) could be predicted through a softmax function. However, generating the questions and answers proved to be extremely time consuming, requiring "*countless hours of effort provided by the workers on Amazon*". After the successful demonstration of such systems, they rapidly expanded, as their weaknesses were discovered and ways to counteract them were developed. Pre-balancing imbalanced datasets by collecting complementary images gave increased accuracy on end-to-end results [19]. Attention mechanisms have been proposed in order to assist the visual component in where to look based on the questions' contexts [45]. Further development on models that can "read" certain parts of the image by focusing on them, is presented in [47]. Regions proposed by a Faster-RCNN [39] network are used in order to eventually answer questions regarding written text on images, such as "What brand are the crayons?" or "What is the time on the bottom middle phone?". However, just as before the accuracy is rather limited and the questions are simple; even by human standards. Further expansion on dynamic scenes (videos) is presented in [29], where clips from popular TV shows are used in order to train the algorithm and in turn be used for inference. At the same time, attempts for general use VQA systems that can work on a variety of datasets have been proposed and implemented [46].

Physical concepts and neural networks, are two fields that have had a difficult time being combined. Recent attempts exist where neural networks attempt to understand the underlying physics of given simulations [24]. Neural networks can also perform analytic function extraction by looking at simulated samples [1] as well as detect interactions of various objects by looking at data with noisy or missing samples [50].

None of the above work would take place if it weren't for large, widely available datasets, and many hours of manual labeling. After all, as can be seen from their respective publications, in most -if not all- cases the data required is rather large, and training remains expensive.

Albeit having many benefits, neural networks are not without trade offs. Deep Learning algorithms, for example, are opaque, and usually non-interpretable by a human, making their troubleshooting incredibly difficult. Also, since neural networks rely on statistical models, they are very data hungry, requiring tremendous amounts of (most times expensive to produce) data in order to be trained [15].

2.3 Neuro-Symbolic Artificial Intelligence and Current Trends

One major downturn of classic VQA systems that fully utilize neural networks, is that they require huge amounts of *labeled* data in order to correctly train the end-to-end architecture and provide answers, which in many cases just surpass the baseline by only a few percentile points.

The combination of the two previous concepts of AI has risen significantly in the past few

years, giving birth to Neuro-Symbolic Artificial Intelligence. It overcomes many difficulties that were previously faced by its predecessors. Neuro-Symbolic AI systems utilize the perks of both Symbolic Reasoning, and Neural Networks. Symbolic models are used in order to capture structured knowledge, while neural networks can capture nonparametric statistical relationships [10]. The causal and compositional process behind observations is something that a symbolic model is well-suited to represent as well as provide explanations that are similar to peoples' theories.

A field where Neuro-Symbolic Artificial Intelligence (NS-AI) excels, is Visual Question Answering (VQA) where an image or a video is given along with a question and an algorithm has to predict the correct answer. The success of NS-AI has been demonstrated by many publications within the last 5 years. As David Cox, IBM Director at MIT-IBM Watson AI Lab said: *"We can create hybrid AI systems that are much more powerful than the sum of their parts"* [7].

Neuro-Symbolic AI provides certain very powerful attributes, that are non-existent in neural approaches. In many cases of VQA, rephrases of the same question, while having the *exact* same meaning, will be answered differently [44], as they will be encoded to different latent vectors, which will change the output of the network. This is due to language models capturing superficial linguistic correlations. Attempts to counteract this have been made by focusing mainly on producing counterfactual questions (based on the input images, as well as the original question) [5]. Neuro-Symbolic applications can completely counter this, as their symbolic aspect takes over the part of meaning extraction based on a rule-driven approach.

Neuro-Symbolic AI can be used in a variety of fields, from linguistic understanding [6], Enterprise Risk Management [9] as well as Art Generation [2]. Neuro-Symbolic AI allows interpretation and provides explanatory capabilities to algorithms, in cases where the steps of the process that have produced the result can be more important than the result itself.

The Neuro-Symbolic Concept Learner [35], is a hybrid implementation, a concept learner able to learn visual concepts, words and semantic parsing of a sentence, without explicit supervision, meaning it can understand what a question asks, and by also seeing the image, create a symbolic sequence of commands (a program) by itself. The Neural State Machine [23] builds a probabilistic graph representing the underlying semantics of the scene, and performs symbolic operations upon the graph to answer questions. These operations are called *programs*, and can either be given along with the questions, in order to train a sequence-to-sequence model to translate to them, or can be generated through Reinforcement Learning means. Certain approaches exist in order to overcome the difficulty of those Reinforcement Learning algorithms to be trained, as well as their hard to interpret and evaluate outputs [38]. Even more abstract entities such as grammar and mathematics rules can be assisted by Neuro Symbolic approaches, by encoding symbolic priors about composition rules [31].

CLEVR [25] is a dataset with various images and questions regarding their positions within the image, as well as their attributes (color, texture, size) and their numbers. After using a visual component to locate the objects and classify them based on their attributes, a data structure is build, upon which a series of simple commands, called *programs*, is executed. These *programs* are a language of their own, a *Symbolic* language, a *Domain Specific Language* and they are a direct translation from their Natural Language counterparts that are being initially asked. CLEVR relies strongly on its visual component, and can only

provide answers to static images.

Improving upon the CLEVR dataset, CLEVRER [54] is a dataset of simulated and rendered *videos*, involving simulations of simple objects that collide together. The questions being asked are dynamic in nature, -meaning they require information of the items' interaction in order to be answered correctly- as well as counterfactual and predictive. Counterfactual and predictive questions heavily rely on a good predictor (which in this case is PropNet [32]), as well as data with simplified physics and fewer objects. HySTER [43] further improves the accuracy upon the CLEVRER dataset, by providing background (prior) knowledge at the Symbolic Executor module (See 4.6).

This work aims to prove that Neuro Symbolic VQA can work and operate at far smaller datasets in many cases, since latent distributions are no longer required to be calculated. This gives a massive advantage to real world problems, where data will most likely *not* be labelled, and certainly not always available upon request.

This Thesis further expands the notion of understanding dynamic behavior from a video. It adds more events, rather than just simple collisions between participants, and complicates many questions, while at the same time simplifying the visual aspect by creating a custom made dataset in order to focus more on dynamic behavioral understanding. Many problems that are faced are analyzed, and proposals about future improvements on this specific task, as well as related tasks in general are proposed.

3 Dataset: Physics Engine, Simulations and Question Generation

Datasets that are widely available will always have limitations. This is usually due to the fact that the dataset is as-is, without the ability to generate more examples or expand the existing ones. Moreover, many times a researcher might need to further add attributes or try techniques in his algorithms that the dataset simply doesn't support.

In the case of the experiments that we want to attempt, a new dataset had to be created. Since primary focus is given in the event understanding under different circumstances as well as the physical quantity predictions, aspects that might include prior bias or hinder accuracy determination such as the visual part are simplified, with more focus given on the interactions and events taking place.

So in this case, a square table with balls of variable sizes and various elements, like obstacles, walls and gravity wells (positive or negative, called gravity fields), are simulated for a maximum duration of 10 seconds. Two aspects of this physics engine are worth discussing. The first is the actual physical interactions that take place, and the second is the way that events are tracked.

As expected, simple numerical methods are used during the simulation. The positions of the balls are calculated by the simple equation (Eq. 3.1):

$$x^{n+1} = x^n + u\Delta t \tag{3.1}$$

where Δt is the time discretization that we use, u the velocity on the x axis and n the timestep of our simulation. In the numerical calculation of the simulations, that is one thousandth $\left(\frac{1}{1000}\right)$ of a second.

Equation 3.1 describes a movement of a body, upon which no forces are acting. In the case of any form of interaction between two objects, this equation cannot *fully* describe the movement, and more phenomena will need to be taken into account:

3.1 Ball collisions

The simplest form of interaction is the collision between two balls. Fully elastic collision will be assumed, therefore no energy is lost in the process. The equations that describe the movement of the balls before and after the collision are: 1) the Conservation of Momentum, and 2) the Conservation of Energy. These two equations in the 1-dimensional case are:

Conservation of momentum:

$$m_1u_1 + m_2u_2 = m_1u_1' + m_2u_2' \tag{3.2}$$

Conservation of energy:

$$\frac{1}{2}m_1u_1^2 + \frac{1}{2}m_2u_2^2 = \frac{1}{2}m_1u_1'^2 + \frac{1}{2}m_2u_2'^2 \tag{3.3}$$

where the ($'$) annotation symbolizes the speed of the object after the collision. These equations are valid for all dimensions (x, y), and therefore can formulate the collision in both

these directions. By combining them, we can get explicit forms of the new velocities of both bodies (body 1 and body 2) after the collision since the velocities before the collision are known (Eq. 3.4, 3.5, 3.6, 3.7) :

$$u_1' = \frac{m_1 - m_2}{m_1 + m_2} u_1 + \frac{2m_2}{m_1 + m_2} u_2 \tag{3.4}$$

$$v_1' = \frac{m_1 - m_2}{m_1 + m_2} v_1 + \frac{2m_2}{m_1 + m_2} v_2 \tag{3.5}$$

$$u_2' = \frac{2m_1}{m_1 + m_2} u_1 + \frac{m_2 - m_1}{m_1 + m_2} u_2 \tag{3.6}$$

$$v_2' = \frac{2m_1}{m_1 + m_2} v_1 + \frac{m_2 - m_1}{m_1 + m_2} v_2 \tag{3.7}$$

These equations can be used for the simulation, however, the results will end up being unrealistic, since the bodies will behave like singular masses. A vectorized approach has to be followed in order to provide the correct equations. Two colliding balls with arbitrary velocities, are shown in Figure 3.1:

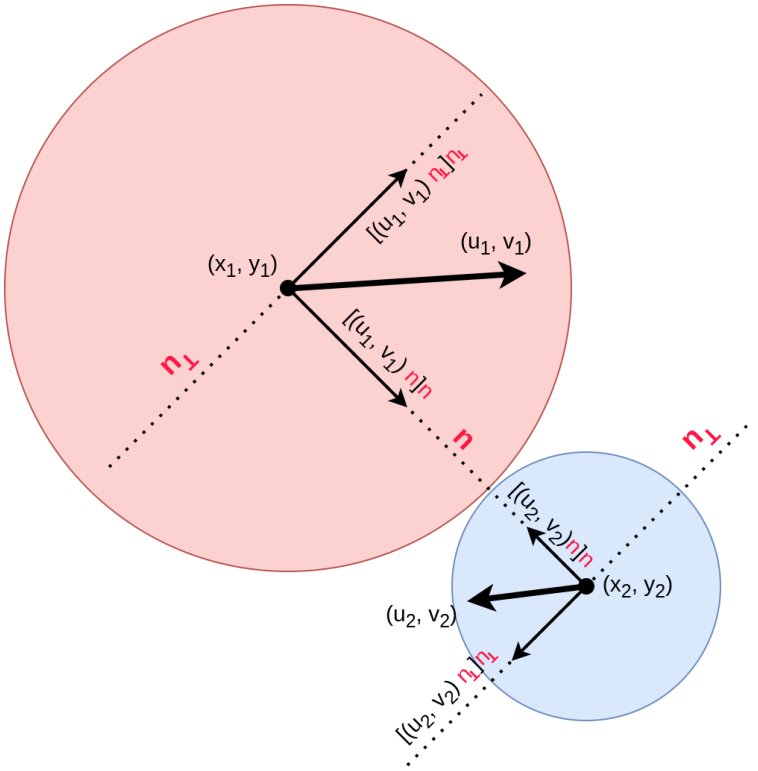

Figure 3.1: Two dimensional collision of balls.

We can define two different vectors. The first is the vector that connects the centers of each ball (stylized as n in Figure 3.1):

$$\vec{n} = \frac{(x_2 - x_1, y_2 - y_1)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \tag{3.8}$$

as well as its perpendicular (stylized as $n_\perp$ in the figure):

$$\vec{n_\perp} = \frac{(y_2 - y_1, x_1 - x_2)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \tag{3.9}$$

Upon projecting the velocities of each ball on those vectors, we can take the components of the velocities that are parallel to vector n (Eq. 3.10). Since they are the ones that will be affected by the collision (assuming no friction takes place), the new velocities can be calculated by the 1D collision equation (Eq. 3.11).

$$\begin{aligned} u_{1\vec{n}} &= [(u_1, v_1)\,\vec{n}] \\ \vec{u}_{1\vec{n}} &= u_{1\vec{n}}\vec{n} \\ u_{2\vec{n}} &= [(u_2, v_2)\,\vec{n}] \\ \vec{u}_{2\vec{n}} &= u_{2\vec{n}}\vec{n} \end{aligned} \tag{3.10}$$

$$\begin{aligned} u_{1n}^{'} &= \frac{m_1 - m_2}{m_1 + m_2} u_{1\vec{n}} + \frac{2m_2}{m_1 + m_2} u_{2\vec{n}} \\ u_{2n}^{'} &= \frac{2m_1}{m_1 + m_2} u_{1\vec{n}} + \frac{m_2 - m_1}{m_1 + m_2} u_{2\vec{n}} \end{aligned} \tag{3.11}$$

Finally, we can determine the new velocities in the global frame of reference ((x, y) instead of $(\vec{n}, \vec{n}_\perp)$) (Eq. 3.12):

$$\begin{aligned} u_1^{'} &= u_{1n}^{'}\vec{n} + \vec{u}_{1\vec{n}_\perp} \\ u_2^{'} &= u_{2n}^{'}\vec{n} + \vec{u}_{2\vec{n}_\perp} \end{aligned} \tag{3.12}$$

In order to calculate when a collision takes place, a simple radius rule is imposed. If the distance separating the centers of the two balls is smaller than the sum of their radii, a collision takes place (Eq. 3.13):

$$D_{coll} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \leq R_1 + R_2 \tag{3.13}$$

3.2 Wall collisions

Wall collisions are rather simpler to formulate, since one body (the wall) is stationary, therefore only one velocity needs to be calculated. In other words, the perpendicular-to-the-wall component of the velocity is mirrored. So, if a wall has a normal vector of $\vec{n}$ then the velocity of the ball can be written in the new frame of reference as:

$$\begin{aligned} \vec{u}_{\vec{n}} &= (\vec{u} \cdot \vec{n})\vec{n} \\ \vec{u}_{\vec{n_\perp}} &= (\vec{u} \cdot \vec{n_\perp})\vec{n_\perp} \end{aligned} \tag{3.14}$$

And, similarly to the case of the balls colliding, the new velocity in the (x, y) frame of reference, will be (Eq. 3.15):

$$u^{'} = -u_{\vec{n}}\vec{n} + \vec{u}_{\vec{n}_\perp} \tag{3.15}$$

The condition of a wall collision is simply a check on weather a ball is closer to the boundary than the wall width.

3.3 Obstacle collisions

Three different types of obstacles have been added in the run cases. Since the simulation is completely vector-based, and doesn't implement numerical methods such as finite element methods, each object type during a collision should be dealt with in a specific manner. The obstacles used in these simulations are *circular*, *triangular*, and *square* obstacles.

3.3.1 Circular obstacles

Circular obstacles are the simplest of all to formulate in a collision. The equations are very similar to a ball-to-ball collision, but the mass of the obstacle tends to infinity. With De L'Hospitals rule, the new velocity (as in Eq. 3.11), is (Eq. 3.16):

$$u_{1n}^{'} = -u_{1\vec{n}} \tag{3.16}$$

Which is of course nothing more than a simple reflection on the vector formed by the balls and the obstacles centers.

3.3.2 Square obstacles

Obstacles with corners can be a bit more tricky to formulate, since the have two different types of collisions. The first type of collision is a simple, side collision that takes place between a ball and a side of a square obstacle. This collision is very similar in nature with the wall collision, where the balls velocity is simply reflected. The condition for a ball-to-square obstacle collision to be a side collision, is for the perpendicular projection of the center of the ball to be within the sides of the obstacle, as shown in Figure 3.2.

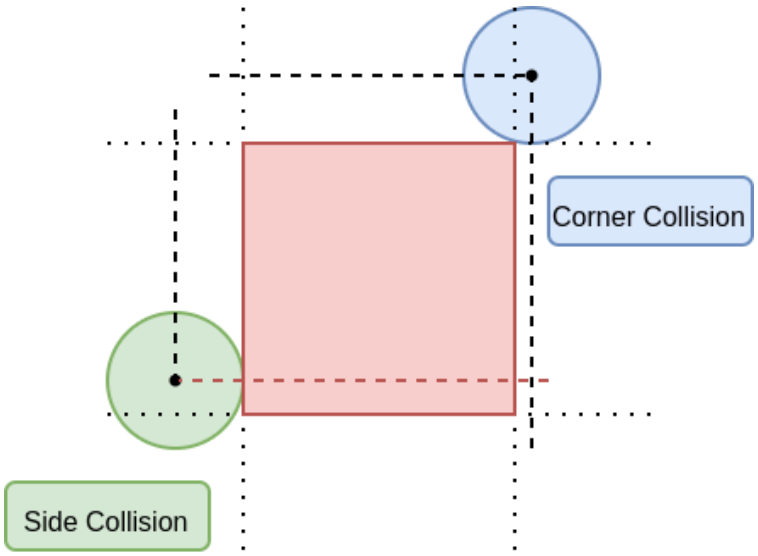

Figure 3.2: Corner and side collision with a square obstacle.

Similarly to the wall collision, the new velocity of a ball colliding with a side of a square obstacle can be computed through Equation 3.17:

$$u^{'} = -u_{\vec{n}}\vec{n} + \vec{u}_{\vec{n}_{\perp}} \tag{3.17}$$

Treating a collision with a corner however, is a bit different. The inelastic corner can be assumed to be equal to a point, or in other words, a collision with a circular obstacle of infinitesimally small radius. Therefore, if the coordinates of that corner are (x_c, y_c), then we can create a vector that connects the center of the ball to the corner (stylized as $\vec{n}$, and, very similarly to a collision with a circular obstacle, the new component of the velocity can be calculated as shown in Equation 3.18:

$$u_{1n}^{'} = -u_{1\vec{n}} \tag{3.18}$$

3.3.3 Triangular obstacles

Very much similar to square obstacles, triangular obstacles have the same two types of collisions. A corner collision, and a side collision. The only difference lies on the number of points and sides with which a ball can collide with a triangle, since they are 3 instead of 4. However, the equations remain completely invariant. So, for a side collision with a triangle, Equation 3.17 can be used to calculate the new velocities, whereas in the case of a corner collision, we use Equation 3.18.

3.4 Gravity fields

In order to further complicate the physics of the simulation, an additional phenomenon, the use of localized gravity fields, is added. These are areas where the ball is either going "up a hill" or "down a slope", in a pseudo-3D manner. Their aim is to test the capabilities of the models in dealing with non-linear trajectories, since their interactions with the balls are non-linear (albeit not chaotic like the collisions). Gravity fields do not participate in any kind of events, and are not part of any questions. For simplicity purposes, each area with a non-zero gradient will be rectangular, and either a *protuberance* or a *dent*. The different gradient is implemented as a sinusoidal curve of one period, so as to be continuous and derivable at all positions.

3.5 Special treatment during numerical simulations

A few special ways to treat irregularities are proposed, in order to be able to reproduce the abovementioned results in a custom simulation.

- In the case of ball-to-obstacle collisions, sometimes, and heavily dependent on the angle and velocity of the collision, after the new velocity is calculated and propagated through the next time step, the two objects might be *slightly* inside each other, leading to them merging, and producing completely wrong results. As a solution, a time sub-step is proposed, that propagates the object towards the new direction for 1/10 of its initial computational timestep (1/10000-th of a second) until the actual propagation does not lead them to merging.

- Each simulation is fully parametric, and the number and type of objects is completely custom. This leads to many different "families" of simulations, each having their own "difficulty". The scaling of the difficulty is mostly arbitrary, with level 1 being

the easiest to understand as a human, while level 8 being the most complex. These "families" will be called levels, and they are as follows (Table 3.1):

Level	No. of balls	No. of walls	No. of obstacles	No. of gravity fields
1	2	1	-	-
2	3	2	-	-
3	4	3	-	-
4	3	2	1	-
5	4	3	2	-
6	3	2	1	1
7	4	3	2	2
8	5	3	3	2

Table 3.1: Levels of simulation, and the items contained in each. The levels are roughly scaled to a human-level perception. Level 1 is easier to understand and answer questions (as a human) than level 8 is.

- In order to make the whole dataset fully reproducible, careful RNG seed management has been implemented, in order to be able to re-generate exactly the same dataset, without the need of downloading it. The generation is done in parallel (multiple *Python 3.8* instances), and is fully scalable to an arbitrarily large number of threads.

3.6 Event Tracking

During the simulation, all types of interesting events are recorded, in order to be used in the question generation and answering, as well as to train the event tracker network (PropNet [32]). Most events are rather easy to track. Ball-to-ball collisions, ball-to-wall and ball-to-obstacle are simple to record, since they happen for a single instant, and at a very specific timestep. As these events affect the simulation, it means that the physics engine is capable of tracking them through numerical means (see Chapters 3.1, 3.2, 3.3), therefore the exact timestep of their happening can be located.

Domain exits are a bit more tricky to formulate. Since a ball will gradually cross over the border of the domain, it is debated which time is the one that characterizes its exit. The time it first crosses over, or when it has fully crossed over? In this scenario, we assume the timestep that the ball exits the domain is the timestep that the ball stops appearing in our simulation video; that is when the ball has fully exited the domain. This assumption might be considered correct at first glance, it might however pose a problem in later steps, as explained in Chapter 8. Other types of data that require tracking, are the objects positions and velocities, which will be used for the Mass Estimator (Chapter 4.4).

Events regarding ball collisions and domain exits, are recorded in a standardized way, as shown in Table 3.2. The first column denotes the time that the event takes place. Note that the time accuracy is that of the numerical simulation, which gives us a time discretization of 1/1000-th of a second. The second and third columns are the two items that participate in the event, with a numbering that corresponds to their data-table, as generated by the

solver. In the event of a collision, it is obvious which items participate. However, in the event of a domain exit, the second item is a "phantom" item, the side. This side, as will be seen later, does not have the usual attributes of an item, however, for tracking purposes, it is considered one.

In order to understand the event tracking concept, a few frames of a simple simulation (with 3 walls and 4 balls) can be seen in Figure 3.3.

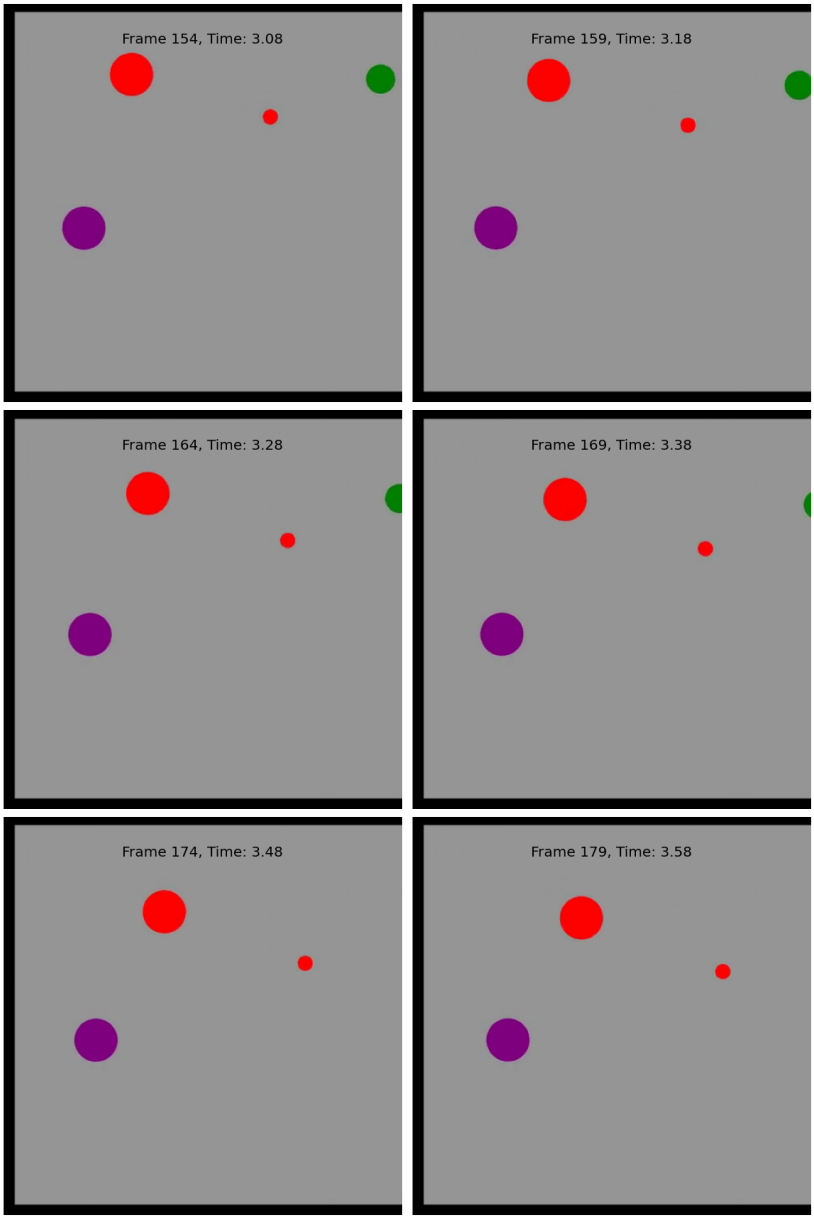

Figure 3.3: Level 3 simulation at a random point, every 5 frames ($\Delta t = 0.1$ sec)

In the same animation, a few other files are generated (apart from the video). The first, is the logfile that contains all the types of interesting events that take place. Its layout is similar to what is shown in Table 3.2.

Event ID	**Time**	**Item 1**	**Item 2**	**Event Type**	**Order 12**	**Order 1**	**Order 2**	**Order Type**
1	0.104	3	1	ball2ball	1	1	1	1
2	0.453	3	0	ball2wall	1	1	1	1
3	0.707	2	0	ball2ball	1	1	1	2
4	1.15	3	0	ball2ball	1	2	2	3
5	1.441	1	0	ball2wall	1	1	2	2
6	1.893	3	0	ball2wall	2	2	3	3
7	1.895	0	3	ball2wall	1	1	1	4
8	2.219	2	3	ball2wall	1	1	2	5
9	2.386	3	3	ball2wall	1	3	3	6
10	**3.476**	**2**	**2**	**ball_exits**			**1**	**1**
11	4.697	0	2	ball_exits			2	2
12	6.67	3	2	ball_exits			3	3

Table 3.2: Events recorded during the simulation

Notice that every item is given a different ID, depending on its characteristics. The same ID however can belong to many items in different categories, but since the category of each event is known, the items' are as well. Additionally, four extra columns of various types of "orders" are added. These basically order the events based on their participants. *Order 12* denotes the order that *Item 1* and *Item 2* have participated in the event. In Table 3.2 event 2 and event 6 have the same participants, and have *Order 12* attributes 1 and 2 respectively. *Order 1* denotes the ascending order in which *Item 1* has participated in an event, while the same applies for *Order 2* and *Item 2*. *Order Type* denotes the order of the specific type of interaction, based upon the *Event Type* column.

As can be seen in the above frames, the blue ball exits the domain some time between *3.38* and *3.58* seconds. Indeed, the 10^{th} event in Table 3.2 is ball *2* that exits from side *2* at *3.476* seconds (reminder that this is the time that the numerical simulation runs, and not the video frame time; the former has a time discretization of 0.001 sec, while the latter 0.02). That means that a ball with ID *2* will exit from the side with ID *2*. While there are many different combinations of colors and sizes, the ID of each ball is given dynamically in each video. However, since there will always be 4 sides, their ID remains constant, and it is as follows: 0: left, 1: lower, 2: right, 3: upper side. The ID of the ball through its characteristics and vice versa, can be found through the corresponding data table that is produced during the simulation. In the example above, the data table looks like the one shown in Table 3.3.

Ball ID	**Size**	**Color**
0	small	red
1	big	purple
2	medium	green
3	big	red

Table 3.3: The balls, their IDs and their characteristics

In addition to a data table for the balls, other data tables, for walls, obstacles, gravity fields, as well as position and velocities, are generated.

3.7 Mask Dataset

Apart from the standard dataset, a second dataset -which is based on the first- can also be generated. This is the "Mask Dataset", which contains the images along with each objects mask and category, compatible with the *PASCAL Annotation Version 1.00* standard.

The "Mask Dataset" is used in order to train the visual component that extract the elements' attributes. This -as will be seen in Chapter 4.1- is a MaskRCNN [21]. As the geometry of all the elements is fully known, the masks (shaded areas that an item covers) are known as well, and can therefore be produced. Along with the masks, a label specific for that object is generated. These labels are limited to include the physical quantities of the object ($18+1$ labels, 18 categories (Table 3.4) + background), therefore the color is omitted, as it can be inferred through simpler means, as will be seen later in Chapter 5.1. Would the color be included, there would be a total of $12n + 7$ labels, where n is the number of colors used (there are 12 categories of objects that can contain color), implicating classification. The "mask" will be 0 at a coordinate, if no item is located, 1 for the first item, 2 for the second item and so on.

Category ID	Description	Category ID	Description
1	left wall	10	big circular obstacle
2	lower wall	11	small triangular obstacle
3	right wall	12	medium triangular obstacle
4	upper wall	13	big triangular obstacle
5	small square obstacle	14	small ball
6	medium square obstacle	15	medium ball
7	big square obstacle	16	big ball
8	small circular obstacle	17	positive gravity field
9	medium circular obstacle	18	negative gravity field

Table 3.4: The 18 categories/labels of items used to train the MaskRCNN. Category *0* is the background image.

For every image, a series of masks will be produced, as shown in Figure 3.4. These masks in turn, will be encoded in RLE (Run Length Encoding) format, since their visual entropy is low and can therefore be successfully compressed without degrading. Notice how each color corresponds to a different object. This is due to the fact that the mask is represented by an integer array, where each different object is represented by a different integer (and therefore different color), except for the background, which is labeled with *0* (and purple color).

In the situation shown in Figure 3.4, we can see multiple circular *static* objects. After all, image segmentation takes place in a static frame. For simplicity purposes, circular obstacles and balls do not have overlapping dimensions, so that a single image of one can lead to successful classification. However, by adding mass, or another temporal attribute to each item, it would be possible to simply classify it as a *circular object*, instead of a *circular*

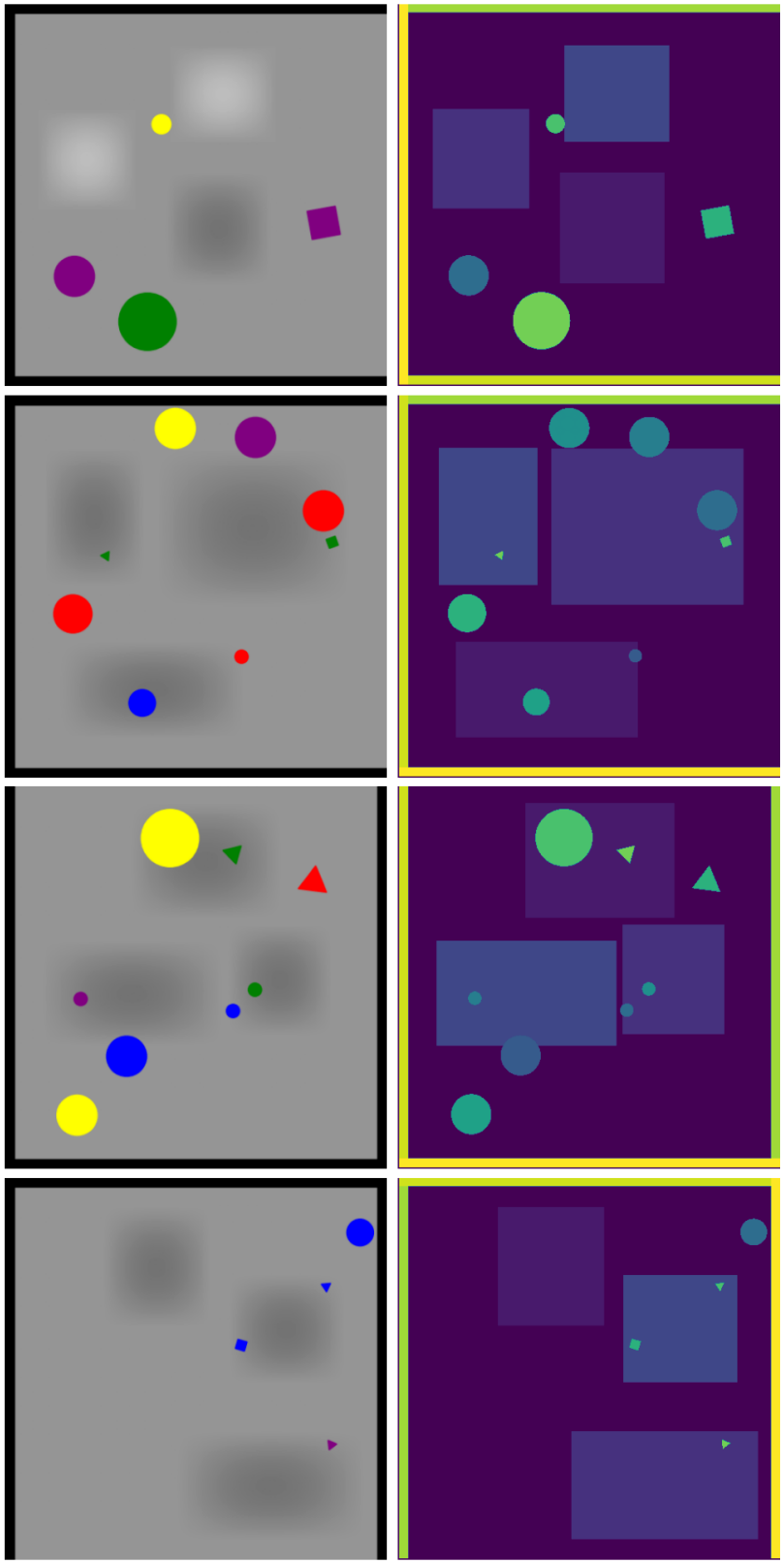

Figure 3.4: Frames (on the left), and their corresponding masks (on the right) as they are generated from the dataset generator.

obstacle, or a *big ball* and let the actual dynamics of the scene decide upon its identity. However, such implementation is left for future work (See more at 4.4).

3.8 Question Generation

In order to be able to teach a network to reply to human-language questions, these have to be generated along with the dataset. So, for every simulation (each video), a number of questions are generated along with their answers (QnA). This leads to a set of video-QA pairs, that can be used to train the whole network in the subsequent steps.

The question generation is performed with hand-crafted, templated questions. The questions are broken into several different categories, depending on the nature of what is being asked, as well as the complexity of the question or the physics behind it. A number of templated questions that involved strictly static information are taken from the CLEVR dataset [25]. The rest -that is questions involving temporal reasoning and event tracking- were novel, custom, hand-crafted questions created for this dataset.

3.8.1 Question Categories

Questions can be broken up into ten (10) different categories:

1. **Integer Comparison**: These types of questions track the quantities of different groups of objects and compare them. These questions, as well as their rephrases were taken almost completely unchanged from the CLEVR dataset. (e.g. "Are there more green balls than yellow small balls?")

2. **Same Relation**: These types of questions track the number of objects of similar nature that exist. These questions were also taken from the CLEVR dataset. (e.g. "Are there any other things that have the same size as the green ball?")

3. **Comparison**: These questions take two different items within the domain (either balls or obstacles), and ask whether they have a similar characteristic in common (e.g. color, size, shape). These questions were taken from the CLEVR dataset. (e.g. "Do the blue obstacle and the green ball have the same size?")

4. **Single-or**: These questions take two different elements of the domain (balls or obstacles), and ask how many of one or the other exist, while also conditioning them to a specific characteristic. These questions were taken from the CLEVR dataset. (e.g. "How many things are either blue balls or green obstacles?")

5. **Zero hop**: These questions take a single element from the domain, and ask a simple question regarding it, such as its color, shape or size. It should be noted, that the chosen element must be unique in order for the algorithm to give a definitive answer. These questions were taken from the CLEVR dataset. (e.g. "What color is the small triangular obstacle?")

6. **Domain Exits**: These questions are the first to take dynamic reasoning into account. They ask questions regarding the absolute order of exit from the domain (e.g. "Which ball will exit the domain third?", "Will the big green ball still be in the domain at the end?", "Which ball will exit from the upper wall second?"), as well as relative information regarding domain exits (e.g. "A ball will exit after colliding with the big triangular obstacle; which ball will it be?", "A ball will collide with the small square obstacle second; from which side will it exit?").

7. **Ball Collisions**: These type of questions track the collisions of the balls (e.g. "Which two balls will collide together third?", "Which ball will hit a wall or an obstacle for the second time?", "With which ball will the blue square obstacle collide for the third time?").

8. **Physics understanding**: These questions are asking for information on a higher level that the ones before. The relative weight of the balls (which *cannot* be inferred in all cases), the resistance the balls face, or the balls' velocity at some point during the simulation are attributes which are hard to encode, let alone infer through a Symbolic Executor alone. For this purpose, different neural networks are used for this stage in order to infer the required quantities (e.g. "Which ball is the heaviest?", "What is the type of the resistance?").

9. **Quantitative answers**: These questions ask information about the number of times that events will take place (e.g. "How many collisions will happen with the upper wall?", "How many times will the small blue ball and the big yellow ball collide?", "How many ball to obstacle collisions will take place in total?").

10. **Binary Answers**: These are YES/NO questions, that ask about various temporal events (e.g. "Will the big green ball hit the red small ball more that three times?", "Will the medium red ball hit the upper wall more that two times?", "Will the small green ball hit the big yellow ball before the medium red ball hits the upper wall?", "Will the small ball collide with an obstacle of the same color?").

Some sample questions per category can be seen in Table 3.5. For all the questions, their templates and their programs used, see Appendix A.1.

In all the above-mentioned categories, the answers are generated along with each question. Since (as explained in Chapter 3.6) for every video/simulation, a number of data tables are generated and all the available and required info is saved into those data tables. These, in turn, are used to generate the answers of the questions in a purely algorithmic way. As will be seen in a Chapter 4.6, this is not the only way to generate answers through the data tables, but it is a good way to validate the correctness of later methods.

Along with each question in natural language, a similar question written as a series of "programs" is also given. These questions will be used to train the sequence-to-sequence model used at a later stage (see Chapter 4.2).

Category	Questions
Integer Comparison	• Are there more green balls than yellow small balls?/Is the number of green balls greater than the number of yellow small balls? • Is the number of obstacles the same as the number of balls?/Are there the same number of obstacles and balls?
Same Relation	• Are there any other things that have the same size as the green ball?/Is there anything else that is the same size as the green ball? • How many other objects are the same size as the blue square obstacle?/ What number of other objects are there of the same size as the blue square obstacle?
Comparison	• Do the blue obstacle and the green ball have the same size?/"Is the blue obstacle the same size as the green ball?" • Is the shape of the green obstacle the same as the blue obstacle?/ "Does the green obstacle have the same shape as the blue obstacle?
Single-Or	• How many things are either blue balls or green obstacles?/What number of objects are blue balls or green obstacles? • How many red objects are square obstacles or small balls?/What number of red things are either square obstacles or small balls?
Zero-Hop	• How many small balls are there?/What number of small balls are there? • What color is the small triangular obstacle?/ The small triangular obstacle is what color?
Domain Exits	• Which ball will exit the domain second? Which ball will stop being within the area second? • Which ball will exit from the left wall first?" A ball will exit from the left wall first; which ball will it be?
Ball Collisions	• Which two balls will collide together third?/ Between which two balls will the third collision occur? • With which ball will the small blue obstacle collide for the fifth time?/ The small blue obstacle will collide with a ball for the fifth time; which ball will that be?
Physics Understanding	• Which ball is the heaviest?/Which ball has the largest mass?" • Which ball will be the second fastest?/One ball will achieve the second highest velocity at some point. Which one will it be?
Quantitative Answers	• How many collisions will happen with the upper wall?/How many times will a ball collide with the upper wall? • How many ball to obstacle collisions will take place in total?/How many times will a ball and an obstacle collide?
Binary Answers	• Will the small blue ball hit the green big ball before the medium red ball hits the upper wall? • Will the big green ball collide with an obstacle of the same color?

Table 3.5: Sample questions from each category. Two are shown for each category. Rephrases are shown in red

3.8.2 Question Templates

All questions are generated through templates, meaning that entities such as color, size, shape, item type, side etc., can be changed depending on the simulation, in order to generate multiple questions of the same type; albeit different. In addition to generating numerous different questions, some entities can be omitted if they over-describe the item that needs to be described. For example, if there are 3 balls in the scene, and they are all different colors, their sizes will not matter when you will need to describe them. Therefore, "*the big blue ball*", and "*the blue ball*" will describe the same item. This gives us the benefit of testing the Symbolic Executor (Ch. 4.6) with less commands, that better, and more directly describe our scene. All the templated questions, their rephrasals, as well as their programmes that will be used in Chapter 4.2, can be found in Appendix A.1. The answer distributions for specific levels of simulation can be found in Appendix A.12.

As mentioned, all questions are templated, in order to be procedurally generated from every possible simulation, regardless of the difficulty of the simulation. These templates, have "variables" that can change depending on the attributes of the participants of the simulation. Attributes such as *color*, *size*, *object type*, *side*, *wall*, and *order* can be replaced in order to describe all possible objects in the simulation, leading to a myriad of different questions. For each simulation, a number of questions will be generated, randomly sampled from all existing questions (the relative frequency of the sampling can be chosen beforehand for testing purposes, but in their default form, they are equiprobable). Duplicates are discarded. In addition to the questions themselves, the "programs" are templated as well with a very similar logic. A more in depth approach will be given in Chapter 4.2, but in a similar manner, for each question a program is generated. Unlike the questions where multiple rephrases can have the exact same meaning, the programs are unique for each set of questions of the same meaning, leading to the translator performing a many-to-one translation for certain rephrased questions. The answers of the questions will be given in natural language, just as they have been computed in the code that generated the questions as well as the answers. In certain cases, the information given will not suffice to provide with an answer, or the answer will be a singularity (e.g. Q: "Between the green ball, and the small blue ball, which one is the heaviest?" A: "I cannot tell (Due to disconnected adjacency matrix (5.4))") In these cases, the answer will be preceded by the word "error", in order to be able to validate them with the Symbolic Executor (Ch. 4.6).

3.9 Generated Datasets and Variations used

In total, a maximum of 100 samples are generated for each level, with the exception of level 8, which 240 samples were generates in order to test scaling capabilities of the Dynamics Predictor [4.3]. In all cases, a maximum of 1000 questions are generated along each simulation with duplicates being discarded. In the final VQA results, 40 test samples per level are used. In order to train the Mass Estimator and the Friction Estimator, different size datasets were used (these networks did not utilize symbolic reasoning, therefore needed larger amounts of data), however videos were not rendered in those cases (Further explanations can be found in Chapters 5.4, 5.5 and 8).

A Neuro Symbolic approach has the added benefit of allowing monitoring in interme-

diate steps of the ensemble architecture, such as before and after the Translator, before and after the Frame Interpreter, before and after the Dynamics Predictor etc. Since -as will be explained in Chapter 4-, each module is trained separately, different datasets -all variations of the same- are used in each different step. Four main editions of the dataset exist:

1. The *Original Dataset* that contains the videos and the questions along with all the respective data tables.

2. The *Frames Dataset*, which contains the videos in the form of frames (separate images). This dataset is used as input to the Frame Interpreter (4.1) as well as for training the Dynamics Predictor.

3. The *Mask Dataset*, which is used to fine-tune the MaskRCNN used in the Frame Interpreter.

4. The *Interpreted Frames Dataset*. This is used in order to train the Dynamics Predictor, as it contains the interpretations of the video, in symbolic form.

4 Proposed Model

Upon taking into account the already existing technologies, we wanted to add more physics concepts and higher level understanding in the VideoQA tasks of NS-AI. For this purpose, the **PhAQ** dataset, along with the **NewtoNN** architecture were developed.

The architecture combines known elements of already existing VQA methods, as well as adding new aspects, in order to be able to answer questions posed in the new dataset. A simplified macroscopic image of the model can be seen in Figure 4.1.

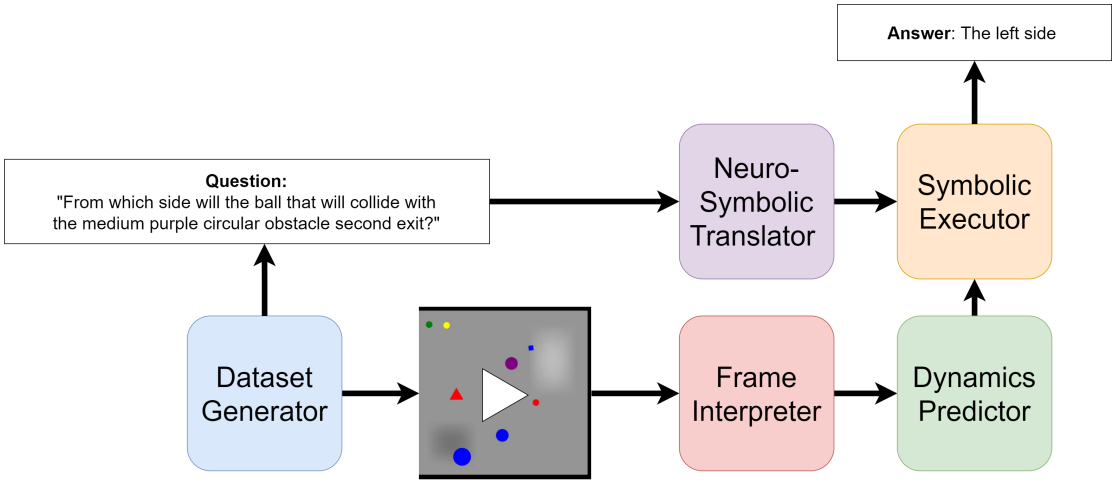

Figure 4.1: Macroscopic view of Neuro-Symbolic NewtoNN model.

As can be seen from the figure above, the model is comprised of the following modules:

- **Dataset Generator**: This module generates the dataset. It simulates the trajectories and events of the balls and renders them into small (10 sec. maximum duration) videos. Additionally, it generates questions regarding each simulation, as well as the answers and programs of each question.
- **Frame Interpreter**: This module interprets the video in a frame-by-frame manner, turning it into a symbolic representation, with each object's attributes, and positional information.
- **Dynamics Predictor**: This module locates the events that take place given the sequence of frames. Additionally, when paired with other sub-modules, it can infer the mass of the balls and the friction that they are subject to.
- **Neuro-Symbolic Translator**: This module transforms the natural language question into a series of commands called programs.
- **Symbolic Executor**: This module pairs the series of programs with the output of the Dynamics Predictor and provides answers in Natural Language.

These modules, are all trained separately, with the exception of the Symbolic Executor, that doesn't require any training as it relies on symbolic (rule-based) logic. In the following sections, each module will be analyzed.

4.1 Frame Interpreter

In order to be able to interpret the video frames into something a neural network can understand, they will have to be transformed into another format, one interpretable by the next component. Recurrent-CNNs [37] (where each frame is passed through a CNN, and then the temporal information is used for inference through a RNN) have two very limiting factors. Firstly, they require a lot of training data in order to capture the latent distributions successfully, and secondly their encodings (that take place inside the CNN) are not interpretable to humans, and therefore it is impossible to tell if the network overfits, uses previous bias for inference, or learns characteristics that it shouldn't without large amounts of training data to test it.

For the reasons above, a different approach will be followed. Neuro-Symbolic AI, uses symbolic representations for both commands/operations, as well as items. Therefore, a latent representation of the positions and attributes of all the objects, would be useless. The first part of the total architecture consists of being able to locate and classify the objects (mobile, or immobile) within the video. This is achieved through a Mask Region-based Convolutional Neural Network (MaskRCNN) [21], that takes as input each separate frame of the video, and outputs information such as its position (in the form of a "mask", as a highlight upon the original image), bounding box and attributes in the form of classification (e.g. type, shape and size). The MaskRCNN is considered an extension of Faster RCNN [39], which in turn is considered an extension of Fast RCNN [13], which itself is an extension of the initial proposition, the RCNN [14].

Unlike its predecessors, a MarkRCNN does not rely on a moving window, nor on simple bounding box segmentation. Instead, it produces pixel-level segmentation masks that accurately highlight the location of each object in the frame, as well as providing a label for each mask so as to classify any object it detects.

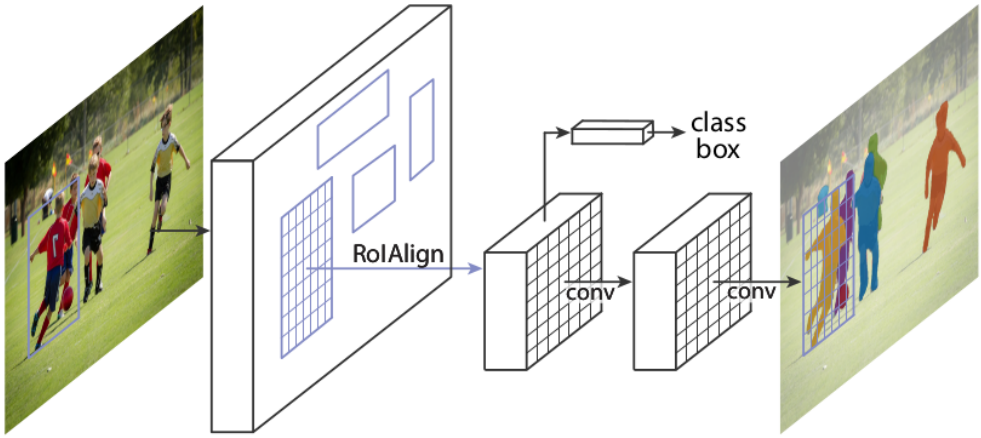

Figure 4.2: Mask R-CNN architecture.

MaskRCNN achieves this by adding a branch for predicting segmentation masks for each Region of Interest (RoI), in parallel with the existing branch. This mask branch, is a small FCN that predicts a segmentation mask in a pixel-to-pixel manner. A FCN is added on top of the features, in order to generate the masks in parallel to the classification and

bounding box regressor.

As previously proposed in the FastRCNN, a RoI-pooling operation takes place. However, when MaskRCNN is run without any modifications, the regions are misaligned due to the Max Pooling taking place inside the convolutions. Therefore, a slightly altered version of RoI-Pooling is implemented. RoIAllign aims to project the regions to a map of fixed-size. However instead of using the max function, it uses a linear interpolation method in order to retain positional data. A demonstration of how RoIAllign works, can be seen in Figure 4.3.

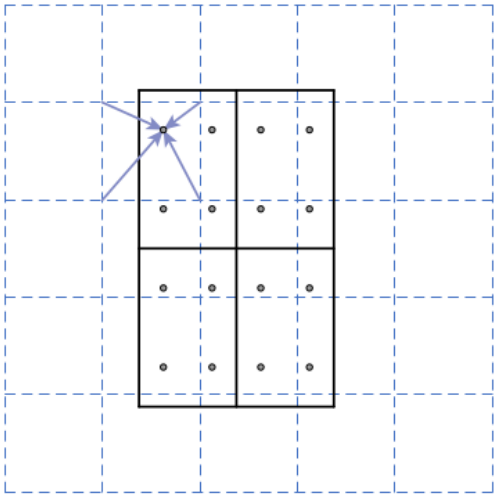

Figure 4.3: RoIAllign in MaskRCNN

In all cases, the MaskRCNN architecture is paired with a standard Image Recognition network, such as ResNet [22], in order to extract the characteristics of each image. Since a MaskRCNN is a rather large model ($\sim$ 44M parameters), a large dataset has to be used to train it from scratch. However, in most cases, fine-tuning an already trained model is fast and incredibly efficient.

4.2 Question Parsing and Translation

In order to be able to answer a question, an algorithm first has to understand that question. This takes place with a question parser, which extracts the meaning of that question in some form. A large number of studies in the realm of VQA have revolved around parsing questions in Natural Language with most attempts aiming to transform each question into a latent vector that represents its meaning.

As mentioned in Chapter 2, by using latent encodings to describe the contents of an image and the context of a question, good results can be achieved, however we still rely on huge statistical models to capture the real distributions and answer *rare* questions. Moreover, we have no supervision on what the visual component and the question parser have understood, and can only evaluate the model end-to-end, without possible supervision in the intermediate uninterpretable steps.

Dissimilarly from a standard semantic parser, a Neuro-Symbolic approach follows a different path. The semantics of a sentence/question are not directly extracted, but rather

translated into a *new* language, that comprises of simple, human-interpretable commands. These commands, are called "programs", and are simple operations that are performed on one or more data tables, which includes all the data requires to answer the question. In practice, they are a *Domain Specific Language*. These programms have two main benefits. Their number is rather small, since a plethora of commands can be created by joining the programs in different order. Additionally, each program represents a very simple, yet known operation, which allows the addition of prior knowledge in the Symbolic Executor 4.6, without however introducing bias.

Generated from the dataset generator (for training purposes), or produced by the MaskRCNN ([21], Ch. 4.1) and the Propagation Network ([32], Ch. 4.3) (during inference), are a set of data tables containing all the information about the participating objects, as well as the events that take place during the simulation. These are the inputs of the Symbolic Executor. A data table has the added benefit of being easy to manipulate with simple commands, like filtering. For this exact reason, a Natural Language question is translated into a series of these commands that will -in turn- be run on the data tables, by the Symbolic executor and provide the answer. Rather than going into depth, three direct examples will be provided. In this case, a Natural Language question will be translated into a series of commands:

1. **Do the big circular obstacle and the big ball have the same color?**
   This question asks whether two entities in a scene (the blue circular obstacle and the big ball) have the same color. As shown in Chapter 3, each simulation (either generated or inferred) provides us with a data table containing the attributes of the participating objects, in this case the balls. Therefore a series of "commands" can be performed on that data table in order to give us the answer. The above question can be translated into the Symbolic order of commands:

   objects, big, filter_size, circle, filter_shape, obstacle, filter_type, unique, query_color, objects, big, filter_size, ball, filter_type, unique, query_color, are_equal

   From an initial point of view, it is just an unintelligible series of random words. However, taking a look at Figure 4.4 might give some extra insight.

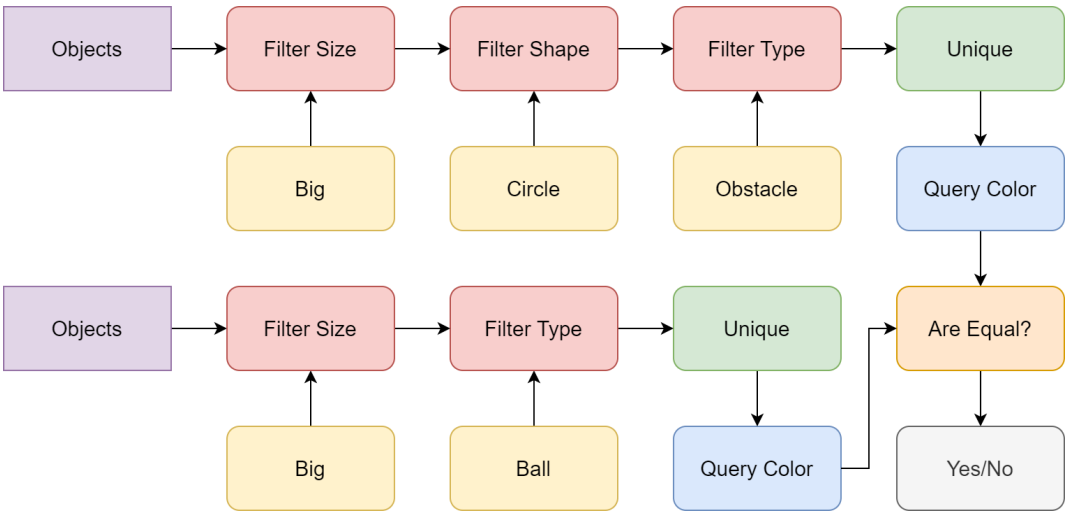

Figure 4.4: Logic diagram as interpreted from the programme series.

In the above diagram, we can see that the purple "objects" tab generates a list of all the objects that exist in the scene, while each filter (depicted in red), filters the objects by its relevant attributes (shown in yellow). The "unique" boxes (shown in green), return a single item from the list of filtered objects, and the blue "query" boxes, cross reference the ID of each proposed object with its attributes, and returns the attributes. Finally, the orange "are equal" box, compares the attributes, and provides an answer (shown in gray). The series of those commands is simply a linearized form of the above flowchart.

2. **Which ball will exit from the left wall second?**

   Has the equivalent symbolic sequence of:

   events, ball_exits, filter_event_type, sides, left, filter_side, filter_events, second, filter_order, ball, query_object, unique, describe_ball

   Which in turn can be better understood with the diagram depicted in Figure 4.5.

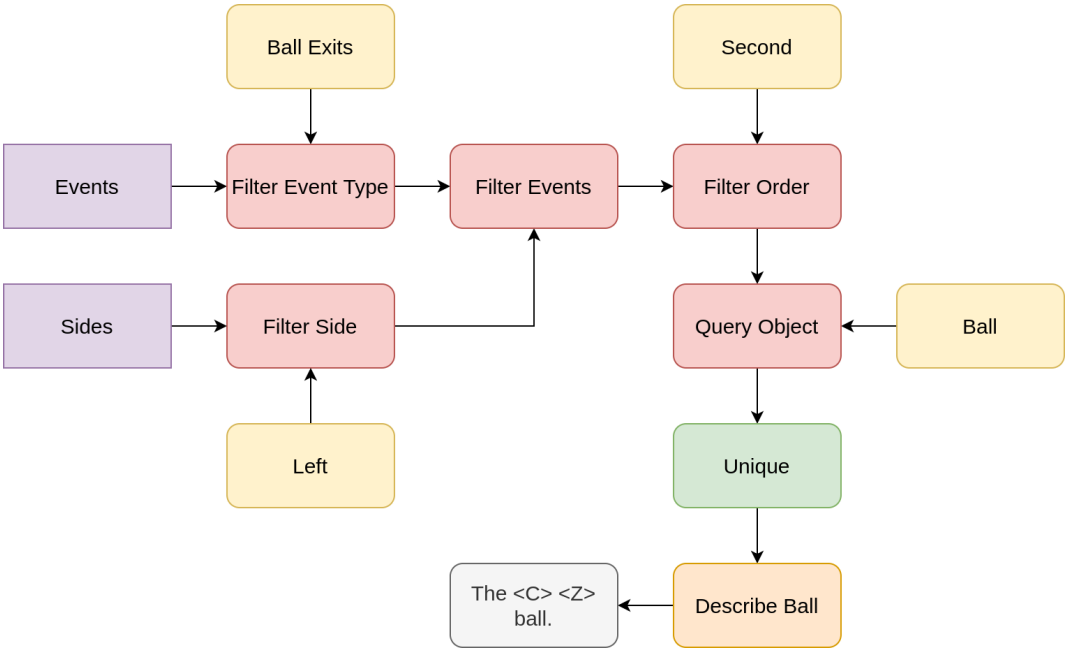

Figure 4.5: Logic diagram as interpreted from the programme series.

Similar to the previous example, the events and the sides are generated by the purple tabs. Then, only the events that are "ball exits" are passed through, and from those, only those that exit from the *left* side are passed to the next stage. Since the events all have a timestamp and are ordered by the timestep they have happened, we can filter them based on this order. In this case, we choose the *second* event, and from that, we query its participating ball object (the other participating object is the side, which we have already filtered by, and is the *left* side). Then, the natural language answer is provided by simply describing the ball.

3. **How many times will a red big ball and a medium purple ball collide?**

   Can be translated in the following symbolic sequence:

   events, ball2ball, filter_event_type, objects, red, filter_color, big, filter_size, filter_events, events, ball2ball, filter_event_type, objects, purple, filter_color, medium, filter_size, filter_events, intersection, count

   Which in turn forms the flow chart shown in Figure 4.6.

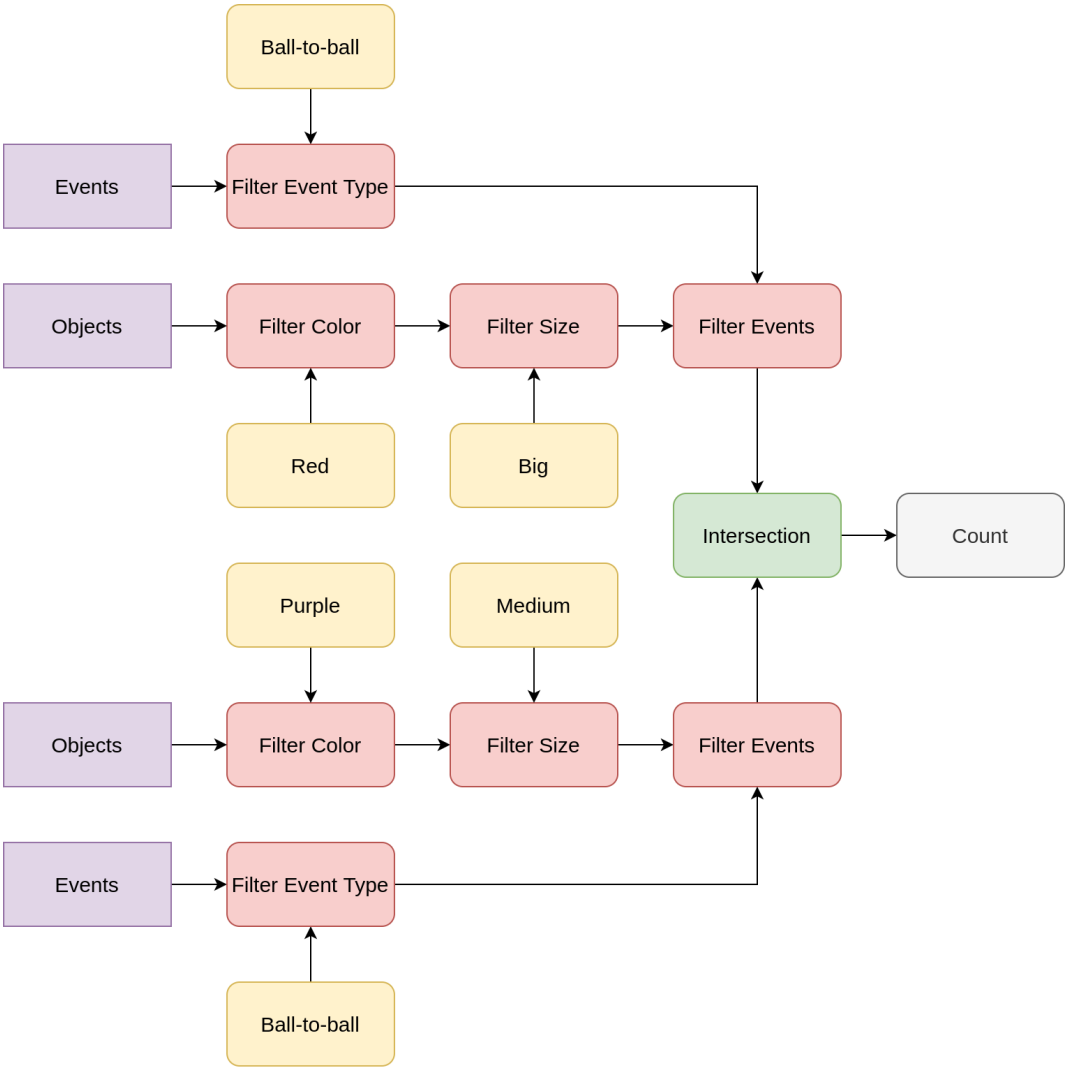

Figure 4.6: Logic diagram as interpreted from the programme series.

In this example, we filter both balls based on their attributes from the list of available objects. Then we separately filter all events of type *ball-to-ball* based on each balls attributes. The *intersection* operation finds the common events between the two filtered lists. These events are the ones that both balls participate in, and by counting them, we get final answer.

All questions are translated into their respective Symbolic sequences. The Symbolic sequences are generated in templated format along with each question during the generation of the dataset and are in turn used to train the Neuro-Symbolic translator.

4.3 Event Interpretation

The main part of this work, is to create an algorithm that will be able to *identify* the events that take place by receiving the interpreted frames from the Frame Interpreter module (Ch. 4.1) as input, by understanding the trajectories and attributes of the objects in relation to their surroundings.

This might -at first glance- be seen as something simple. However, one must realize that rigid-body collisions are a chaotic phenomenon. A slight change in the initial conditions can change whether a collision will take place or not, thus changing the whole outcome of the simulation. Perhaps a solution would be a sequence-to-sequence network that would receive the temporal interpreted frames, and output a sequence of events. However, this would be overly complicated since the relations in a physical simulation only have dependencies in the past (and strictly speaking only to their direct past), and sequence to sequence networks are build in order to retain information from arbitrarily long time steps. In addition to this, force propagation poses an extra problem since it cannot happen in a single timestep between multiple objects (as proposed in Interaction Network [4]). For all the above mentioned challenges (event identification and prediction) as well as to mitigate potential misses and wrong interpretation of phenomena, a Propagation Network (PropNet [32]) is used in order to successfully surpass them. A Propagation Network is a differentiable, learnable, dynamics model, that aims to handle physical simulations, as well as partially observable scenarios, by representing them as a graph. All of this action takes place in a symbolic manner, where its inputs are fully interpretable, while the network itself transforms them into latent representations, in order to inherit the numerous perks of neural networks. PropNet's perks mainly include instantaneous propagation of signals, which, in the case of colliding balls is incredibly important, since rigid body collision systems can be highly nonlinear (chaotic), therefore force propagation should be given more attention than in other cases.

4.4 Mass Estimation

An interesting aspect of the physics simulations, is to be able to extract physical quantities from the video of the simulation. Apart from obvious, easy to extract attributes, like position and velocity (position is given by the MaskRCNN, and velocity can be calculated by the difference of two consecutive positions), other, far more difficult attributes are mass and resistance. The focus of this section will be on an object's mass. As can be inferred from Equations 3.2 and 3.3 in Chapter 3, mass and velocity are correlated through the two equations that describe a collision, therefore, since velocity is known, we could train a network to be able to classify objects based on their mass.

The way this can be performed is through a branched LSTM [12]. That is a network that is trained upon the positions of the balls as a sequence, then the hidden state is passed through a branched classifier that classifies each ball (we train the network for 5 balls, since that is the max number used in the simulations) in a category. There are a total of 5 categories:

1. "None"
2. Small mass
3. Medium mass
4. Big mass
5. Infinite mass

The 1st category exists, so that cases with less than 5 balls are manageable (0 inputs everywhere is "None" mass). The 5th category, intends to be able to manage cases where obstacles are not separate entities from balls, but dynamic objects as well, albeit with infinite

mass. This -as has been mentioned before in Chapter 3- gives the added ability to have balls and obstacles of the same sizes by being able to discern between them strictly based on their dynamic behavior [1].

The aim of such a network is to be able to classify all the balls in a simulation to their correct masses, and in turn complete VideoQA tasks successfully. Such an attempt is not only important in order to understand the limits of Machine Learning, but to also further develop dynamic understanding of perceived objects in real life scenarios, such as autonomous car driving.

4.5 Friction Estimation

Apart from tasks like estimating the mass of the participants in a simulation, we add a sub-network that aims to understand the underlying physics regarding the *type* and *level* of air resistance within the simulation. In the simulations [2], there can be three (3) separate types of resistance: 1) No resistance 2) Dry Friction 3) Aerodynamic Drag, with the latter two having three (3) separate levels each. Their underlying physics are fundamentally different. When an object is facing dry friction, the opposing force is directly proportional to its weight, while an object facing aerodynamic force has force that is proportional to a power of its velocity as well as its radius. In our case, we use Stoke's Law, where the friction is proportional to the 1st power of the velocity, while the intensity of the drag can change with the viscosity parameter of the fluid that the balls move in. Since all balls face the same resistance [3], the resistance is global for each simulation, therefore the questions that can be asked are limited to asking the type and level of resistance, since no comparisons can be made inbetween balls.

4.6 Symbolic Executor

The symbolic executor is the final module in the end-to-end architecture, and works in a purely symbolic manner (no prior learning required). What is interesting about this module, is that the language itself provides a "prior" knowledge regarding the operations that need to take place, without however resorting to biases, since it remains fully rule-based.

At this point, there is a set of data-tables, containing the events and the object attributes that have been extracted by the Frame Interpreter and the Dynamics Predictor, and a set of commands that have to be applied to these data-tables in order to extract an answer. These commands have been tailored to be specifically applied to a data table, therefore this step is rather straightforward.

This takes place as a sequential, stack-based approach. Each program receives a variable number of inputs, and creates a specific output. The active inputs (that is the ones that will be used by later programms) are placed in a stack. The final program should reduce the number of items in the stack to one; the answer to the question.

[1] This is left as potential future work.

[2] In practice a different dataset is used to train and test the friction estimator so that it doesn't interfere with the already developed components.

[3] An interesting future development would be for each ball to have completely separate dynamic behavior

5 Experimental Setup

This chapter will be dedicated to thoroughly explaining the network's ensemble setup, as well as the types of training data that were used and the different setups used in each sub-module of the network. Analysis will take place for the image segmentation component, the dynamics predictor along with the mass and friction estimators, the neuro-symbolic translator, as well as the symbolic executor. Results for each module separately, as well as end-to-end VQA results, will be presented in Chapters 6 and 7.

5.1 Frame Interpreter

In order to transform our simulations from video format into a symbolic representation, each simulation needs to be converted into a series of symbolic attributes, that can in turn be used and processed in order to extract answers. This is done through a MaskRCNN [21], since, apart from bounding boxes and classification labels, it generates masks which can be used to extract precise positional data for each object.

Training a MaskRCNN is expensive. In order to bypass this, we used a pre-trained MaskRCNN on the *Penn-Fudan Database for Pedestrian Detection and Segmentation* dataset [52], and by fine-tuning it we tailored it to our task. As mentioned in Chapter 4.1, a MarkRCNN requires for training a set of still images, along with each objects mask and label. Due to the nature of this custom made dataset, these can be produced with slight modifications to the already existing code used for the generation. Thus, a number of training samples are generated, as described in Chapter 3. This is a number of images, as well as their masks in binary format, and the labels accompanying each mask. The masks are, in simple terms, a 2D array with the same resolution as the image, but in the position of each object, instead of *zeros*, there are *ones* for the first item, *twos* for the second, and so on. This gives the positional data of every item, while also providing information regarding its shape and size. Apart from the mask, a series of annotations are given. These include the type of the items that exist in the image (the labels that the masks point to), as well as the bounding boxes of each item, in the form of coordinates of the bottom left, and upper right corners of their respective rectangles. All annotations are compatible with *PASCAL Annotation Version 1.00*.

A total of 18+1 classification categories are used. The 18 classification categories each correspond to a class of items (e.g. left wall, small ball, medium triangular obstacle), as depicted in Table 3.4. The zero-th class is the Null class, which is the background.

As can be seen from the number of categories, the colors were not included in the classes used by the MaskRCNN. As mentioned in Chapter 4.1, this would only complicate things and for this purpose, the colors of each object were calculated by averaging the RGB value of the pixels of an image, after being masked with each items corresponding mask, and classifying the result with a Support Vector Machine (SVM). As the visual component of the dataset is simple, color classification accuracy was 100%.

Since a pre-trained model is used, it is expected that a rather small number of epochs will be required. For this training, we used a total of 1080 frames on an 80-20 split, with each frame being randomly sampled from a simulation, with 10 frames taken per simulation. All

frames are taken from simulations of difficulty level 8 [4], in order to include the most objects per frame possible. Since balls might exit between frames, the number of items depicted in each frame varies. 900 frames were used for *training*, while another 180 for *validation*.

5.1.1 Contents of predicted outputs, and file layout

An interesting aspect that will provide insight into the networks operation, is the output of the Image Segmentation component. The MaskRCNN is simply a tool used, and not the whole component itself.

Each video is broken into its respective frames (50 fps * 10 sec = 500 frames, with videos ending sooner having less frames). Each frame is independently passed through the MaskRCNN. After the output of the model, an SVM is used to determine the color of the objects that have colors. The output of each frame consists of:

- Frame filename
- Frame index
- **Objects**. This is a list of all the objects detected by the MaskRCNN. Each object has the following attributes:
  - Score. This is a number between 0 and 1. It determines the certainty of the correctness of the proposed object. We set a threshold, so that proposed objects with low score are discarded. This threshold is set at 0.8, meaning that an object with a score of e.g. 0.7 would be discarded and would never appear in the final proposals. In most cases, the scores were above 0.95.
  - Mask. This is the item's mask, in RLE (Run Length Encoding) format. The dimension of the initial frame is given, in order to be able to decode it.
  - Video Index
  - Frame Index
  - Frame filename
  - Type of object
  - Rest of the attributes. Balls have *size* and *color*, obstacles have *shape*, *size* and *color*, walls have *side* etc.

All the above information is stored in a single **.json* file, that in practice contains all the interpreted information of the whole video; in practice an *un-rendered* video.

5.2 Neuro-Symbolic Translator

A Neuro-Symbolic translator aims to transform questions from Natural Language to their counterparts in a new Symbolic program format. As can be seen in Appendix A.1, while there are a plethora of questions, the number of tokens (different words) in both Natural Language questions as well as their programms, is limited. For this reason, the model used in this case is an Encoder-Decoder GRU-RNN, with the decoder containing an attention mechanism. As with most sequence-to-sequence models, the embedding layer is included in the encoder (instead of "feeding" it with pre-calculated embeddings). Then, the

[4] In the training set, three gravity fields instead of the standard two were used.

resulting context embedding is passed through a GRU unit, outputting the encoded temporal representation of the natural language sentence.

The attention layer is applied directly to the outputs of the encoder, and concatenated with the non-weighted encoders embeddings. The resulting vector, is passed through a linear layer, and the *new* embedding, that better describes the sequence, is passed through a non-linear unit (ReLU), and finally through a GRU unit, in order to generate the output Symbolic sequence. The network can be seen in Figure 5.1.

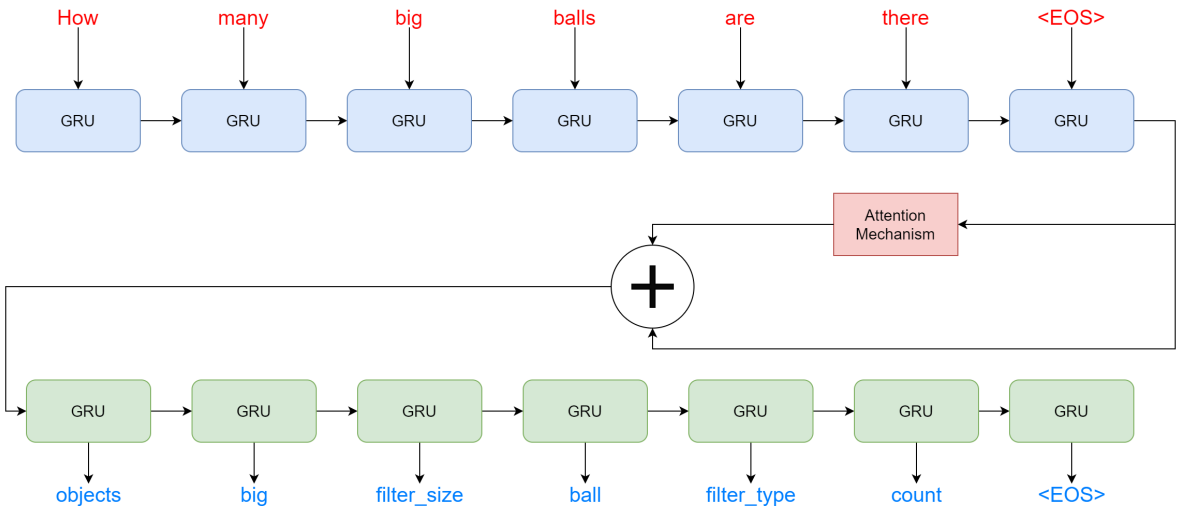

Figure 5.1: Sequence-to-sequence GRU network, with attention mechanism

As the vocabulary is limited, it is expected of the network to be able to reach high accuracies with limited data. Since the output of a Symbolic Executor can change dramatically if a single program is different, it is imperative that the sequence is exactly the same as the one intended. This is the reason for a strict definition of accuracy (instead of e.g. the Levenshtein distance [30]). For this purpose, the accuracy is defined as 5.1:

$$\text{accuracy} = \frac{\text{correct predictions of sequence}}{\text{total number of predictions}} \cdot 100\% \tag{5.1}$$

5.3 Propagation Network

A Propagation Network (PropNet) is a differentiable, learnable, dynamics model, that aims to handle physical simulations, as well as partially observable scenarios, by representing them as a graph. Its perks mainly include instantaneous propagation of signals, which in the case of colliding balls is incredibly important, since rigid body collision systems can be highly nonlinear (chaotic), therefore force propagation should be given more attention than in other cases.

Similarly to most graph networks, there are a number of objects (O), and relations (R) that "connect" those objects together. The aim is to train the network in order to understand these relations correctly given an input. More can be found in Appendix A.13.

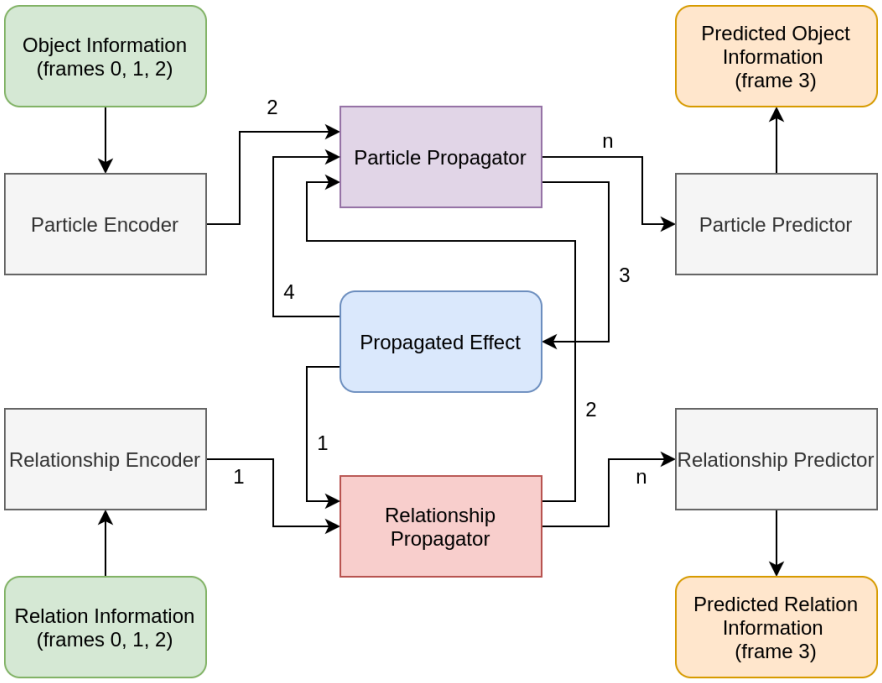

Figure 5.2: Architecture of PropNet. Encoders and decoders are annotated with grey color, propagators with purple and red, and states/data with rounded-edge rectangles.

5.3.1 Training, Inputs, Outputs

A Propagation Network aims to produce a sequence of events, as well as predict the positions of items, when given a sequence of inputs. In more detail, it requires two different versions of the same dataset in order to operate. The first dataset, is the sequence of interpreted frames, as given by the Image Processing component (MaskRCNN + color classifier (Ch. 4.1)). Unlike most networks where pre-processed data is given in a latent vector, these inputs are fully interpretable by a human, allowing monitoring of the input and output processes of the PropNet more closely than if a Neuro-Symbolic approach was not followed. The second dataset used, is the extracted frames from the videos. These are used to cross-reference the outputs, and increase the output accuracy by some degree.

Each frame is given in a very precise manner. First, the frame is down-scaled to a specified dimension, and in every x, y position, attributes are allocated. The symbolic characteristics are encoded into a 22-dimensional vector, as shown in Table 5.1.

Position	Attribute	Position	Attribute
1	ball	12	medium
2	obstacle	13	big
3	wall	14	circle
4	gravity field	15	square
5	side	16	triangle
6	red	17	left
7	blue	18	lower
8	yellow	19	right
9	purple	20	upper
10	green	21	positive
11	small	22	negative

Table 5.1: Attribute encodings used in PropNet

The mask of each object allows the extraction of the coordinates in (x, y) format. The image is down-scaled into a predetermined dimension based on the size of the networks encoders. Then, the information is added into a 4D tensor, where the first dimension is in practice the ID of each item, the 2nd is the encoding of the 22-dimensional vector, and dimensions 3 and 4 are the x and y coordinates. This way, the network has the ability to retain pixel level attributional information during the input. Additionally, positional information in the form of (x, y) is given per object, per timestep. All the information above is concatenated, so that all three previous time steps (-2, -1, and 0) are given as as single tensor. As will be mentioned, elements that fail to be detected inbetween frames or elements that no longer exist in the frames (balls that have exited) are replaced by dummy tensors in their respective positions in order to not discard those cases. Finally, the relations between the objects are given into a 1-hot encoding N-by-M matrix, where each (i, j) and (j, i) position has a positive value if an event takes place, or a negative value if nothing happens. Dimension N is the number of potential relations ($number\ of\ objects^2$), and M is defined as:

$$M = relation\_dim * (n\_his + 2) \tag{5.2}$$

where *relation_dim* is the number of types of potential relations that can take place. In our case, that is 6 (4+2): 1) *ball-to-ball* collision, 2) *ball-to-wall* collision, 3) *ball-to-obstacle* collision and 4) *ball exits the domain*. The other 2, are the Δx and Δy between the center masses of the two objects. The *n_hist* is the number of previous timesteps given to the network, in order to get a temporal image of the situation. In our case, this is set to 2. This matrix gives us information regarding the events that took place in the past 3 frames, as well as the information that needs to be predicted for the next frame. The purpose of this whole process is to be able to predict the information contained in frame $ts + 1$.

For this purpose, certain losses have to be defined. The first, is the *mask loss*, which compares the predicted mask (position) of the object, and the actual mask. The second is the *positional loss*, which compares the predicted position of the object, and the ground truth. The third is the *image loss*, which correlates the predicted sub-sampled image, to the predicted images. The last error, is the *event loss*, which compares the actual events that

take place, to the predicted events.

All losses are defined as *Mean Square Error Loss*. The network is trained by weight-adding the four losses. Even though the first three losses help during the training and provide extra feedback loops to the network, their results will not be used within the framework of this Thesis (see Chapter 8).

5.3.2 Additional work and differences from existing implementations

PropNet [32] is a very versatile network, meaning it can locate temporal events and predict future states by minimizing the *RMS* error. However, the number of objects that are used in the simulation has to be *invariable*. In its implementation in CLEVRER, even though two types of events were recorded by the dataset generator (collisions and exits), only one of those was eventually used for the temporal understanding and prediction aspects of the dataset.

In our case, this cannot be overlooked, since collisions (between different types of objects) and domain exits are incredibly important. For this purpose, the final network is substantially different than the original PropNet, mainly regarding its data preprocessing and event management. The main characteristics that have changed are:

1. **Different (physical) object management**
   In the case of CLEVRER, a total of 4 frames are used as each training example. 3 frames are the input, while the fourth is the one that has to be predicted, along with any types of events that will lead to it. These frames are usually not continuous, but are instead sampled from the video every 5 frames, in order to capture temporal information. However, since the contents of each frame heavily depend on whether the visual aspect of the architecture (MaskRCNN) will successfully locate all the objects or if an object will exit the domain at some point between the frames, the number of objects inbetween frames might vary. This issue is not addressed in the variation of PropNet in CLEVRER, and these cases are simply discarded. In our case, we carefully manage the objects in all the frames, in order to retain frames with variable object count. This is done by creating a super-set of the objects detected in each frame, and simply filling any non-existing cases of objects with dummy arrays. This allows the network to handle cases with variable objects per group of frames, and in turn allows the detection of events with variable objects (ball exits), as well as handle failures of the visual component (See Table 5.2).

Objects	Frame -2	Frame -1	Frame 0	Frame 1	Global Objects
Ball 1	✓	✗	✗	✗	✓
Ball 2	✗	✓	✓	✓	✓
Ball 3	✓	✓	✓	✓	✓
Ball 4	✓	✓	✗	✓	✓
Wall 1	✓	✓	✓	✓	✓
Wall 2	✓	✗	✓	✓	✓
Wall 3	✓	✓	✓	✓	✓
Gravity Field 1	✓	✓	✓	✗	✓
Gravity Field 2	✓	✓	✓	✓	✓
Obstacle 1	✓	✓	✗	✓	✓
Obstacle 2	✓	✓	✓	✓	✓
Obstacle 3	✗	✓	✓	✓	✓

Table 5.2: Objects located in sequential frames, and objects in all the frames. ✓ denotes successful detection of an object, while ✗ denotes failure by the visual component.

2. **Expanded event encoded representation**
   The way events are encoded in CLEVRER is rather simple. A standardized one-hot encoding is used (instead of 0,1 it is scaled to -0.5,0.5). When an event takes place, that number is positive, and a few more characteristics regarding said event are given. When an event is not taking place in these frames, that number is -0.5. This is rather simplified, since only a single type of event is tracked in CLEVRER (object-to-object collision). In our case, we have to expand this in order to include all types of events, therefore that vector has to be further expanded.
3. **Event tracking evaluation**
   In the original paper, whether an event was tracked successfully or not, could only be evaluated with the MSE Loss that was used for training. In this case we can also track the events' Precision, Recall, False Discovery Rate and Miss Rate as can be seen in Chapter 5.3.3.

The output of the evaluation step of the network, generates a **.csv* file as the log of the events. This file is similar to the one being produced during the initial simulation, and can be compared to it through various ways. There are two different methods to evaluate PropNet's performance: as a separate entity, or as a part of the total VQA architecture (end2end approach).

The first way is the preferred, since PropNet is used for event tracking, and the accuracy of event detection needs to be evaluated. As mentioned in Chapter 4.3, we can compare the ground truth events with the predicted ones, and consider an event as found, if it is found successfully within a certain timeframe of the original (see Chapter 5.3.3). The end-to-end evaluation could be used as well, however it would not give insight to PropNet's performance, therefore negating the perks of the Neuro-Symbolic approach.

5.3.3 Event detection evaluation

In order to validate the event detection accuracy of the Propagation Network, the predicted events are compared to the ground truth events in order to extract certain metrics.

There can be a total of four (4) cases of events:

- **True Positive** (TP): An event that is correctly identified
- **False Positive** (FP): An event that is identified, but doesn't really exist. This can happen if a non-existent event is detected, or if a correct event is detected twice.
- **False Negative** (FN): An event that does happen, but isn't detected.
- **True Negative** (TN): Cannot be defined in our case, since events are sparse events. But strictly speaking, it would mean that an event that doesn't happen, indeed doesn't get detected (which, due to being in most timesteps, is pointless to define). This also prohibits the use of "accuracy", since it requires TNs to be correctly defined.

With the above three (3) metrics (since TN is rejected in our case), we can define the following metrics to give us some insight:

- $Precision(PR) = \frac{TP}{TP+FP}$
- $Recall = \frac{TP}{TP+FN}$
- $FalseDiscoveryRate(FDR) = \frac{FP}{FP+TP}$
- $MissRate(MR) = \frac{FN}{FN+TP}$

Lastly, an event is considered successfully detected, if its *type*, *item_1* and *item_2* are identical, while the timestamp satisfies the following condition:

$$|t_{pred} - t_{gt}| \leq p_t \frac{\text{frame offset}}{\text{fps}} \tag{5.3}$$

Where t_{pred} is the predicted time that the event will take place, t_{gt} is the ground truth time that the event will take place, $frame\_offset$ is PropNets sampling rate (5 is used in this case), fps are the frames-per-second used in the videos simulation (which is 50), and finally p_t is a window parameter. If $p_t = 1$, it means that *only* the events inside a window of $\pm\frac{frame\ offset}{fps} = 0.1sec$ will be considered valid. Since this restriction is rather strict and will lead to incorrect metrics, for evaluation purposes $p_t = 2$ will be used.

5.3.4 Parameterization

As with any Neural Network, Prop-Net can be tuned with various meta-parameters, that will change its output.

The most important parameter that can be "tuned" in PropNet, is the frame offset. That is the rate at which time-steps/frames will be sampled from the video/frames in order to be trained/inferred. A larger frame offset, means a lower sampling rate (less frames sampled per second). A bigger frame offset has the added benefit that training as well as inference take less time to perform, since the dataset ends up being smaller. Also, the

timestep at which events are located is less strict, since each time window in which an event needs to be located is larger (see 5.3.3). Additionally, slower events that the solver attributes to a single frame (like ball exits) are now easier to locate, since the visual component's ability to detect them is more guaranteed, as the ball will likely be fully within one frame, and fully outside in the next. After all, in a gradual exit a ball drastically changes shape (a ball gradually turns into a *part* of a ball, which the MaskRCNN might not be able to correctly classify). On the other hand though, a smaller frame offset, allows more accurate event location (time-wise), and also allows ball trajectory prediction to be more accurate, since practically the timestep of the simulation of PropNet (this time as a physics engine) is smaller and therefore more accurate, however temporal information is gradually lost as consecutive frames are very similar to each other, and the prediction accuracy is low. Through trial and error, a good value of frame-offset was deemed to be *5*, with a video fps of *50*.

Other meta-parameters that can be tuned, are the number of propagation steps taking place (Interaction Network: 1 propagation step, Propagation Network: 2 steps), as well as the hidden dimensions of the inputs and outputs of the networks encoders and decoders.

5.4 Mass Estimation

The purpose of the Mass Estimator, is to classify each of the participating balls in one of the 5 categories ("none", "low mass", "medium mass", "big mass", and "infinite mass"). In order for such a network to be trained, incredible attention should be given to the training data. If *all* simulations are used in order to classify their balls' masses, the result on the validation set will barely be above the baseline of 33% (3 equiprobable mass categories are used). For this purpose, it should be thoroughly explained under which conditions the mass can be inferred through the interactions alone.

The adjacency matrix of the ball-to-ball collisions is a square matrix where each (i,j) and (j,i) position contains the amount of times that ball i has collided with ball j. This matrix is symmetrical. One can look at this matrix as an undirected graph, where each collision produces a vertice between node i and node j. If all nodes are interconnected, directly or indirectly, one can infer the relations between their masses, since a collision between node i and node j means that their mass ratio is now known. The logic behind is is rather simple. If ball 1 is heavier than ball 2, and ball 2 is heavier than ball 3, then ball 1 is heavier than ball 3. Since the ratios of the masses are logarithmic[5], as each category is double the mass of the previous, the two extremes of all the categories have to be included in a training sample, in order to correctly classify them. If this is not the case, two identical simulations with balls belonging to categories *small, medium*, and *medium, big* will have the exact same dynamic behavior (same image on the simulation) hindering the classification capabilities, and therefore corrupting the training data. For this purpose, we demand that each training sample has data from at least the lightest and heaviest category.

One simple way to find if a simulation is adequate for inferring the masses, is to find the sign of the Fiedler value [11] of its Laplacian. A graph's Laplacian is defined as:

$$L = D - A \tag{5.4}$$

[5]The masses used in this scenario are 0.5, 1 and 2 kg

where D is the graph's degree matrix, and A the adjacency matrix. The degree matrix contains the number of other nodes that node i is connected to, while the adjacency matrix contains the number of times that node i is connected to node j. In our case, if a position is non-zero, it can simply be replaced with one (1).

After the Laplacian matrix has been calculated, we calculate its eigenvalues in ascending order. The second eigenvalue will be the Fiedler value. If its sign is positive, it means the graph is connected. This is the condition under which the selection of the data will take place. Two examples, one valid and one invalid, will follow:

- **Valid example**
  Consider the graph:

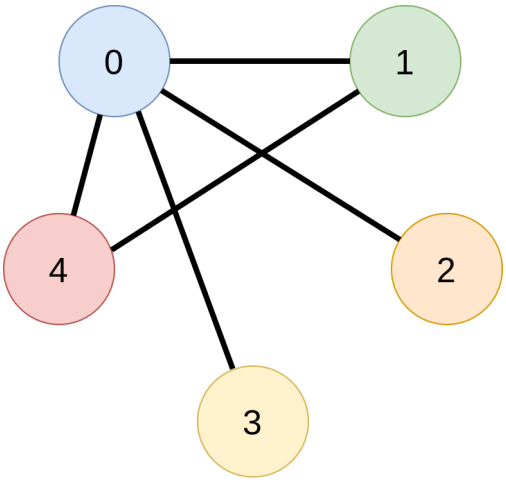

Figure 5.3: A connected graph.

Which has the following adjacency matrix (assuming multiple ball collisions between some pairs of balls):

$$A = \begin{bmatrix} 1 & 1 & 2 & 1 & 3 \\ 1 & 0 & 0 & 0 & 1 \\ 2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 \end{bmatrix}$$

And the following degree matrix:

$$D = \begin{bmatrix} 4 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

The Laplacian, is:

$$L = \begin{bmatrix} 4 & -1 & -2 & -1 & -3 \\ -1 & 2 & 0 & 0 & -1 \\ -2 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 \\ -3 & -1 & 0 & 0 & 2 \end{bmatrix}$$

Which has the following eigenvalues (in ascending order):

$$\lambda_i = (-1.546), (1), (1), (2.767), (6.779)$$

The second eigenvalue (λ_2) is 1, which is positive, therefore the graph is connected, which is indeed true, so the simulation is considered a *valid* training sample.

- **Invalid example**
  Consider the graph:

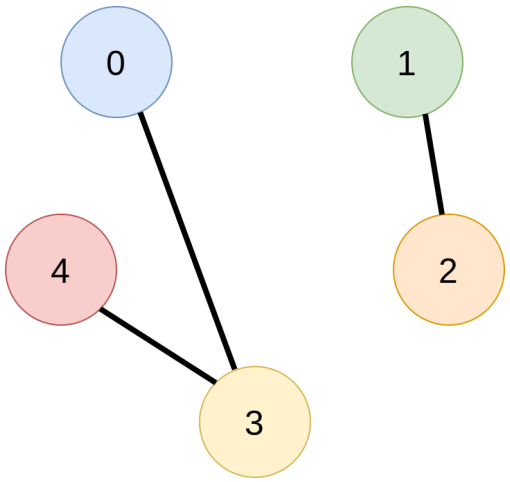

Figure 5.4: A disconnected graph.

Which has the following adjacency matrix (as calculated by the collision of the simulation):

$$A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

As well as the following degree matrix:

$$D = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The Laplacian matrix, as defined by $L = D - A$ is:

$$L = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 \\ 0 & 1 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Whose eigenvalues in ascending order are:

$$\lambda_i = (0), (0), (1), (2), (3)$$

The second eigenvalue (λ_2) is the Fiedler value. In this case, it's zero, therefore the graph is not connected, which is true, thus the interaction producing the graph is an *invalid* interaction.

5.4.1 Training Data

Initial attempts to train the Mass Estimator with the same number of samples as the Dynamics Predictor failed, with a marginal pass of the baseline accuracy. Since this is not a Neuro-Symbolic Model, standard Machine Learning techniques had to be used in order to achieve good accuracy, and this involved using *many* samples. With the rules posed above, not all samples are considered valid and/or interpretable. For this purpose, 500k samples of each of levels 1,3,4 and 8 were generated (total of 2M samples). However, after the filtering strategy used above, only about 90k samples were considered valid and therefore used for training. The training data consisted of the coordinates generated by the physics engine, and not by the MaskRCNN output. The reason for this was mainly the fact that the latter case would require rendering of the videos, and interpretation of each frame, which would add a huge time delay (a few weeks for around 90k samples). However, by following this tactic it took around 1-2 days to generate 2M samples on 3 separate multi-processor machines.

5.4.2 Model used

The model used to predict the masses of the participating balls, is an LSTM that branches into 5 classifiers, one for each different ball, with each classifier having 5 different categories. The coordinates are given as a time series, meaning that each timestep comprises of 25 values, x, y, u, v and $|u|$ of the 5 balls. Since the predictor operates with timesteps of 0.1 seconds, the model was trained on the same timesteps. In all cases the final 1000 elements of the dataset were used for validation, while a varying number of samples was used for training in order to test the model's scaling abilities. All the data used was separately generated from the initial dataset, so as to not interfere with overfitting on data used to test it [6].

5.5 Friction Estimation

The previous chapter analyzed the way that a neural network can learn dynamic behavior based on the movement of the balls. Dynamic behavior can be expanded into many areas. Another interesting area is that of movement type, when it is affected by a drag.

[6] More info on how the separate generation takes place can be found in the code's documentation

In order to set up this experiment [7], three different types of opposing forces were used:

- None: Vaccum like simulation (like the initial dataset)
- Dry friction
- Aerodynamic friction

Each simulation only faced a single type of friction, meaning that all balls face the same type of friction. The types are equiprobable, with a probability of 33.3%.

Dry friction is defined as an opposing to the movement force, that is proportional to the weight of the object (ball). In this case, the balls are assumed to be non-rotating, since a rotating ball has different dynamics. The force of dry friction is defined as:

$$F_d = \mu N \tag{5.5}$$

where μ is the friction coefficient, and N is the weight of the object, defined as gm, where m is the mass of the ball. This force is always the same, and depends on the mass, and *not* the object's speed (apart from when the object is stationary, where the force is zero).

Aerodynamic friction on the other hand can be defined in many ways. In this case, we opted to use Stoke's Law [48] which is perfectly tailored to our needs. It simulates the force acted upon a moving ball within a viscous fluid. The force acted upon the object is:

$$F_{drag} = 6\pi\nu|u|R \tag{5.6}$$

where ν is the viscosity of the fluid, R the radius of the ball, and $|u|$ the magnitude of its velocity.

During the setup of the problem, we additionally added the *degree* of each resistance as a parameter to be inferred. This means that the coefficients (μ and ν) are not constant, but sampled from three different values with equal probabilities:

- $\mu \in [0.002, 0.004, 0.008]$
- $\nu \in [0.6, 1.0, 1.4]$

In this scenario, two different approaches can be followed. The first comprises of two different classifiers, with each tailored to classifying each simulation into a class based on the *type* of resistance, while the second one determined the *level* of resistance (when there is no resistance, the level is considered as zero (0)), giving a total of 3 and 4 classes for classification respectively. However, this approach, does not provide easy understanding of the difficulties that the classifier faces, so we opted for a meta-class approach. This approach uses a single classifier with a total of 7 classes:

- Class 1: No resistance
- Class 2: Dry friction with $\mu = 0.002$

[7] Since this experiment requires changing the dynamics of the solver, any tests and results will be performed on a different dataset in order to not interfere with the existing datasets (that don't use drag or friction)

- Class 3: Dry friction with $\mu = 0.004$
- Class 4: Dry friction with $\mu = 0.008$
- Class 5: Drag (Stokes Law) with $\nu = 0.6$
- Class 6: Drag (Stokes Law) with $\nu = 1.0$
- Class 7: Drag (Stokes Law) with $\nu = 1.4$

The network used in both cases, was a bidirectional LSTM [8]. The results of the network, were used to answer questions of the type: "What type of resistance do the balls face in the simulation?" or "How much resistance do the balls face in the simulation?". The results of the trained networks, their accuracies, their generalizations, as well as their VQA answering successes can be found in Chapter 6.5.

5.6 Symbolic Executor

The Symbolic Executor is a pure rule-based operator that extracts information from the generated (or ground truth) data-tables, by applying commands on them, which are encoded in a series of symbolic sequences, called *programs*. The whole process takes place as a pipeline with a stack, that contains all the important information from any previous command applications. A stack can include multiple entities. And these entities are "consumed" each time a program is called (each program requires a varying number of entities to function). These entities can either be parts of the data table (a few objects), a characteristic or even an argument for another program. After a command is performed, its result goes on top of the stack, replacing any kind of required entities. An example of the stack can be seen in Figure 5.5

[8]The original source code of the network used can be found at: https://github.com/georgepar/slp

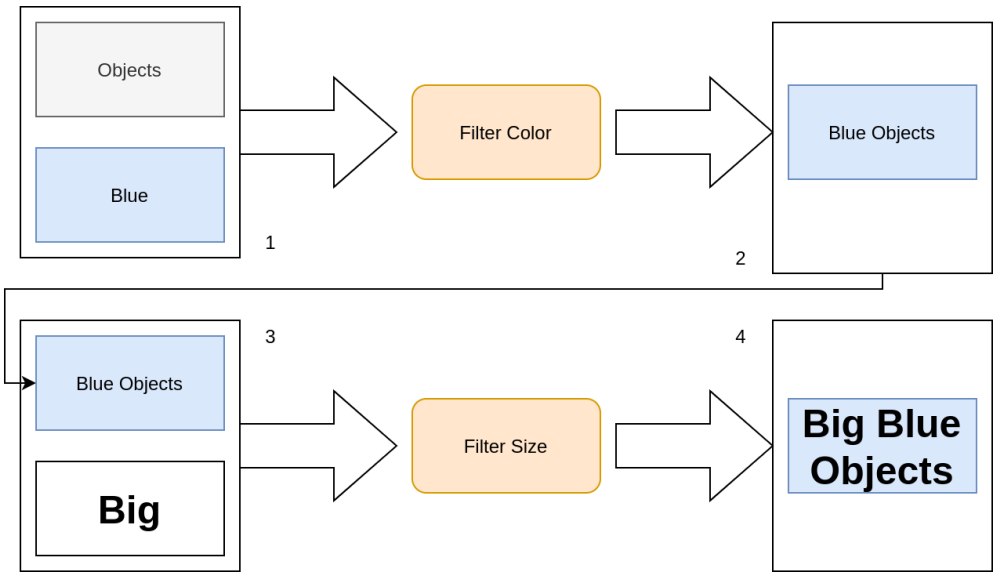

Figure 5.5: Stack during and after commands. In the initial state (1), the stack contains two entities, all the *objects*, and the attribute *blue*. After the operator *filter color*, the new stack (2) has a single element, just the *blue objects*. After that (3), another attribute is added to the stack: *big*. After the operator *filter size* is implemented, then the stack again is reduced to a single entity, that of the *big blue objects*.

A total of 39 commands exist, and each of them is applied to a corresponding data table, similarly to how it is shown in Figure 5.5. The commands used, are the following:

1. **objects**: Lists all the objects that participate in the simulation as a list of indices. Doesn't require any input.
2. **events**: Lists all the events that that participate in the simulation as a list of indices. Doesn't require any input.
3. **xy_vel**: Returns a data table with all the positions and velocities of all the balls in the simulation. Doesn't require any input.
4. **sides**: Lists all the sides that are not blocked by walls. Doesn't require any input.
5. **resistances**: Returns the data table with the resistance of the simulation. Doesn't require any input.
6. **filter_size**: It filters the objects based on the size attribute given. Requires 2 inputs (objects and size attribute).
7. **filter_color**: It filters the objects based on the color attribute given. Requires 2 inputs (objects and color attribute).
8. **filter_shape**: It filters the objects based on the shape attribute given. Requires 2 inputs (objects and shape attribute).
9. **filter_type**: It filters the objects based on the type attribute given. Requires 2 inputs (objects and type attribute).
10. **filter_side**: It filters the objects based on the side attribute given. Requires 2 inputs (objects and side attribute).
11. **count**: Counts the number of existing entities in its given list. Requires 1 input (the list).

12. **is_less**: Returns *Yes* if number_1 is less than number_2. Else, it returns *No*. Requires 2 inputs (number_1 and number_2).
13. **is_more**: Returns *Yes* if number_1 is more than number_2. Else, it returns *No*. Requires 2 inputs (number_1 and number_2).
14. **is_equal**: Returns *Yes* if number_1 is equal to number_2. Else, it returns *No*. Requires 2 inputs (number_1 and number_2).
15. **reduce_by_one**: Reduces the number given by one. Requires 1 input (the number given).
16. **more_than_one**: Checks if the list has more than one items inside it. Requires 1 input (the list).
17. **exists**: Checks if the list has any items inside it. Requires 1 input (the list).
18. **find_min**: Finds the index that minimizes a quantity. Receives 2 inputs (the data table, and the quantity).
19. **find_max**: Finds the index that maximizes a quantity. Receives 2 inputs (the data table, and the quantity).
20. **unique**: Returns the single element of the list that is given. Requires 1 input (the list).
21. **inv_boolean**: Returns *No* if the input is *Yes*, and vice versa. Requires 1 input (Yes or No).
22. **sort_xyv**: Orders a column of a data table of positions and velocities in ascending or descending order per timestep. Receives 3 inputs (the data table, the column variable, and the type of ordering; ascending or descending).
23. **filter_ts**: Returns a specific subsection of the position and velocities data table, with a specific timestep. Receives 2 inputs (the data table, and the timestep to be returned).
24. **query_color**: Returns the color of the given item. Requires 1 input (the item).
25. **query_shape**: Returns the shape of the given item. Requires 1 input (the item).
26. **query_size**: Returns the size of the given item. Requires 1 input (the item).
27. **query_mass**: Returns the mass of the given item. Requires 1 input (the item).
28. **query_type**: Returns the type of the resistance. Requires 1 input (the resistance data table).
29. **query_level**: Returns the level of the resistance. Requires 1 input (the resistance data table).
30. **union**: Returns the union of the two input sets (lists) of indices. Requires 2 inputs (the two lists/sets of indices).
31. **intersection**: Returns the intersection of the two input sets (lists) of indices. Requires 2 inputs (the two lists/sets of indices).
32. **filter_event_type**: Returns the events from the list of given events, that are of type *type*. Requires 2 inputs (a list of events, and an event type).
33. **filter_order**: Extracts the i-th element of a series of events, based on the order attribute. e.g. [events, fifth] will return the fifth event in the list of events (since events are naturally sorted). Requires 2 arguments (the list of events, and the order in *text*).
34. **df_order**: Returns the i-th item in a data table. Requires 2 inputs (the data table, and the order in text (e.g. "second")).
35. **filter_particip**: Returns the participating items in an event. Requires 1 argument (the event).

36. **describe_ball**: Returns a description of the *ball* participating in the event. Requires 1 argument (the event).
37. **describe_side**: Returns a description of the *side* participating in the event. Requires 1 argument (the event).
38. **describe_particip**: Returns a description of the *balls* participating in the event. Requires 1 argument (the event).
39. **query_object**: Returns the items of type *item* that participate in the events *events*. Requires 2 arguments (the events, and the item type).

The above commands (programs) are the operators that will be used. The software responsible for receiving the series of commands and data tables, and outputting an answer in natural language, is the *Symbolic Executor*.

In Appendix A.1, one can see that most programs are templated, and can use arguments such as color, shape and size, to be tailored to each question, much like the questions themselves.

In total, in the programs vocabulary there are around 80 different words (including programs, and attributes, such as "first", "red", "left", "big", "square"), while the Natural Language questions contain approximately 129 words.

After taking a look at Section 3.8 and Appendix A.1, one will see that perhaps there are multiple ways to formulate a question in the form of programs. This is true. However, for simplicity purposes we will use a single sequence of programs per question (and its rephrasals) even though in cases there might be more optimal sequences of commands.

In our case, an attention encoder-decoder GRU-RNN was used in order to achieve the translation from the Natural Language questions to their Symbolic sequences of commands (Ch. 6.2).

5.7 Macroscopic view of model

A more detailed macroscopic view of the full ensemble model can be seen in Figure 5.6:

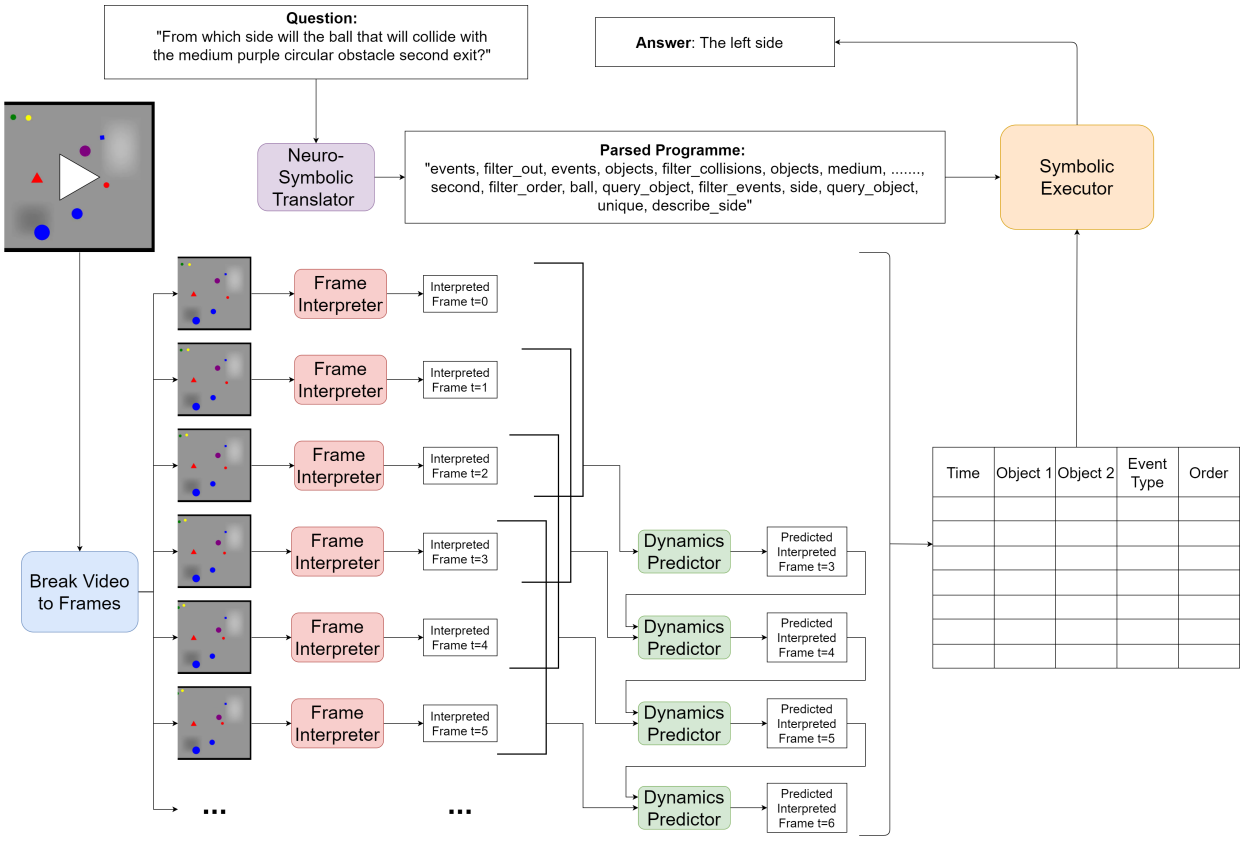

Figure 5.6: End-to-end process for answering questions. Each different element is shown with a different color. *Blue* is the video-to-frame breaker, *red* is the frame interpreter (MaskR-CNN, color classifier and all other related code), *green* is the Dynamics Predictor (PropNet), *purple* is the sequence-to-sequence model (Attention-GRU), and *orange* is the Symbolic executor.

6 Results for each module

In the previous chapters, a macroscopic view of the model was given, along with an analysis of the theory behind each module as well as the setup that was used in these experiments. In this chapter the results of each module will be presented separately from the end-to-end approach (which will in turn be shown in Chapter 7). These sub-modules are the Frame Interpreter (MaskRCNN), the Neuro-Symbolic translator, the Dynamics Predictor (PropNet), the Mass Estimator, the Friction Estimator and finally the Symbolic Executor.

As mentioned in previous chapters, the ensemble network cannot be trained end-to-end. However being able to train and monitor each module independently provides us with the large advantage of monitoring and calibrating its ability independently from all the other modules.

6.1 Frame Interpreter

As previously mentioned, MaskRCNN [21] aims to provide pixel level segmentation on each given image, along with classification for given objects. The dataset consisted of 1080 items, with a 9:2 train-validation split (900 train samples, 180 validation samples), which were generated from the original dataset generator. Training was done using Colabs [17] GPUs, and training time was approximately 30-35 mins. Convergence was achieved after about 4-5 epochs (Figure 6.1).

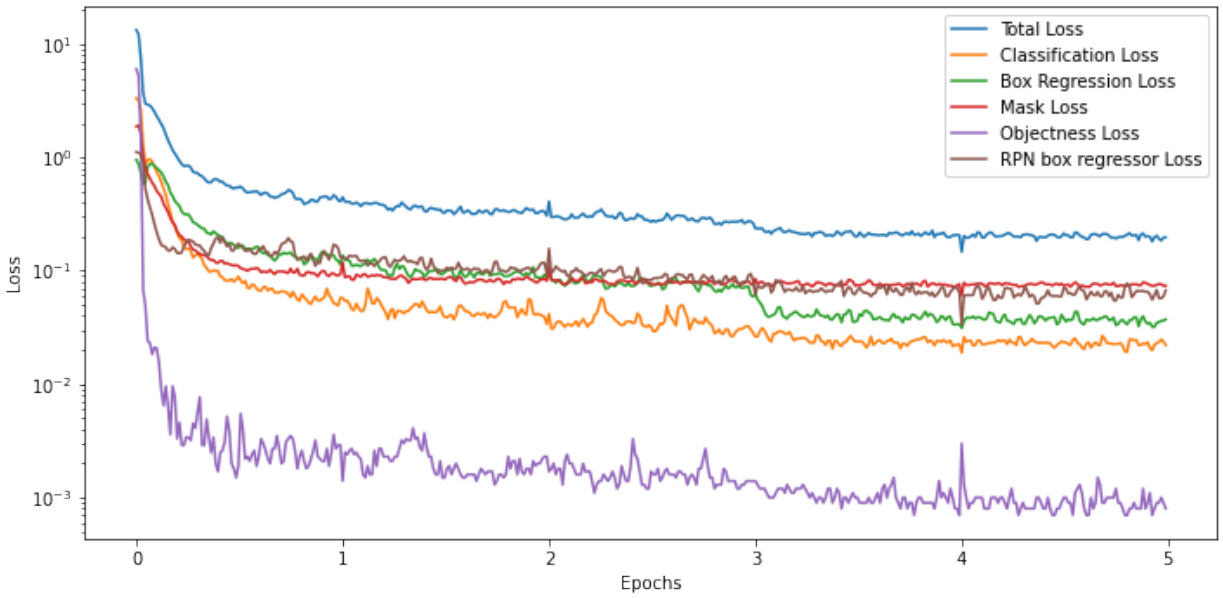

Figure 6.1: Losses on training set during MaskRCNN training.

We evaluated the results of the MaskRCNN by accuracy of the classifier. Since the order with which the different items per training sample are given during training differs from the order during inference, matching between the items was performed with an IOU (Intersection Over Union) metric on the initial and predicted binary masks. The pair with

the highest IOU score was evaluated, as it was considered to be the same item in the ground truth and in the predicted labels. A sample of the predicted masks from the output of the MaskRCNN can be seen in Figure 6.2.

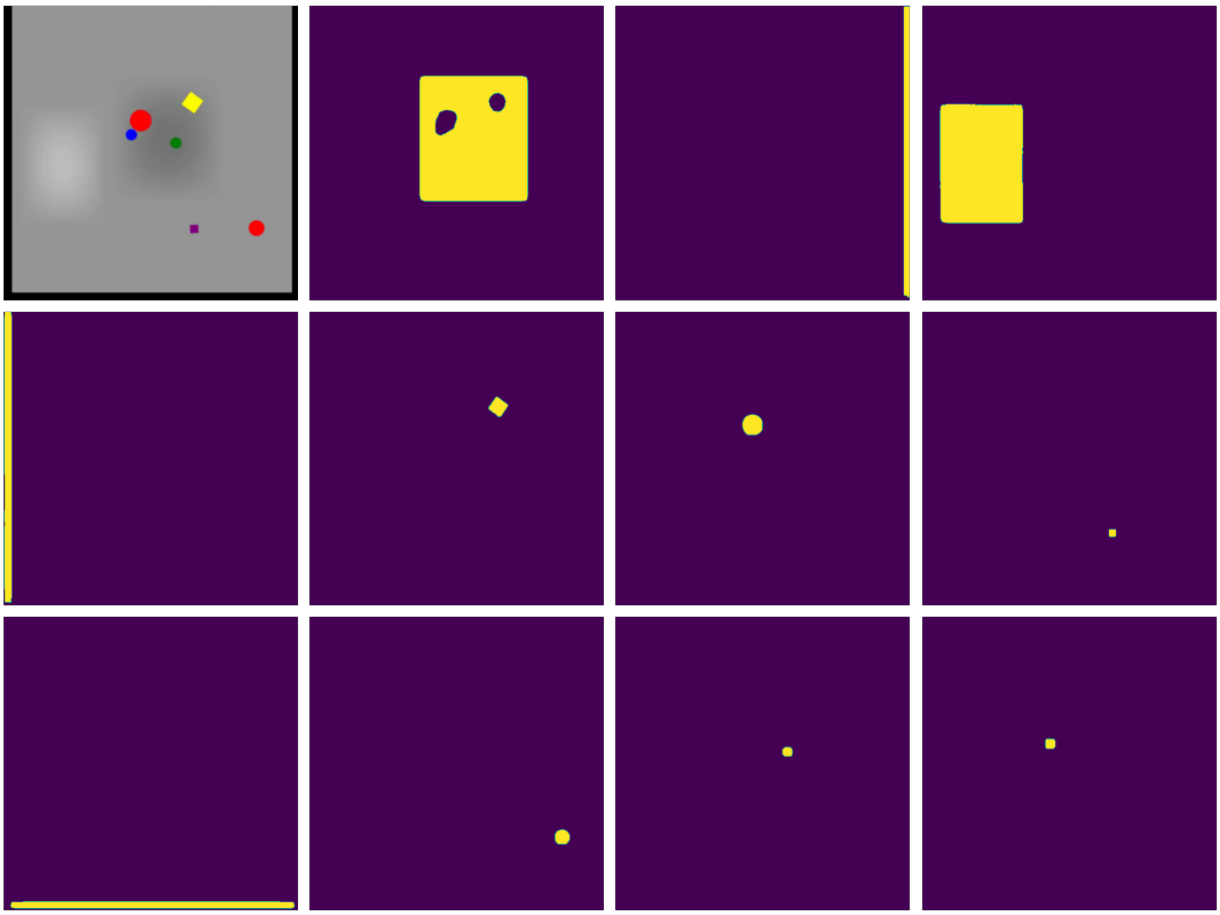

Figure 6.2: Frame from simulation, and generated masks from Mask R-CNN. Notice how one of the two gravity fields contains "gaps" where some of the objects are located.

The classification accuracy was above 99% and the classifier's confusion matrix can be see in Figure 6.3.

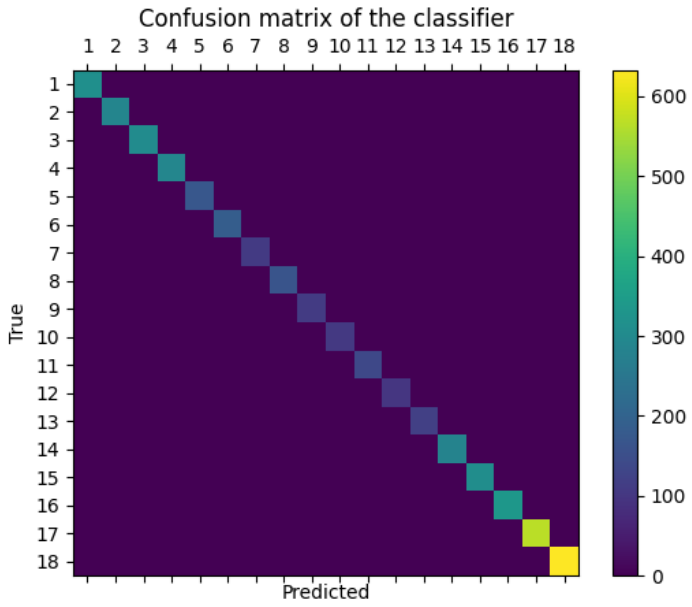

Figure 6.3: Confusion Matrix of MaskRCNN classification. The 18 categories can be seen in Table 3.4

A sample image and an inferred masked image can be seen in Figures 6.4 and 6.5 respectively.

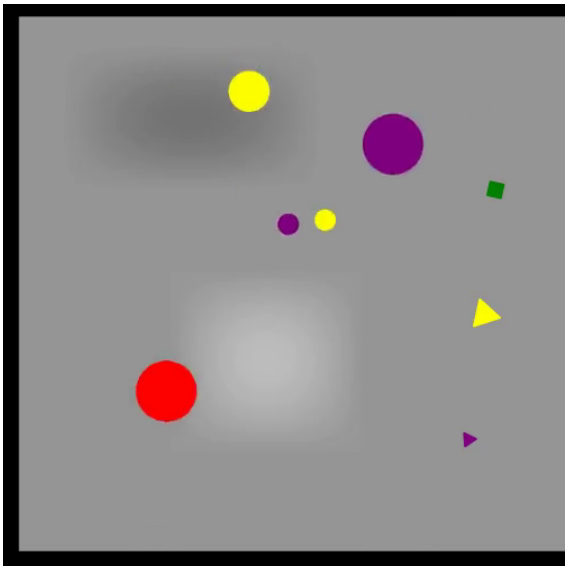

Figure 6.4: Frame from simulation

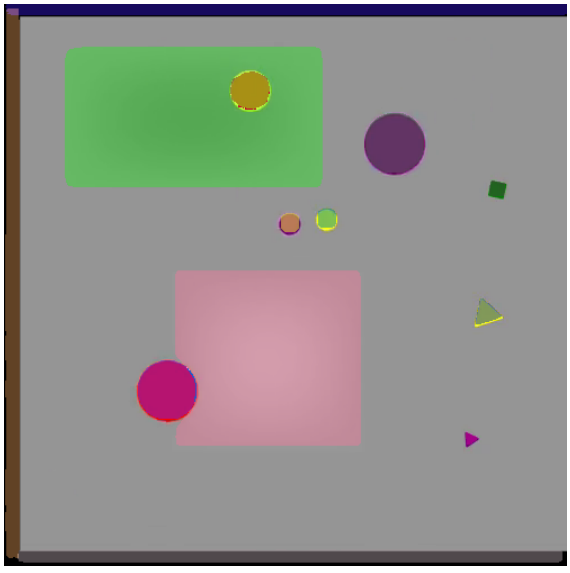

Figure 6.5: Frame from MaskRCNN

6.2 Neuro-Symbolic Translator

Along with the dataset, a huge set of questions has been generated (Chapter 3.8). However, in order to follow a symbolic approach, these questions need to be converted from

Natural Language (English), to a series of Symbolic "programs", that can be used to extract information from the output of the Propagation Network. In practice, we translate English into a new, *Symbolic* language (a Domain Specific Language). Along with the templated questions generated with the dataset, a templated series of programs has also been generated (for more information, see Appendix A.1). This will be used to train and evaluate a network that will perform the translation.

A Neuro-Symbolic parser is used in order to translate the questions from Natural Language (English) to their symbolic counterparts (programms). This is done using a sequence-to-sequence model, as explained in Chapter. 4.2. That is a GRU model, with an attention mechanism on its decoder's input. For this to happen, a dataset with pairs of Natural Language questions and their respective Symbolic programms needs to be created. This dataset, as mentioned in Chapter. 3 is generated alongside the simulations [9].

The number of questions is large, since they are procedurally generated. Duplicates are guaranteed to exist, therefore are discarded in order to correctly evaluate training and avoid overfitting. In the end, a total of 30k+ different question-program pairs were left. A variable number of question pairs is used in order to test the parsers ability to generate correct NS sequences from limited training data. In all cases, an 80-20 split is used for training and validation set.

A maximum number of 100000 epochs were used, however the model will almost always converge much earlier. The data is split with a 80:20 ratio, therefore for the 2000 samples case, 1600 were used for training, while 400 for validation. The model is evaluated every 50 epochs [10], and the accuracies of the train and validation set are shown in Figure 6.1. Early stopping is implemented with the accuracies that are calculated every 50 epochs. If more than 20 evaluations pass ($20 * 50 = 1000$ epochs) without improvement of the validation accuracy, the model is considered trained and the best validation epoch is saved for inference (a.k.a. early stopping with a patience of 20 evaluations). In Table 6.1 the best accuracies of the train/validation set are displayed, along with the number of epochs required to achieve them, as well as the training time required (time was calculated when computing on a GTX-1650).

[9] It is not actually a different dataset, as each generated question comes paired with its respective program, which can be directly used to train a translation model.

[10] Training would be faster if the evaluation was sparser, but for development purposes, a more dense evaluation is used.

Training Samples	Train Acc.	Validation Acc.	Epochs	Time (mins)*
100	100.00 %	5.00 %	3150	3
200	79.375 %	15.0 %	3600	6
500	89.25 %	60.0 %	7500	23
1000	96.5 %	77.0 %	11950	61
2000	86.62%	83.75 %	13900	129
5000	99.75 %	97.39 %	83000	138
7500	99.26 %	98.40 %	77000	166
10000	99.08 %	98.10 %	92000	248
20000	98.72 %	997.94 %	100000	500

Table 6.1: Training samples, train and validation accuracy, epochs and time required (on a GTX-1650) for training the sequence-to-sequence English to Symbolic translator. *Training with 5000, 7500, 10000 and 20000 samples was evaluated every 1000 epochs in order to save training time.

The convergence graphs (with regards to sentence accuracy, as defined in Eq. 5.1) of all the training examples, are displayed in Figure 6.6.

What is evident from Figure 6.6 is that the accuracy on the validation set increases with the number of training samples, therefore the network is able to generalize better (as was expected). Any experiment with a set size of 5k+, can be considered adequate to perform inference with.

Even through no duplicate questions exist in our dataset (as they have been filtered), in certain questions that do not receive arguments (such as color or size) and do not have rephrases, the network might fail to generate their programs successfully. This is due to the fact that after seeing many samples, the encoder understands that "red" and "blue" belong to the same group (color) that goes into specific positions, while "small" and "big" belong to another group (size) that goes into different positions. This way, it understands language rules and can generalize well on similar things that it has seen. In the case of a question such as "What is the level of the friction?", since there are no similar questions with similar programs, it might fail to answer it. Such cases could be solved by introducing numerous rephrasals, or using pre-trained embeddings, but in order to generalize better the total accuracy would most likely be reduced.

In our experiments, the translator worked incredibly well, and the end-to-end accuracies are incredibly close to the ones calculated by using the ground truth sequences of the programs. However, in more expanded cases, the abovementioned potential issue should be taken into account when generating the questions and training the network.

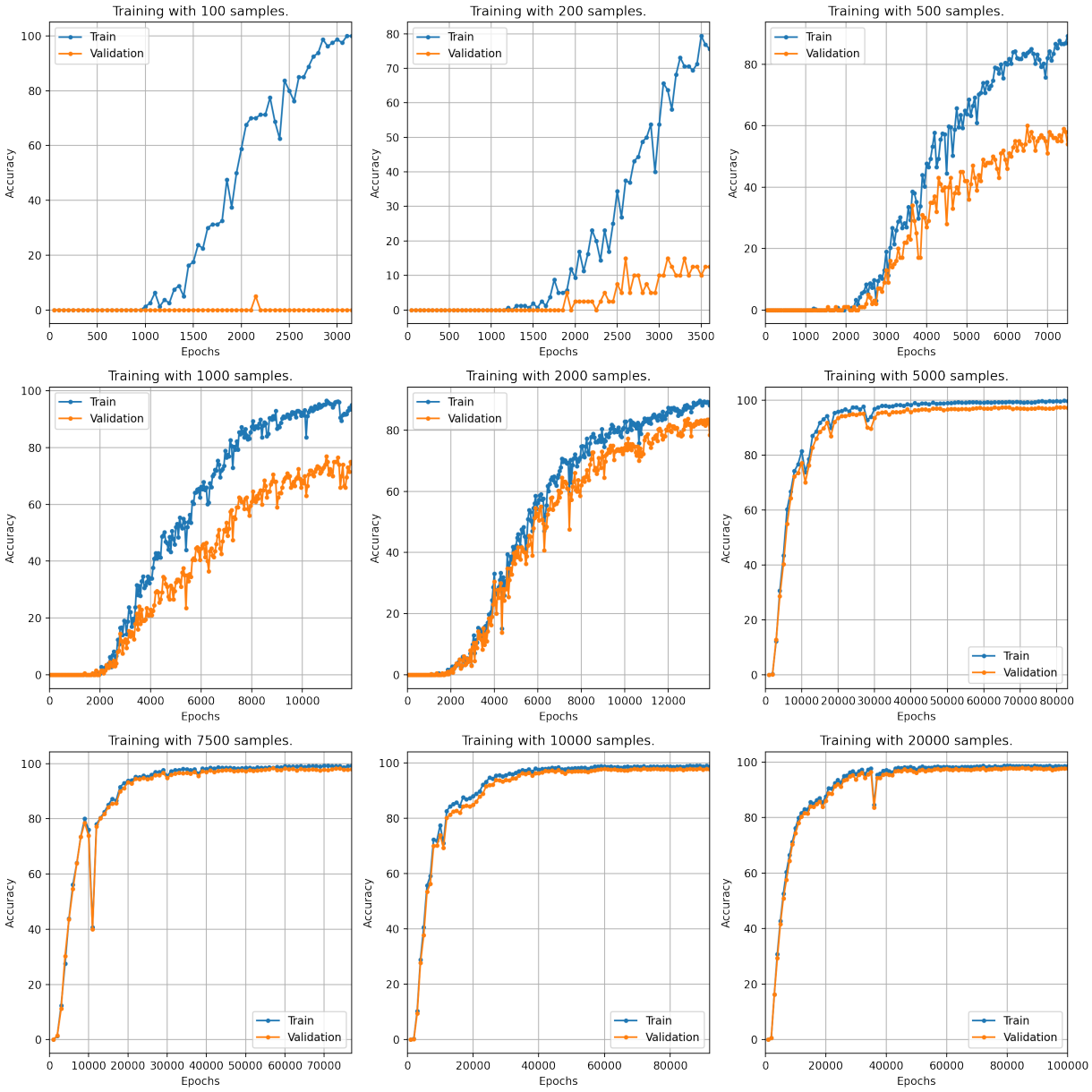

Figure 6.6: Accuracies of training and validation set for the sequence-to-sequence translator. Results are displayed for total size (train+validation) of 100, 200, 500, 1000, 2000, 5000, 7500, 10000, 20000.

6.3 Dynamics Predictor

The Dynamics Predictor aims to understand the dynamic events that take place in a scene by receiving a sequence of interpreted frames as input. These frame interpretations are generated by the frame interpreter (see Ch. 4.1). The output of the evaluation step of the Dynamics Predictor generates a **.csv* file as the logfile of the events. This file is similar to the one being produced during the initial simulation and can be compared to it through various ways. There are two different methods to evaluate PropNet's performance: as a separate entity, or as a part of the total VQA architecture (end-to-end approach).

The first way is the preferred, since PropNet is used for event tracking and the precision (accuracy cannot strictly be defined) of event detection needs to be evaluated. As mentioned in Chapter 4.3, we can compare the ground truth events with the predicted ones and consider an event as found if it is found successfully within a certain time-window of the original (Ch. 5.3.3). The end-to-end evaluation could be used to evaluate PropNet's performance as well, however it would not give insight to PropNet's performance as s single module, therefore negating the perks of the Neuro-Symbolic approach. It will be used in order to evaluate the total end-to-end architecture.

By training PropNet for 100 epochs, with 120 samples (80 training, 20 validation, 20 test) of difficulty of level 8, we received the following per-event precisions and recalls:

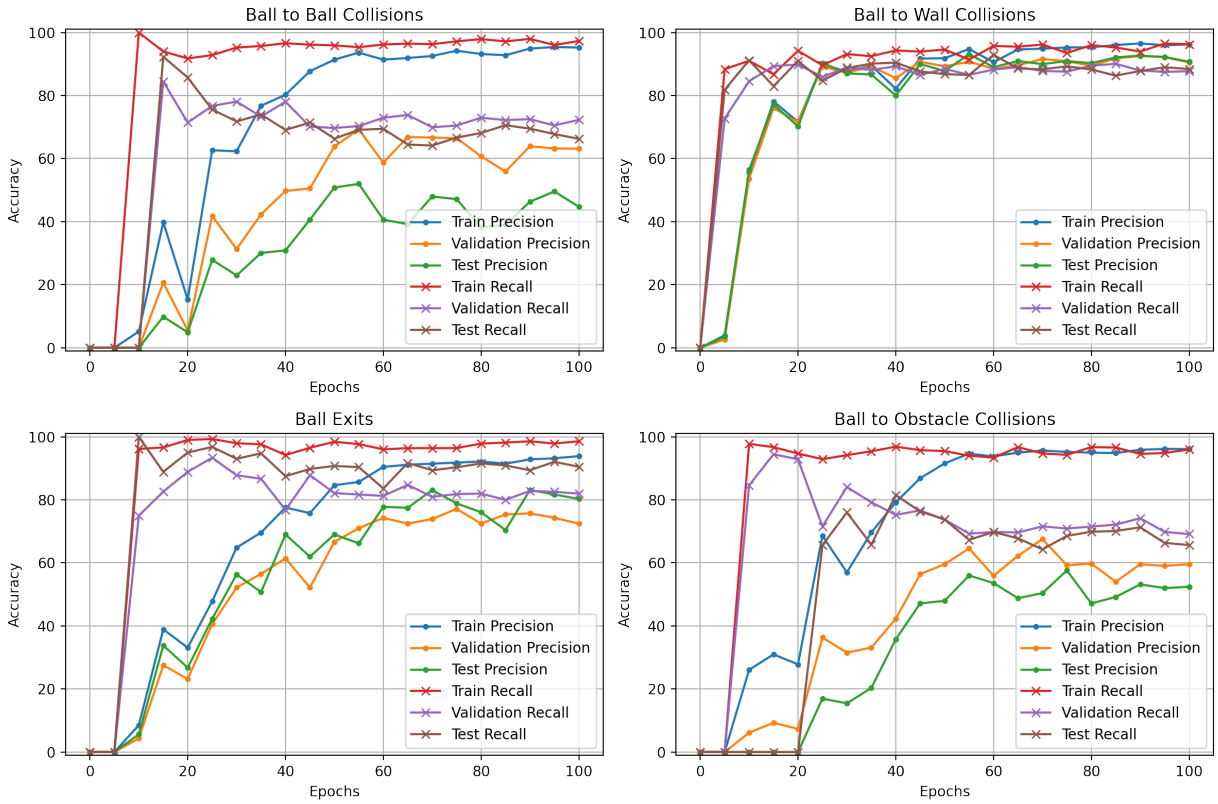

Figure 6.7: Per-event accuracies with PropNet trained on 120 samples (80 of which are the training set). Execution time was 30.5 hours on a GTX 1650 with batch size 1.

We can easily see that while *ball-to-wall* collisions, and *ball exits* generalize well and with relatively few epochs -even with the limited 80 training samples-, cases such as *ball-to-ball* and *ball-to-obstacle* do not. This is probably due to the fact that the latter cases involve different, non-static items, and the network requires more samples in order to be able to generalize from that behavior. A ball that exits, simply stops being inside the frame. A ball that collides with a wall rapidly changes velocity at some side (each side has a specific location within the domain). However, balls are not static objects, and they move at every timestep. Obstacles, even though static in all the frames of a single simulation, are vastly different between different simulations, thus the necessity for larger datasets to train arises. However, the precision -considering the limited number of 80 training samples- supports the concept of the Neuro Symbolic approach; *less training data is required* in order to achieve better results.

This training took place with 80 training samples, 20 for validation, and 20 for test. The next step is to train the Network with a larger dataset, to check if the lack of precision is data-driven. By further training the network (keeping as a checkpoint the 70-th epoch, as it gives the highest precision (Figure 6.7)) with double the number of training samples (160:40:40 split) for 50 extra epochs, we receive the following precisions and recalls (Fig. 6.8):

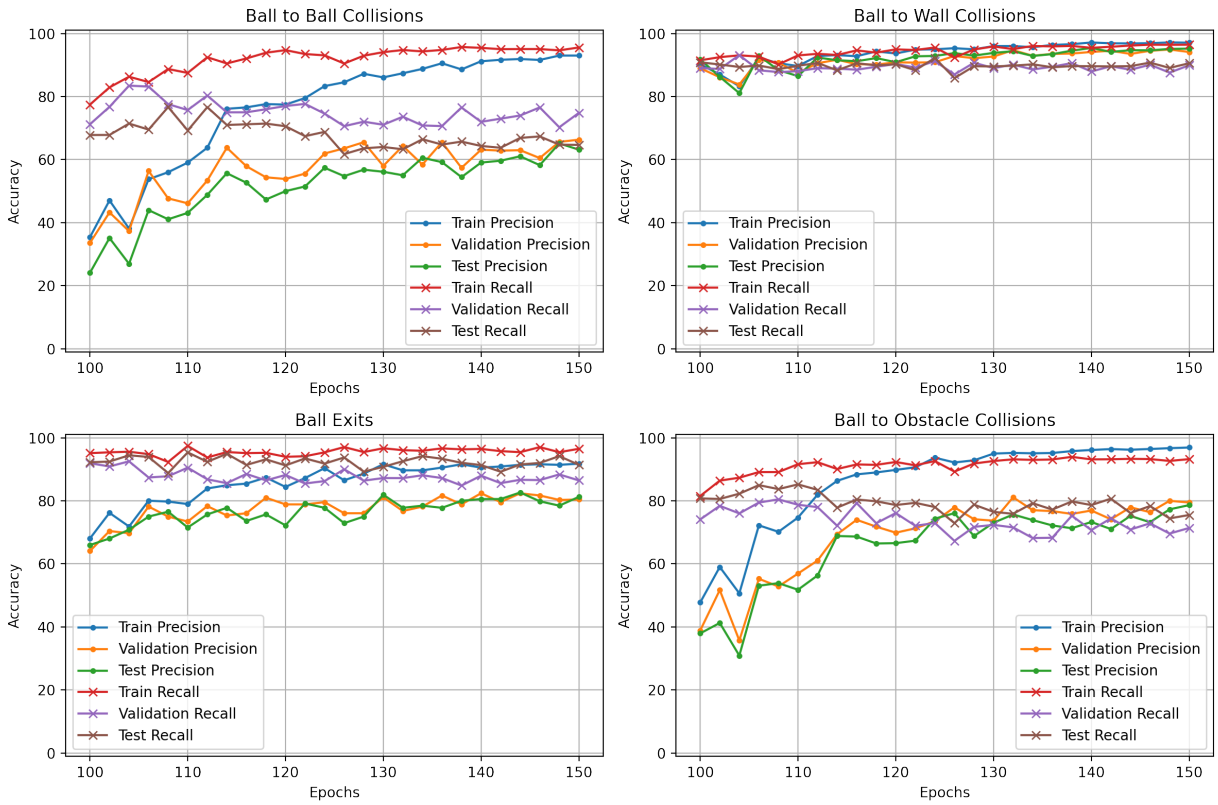

Figure 6.8: Per-event precisions with PropNet trained on 240 samples (160 of which are training set). Execution time was 30 hours on a GTX 1650 with batch size 1. The network was initialized based on the weights calculated from the previous run, with 100 epochs (See Fig. 6.7)

Note that since the dataset is now larger, the starting precision -even on the train set- will be lower, since half the new training set is now new, and never before trained-upon data (it is seen as test data).

We can see that the precisions further improve upon doubling the data, and not by an insignificant amount. The recall remains high, meaning events can be detected, however, the lower precision means that some events are detected multiple times, which hinders certain VQA tasks which heavily rely on the correct number of event detections. The per-event Precision, Recall, False Discovery Rate and Miss Rate of epoch 150 can be seen in Table 6.2. This checkpoint will be used for the end-to-end inference in Chapter 7.

Further training with even more data would most likely further increase the precisions and recall of the located events, however, taking into account the training time required, something such as that will be left for future work.

Set	Event Type	Precision	Recall	FDR	MR	TP	FP	FN
Train	Ball-to-ball	93.00%	95.56%	7.00%	4.44%	904	68	42
	Ball-to-wall	97.08%	96.44%	2.92%	3.56%	2195	66	81
	Ball Exits	91.77%	96.50%	8.23%	3.50%	524	47	19
	Ball-to-obstacle	96.91%	93.22%	3.09%	6.78%	1444	46	105
	All Events	95.71%	95.35%	4.29%	4.65%	5067	227	247
Validation	Ball-to-ball	66.32%	74.81%	33.68%	25.19%	193	98	65
	Ball-to-wall	94.13%	89.91%	5.87%	10.09%	481	30	54
	Ball Exits	80.42%	86.47%	19.58%	13.53%	115	28	18
	Ball-to-obstacle	79.49%	71.38%	20.51%	28.62%	217	56	87
	All Events	82.59%	81.79%	17.41%	18.21%	1006	212	224
Test	Ball-to-ball	63.16%	64.67%	36.84%	35.33%	108	63	59
	Ball-to-wall	95.21%	90.55%	4.79%	9.45%	556	28	58
	Ball Exits	81.25%	91.41%	18.75%	8.59%	117	27	11
	Ball-to-obstacle	78.64%	75.50%	21.36%	24.50%	265	72	86
	All Events	84.63%	83.02%	15.37%	16.98%	1046	190	214

Table 6.2: Precision, Recall, False Discovery Rate, and Miss Rate of PropNet's training, at epoch 150, on all three sets (train, validation and test). True Positives, False Positives, and False Negatives are also displayed.

6.4 Mass Estimation

This, and the following sub-chapters will focus mainly on the training and performance of the models that are aimed at understanding physical quantities that are present in the simulations. Unlike all other questions, these will *not* be included in any end-to-end approach, since they will be more thoroughly explored here. These types of questions involve three different aspects. The first is the mass estimation of the balls, based on their perceived interactions (Ch. 4.4), the second one is the friction type and magnitude estimation (Ch. 4.5) and the third is the velocity order estimation (due to its simplicity, it has not been explored in theory, and will briefly be explained here, along with its results).

In order to train a mass estimator (which in practice is an LSTM paired with a fully connected network that simultaneously classifies the 5 balls), a rather large dataset will need to be generated. Even though the previous Neuro-Symbolic approaches (event detection through Prop-Net) requires a rather small number of training samples to achieve acceptable results, this is not the case here. Initial attempts with 20-100k initial samples (reduced to about 10% of that after their filtering as described in 5.4) barely passed the baseline [11]. Since the composition of the problem was sound, more data was needed. For this purpose, 500k samples from each of the categories of levels 1, 3, 4, and 8 were generated [12], leading to a total of 2M samples, of which around 90k were valid. In order to test the scaling capabilities (and in practice validate that this indeed is a data-driven issue), different runs were made,

[11] The baseline with three mass categories is 33.3%

[12] The generation was computationally intensive, and required around 2 days to perform on three different multi core processor machines

with different amounts of data. Since these samples were generated from different difficulties (therefore different distributions) their classes were not balanced. In total, there were:

- Class 0 ("none"): 58606 samples
- Class 1 ("low mass"): 132661
- Class 2 ("medium mass"): 132031
- Class 3 ("big mass"): 132777
- Class 4 ("infinite mass"[13]): 0 samples

Assuming that Class 0 can always be classified correctly (since it is made up of zeros), this leaves a baseline accuracy of:

$$acc_{BL} = \frac{58606 + 132777}{58606 + 132661 + 132031 + 132777} = 0.4196 = 41.96\%$$

In all cases, a validation set of the *same* 1000 samples was used, and the training set consisted of an increasing set of samples, namely at 1k, 5k, 10k, 20k, 30k, 60k and 90k samples. The accuracy of the classifier as a function of the number of training samples can be seen in Figure 6.9.

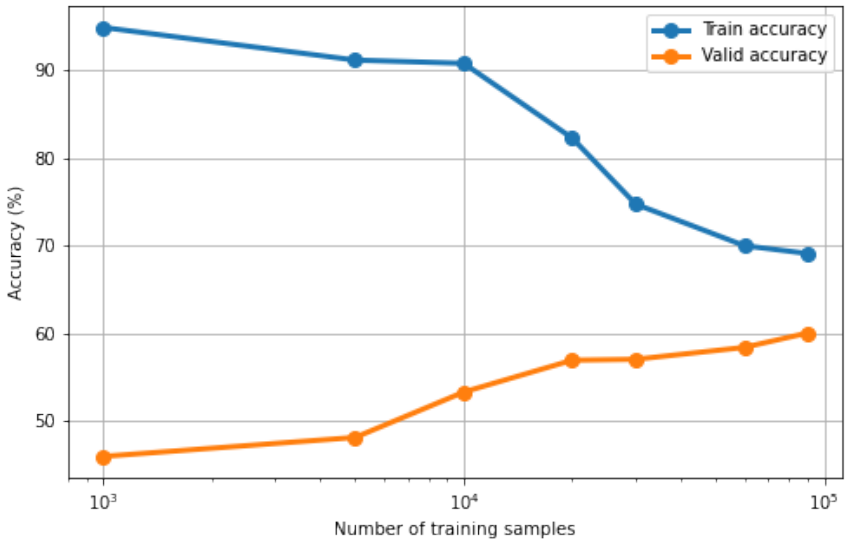

Figure 6.9: Accuracy of train and validation set for the mass estimator. The runs comprised of 1k, 5k, 10k, 20k, 30k, 60k and 90k training samples, and the same validation set, that consists of 1000 samples.

From Figure 6.9 we can see that for small datasets, the model overfits to the training data and the accuracy is barely above the baseline ($\sim$ 46%). As the dataset increases in size, the model learns better and better, finally reaching an accuracy of $60 + \%$.

[13]This is not used in these runs, but rather left as a placeholder for future development

6.4.1 Mass Estimation Generalization

As mentioned, the classifier was trained on a joined dataset, comprised of samples from levels 1, 3, 4 and 8. In order to test its classification capabilities, we will create 2000 samples from each difficulty (different than the trained ones), and attempt to find the classification accuracy. This will both work as a test set for simulations of levels 1,3,4 and 8 (rather than a validation), as well as a way to test the model's generalization capabilities on unseen data. The test data, similarly to the training data, has been filtered as mentioned in Chapter 5.4. Based on the distributions of the test sets, a baseline can also be calculated, which is displayed as well.

Level	Accuracy	Baseline	Valid Samples	Cat. 0	Cat. 1	Cat. 2	Cat. 3
1	84.44 %	77.04 %	27	81	23	15	16
2	75.29 %	65.88 %	17	34	17	22	12
3	70.72 %	47.21 %	97	97	132	130	126
4	68.89 %	64.44 %	18	36	16	22	16
5	71.13 %	48.04 %	97	97	130	136	122
6	68.70 %	63.47 %	23	46	26	27	16
7	51.98 %	47.44 %	172	172	220	232	236
8	51.07 %	34.02 %	224	0	362	381	377

Table 6.3: Accuracies on test samples. All test sample levels have been generated from an initial 2000 samples per level, but through the filtering explained in Chapter 5.4 only the number of samples displayed on the column "Valid Samples" end up being used. Category 0 is "None", category 1 is "low" mass, category 2 is "medium" mass, and category 3 is "big" mass. While category 4 exists ("infinite mass", since it is never used as training, it is never predicted in the classifier, therefore omitted.

As seen from Table 6.3, while all accuracies are higher than the baselines, the high fluctuation of the differences is due to the limited number of samples being used for testing. However, it is evident that the network does indeed manage to learn to infer the masses of the participating objects.

6.4.2 Mass Estimation VQA Results

A similar testing approach will take place now, but instead of evaluating the accuracy of the masses, we will evaluate the accuracy of the VQA questions regarding the mass. However, in this case since the results are end-to-end, and the video generation is costly, no filtering will take place[14]. The trained model will be the same as used before, and evaluations will take place for all levels (1-8), for all questions regarding mass. Note that these results are completely end-to-end results, meaning that the initial videos, go through the whole structure shown in Figure 5.6. This means that the inputs that are given to the mass predictor are no longer the ground truths as directly provided by the simulator, but as exports from the MaskRCNN and PropNet. During the dataset generation, *no samples*

[14]In future work, complete comparison with larger VQA tests sets can take place.

were filtered, meaning that many (or *most* to be precise) examples might contain balls whose masses were impossible to predict. The results of the above end-to-end runs can be seen in Tables 6.10 and 6.11.

	Level 1		**Level 2**		**Level 3**		**Level 4**	
Question ID	Accuracy	Total	Accuracy	Total	Accuracy	Total	Accuracy	Total
9.0	34.00	100	20.00	100	20.00	100	28.00	100
9.1	34.00	100	22.00	100	31.31	99	30.30	99
9.2	54.00	100	46.39	97	58.00	100	47.47	99
9.3	32.00	100	25.00	100	38.00	100	29.00	100
9.4	32.00	100	28.00	100	37.00	100	28.00	100

Table 6.4: VQA accuracy results on mass estimation questions for levels 1 to 4

	Level 5		**Level 6**		**Level 7**		**Level 8**	
Question ID	Accuracy	Total	Accuracy	Total	Accuracy	Total	Accuracy	Total
9.0	14.00	100	30.00	100	22.00	100	48.00	100
9.1	49.49	99	30.30	99	46.00	100	62.00	100
9.2	54.00	100	43.88	98	58.00	100	64.29	98
9.3	36.00	100	26.00	100	32.00	100	37.00	100
9.4	46.00	100	22.00	100	30.00	100	36.36	99

Table 6.5: VQA accuracy results on mass estimation questions for levels 5 to 8

From the above results, the first thing that comes to mind is the incredible low accuracy of the end-to-end VQA for the questions on the lower levels [15]. As mentioned, the test set is *unfiltered*, meaning that most samples that will be given will be incomplete, and not actually inferrable. The mass estimator has been trained in ball interactions that create connected graphs, and since these samples do not satisfy these conditions, they are seen as completely unknown out-of-distribution data, and are incorrectly classified [16]. However, the results as the levels increase are substantially better than the baselines, as was expected, since more interactions take place, and the mass can be inferred. Additionally, even though we trained the network on simulations with fully connected graphs, it fares significantly better than expected, which means perhaps that even though the full classification of all the balls might not be able to take place, partial correct classification does occur for the balls that *do* actually interact. Additionally, the coordinates given to the mass estimator, are the ones that come from from the MaskRCNN masks, and are therefore subject to pixel quantification noise [17] hindering the learned behavior from the initial predictor.

In any case, since more in-depth analysis and experimenting would require tremendous amounts of data and computational time, the progress of such a feat is left for future work. However, what can be proposed is the use of larger datasets in the end-to-end manner, in

[15]The actual questions can be found in Appendix A.1

[16]The (not-so-simple) rules of determining the accuracy of each question can be seen in the source code.

[17]Noise that comes from the quantization of the image to the mask grid.

order to correctly filter and validate the answers in the test set as well, and attempt to infer whether the neural network can indeed learn partial interactions, when the graph is incomplete. Additionally, more in depth specialized training can take place where there is an extra category of an "uninferrable" mass. This would however, require careful handling of data balancing, as well as specialized pre-processing in order to correctly label each ball's mass.

6.5 Friction Estimation

After seeing the (costly) success of being able to estimate the mass of the participating objects simply by looking at the video and their interactions, a set of experiments regarding friction estimation will take place, as mentioned in Chapter 4.5. All training data consists of samples taken from difficulty of *level 3*, where the ball trajectories are straight lines, as gravity fields don't exist. A total of 20k samples were generated, with 1k being used for validation, while the other 19k for training The bidirectional LSTM used, has an input size of 25, where the coordinates of each ball (x,y), the velocities (u,v) and the magnitude of the velocity ($vel = \sqrt{u^2 + v^2}$) are given for each ball. There can exist a maximum of 5 balls, and when some are missing, their values are replaced by zeros. The LSTM has a hidden size of 16, while the classifier is a simple FC network, with an input size of 32 ($2 * 16$ since the network is bidirectional), and an output size of 7, one for every possible class. The optimizer used is the ADAM [27] optimizer with its default parameters. Early stopping was used with a patience of 30 epochs, and a maximum number of 1000 epochs, although most experiments had converged in far less than 200 epochs. The *number of samples - accuracy* graph, can be seen in Figure 6.10, while the confusion matrix is displayed in Figure 6.11.

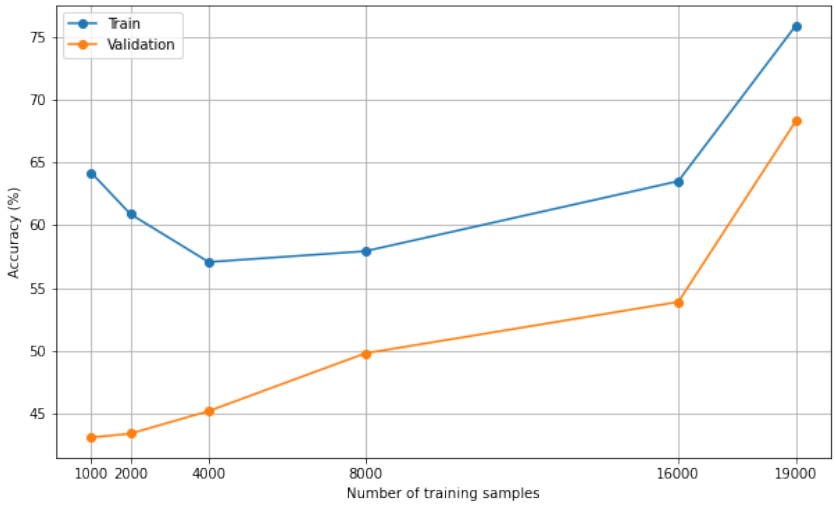

Figure 6.10: Accuracy of resistance estimator on the training and validation set for a varying number of samples. 1k, 2k, 4k, 8k, 16k and 19k samples were used for training, while the same 1k samples were used for validating.

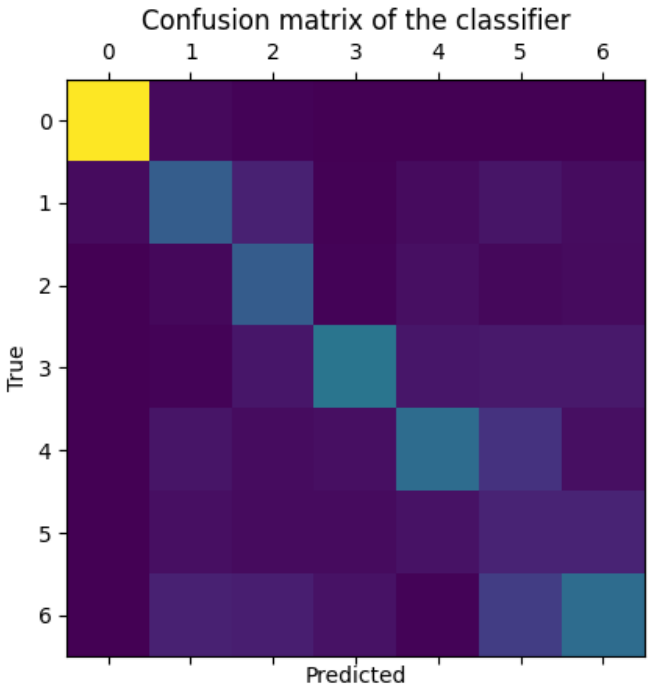

Figure 6.11: Confusion matrix of the best classifier (19k training samples), validated on the validation set. Category 0 is without any type of friction, while (1,2,3) and (4,5,6) are various degrees of dry friction and viscous drag respectively.

The *per-type* and *per-level* accuracies can be seen in Tables 6.6 and 6.7 respectively:

Type	**Accuracy (%)**
none	87.98
dry friction	70.00
drag	66.15

Table 6.6: Accuracy of per type classification

Level	**Accuracy (%)**
0	87.98
1	63.44
2	63.79
3	67.11

Table 6.7: Accuracy of per level classification

While for all other question types the results will be presented in Chapter 7, in this case (similarly with the mass above), the end-to-end results will be shown into more detail here, since many comments can be made upon observing them.

Namely, for different levels of simulation, the accuracies of the multi-class classification abilities of the classifier will be shown. Additionally, we will investigate how well it can generalize on different levels on the VQA task as well.

6.5.1 Friction Estimation Generalization

The point of this simulation, is to check if the classifier can correctly generalize on all the other levels. As has been mentioned, it has been trained in samples from *only* level 3 (4 balls, 3 walls). Even though a big decrease is accuracy is expected (since the number of balls differs), it would be interesting to check upon the actual accuracies on all the other levels. For this purpose, 2000 test samples from each level were generated (without the video renders, just the coordinates of the balls), in order to determine their accuracy. This generalization test takes place with data generated from the physics simulator, and not the results that have been generated by the MaskRCNN + PropNet architecture, as those whould be too costly to produce. The results for 200 samples from each level, can be seen in Table 6.8.

Level	**Accuracy (%)**	**Level**	**Accuracy (%)**
1	56.00	5	72.00
2	61.00	6	45.50
3	64.50	7	45.50
4	64.50	8	34.50

Table 6.8: Generalization of friction estimator on test data on levels 1-8, with 200 samples per level.

From the results displayed on Table 6.8 it's evident that the model can generalize quite well on unseen data. What is interesting, is the direct decline in accuracy displayed on levels 6,7 and 8. This is largely due to the fact that in these levels the gravity fields are introduced, which significantly modifies the trajectories of the balls, and since friction estimation relies on the trajectories, this hinders the generalization capabilities since the network has not been trained on cases with gravity fields.

6.5.2 Friction Estimation VQA Results

In this subchapter, we will evaluate the end-to-end results of the friction estimator. The purpose of this experiment is to see how much the noise that is inserted by the MaskRCNN can hinder the prediction, as it slightly changes the coordinates, upon which the predictor aims to classify the friction type and level. For this experiment a total of 100 samples with friction were generated from level 3, and evaluated upon.

Question ID	Accuracy	Total
9.9	83.24 %	164
9.10	60.38 %	143

Table 6.9: VQA accuracy results on friction estimation questions for level 3 (100 samples used)

We can see that friction estimation is inhibited by the quantification noise inserted during the MaskRCNN phase, as well as the lower sampling rate used by the Dynamics Predictor. This is evident especially during the inference of the *level* of the friction and reduces the accuracy of the end-to-end results of the VQA problem.

6.6 Velocity Order Estimation

This task mainly involves answering questions in regards to the ordering of the balls' velocities, such as "Which ball will be the third fastest in the beginning?" [18].

This task is significantly easier than the two previous tasks, since no specific neural network needs to be trained in order to be able to answer it. The interpreted frames include positional information about each ball and the velocities are nothing more than the numerical derivative of the positions, similar to how they have been used as data in the previous two cases. In all intermediate timesteps, a 2nd-order scheme is used to calculate the x and y components of the velocity:

$$\begin{aligned} u &= \frac{x_{i+1} - x_{i-1}}{2dt} \\ v &= \frac{y_{i+1} - y_{i-1}}{2dt} \end{aligned} \tag{6.1}$$

While at the first timestep, and the last timestep, we use 1st-order forward (Eq. 6.2) and backward (Eq. 6.3) schemes respectively:

$$\begin{aligned} u &= \frac{x_{i+1} - x_i}{dt} \\ v &= \frac{y_{i+1} - y_i}{dt} \end{aligned} \tag{6.2}$$

$$\begin{aligned} u &= \frac{x_i - x_{i-1}}{dt} \\ v &= \frac{y_i - y_{i-1}}{dt} \end{aligned} \tag{6.3}$$

In all cases, the value of the velocity itself is defined as:

$$vel = \sqrt{u^2 + v^2} \tag{6.4}$$

Since no network is involved in this case and a purely symbolic approach is followed, the results are going to be directly evaluated on the VQA task, as evaluating them on the outputs of the dataset generator (physics engine) would result in a 100% accuracy.

[18] All the questions and their rephrasals can be found in Appendix A.1

6.6.1 Velocity order estimation VQA Results

In this test case, the exact same dataset -as the one used to evaluate the mass prediction- was used. In total, there are 4 different questions that query upon the velocities of the objects.

	Level 1		**Level 2**		**Level 3**		**Level 4**	
Question ID	Accuracy	Total	Accuracy	Total	Accuracy	Total	Accuracy	Total
9.5	87.40	127	85.86	99	92.13	89	87.25	102
9.6	74.44	133	66.33	98	67.42	89	66.67	99
9.7	87.37	95	75.28	89	83.56	73	77.27	88
9.8	88.66	97	87.36	87	85.94	64	92.13	89

Table 6.10: VQA accuracy results on velocity estimation questions for levels 1 to 4

	Level 5		**Level 6**		**Level 7**		**Level 8**	
Question ID	Accuracy	Total	Accuracy	Total	Accuracy	Total	Accuracy	Total
9.5	90.22	92	88.66	97	82.61	92	81.69	71
9.6	39.77	88	75.76	99	51.22	82	55.88	68
9.7	86.49	74	62.50	88	36.00	75	25.00	68
9.8	86.36	66	73.26	86	61.64	73	50.85	59

Table 6.11: VQA accuracy results on velocity estimation questions for levels 5 to 8

Even through this would be considered a relatively easy task, in some cases, we see that the accuracies are not as good as expected. Lets start off from the lower levels. The velocities in many cases are stable, since far less collisions occur. This however, also means that balls exit the domain quicker, and might not be detected in time, since the inputs of PropNet (from which the velocity tables are build) are 1) sampled every 5 frames ($frame\_offset = 5$) and 2) discard the first three frames in order to successfully operate, as explained in Chapter 5.3. At higher levels, even though the initial velocities remain the same, they have far more chances to interact, and therefore change. PropNet's sampling rate does not allow many of those events to be captured, leading to a reduced accuracy. In the intermediate levels, the accuracies are in fact rather high. Finally, at the higher levels where gravity fields are introduced, the trajectories become increasingly erratic, and the reduced sampling rate, as well as the noise inserted from the estimated position from the MaskRCNN, reduce the accuracy furthermore.

7 VQA results

In this chapter, the end-to-end results in all the categories (except the ones that were analyzed in Chapter 6) will be shown. Initial results will be shown for the level that they were trained upon (level 8), while generalization on other levels will be displayed as well. This chapter will mostly consist of tables with accuracies per category per question, as well as various comments regarding the results.

7.1 End-to-end results for level 8

The end results will be displayed as the percentage of the questions that have been answered correctly. In this section, results from the trained upon *level 8* will be displayed (excluding questions regarding Physics Understanding that have been showed in their own sections). No comments will be made in this chapter, but will be cumulatively presented in Chapter 7.2 in order to compare accuracies across various levels.

Compare Integer				
Question ID	Correct	Incorrect	Total	Accuracy
0	122	0	122	100.00 %
1	82	0	82	100.00 %
2	82	0	82	100.00 %
3	118	0	118	100.00 %
4	82	0	82	100.00 %
5	82	0	82	100.00 %

Table 7.1: Compare Integer accuracy results.

Same Relate				
Question ID	Correct	Incorrect	Total	Accuracy
0	214	0	214	100.00 %
1	238	0	238	100.00 %
2	141	0	141	100.00 %
3	239	0	239	100.00 %
4	244	0	244	100.00 %
5	161	0	161	100.00 %
6	134	0	134	100.00 %
7	116	0	116	100.00 %
8	27	0	27	100.00 %
9	149	0	149	100.00 %
10	136	0	136	100.00 %
11	110	0	110	100.00 %

Table 7.2: Same Relate accuracy results.

Comparison				
Question ID	Correct	Incorrect	Total	Accuracy
0	186	4	190	97.89 %
1	124	0	124	100.00 %
2	4	0	4	100.00 %

Table 7.3: Comparison accuracy results.

Single Or				
Question ID	Correct	Incorrect	Total	Accuracy
0	163	1	164	99.39 %
1	163	1	164	99.39 %
2	162	1	163	99.39 %

Table 7.4: Single Or accuracy results.

Zero Hop				
Question ID	Correct	Incorrect	Total	Accuracy
0	82	0	82	100.00 %
1	123	0	123	100.00 %
2	243	0	243	100.00 %
3	163	0	163	100.00 %
4	201	1	202	99.50 %

Table 7.5: Zero Hop accuracy results.

Domain Exits				
Question ID	Correct	Incorrect	Total	Accuracy
0	119	43	162	73.46 %
1	149	12	161	92.55 %
2	144	18	162	88.89 %
3	59	23	82	71.95 %
4	4	4	8	50.00 %
5	75	41	116	64.66 %

Table 7.6: Domain Exits accuracy results.

Collision Order				
Question ID	Correct	Incorrect	Total	Accuracy
0	20	62	82	24.39 %
1	46	36	82	56.10 %
2	33	49	82	40.24 %
3	35	47	82	42.68 %

Table 7.7: Collision Order accuracy results.

Quantitative Answers				
Question ID	Correct	Incorrect	Total	Accuracy
0	34	41	75	45.33 %
1	28	13	41	68.29 %
2	36	5	41	87.80 %
3	11	30	41	26.83 %
4	3	119	122	2.46 %
5	21	101	122	17.21 %
6	21	101	122	17.21 %

Table 7.8: Quantitative Answers accuracy results.

Binary Answers				
Question ID	Correct	Incorrect	Total	Accuracy
0	137	10	147	93.20 %
1	36	1	37	97.30 %
2	39	2	41	95.12 %
3	36	5	41	87.80 %
4	20	14	34	58.82 %
5	22	12	34	64.71 %
6	21	11	32	65.62 %
7	21	10	31	67.74 %
8	11	9	20	55.00 %
9	14	6	20	70.00 %
10	38	1	39	97.44 %
11	36	3	39	92.31 %

Table 7.9: Binary Answers accuracy results.

7.2 End-to-end results for levels 1 to 8

As mentioned, all training has been performed on training samples from level 8 simulations, and generalized on all the other levels. It is evident that many times (especially in lower levels) certain questions do not exist, since the conditions to generate them -due to

Question Family	Question ID	Levels							
		1	2	3	4	5	6	7	8
Compare Integer	1.0	95.83	98.33	100.00	97.50	100.00	97.50	100.00	100.00
	1.1	100.00	98.75	100.00	100.00	100.00	100.00	100.00	100.00
	1.2	97.50	97.50	100.00	98.75	100.00	98.75	100.00	100.00
	1.3	-	-	-	100.00	100.00	97.32	100.00	100.00
	1.4	-	-	-	100.00	100.00	97.47	100.00	100.00
	1.5	-	-	-	100.00	100.00	97.50	100.00	100.00
Same Relate	2.0	96.59	98.65	100.00	97.71	100.00	96.33	100.00	100.00
	2.1	95.94	98.50	100.00	99.25	100.00	99.29	100.00	100.00
	2.2	-	-	-	100.00	100.00	100.00	100.00	100.00
	2.3	99.46	98.80	100.00	98.44	100.00	98.80	100.00	100.00
	2.4	95.60	97.93	100.00	98.05	100.00	97.04	100.00	100.00
	2.5	-	-	-	100.00	100.00	100.00	100.00	100.00
	2.6	92.86	98.08	100.00	95.70	100.00	96.70	100.00	100.00
	2.7	87.50	98.25	100.00	97.14	100.00	97.06	100.00	100.00
	2.8	-	-	-	-	100.00	-	100.00	100.00
	2.9	97.09	100.00	100.00	98.56	100.00	98.53	100.00	100.00
	2.10	96.12	97.14	100.00	98.56	100.00	97.12	100.00	100.00
	2.11	-	-	-	100.00	100.00	100.00	100.00	100.00
Comparison	3.0	96.67	95.24	100.00	99.50	100.00	96.50	100.00	97.89
	3.1	96.15	90.62	100.00	98.61	100.00	97.22	100.00	100.00
	3.2	-	-	-	-	100.00	-	100.00	100.00
Single Or	4.0	95.00	96.25	100.00	98.12	100.00	98.12	100.00	99.39
	4.1	94.97	96.25	100.00	98.75	100.00	99.38	100.00	99.39
	4.2	96.25	96.88	100.00	98.12	100.00	99.38	100.00	99.39
Zero Hop	5.0	96.25	96.25	100.00	98.75	100.00	97.50	100.00	100.00
	5.1	99.17	96.67	100.00	100.00	100.00	99.17	100.00	100.00
	5.2	-	-	-	100.00	100.00	100.00	100.00	100.00
	5.3	99.04	98.61	100.00	98.72	100.00	99.38	100.00	100.00
	5.4	98.67	98.49	100.00	100.00	100.00	98.97	100.00	99.50
Domain Exits	6.0	61.88	65.19	57.50	70.25	68.99	67.50	75.32	73.46
	6.1	65.19	67.50	85.62	75.80	82.91	73.08	94.94	92.55
	6.2	70.44	72.33	78.75	68.55	83.54	74.05	89.24	88.89
	6.3	57.50	65.00	58.75	57.50	67.50	56.25	78.75	71.95
	6.4	-	-	-	66.67	0.00	100.00	60.00	50.00
	6.5	-	-	-	50.00	52.94	54.55	59.38	64.66
Collision Order	8.0	95.00	85.00	49.37	83.75	48.10	76.25	30.38	24.39
	8.1	-	-	-	66.67	69.57	83.33	81.94	56.10
	8.2	76.25	51.25	56.25	55.00	56.25	58.75	51.25	40.24
	8.3	76.25	35.00	40.00	41.25	33.75	47.44	45.00	42.68
Quantitative Answers	10.0	85.00	80.82	66.67	82.19	60.53	82.43	61.54	45.33
	10.1	97.50	92.50	85.00	92.50	75.00	90.00	80.00	68.29
	10.2	97.50	100.00	95.00	95.00	92.50	97.50	95.00	87.80
	10.3	97.50	92.50	67.50	90.00	62.50	75.00	45.00	26.83
	10.4	65.00	20.17	0.00	19.66	2.52	20.17	2.52	2.46
	10.5	87.29	65.55	39.50	65.55	37.29	70.59	40.34	17.21
	10.6	100.00	100.00	100.00	89.74	52.14	89.92	37.82	17.21
Binary Answers	11.0	97.50	98.75	95.00	96.23	92.41	95.51	93.38	93.20
	11.1	100.00	100.00	96.00	100.00	100.00	100.00	93.75	97.30
	11.2	97.50	95.00	100.00	97.50	100.00	95.00	97.50	95.12
	11.3	92.50	100.00	100.00	95.00	95.00	92.50	95.00	87.80
	11.4	100.00	80.00	66.67	60.00	57.89	62.50	61.54	58.82
	11.5	0.00	80.00	42.86	60.00	63.16	62.50	69.23	64.71
	11.6	-	-	-	100.00	38.46	50.00	61.11	65.62
	11.7	-	-	-	100.00	38.46	50.00	55.56	67.74
	11.8	-	100.00	50.00	-	57.14	33.33	72.73	55.00
	11.9	-	100.00	50.00	-	57.14	33.33	72.73	70.00
	11.10	-	-	-	100.00	93.94	94.74	93.94	97.44
	11.11	-	-	-	100.00	97.30	100.00	89.19	92.31

Table 7.10: End-to-end accuracies per question for all levels. 40 test samples/simulations are used per level for evaluation. The Dynamics Predictor has been trained only on level 8 samples.

Question Family	Levels							
	1	2	3	4	5	6	7	8
Compare Integer	97.78	98.19	100.00	99.38	100.00	98.09	100.00	100.00
Same Relate	95.14	98.42	100.00	98.49	100.00	98.26	100.00	100.00
Comparison	96.41	92.93	100.00	99.06	100.00	96.86	100.00	99.30
Single-Or	95.41	96.46	100.00	98.33	100.00	98.96	100.00	99.39
Zero Hop	98.28	97.51	100.00	99.49	100.00	99.00	100.00	99.90
Domain Exits	63.75	67.50	70.16	64.79	71.18	70.90	76.27	73.58
Collision Order	82.50	57.08	48.54	61.67	51.92	66.44	52.14	40.85
Quantitative Answers	89.97	78.79	75.61	76.38	54.64	75.09	51.74	37.88
Binary Answers	97.50	94.22	75.07	90.87	74.24	72.45	79.64	78.76

Table 7.11: End-to-end accuracies per category per level. 40 test samples/simulations are used per level fo evaluation. This is a compacted form of Table 7.10 with averaged results per category.

the simplicity of the simulation- are not met. Questions of static categories (1,2,3,4,5) have incredibly high accuracies, since they only rely on the item identification properties of the MaskRCNN, as they query about static image relations ("Are there more blue balls than green obstacles?"). However, in these (rare) cases where the MaskRCNN fails to correctly recognize an object, these questions will be answered incorrectly.

In the dynamic categories (categories of Domain Exits, Collision Order, Quantitative Answers and Binary Answers), we can see that questions in Domain Exits are somewhat consistent in their accuracy throughout the different levels. As mentioned during training PropNet, detection of ball exits is high. In the case of Collision Order, lower levels have higher accuracies, since fewer collisions take place and in total are easier to distinguish, since the number of "orders" (first, second, third) is limited. In higher levels the results are generally lower since *ball-to-ball* collisions are harder to detect, and also, if a single *ball-to-ball* collision goes undetected, or an additional *ball-to-ball* collision is falsely detected, then the order with which these events will take place is compromised, therefore the question will be answered incorrectly. In Quantitative Answers the results are good in lower levels, however in higher levels certain questions fail, due to the failures of the algorithm to correctly detect *all* events *once*, as many of them require precise counting of events of a specific nature. In the category of Binary Answers, the results are mixed per question. Certain questions require answering if an event will take place more than one time, in which case false positives do not corrupt the answer, while other questions require precise temporal understanding of a scene in order to correctly answer (e.g. "Will the green big ball hit the purple medium ball after the yellow big ball hits the green triangle small obstacle?"), in which case, a single event detected incorrectly will result in a wrong answer. Finally, the (rare) failure of the visual component does in some cases attribute to a lower accuracy. The questions, their rephrases and their symbolic programmes can be found in Appendix A.1.

8 Issued Faced, Future Work and Conclusions

This chapter will be dedicated into analyzing various issues that were faced during the development of this work, as well as VideoQA in general. Additionally, proposals regarding future work will be provided and conclusions that stem from this work will be presented.

8.1 Issues Faced and Future Work

Numerous issues were faced during the initial analysis, development, training and inference.

1. Dataset generation was at times slow. Even though it can be parallelized to an arbitrarily large number of cores, each numerical calculation/video rendering/question generation takes about 25-35 seconds/core (at 50 fps, with each video lasting a maximum of 10 seconds). Additional time is added to generate the questions and their answers. In the cases of mass and friction estimation, the dataset was generated without rendering the videos (keeping only the positional information that was generated by the simulator) in order to save time. Training in this case did not use the positional data generated by the MaskRCNN, leading to potential lower accuracy on the VQA evaluations that did. However, using the MaskRCNN on each frame for 100k videos would take multiple weeks on a single GPU, therefore potential lower accuracy was preferred over exorbitant amounts of execution time). Even though dataset generation on the long run shouldn't be an issue -since the dataset is generated once- if someone wants to generate huge numbers of data for statistical purposes or during development, this can be a limiting factor.

2. In certain questions, especially the ones on static image reasoning, the results were not always 100%, despite their dependence in practice on the MaskRCNN. This is due to failures of the MaskRCNN to classify objects that are not fully within the frame. If a ball exits within 1-2 timesteps since t=0, the MaskRCNN might not be able to correctly detect it during its exit (since part of it would be outside the domain). A potential solution to this issue is to generate an extra number of mask-label pairs to fine-tune the MaskRCNN, where objects are purposefully partially outside the domain. However, since the potential increase in accuracy will be less that 1%, in our case it was mostly arduous to attempt.

3. One of the initial aims of this thesis was to also be able to predict future behaviors of the balls and their interactions, even after the video has ended. However, there were certain factors that prohibited such work. In order to "feed" an image to PropNet, as explained in Chapter 5.3, it needs to be downscaled, in order to not make the network (or the training data) prohibitively large. This downscaling reduces the spacial information contained in each sample. Even though one might assume that this would just lead to a smaller accuracy in the end result, this is not the case. Since the future positional information is predicted through a latent dynamic model based on the previous three (3) frames, a slight change in the positional information will cause the trajectory to turn chaotic incredibly fast, as information is propagated through all timesteps. Therefore,

higher accuracy, as well as potentially a different mechanic might be required in order to successfully predict trajectories (and the events related to them) to some extent. This would require a far larger model, and an incredibly large amount of training data (in the order of tens of thousands). Access to such resources didn't exist, therefore this part is left for future development.

4. As a continuation to the above issue, another type of interesting questions that could be asked, are counterfactual questions. These are questions that ask "*what if*". A theoretical scenario related to the scene is presented, and the dynamic behavior of the scene, based on *that* theoretical scenario, is queried upon. An example would be: "If the left wall didn't exist, from which side would the small blue ball exit?", or "If the small red ball was big, how many times would the blue ball collide with a wall?". In simple cases, such as the ones presented in CLEVRER [54], by removing dependencies of the built interaction graph, would provide an answer, bypassing the baseline prediction of the model. However, in our cases the scenes are not that simple, and such questions rely heavily on a strong predictor, which would sequentially parse the question, rebuild the initial conditions of the scene, and create a new set of outputs, upon which the translated (to a Symbolic language) question would be executed with the Symbolic Executor. However, apart from requiring tremendous amounts of data to train the predictor, such a model goes beyond the framework of this Thesis.

5. A problem posed with the Neuro Symbolic translation, is that every question-program pair has to be handcrafted, leading to the known issue faced with neural networks; that of the cost of creating the training data. It would be interesting if algorithms (Reinforcement Learning) could be implemented to generate the sequences themselves. Apart from allowing the dataset to be far larger in size (since getting simple video-question pairs is rather easy), it would very likely lead to more optimal programms that can filter out the correct answer in less, more optimally chosen commands. An initial attempt was made in the NeuroSymbolic VQA paper [55], however in order to generalize such an attempt, it would be better for the tokens in the Neuro Symbolic language (the programms) to be more arbitrary, and less tied to each task. In an optimal scenario, the tokens would essentially be the least possible, most fundamental functions that can be performed on data tables.

6. All shown accuracies are without a real-world counterpart. Human and machine accuracy comparison is difficult to perform without crowdsourcing the data. It would be interesting to set a timer (e.g. 20 seconds) per question per video, and see at which questions a human fails whereas the algorithm succeeds. It would provide incredible value to compare the accuracy of the model, and that of a human. In our case such tests were not possible to perform, but are proposed for future work.

7. Another problem faced was that, during training PropNet on simpler simulations like those of levels 1-7, more "rare" events, like ball-to-obstacle collisions, would not be learned, since they were rather imbalanced within the datasets of those difficulties, and PropNet perceived them as noise. Our results are trained on level 8 difficulty where the events are far more balanced, and the evaluation on levels of different difficulties

works rather fine, concluding that much care should be taken on the balance of events in the training set. In a future scenario, one could balance the created dataset of *all* levels, by discarding samples in a controlled manner. However, this can also lead to the training samples be tied to a specific nature of interactions, therefore completely failing to generalize on an unfiltered dataset.

8. Finally, a proposal for future work would be the inference of the object *type* from the scene dynamics. This means that obstacles and balls may have exactly the same visual attributes, however due to their distinct dynamic behaviors, they will be perceived differently. The main way to do this is through the mass classifier that is proposed in this work (Ch. 4.4). An obstacle, if it is seen to be immovable, should be classified with infinite mass, whereas a similar ball with a finite mass. This relies heavily on an almost perfectly working mass classifier, which in turn requires careful data manipulation during training. This will be able to further generalize dynamic scene understanding from a given simulation.

8.2 Conclusions

Without a doubt, it is evident that end-to-end results of Neuro-Symbolic Dynamic VQA (or VideoQA) are impressive. Otherwise difficult tasks can be formulated in forms that both humans and machines can understand, be trained upon and reply. While the proposed dataset and the provided results are of a very specific domain, as recent research shows, the path is already paved towards a new approach for AI. Far, far less data is required to train such systems, and training was relatively fast ($\sim 1-2$ days on a laptop's GPU (GTX-1650) to achieve the results shown in Fig. 6.7 and 6.8) [19]. This can only be a very positive step towards implementing Neuro-Symbolic approaches in other areas of AI, where machines struggle to understand or where data is rather limited. But with Neuro-Symbolic AI, one can train more complicated systems, with less expensive data. At the same time, less computational resources are required -against the expanding trend of ever-increasing training cost-, since a large part of the training takes place in the formulation of the problem, in the form of defining the outputs of intermediate steps, and the programs on the Symbolic Executor.

In many VQA tasks, such as those of questions of type "Quantitative Answers" and "Binary Answers", it can be seen (more so from the sample videos provided in Chapter 1 (Fig. 1.3)), that questions of these types are not the easiest to be answered by humans, yet the ensemble system presented can, while at the same time providing the logic behind its decisions and answers, as well as perfect error checking conditions. Therefore, in complicated scenarios where distributions are highly conditional and "messy", Neuro-Symbolic approaches can provide a solution to otherwise very complicated problems.

Neuro-Symbolic AI paves the way to build machines that operate far more similarly to how humans think. Apart from being simpler and easier to analyze and optimize upon, centuries of research in psychology and psychiatry can only further improve the understanding and the efficiency of such systems.

[19] The full neural approach was not even attempted, due to the simply massive amounts of data that would be required to train it, as well as the computational resources needed to achieve such training.

This Thesis has shown that end-to-end approaches are possible to be trained and inferred upon with existing, low-mid range hardware, while at the same time providing **PhAQ**, a highly customizable, novel dataset, that can be used for future projects and benchmarking, while at the same time developing **NewtoNN**, a successful end-to-end architecture that can demonstrate the capabilities of Neuro-Symbolic VQA.

A Appendices

A.1 Appendix 1: Questions, Templates and their Programms

During the dataset generation, each video is accompanied by a large number of questions. These questions belong in 10 categories, namely:

1. **Compare Integer**
2. **Same Relation**
3. **Comparison**
4. **Single-Or**
5. **Zero Hop**
6. **Domain Exits**
7. **Collision Order**
8. **Physics Understanding**
9. **Quantitative Answers**
10. **Binary Answers**

In each ID, different rephrasals of the same question are posed. The programs are the same for all rephrasals, and are at the right column. Within the brackets "<" and ">" are the "variables" of the questions, that is their entities that change depending on each simulation. "<C>" is replaced by a color, "<S>" by a shape, "<Z>" by size, "<E>" by object type (obstacle or ball), "<position_of_wall>" by the position of the wall (left, lower, right, upper), "<order>" by an order attribute (first, second, third etc.). Many times, some of these entities can be completely removed (if the remaining can still fully describe the object in question). Entities with inside brackets ([...]) are words that can be omitted without changing the meaning, but creating a rephrasal. These are removed in 50% of the cases (RNG based, during question generation).

A.2 Compare Integer

ID	Questions	Program
0	• Are there an equal number of <Z> <C> <S> <E> s and <Z2> <C2> <S2> <E2> s? • Are there the same number of <Z> <C> <S> <E> s and <Z2> <C2> <S2> <E2> s? • Is the number of <Z> <C> <S> <E> s the same as the number of <Z2> <C2> <S2> <E2> s?	objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, count, objects, <Z2>, filter_size, <C2>, filter_color, <S2>, filter_shape, <E2>, filter_type, count, is_equal

1	• Are there fewer <Z> <C> <S> <E> s than <Z2> <C2> <S2> <E2> s? • Is the number of <Z> <C> <S> <E> s less than the number of <Z2> <C2> <S2> <E2> s?	objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, count, objects, <Z2>, filter_size, <C2>, filter_color, <S2>, filter_shape, <E2>, filter_type, count, is_less
2	• Are there more <Z> <C> <S> <E> s than <Z2> <C2> <S2> <E2> s? • Is the number of <Z> <C> <S> <E> s greater than the number of <Z2> <C2> <S2> <E2> s?	objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, count, objects, <Z2>, filter_size, <C2>, filter_color, <S2>, filter_shape, <E2>, filter_type, count, is_more
3	• Are there an equal number of <E> s and <E2> s? • Are there the same number of <E> s and <E2> s? • Is the number of <E> s the same as the number of <E2> s?	objects, <E>, filter_type, count, objects, <E2>, filter_type, count, is_equal
4	• Are there fewer <E> s than <E2> s? • Is the number of <E> s less than the number of <E2> s?	objects, <E>, filter_type, count, objects, <E2>, filter_type, count, is_less
5	• Are there more <E> s than <E2> s? • Is the number of <E> s greater than the number of <E2> s?	objects, <E>, filter_type, count, objects, <E2>, filter_type, count, is_more

Table A.1: Questions from the Compare Integer category.

A.3 Same Relate

ID	Questions	Program

0	• Are there any other things that have the same size as the <Z> <C> <S> <E> ? • Is there anything else that has the same size as the <Z> <C> <S> <E> ? • Is there any other thing that has the same size as the <Z> <C> <S> <E> ? • Are there any other things that are the same size as the <Z> <C> <S> <E> ? • Is there anything else that is the same size as the <Z> <C> <S> <E> ? • Is there any other thing that is the same size as the <Z> <C> <S> <E> ?	objects, objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_size, filter_size, count, more_than_one
1	• Are there any other things that have the same color as the <Z> <C> <S> <E> ? • Is there anything else that has the same color as the <Z> <C> <S> <E> ? • Is there any other thing that has the same color as the <Z> <C> <S> <E> ? • Are there any other things that are the same color as the <Z> <C> <S> <E> ? • Is there anything else that is the same color as the <Z> <C> <S> <E> ? • Is there any other thing that is the same color as the <Z> <C> <S> <E> ? • Are there any other things of the same color as the <Z> <C> <S> <E> ? • Is there anything else of the same color as the <Z> <C> <S> <E> ? • Is there any other thing of the same color as the <Z> <C> <S> <E> ?	objects, objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_color, filter_color, count, more_than_one
2	• Are there any other things that have the same shape as the <Z> <C> <S> <E> ? • Is there anything else that has the same shape as the <Z> <C> <S> <E> ? • Is there any other thing that has the same shape as the <Z> <C> <S> <E> ? • Are there any other things that are the same shape as the <Z> <C> <S> <E> ? • Is there anything else that is the same shape as the <Z> <C> <S> <E> ? • Is there any other thing that is the same shape as the <Z> <C> <S> <E> ?	objects, objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_shape, filter_shape, count, more_than_one

3	• How many other things are the same size as the <Z> <C> <S> <E> ? • What number of other things are the same size as the <Z> <C> <S> <E> ? • How many other things are there of the same size as the <Z> <C> <S> <E> ? • What number of other things are there of the same size as the <Z> <C> <S> <E> ? • How many other objects are the same size as the <Z> <C> <S> <E> ? • What number of other objects are the same size as the <Z> <C> <S> <E> ? • How many other objects are there of the same size as the <Z> <C> <S> <E> ? • What number of other objects are there of the same size as the <Z> <C> <S> <E> ?	objects, objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_size, filter_size, count, reduce_by_one
4	• How many other things are the same color as the <Z> <C> <S> <E> ? • What number of other things are the same color as the <Z> <C> <S> <E> ? • How many other things are there of the same color as the <Z> <C> <S> <E> ? • What number of other things are there of the same color as the <Z> <C> <S> <E> ? • How many other objects are the same color as the <Z> <C> <S> <E> ? • What number of other objects are the same color as the <Z> <C> <S> <E> ? • How many other objects are there of the same color as the <Z> <C> <S> <E> ? • What number of other objects are there of the same color as the <Z> <C> <S> <E> ?	objects, objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_color, filter_color, count, reduce_by_one

5	• How many other things are the same shape as the <Z> <C> <S> <E> ? • What number of other things are the same shape as the <Z> <C> <S> <E> ? • How many other things are there of the same shape as the <Z> <C> <S> <E> ? • What number of other things are there of the same shape as the <Z> <C> <S> <E> ? • How many other objects are the same shape as the <Z> <C> <S> <E> ? • What number of other objects are the same shape as the <Z> <C> <S> <E> ? • How many other objects are there of the same shape as the <Z> <C> <S> <E> ? • What number of other objects are there of the same shape as the <Z> <C> <S> <E> ?	objects, objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_shape, filter_shape, count, reduce_by_one
6	• Are there any <C2> <S2> <E2> s that have the same size as the <Z> <C> <S> <E> ? • Is there a <C2> <S2> <E2> that has the same size as the <Z> <C> <S> <E> ? • Are there any <C2> <S2> <E2> s of the same size as the <Z> <C> <S> <E> ? • Is there a <C2> <S2> <E2> of the same size as the <Z> <C> <S> <E> ?	objects, <C2>, filter_color, <S2>, filter_shape, <E2>, filter_type, objects, <C>, filter_color, <Z>, filter_size, <S>, filter_shape, <E>, filter_type, unique, query_size, filter_size, count, exist
7	• Are there any <Z2> <S2> <E2> s that have the same color as the <Z> <C> <S> <E> ? • Is there a <Z2> <S2> <E2> that has the same color as the <Z> <C> <S> <E> ? • Are there any <Z2> <S2> <E2> s of the same color as the <Z> <C> <S> <E> ? • Is there a <Z2> <S2> <E2> of the same color as the <Z> <C> <S> <E> ?	objects, <Z2>, filter_size, <C2>, filter_color, <S2>, filter_shape, <E2>, filter_type, objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_color, filter_color, count, exist
8	• Are there any <Z2> <C2> <E2> s that have the same shape as the <Z> <C> <S> <E> ? • Is there a <Z2> <C2> <E2> that has the same shape as the <Z> <C> <S> <E> ? • Are there any <Z2> <C2> <E2> s of the same shape as the <Z> <C> <S> <E> ? • Is there a <Z2> <C2> <E2> of the same shape as the <Z> <C> <S> <E> ?	objects, <Z2>, filter_size, <C2>, filter_color, <E2>, filter_type, objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_shape, filter_shape, count, exist

9	• How many <C2> <S2> <E2> s have the same size as the <Z> <C> <S> <E> ? • How many <C2> <S2> <E2> s are the same size as the <Z> <C> <S> <E> ? • What number of <C2> <S2> <E2> s have the same size as the <Z> <C> <S> <E> ? • What number of <C2> <S2> <E2> s are the same size as the <Z> <C> <S> <E> ?	objects, <C2>, filter_color, <E2>, filter_type, <S2>, filter_shape, objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_size, filter_size, count
10	• How many <Z2> <S2> <E2> s have the same color as the <Z> <C> <S> <E> ? • How many <Z2> <S2> <E2> s are the same color as the <Z> <C> <S> <E> ? • What number of <Z2> <S2> <E2> s have the same color as the <Z> <C> <S> <E> ? • What number of <Z2> <S2> <E2> s are the same color as the <Z> <C> <S> <E> ?	objects, <Z2>, filter_size, <S2>, filter_shape, <E2>, filter_type, objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_color, filter_color, count
11	• How many <Z2> <C2> <E2> s have the same shape as the <Z> <C> <S> <E> ? • What number of <Z2> <C2> <E2> s have the same shape as the <Z> <C> <S> <E> ? • What number of <Z2> <C2> <E2> s are the same shape as the <Z> <C> <S> <E> ? • How many <Z2> <C2> <S2> <E2> s are the same shape as the <Z> <C> <S> <E> ?	objects, <Z2>, filter_size, <C2>, filter_color, <E2>, filter_type, objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_shape, filter_shape, count

Table A.2: Questions from the Same Relate category.

A.4 Comparison

ID	Questions	Program
0	• Do the <Z> <C> <S> <E> and the <Z2> <C2> <S2> <E2> have the same size? • Is the size of the <Z> <C> <S> <E> the same as the <Z2> <C2> <S2> <E2> ? • Do the <Z> <C> <S> <E> and the <Z2> <C2> <S2> <E2> have the same size? • Is the <Z> <C> <S> <E> the same size as the <Z2> <C2> <S2> <E2> ? • Does the <Z> <C> <S> <E> have the same size as the <Z2> <C2> <S2> <E2> ?	objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_size, objects, <Z2>, filter_size, <C2>, filter_color, <S2>, filter_shape, <E2>, filter_type, unique, query_size, are_equal

1	• Do the <Z> <C> <S> <E> and the <Z2> <C2> <S2> <E2> have the same color? • Is the color of the <Z> <C> <S> <E> the same as the <Z2> <C2> <S2> <E2> ? • Does the <Z> <C> <S> <E> have the same color as the <Z2> <C2> <S2> <E2> ? • Is the <Z> <C> <S> <E> the same color as the <Z2> <C2> <S2> <E2> ?	objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_color, objects, <Z2>, filter_size, <C2>, filter_color, <S2>, filter_shape, <E2>, filter_type, unique, query_color, are_equal
2	• Do the <Z> <C> <S> <E> and the <Z2> <C2> <S2> <E2> have the same shape? • Does the <Z> <C> <S> <E> have the same shape as the <Z2> <C2> <S2> <E2> ? • Is the shape of the <Z> <C> <S> <E> the same as the <Z2> <C2> <S2> <E2> ? • Is the <Z> <C> <S> <E> the same shape as the <Z2> <C2> <S2> <E2> ?	objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_shape, objects, <Z2>, filter_size, <C2>, filter_color, <S2>, filter_shape, <E2>, filter_type, unique, query_shape, are_equal

Table A.3: Questions from the Comparison category.

A.5 Single-Or

ID	Questions	Program
0	• How many things are [either] <Z> <C> <S> <E> s or <Z2> <C2> <S2> <E2> s? • How many objects are [either] <Z> <C> <S> <E> s or <Z2> <C2> <S2> <E2> s? • What number of things are [either] <Z> <C> <S> <E> s or <Z2> <C2> <S2> <E2> s? • What number of objects are [either] <Z> <C> <S> <E> s or <Z2> <C2> <S2> <E2> s?	objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, objects, <Z2>, filter_size, <C2>, filter_color, <S2>, filter_shape, <E2>, filter_type, union, count
1	• How many <Z3> things are [either] <Z> <C> <S> <E> s or <Z2> <C2> <S2> <E2> s? • How many <Z3> objects are [either] <Z> <C> <S> <E> s or <Z2> <C2> <S2> <E2> s? • What number of <Z3> things are [either] <Z> <C> <S> <E> s or <Z2> <C2> <S2> <E2> s? • What number of <Z3> objects are [either] <Z> <C> <S> <E> s or <Z2> <C2> <S2> <E2> s?	objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, objects, <Z2>, filter_size, <C2>, filter_color, <S2>, filter_shape, <E2>, filter_type, union, <Z3>, filter_size, count

2	• How many <C3> things are [either] <Z> <C> <S> <E> s or <Z2> <C2> <S2> <E2> s? • How many <C3> objects are [either] <Z> <C> <S> <E> s or <Z2> <C2> <S2> <E2> s? • What number of <C3> things are [either] <Z> <C> <S> <E> s or <Z2> <C2> <S2> <E2> s? • What number of <C3> objects are [either] <Z> <C> <S> <E> s or <Z2> <C2> <S2> <E2> s?	objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, objects, <Z2>, filter_size, <C2>, filter_color, <S2>, filter_shape, <E2>, filter_type, union, <C3>, filter_color, count
3	• How many <S3> s are [either] <Z> <C> <S> <E> or <Z2> <C2> <S2> <E2> ? • What number of <S3> s are [either] <Z> <C> <S> <E> or <Z2> <C2> <S2> <E2> ?	objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, objects, <Z2>, filter_size, <C2>, filter_color, <S2>, filter_shape, <E2>, filter_type, union, <S3>, filter_shape, count

Table A.4: Questions from the Single-Or category.

A.6 Zero Hop

ID	Questions	Program
0	• How many <Z> <C> <S> <E> s are there? • What number of <Z> <C> <S> <E> s are there?	objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, count
1	• Are there any <Z> <C> <S> <E> s? • Are any <Z> <C> <S> <E> s visible? • Is there a <Z> <C> <S> <E> ?	objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, exist
2	• What shape is the <Z> <C> <S> <E> ? • What is the shape of the <Z> <C> <S> <E> ? • The <Z> <C> <S> <E> has what shape? • What is the shape of the <Z> <C> <S> <E> ? • There is a <Z> <C> <S> <E> ; what shape is it? • The <Z> <C> <S> <E> is what shape?	objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_shape

3	• What color is the <Z> <C> <S> <E> ? • What is the color of the <Z> <C> <S> <E> ? • The <Z> <C> <S> <E> has what color? • The <Z> <C> <S> <E> is what color?	objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_color
4	• What size is the <Z> <C> <S> <E> ? • What is the size of the <Z> <C> <S> <E> ? • The <Z> <C> <S> <E> has what size? • The <Z> <C> <S> <E> is what size? • How big is the <Z> <C> <S> <E> ?	objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, <E>, filter_type, unique, query_size

Table A.5: Questions from the Zero Hop category.

A.7 Domain Exits

ID	Questions	Program
0	• Which ball will exit the domain <order> ? • Which ball will stop being in the domain <order> ? • Which ball will exit the area <order> ? • Which ball will stop being within the area <order> ?	events, ball_exits, filter_event_type, <order>, filter_order, ball, query_object, unique, describe_ball
1	• Will the <C> <Z> ball remain within the domain at the end? • Will the <C> <Z> ball still be in the domain at the end? • Will the <Z> <C> ball remain in the area at the end? • Will the <Z> <C> ball still be in the area at the end?	events, ball_exits, filter_event_type, objects, <Z>, filter_size, <C>, filter_color, ball, filter_type, filter_events, exists, inv_boolean
2	• From which side will the <C> <Z> ball exit the domain? • From which side will the <Z> <C> ball exit the area? • The <Z> <C> ball will exit the domain; from which side? • The <Z> <C> ball will exit the area; from which side?	events, ball_exits, filter_event_type, objects, <Z>, filter_size, <C>, filter_color, ball, filter_type, filter_events, side, query_object, unique, describe_side

3	• Which ball will exit from the <position_of_wall> side <order> ? • A ball will exit from the <position_of_wall> side <order> ; which ball will it be?	events, ball_exits, filter_event_type, sides, <position_of_wall>, filter_side, filter_events, <order>, filter_order, ball, query_object, unique, describe_ball
4	• A ball will exit after colliding with the <Z> <C> <S> obstacle; which ball will it be? • Which ball will exit after colliding with the <Z> <C> <S> obstacle?	events, ball_exits, filter_event_type, events, objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, obstacle, filter_type, filter_collisions, ball, query_object, filter_events, first, filter_order, ball, query_object, unique, describe_ball
5	• From which side will the ball that will collide with the <Z> <C> <S> obstacle <order> exit? • A ball will collide with the <Z> <C> <S> obstacle <order> ; from which side will it exit? • A ball will collide with the <Z> <C> <S> obstacle <order> ; from which side will it exit the area? • A ball will collide with the <Z> <C> <S> obstacle <order> ; from which side will it exit the domain?	events, ball_exits, filter_event_type, events, objects, filter_collisions, objects, <Z>, filter_size, <C>, filter_color, <S>, filter_shape, filter_events, <order>, filter_order, ball, query_object, filter_events, side, query_object, unique, describe_side
6	• Do any balls exit the domain?	events, ball_exits, count, is_zero

Table A.6: Questions from the Domain Exits category.

A.8 Collision Order

ID	Questions	Program
0	• Which two balls will collide together <order> ? • Between which two balls will the <order> collision occur?	events, ball2ball, filter_event_type, <order>, filter_order, describe_particip

1	• Which ball will hit an obstacle <order> ? • A ball will hit an obstacle <order> ; which ball will it be?	events, ball2obstacle, filter_event_type, <order>, filter_order, ball, query_object, unique, describe_ball
2	• Which ball will hit a wall or an obstacle for the <order> time? • A ball will hit a wall or an obstacle for the <order> time; which ball will it be?	events, ball2wall, filter_event_type, events, ball2obstacle, filter_event_type, union, <order>, filter_order, ball, query_object, unique, describe_ball
3	• With which ball will the <position_of_wall> wall collide <order> ? • The upper wall will collide <order> with a ball; which ball is that?	events, ball2wall, filter_event_type, <position_of_wall>, filter_wall, <order>, filter_order, ball, query_object, unique, describe_ball
4	• With which ball will the <C> <S> <Z> obstacle collide for the <order> time? • The <C> <S> <Z> obstacle will collide with a ball for a <order> time; which ball will that be?	events, objects, <C>, filter_color, <S>, filter_shape, <Z>, filter_size, filter_obstacle, order, filter_order, ball, query_object, unique, describe_ball

Table A.7: Questions from the Collision Order category.

A.9 Physics Understanding

ID	Questions	Program
0	• Which ball is the heaviest? • Which ball has the largest mass?	objects, ball, filter_type, mass, find_max, describe_ball
1	• Which ball is the lightest? • Which ball has the lowest mass?	objects, ball, filter_type, mass, find_min, describe_ball

2	• Is the <C> <Z> ball heavier than the <C2> <Z2> ball? • Is the <C2> <Z2> ball lighter than the <C> <Z> ball?	objects, ball, filter_type, <C>, filter_color, <Z>, filter_size, query_mass, objects, ball, filter_type, <C2>, filter_color, <Z2>, filter_size, query_mass, is_more
3	• Between the <C> <Z> ball and the <C2> <Z2> ball, which one is the heaviest? • Between the <C2> <Z2> ball and the <C> <Z> ball, which one is the heaviest?	objects, ball, filter_type, <C>, filter_color, <Z>, filter_size, unique, objects, ball, filter_type, <C2>, filter_color, <Z2>, filter_size, unique, filter_bigger_mass, describe_ball
4	• Between the <C> <Z> ball and the <C2> <Z2> ball, which one is the lightest? • Between the <C2> <Z2> ball and the <C> <Z> ball, which one is the lightest?	objects, ball, filter_type, <C>, filter_color, <Z>, filter_size, unique, objects, ball, filter_type, <C2>, filter_color, <Z2>, filter_size, unique, filter_smaller_mass, describe_ball
5	• Which ball will be the <order> fastest at some point? • Which ball will be the <order> fastest? • One ball will achieve the <order> highest velocity at some point. Which one will it be?	xy_vel, vel, asc, sort_xyv, <order>, df_order, describe_ball
6	• Which ball will be the <order> slowest at some point? • Which ball will be the <order> slowest? • One ball will achieve the <order> lowest velocity at some point. Which one will it be?	xy_vel, vel, des, sort_xyv, <order>, df_order, describe_ball
7	• Which ball will start of with the <order> highest velocity? • Which ball will have the <order> highest velocity at the beginning?	xy_vel, start, filter_ts, vel, asc, sort_xyv, <order>, df_order, describe_ball
8	• Which ball will start of with the <order> lowest velocity? • Which ball will have the <order> lowest velocity at the beginning?	xy_vel, start, filter_ts, vel, des, sort_xyv, <order>, df_order, describe_ball
9	• What type of resistance do the balls face in the simulation? • What is the type of the resistance?	resistances, query_type

10	• What is the level of resistance do the balls face in the simulation? • What is the level of the resistance? • What is the intensity of the resistance in the simulation? • What is the intensity of the resistance?	resistances, query_level

Table A.8: Questions from the Physics Understanding category.

A.10 Quantitative Answers

ID	Questions	Program
0	• How many collisions will happen with the <position_of_wall> wall? • How many times will a ball collide with the <position_of_wall> wall?	events, ball2wall, filter_event_type, <position_of_wall>, filter_wall, count
1	• How many different balls will hit the <C> <Z> ball?	events, ball2ball, filter_event_type, objects, <C>, filter_color, <Z>, filter_size, filter_events, filter_particip, count, reduce_by_one
2	• How many times will the <C> <Z> ball and <C2> <Z2> ball collide?	events, ball2ball, filter_event_type, objects, <C>, filter_color, <Z>, filter_size, filter_events, objects, <C2>, filter_color, <Z2>, filter_size, filter_events, count
3	• How many times will a <C> <Z> ball and a <C2> <Z2> ball collide?	events, ball2ball, filter_event_type, objects, <C>, filter_color, <Z>, filter_size, filter_events, events, ball2ball, filter_event_type, objects, <C2>, filter_color, <Z2>, filter_size, filter_events, intersection, count
4	• How many ball to ball collisions will take place in total? • How many ball to ball collisions will happen in total? • How many ball to ball collisions will take place?	events, ball2ball, filter_event_type, count

5	• How many wall to ball collisions will take place in total? • How many wall to ball collisions will happen in total? • How many wall to ball collisions will take place?	events, ball2wall, filter_event_type, count
6	• How many ball to obstacle collisions will take place in total? • How many ball to obstacle collisions will happen in total? • How many ball to obstacle collisions will take place?	events, ball2obstacle, filter_event_type, count

Table A.9: Questions from the Quantitative Answers category.

A.11 Binary Answers

ID	Questions	Program
0	• Will the <C> <Z> ball hit the <C2> <Z2> at some point? • Will the <C> <Z> ball hit the <C2> <Z2> at any time? • Will the <C> <Z> and the <C2> <Z2> ball ever collide together? • Will the <C> <Z> and the <C2> <Z2> ball ever collide?	events, ball2ball, filter_event_type, objects, <C>, filter_color, <Z>, filter_size, filter_events, objects, <C2>, filter_color, <Z2>, filter_size, filter_events, exists
1	• Will the <C> <Z> ball hit the <C2> <Z2> ball more than <time> time(s)?	events, ball2ball, filter_event_type, objects, <C>, filter_color, <Z>, filter_size, filter_events, objects, <C2>, filter_color, <Z2>, filter_size, filter_events, count, <time>, is_more
2	• Will the <C> <Z> ball hit the <position_of_wall> wall?	events, ball2wall, filter_event_type, objects, <C>, filter_color, <Z>, filter_size, filter_events, objects, <position_of_wall>, filter_side, filter_events, exists

3	• Will the <C> <Z> ball hit the <position_of_wall> wall more than <time> time(s)?	events, ball2wall, filter_event_type, objects, <C>, filter_color, <Z>, filter_size, filter_events, objects, <position_of_wall>, filter_side, filter_events, count, <time>, is_more
4	• Will the <C> <Z> ball hit the <C2> <Z2> ball before the <C3> <Z3> ball hits the <position_of_wall> wall?	events, ball2ball, filter_event_type, objects, <C>, filter_color, <Z>, filter_size, filter_events, objects, <C2>, filter_color, <Z2>, filter_size, filter_events, unique, events, ball2wall, filter_event_type, objects, <C3>, filter_color, <Z3>, filter_size, filter_events, objects, <position_of_wall>, filter_side, filter_events, unique, is_less
5	• Will the <C> <Z> ball hit the <C2> <Z2> ball after the <C3> <Z3> ball hits the <position_of_wall> wall?	events, ball2ball, filter_event_type, objects, <C>, filter_color, <Z>, filter_size, filter_events, objects, <C2>, filter_color, <Z2>, filter_size, filter_events, unique, events, ball2wall, filter_event_type, objects, <C3>, filter_color, <Z3>, filter_size, filter_events, objects, <position_of_wall>, filter_side, filter_events, unique, is_more

6	• Will the <C> <Z> ball hit the <C2> <Z2> ball before the <C3> <Z3> ball hits the <C4> <S4> <Z4> obstacle?	events, ball2ball, filter_event_type, objects, <C>, filter_color, <Z>, filter_size, filter_events, objects, <C2>, filter_color, <Z2>, filter_size, filter_events, unique, events, ball2obstacle, filter_event_type, objects, <C3>, filter_color, <Z3>, filter_size, ball, filter_type, filter_events, objects, <C4>, filter_color, <Z4>, filter_size, <S4>, filter_shape, filter_events, unique, is_less
7	• Will the <C> <Z> ball hit the <C2> <Z2> ball after the <C3> <Z3> ball hits the <C4> <S4> <Z4> obstacle?	events, ball2ball, filter_event_type, objects, <C>, filter_color, <Z>, filter_size, filter_events, objects, <C2>, filter_color, <Z2>, filter_size, filter_events, unique, events, ball2obstacle, filter_event_type, objects, <C3>, filter_color, <Z3>, filter_size, filter_events, objects, <C4>, filter_color, <Z4>, filter_size, <S4>, filter_shape, filter_events, unique, is_more

8	• Will the <C> <Z> ball hit the <C2> <Z2> ball before the <C3> <Z3> ball collides with the <C4> <Z4> ball?	events, ball2ball, filter_event_type, objects, <C>, filter_color, <Z>, filter_size, filter_events, objects, <C2>, filter_color, <Z2>, filter_size, filter_events, unique, events, ball2ball, filter_event_type, objects, <C3>, filter_color, <Z3>, filter_size, filter_events, objects, <C4>, filter_color, <Z4>, filter_size, filter_events, unique, is_less
9	• Will the <C> <Z> ball hit the <C2> <Z2> ball after the <C3> <Z3> ball collides with the <C4> <Z4> ball?	events, ball2ball, filter_event_type, objects, <C>, filter_color, <Z>, filter_size, filter_events, objects, <C2>, filter_color, <Z2>, filter_size, filter_events, unique, events, ball2ball, filter_event_type, objects, <C3>, filter_color, <Z3>, filter_size, filter_events, objects, <C4>, filter_color, <Z4>, filter_size, filter_events, unique, is_more
10	• Will the <C> <Z> ball collide with an obstacle of the same color?	events, ball2obstacle, filter_event_type, objects, <C>, filter_color, <Z>, filter_size, ball, filter_type, filter_events, objects, obstacle, filter_type, <C>, filter_color, filter_events, exists

11	• Will the <C> <Z> ball collide with an obstacle of the same size?	events, ball2obstacle, filter_event_type, objects, <C>, filter_color, <Z>, filter_size, ball, filter_type, filter_events, objects, obstacle, filter_type, <Z>, filter_size, filter_events, exists

Table A.10: Questions from the Binary Answers category.

A.12 Appendix 2: Distribution of Answers

For each question category, each question can have a varying number of answers. These answers can be "Yes/No" answers, numbers, attributes, or object descriptions. In the following tables, answer distributions for the the categories of Integer Comparison (Figures A.1, A.11, A.21, A.31), Same Relate (Figures A.2, A.12, A.22, A.32), Comparison (Figures A.3, A.13, A.23, A.33), Single Or (Figures A.4, A.14, A.24, A.34), Zero Hop (Figures A.5, A.15, A.25, A.35), Domain Exits (Figures A.6, A.16, A.26, A.36), Collision Order (Figures A.7, A.17, A.27, A.37), Physical Understanding (Figures A.8, A.18, A.28, A.38), Quantitative Answers (Figures A.9, A.19, A.29, A.39) and Binary Answers (Figures A.10, A.20, A.30, A.40).

A.12.1 Answer distributions of simulation of level *1*

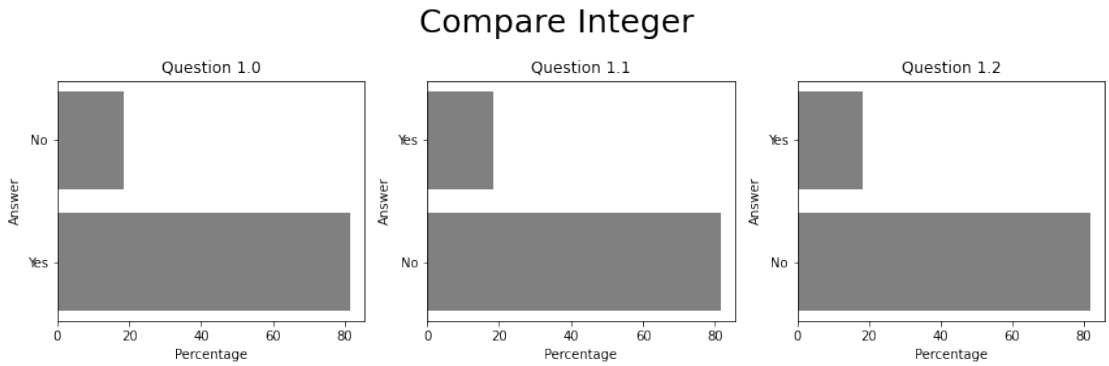

Figure A.1: Answer distribution of questions in the category of Compare Integer (level 1)

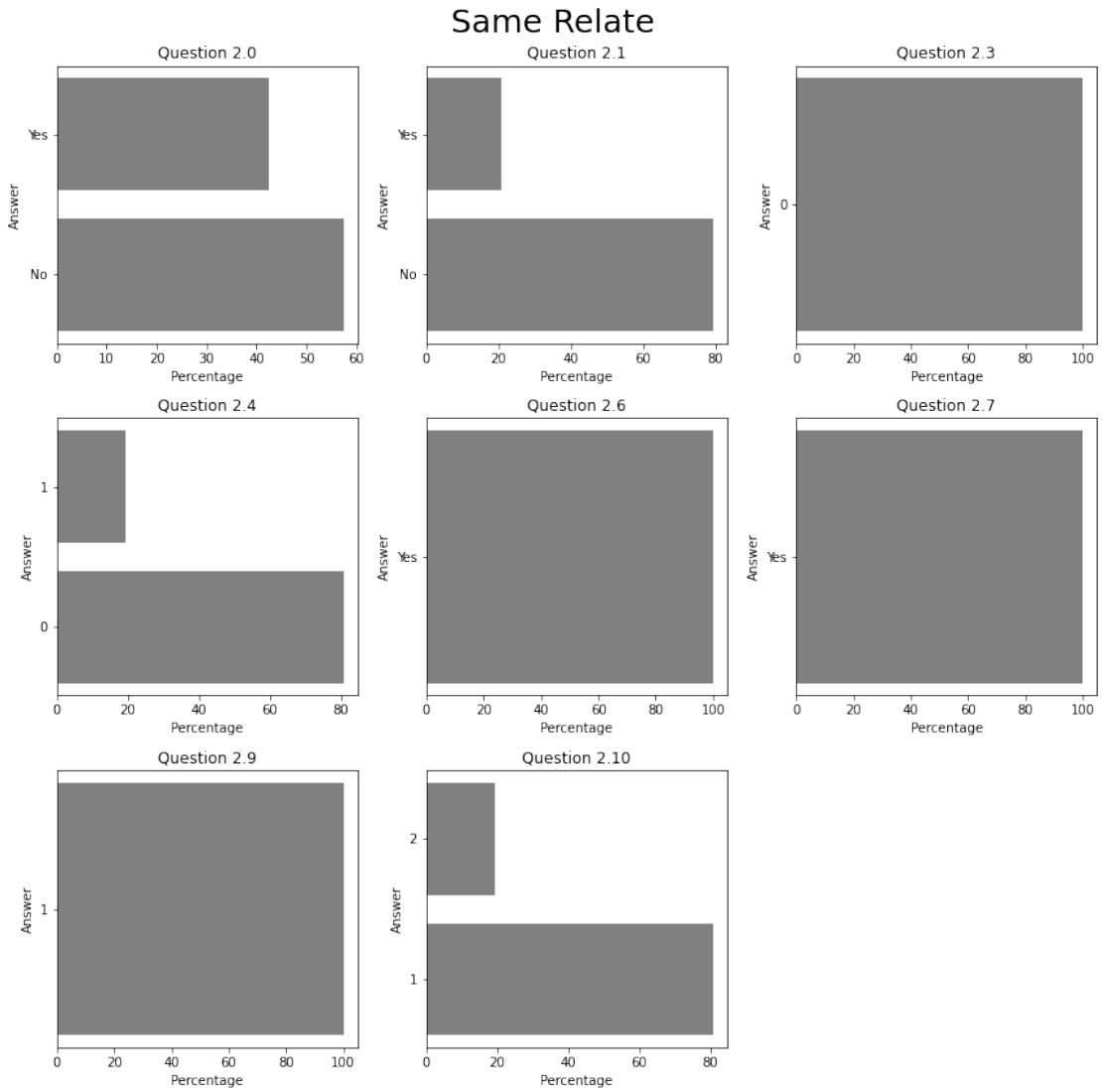

Figure A.2: Answer distribution of questions in the category of Same Relate (level 1)

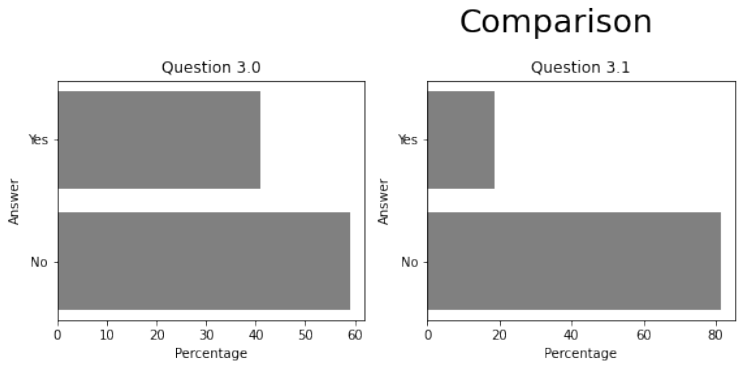

Figure A.3: Answer distribution of questions in the category of Comparison (level 1)

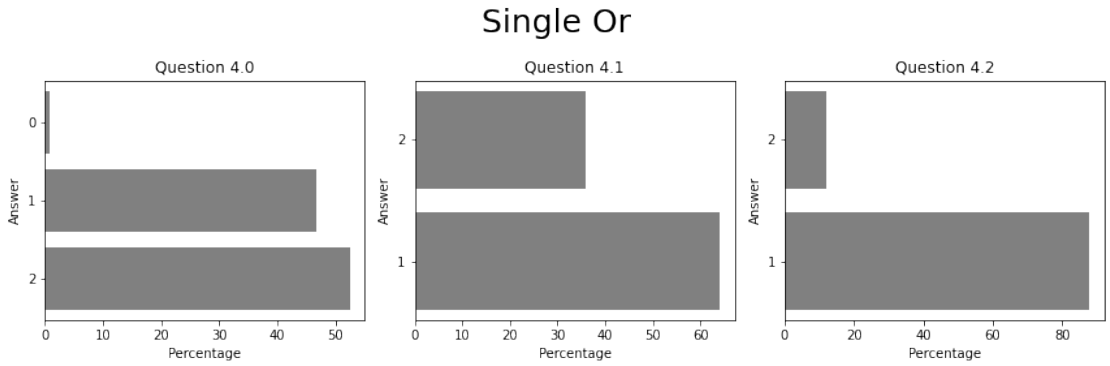

Figure A.4: Answer distribution of questions in the category of Single-Or (level 1)

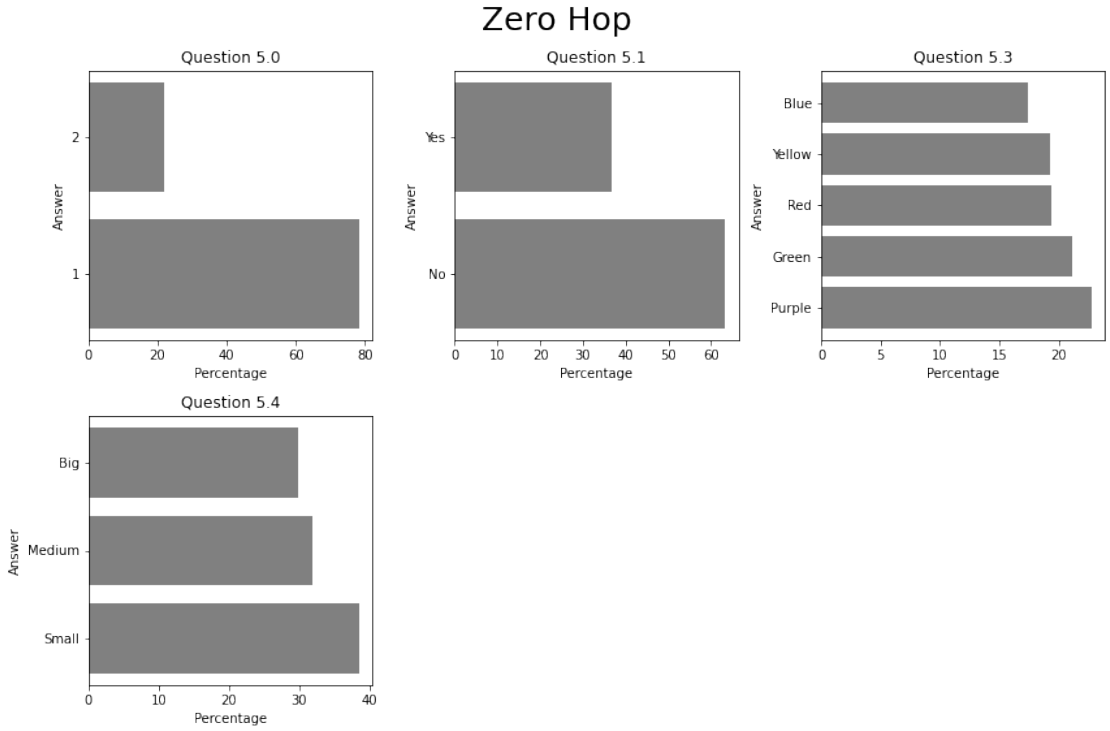

Figure A.5: Answer distribution of questions in the category of Zero Hop (level 1)

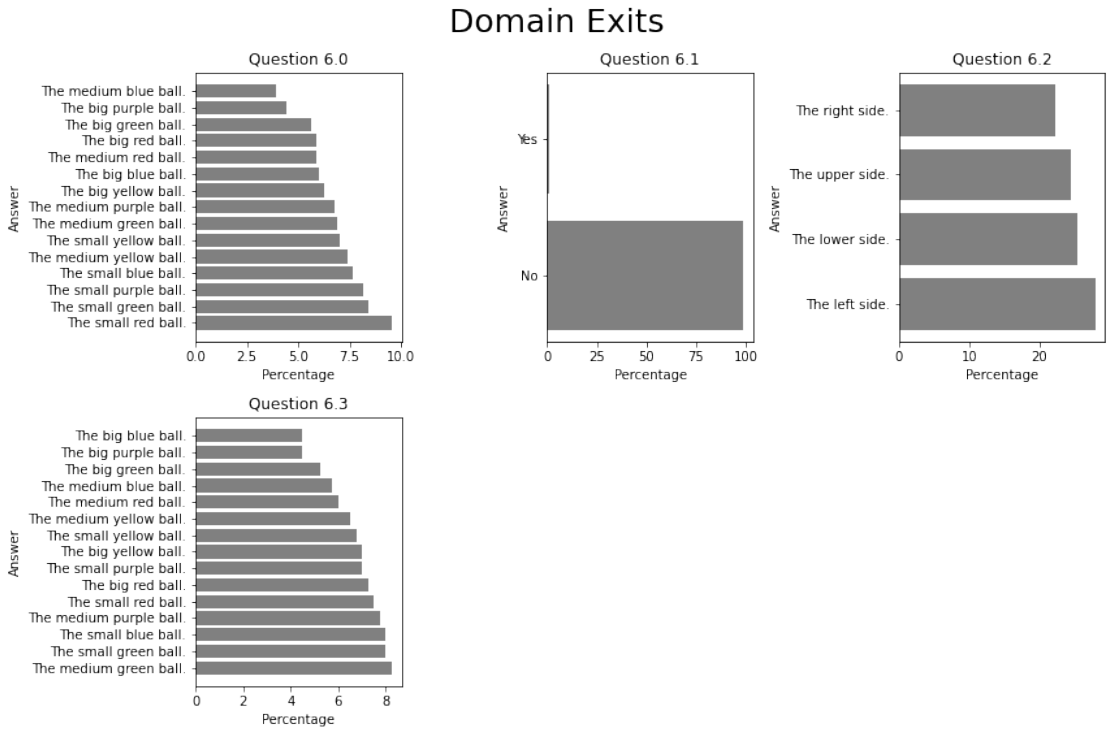

Figure A.6: Answer distribution of questions in the category of Domain Exits (level 1)

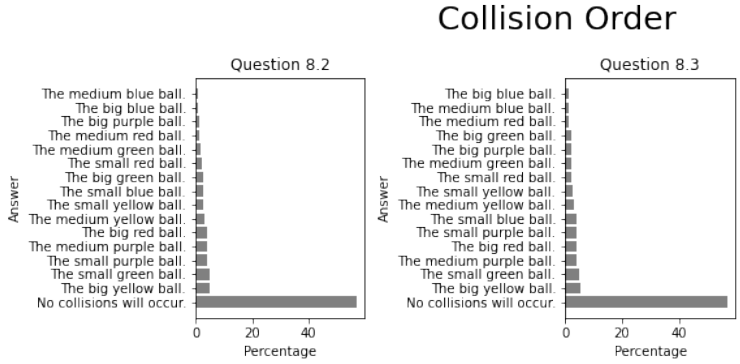

Figure A.7: Answer distribution of questions in the category of Collision Order (level 1)

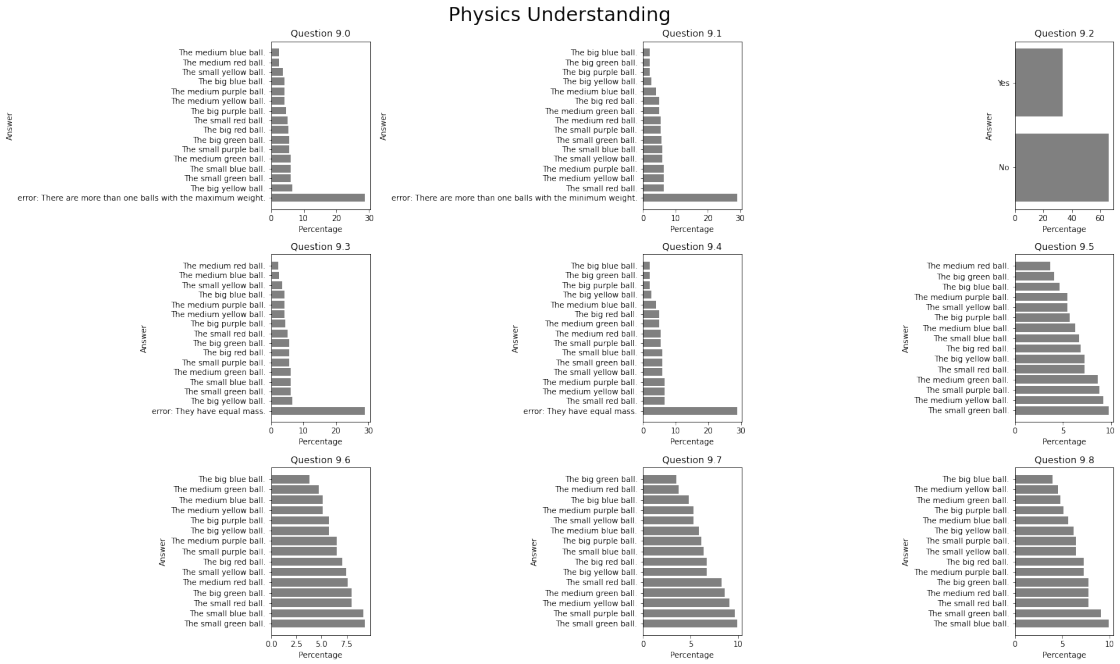

Figure A.8: Answer distribution of questions in the category of Physics Understanding (level 1). Questions 9.9 has equiprobable distribution of the answers, while 9.10 has a 1/3 chance for category 0, and 2/9 for all the others.

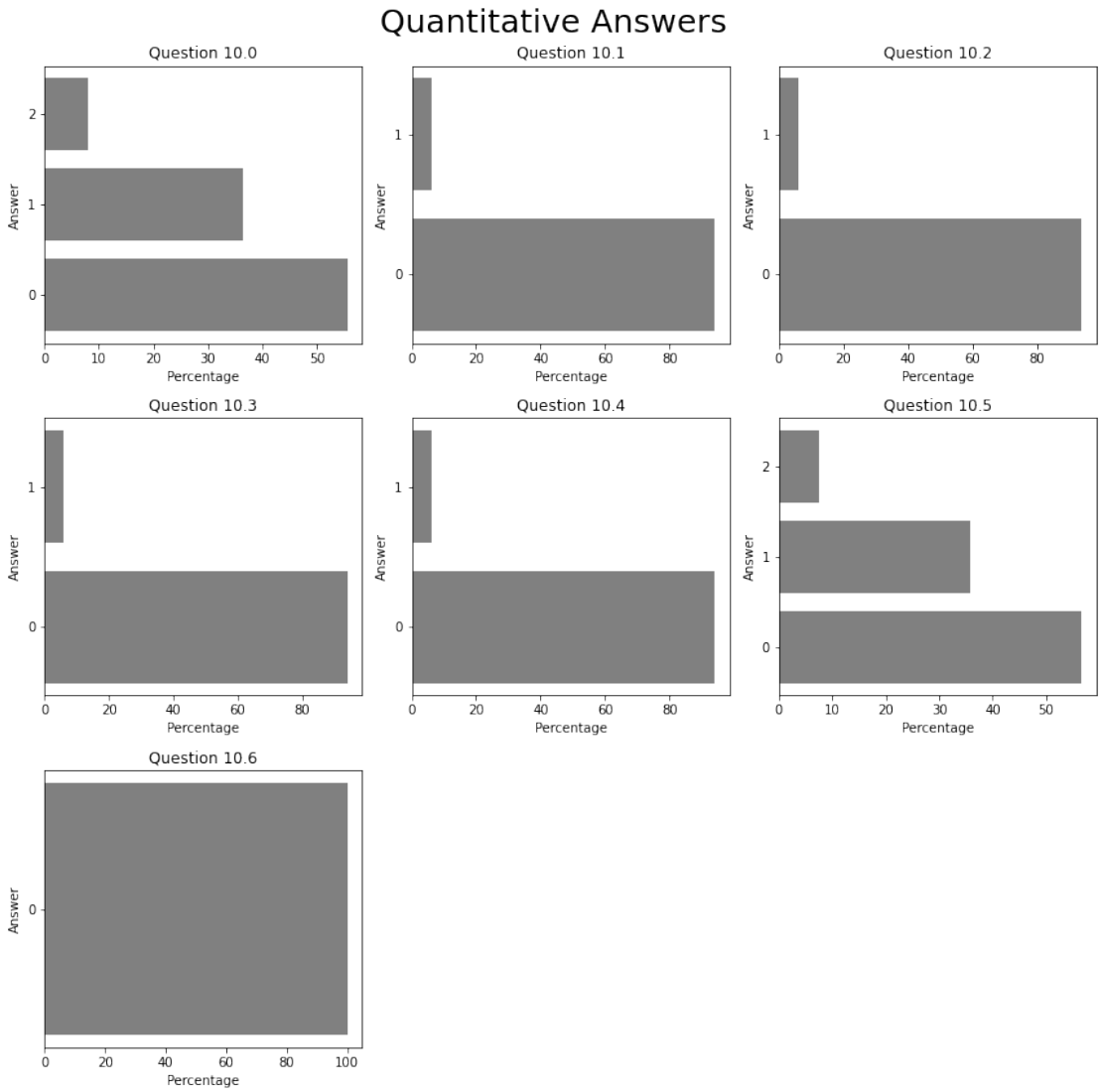

Figure A.9: Answer distribution of questions in the category of Quantitative Answers (level 1)

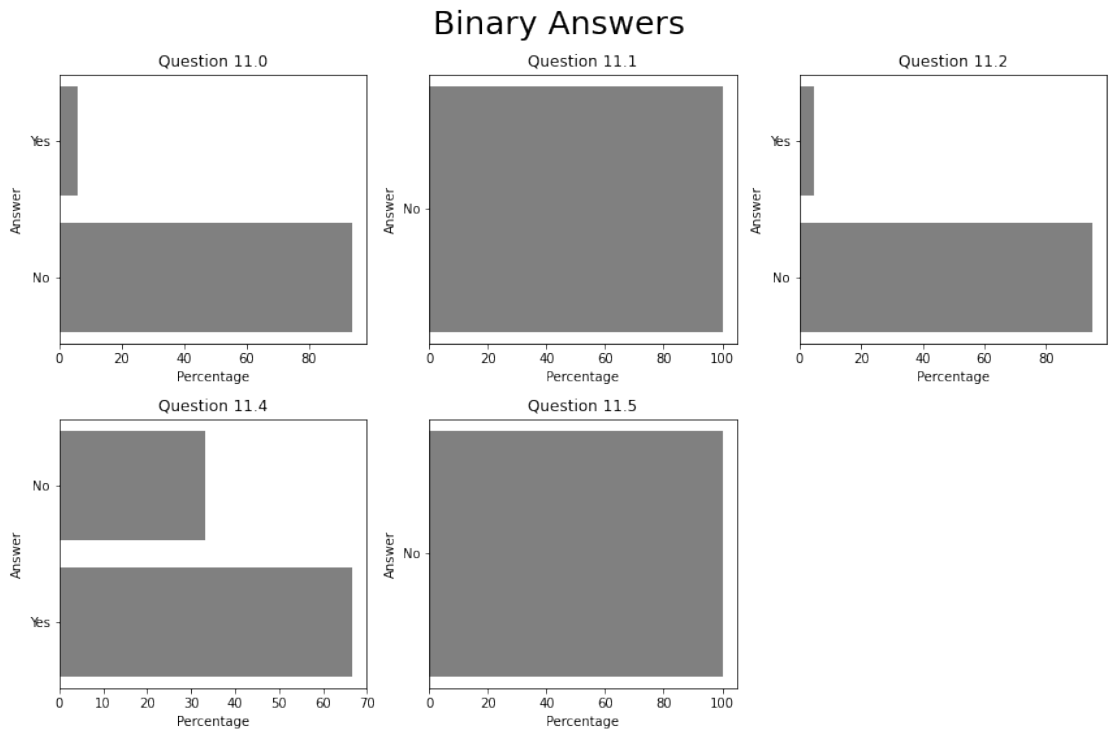

Figure A.10: Answer distribution of questions in the category of Binary Answers (level 1)

A.12.2 Answer distributions of simulation of level *3*

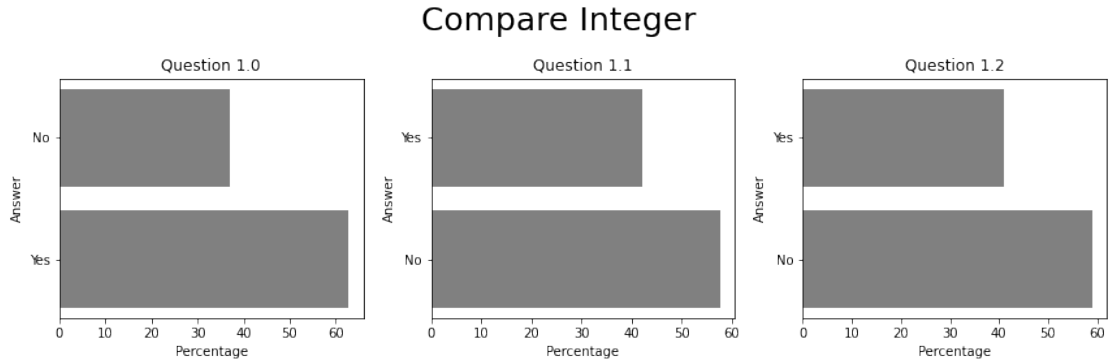

Figure A.11: Answer distribution of questions in the category of Compare Integer (level 3)

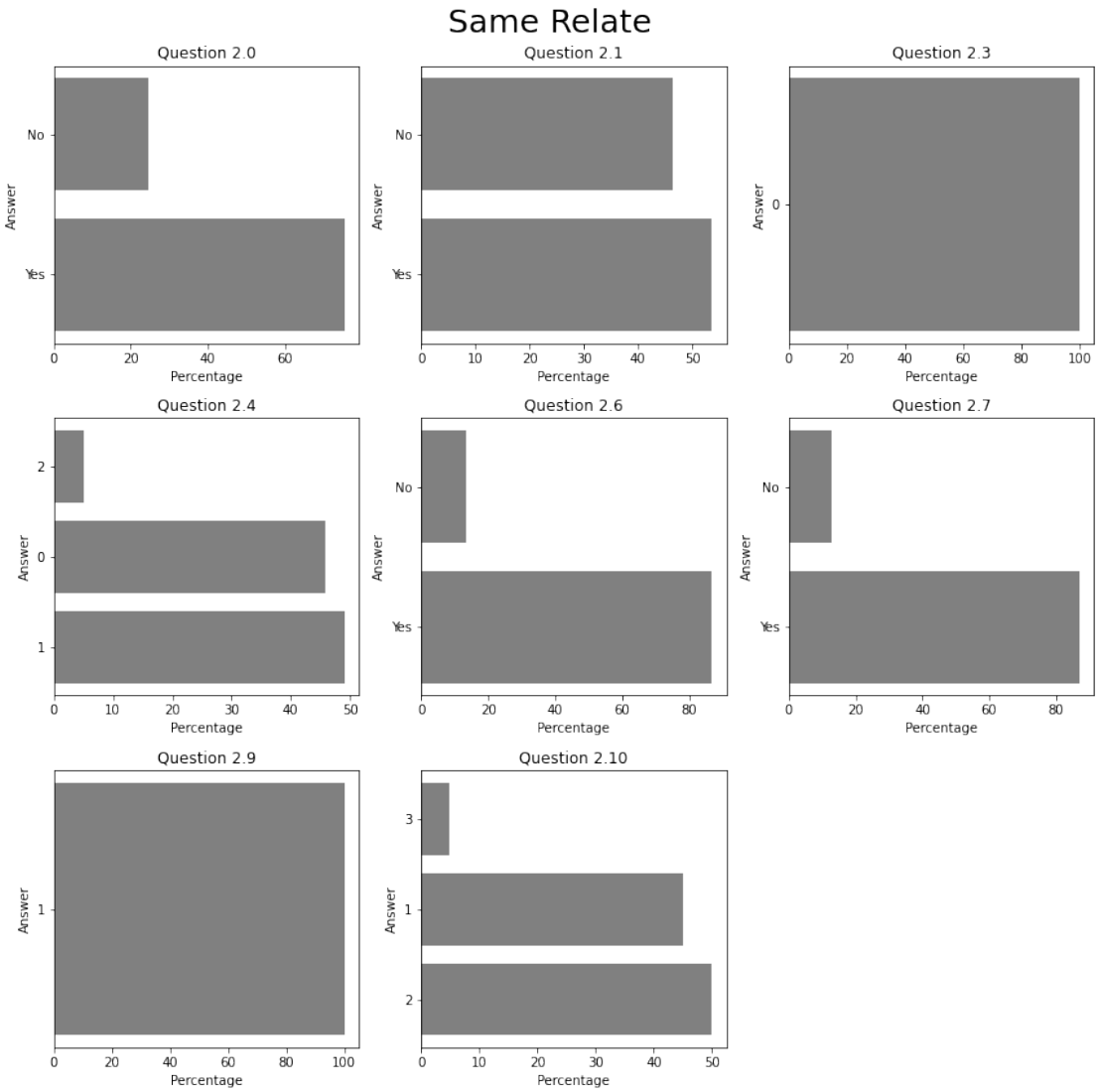

Figure A.12: Answer distribution of questions in the category of Same Relate (level 3)

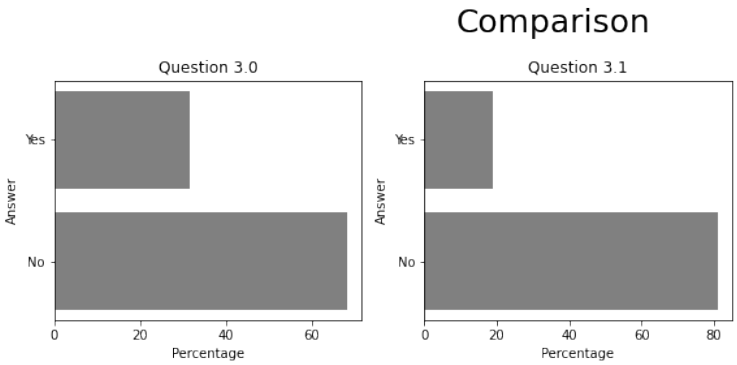

Figure A.13: Answer distribution of questions in the category of Comparison (level 3)

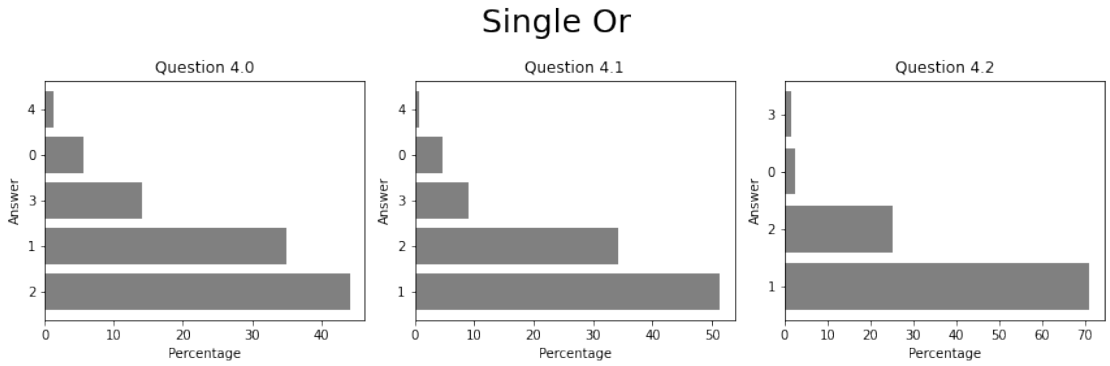

Figure A.14: Answer distribution of questions in the category of Single-Or (level 3)

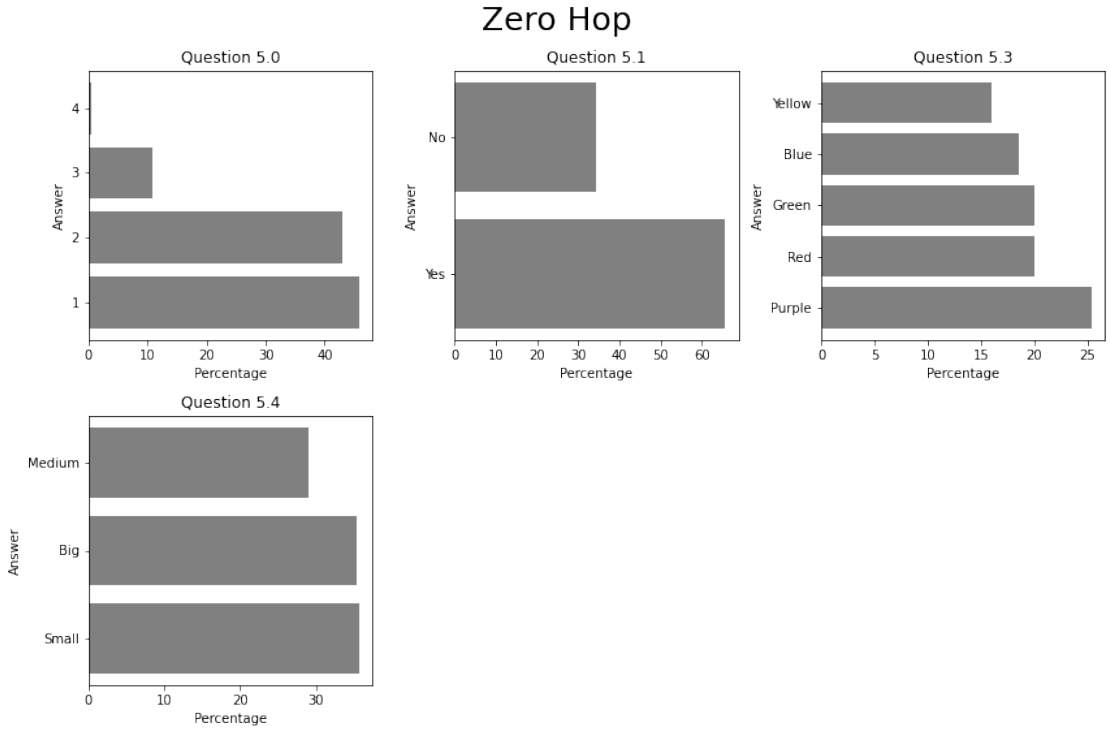

Figure A.15: Answer distribution of questions in the category of Zero Hop (level 3)

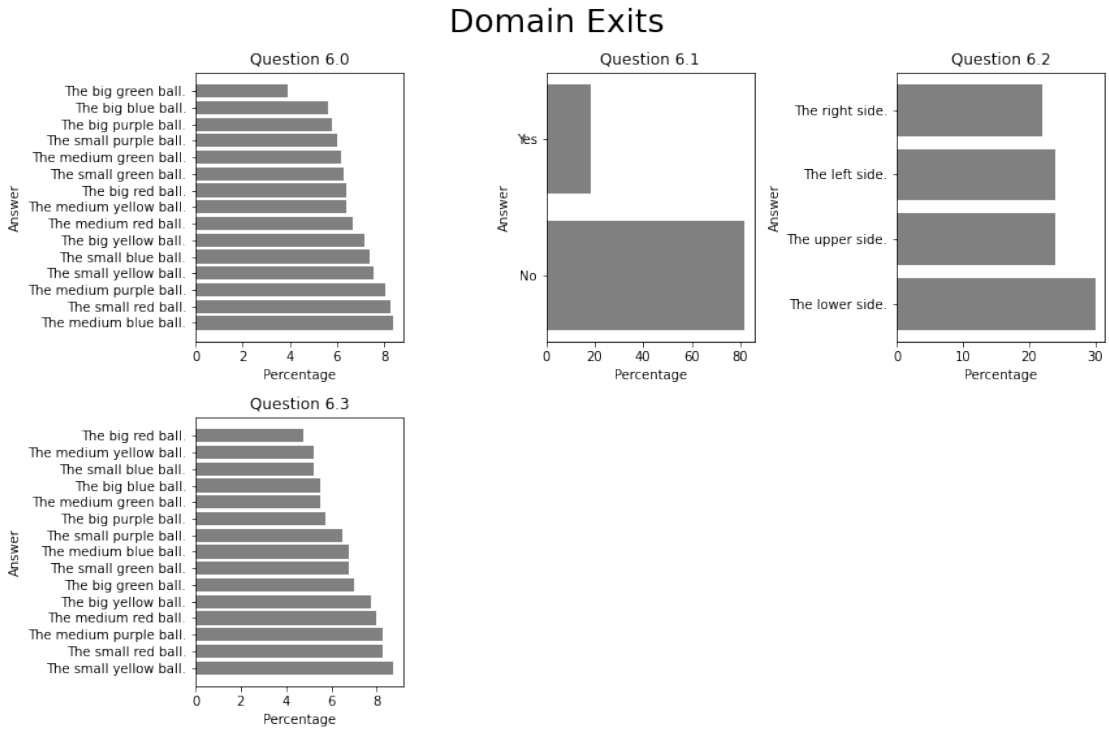

Figure A.16: Answer distribution of questions in the category of Domain Exits (level 3)

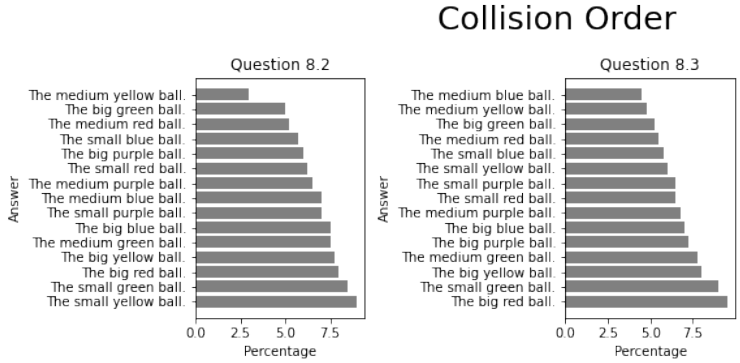

Figure A.17: Answer distribution of questions in the category of Collision Order (level 3)

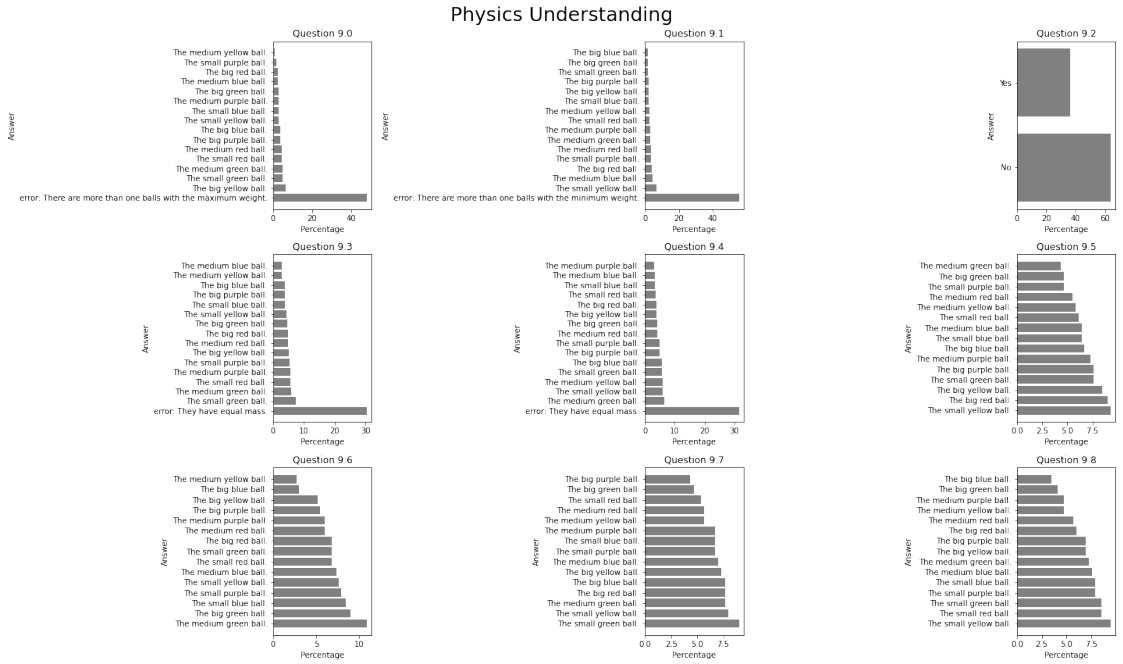

Figure A.18: Answer distribution of questions in the category of Physics Understanding (level 3). Questions 9.9 has equiprobable distribution of the answers, while 9.10 has a 1/3 chance for category 0, and 2/9 for all the others.

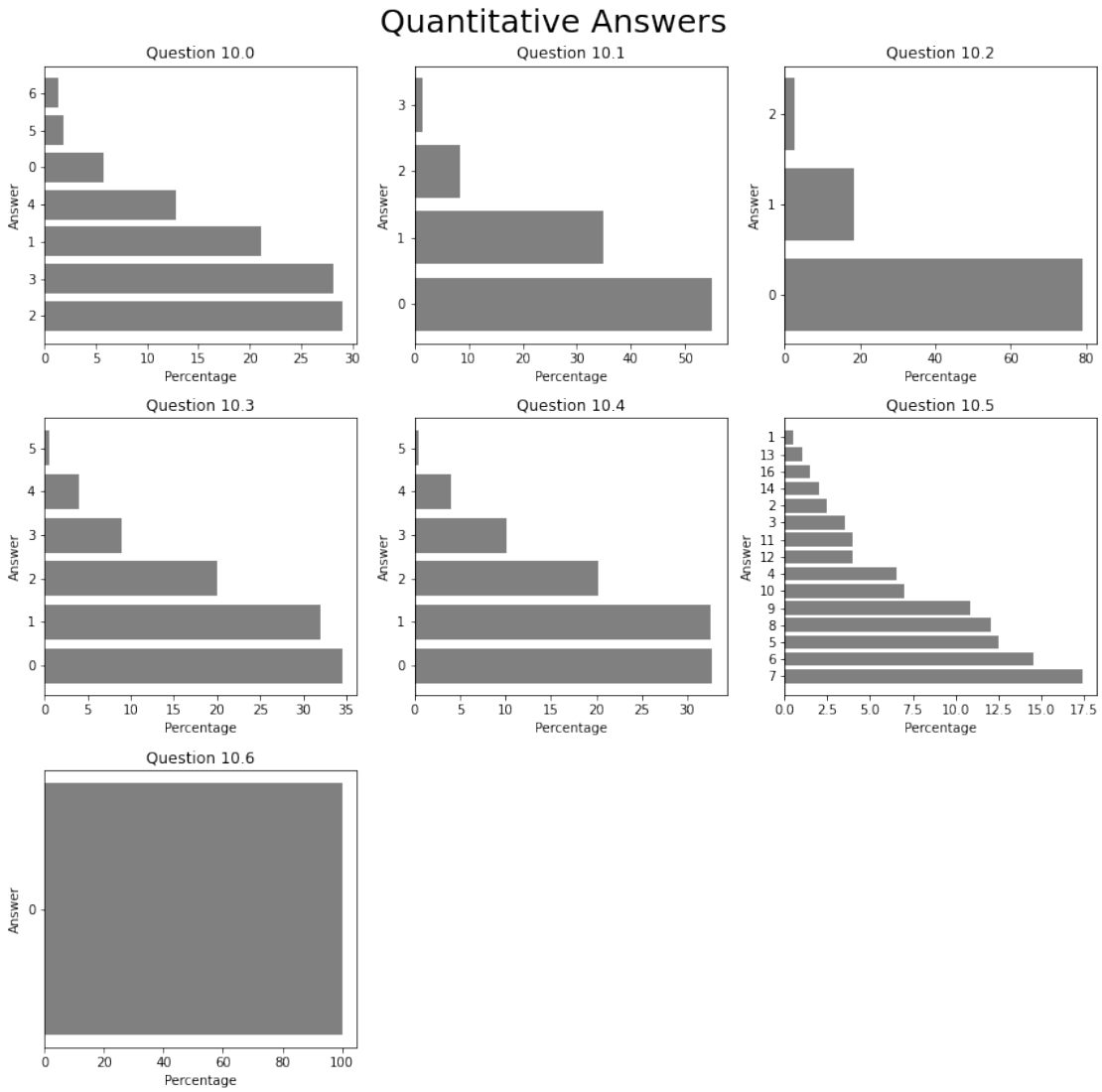

Figure A.19: Answer distribution of questions in the category of Quantitative Answers (level 3)

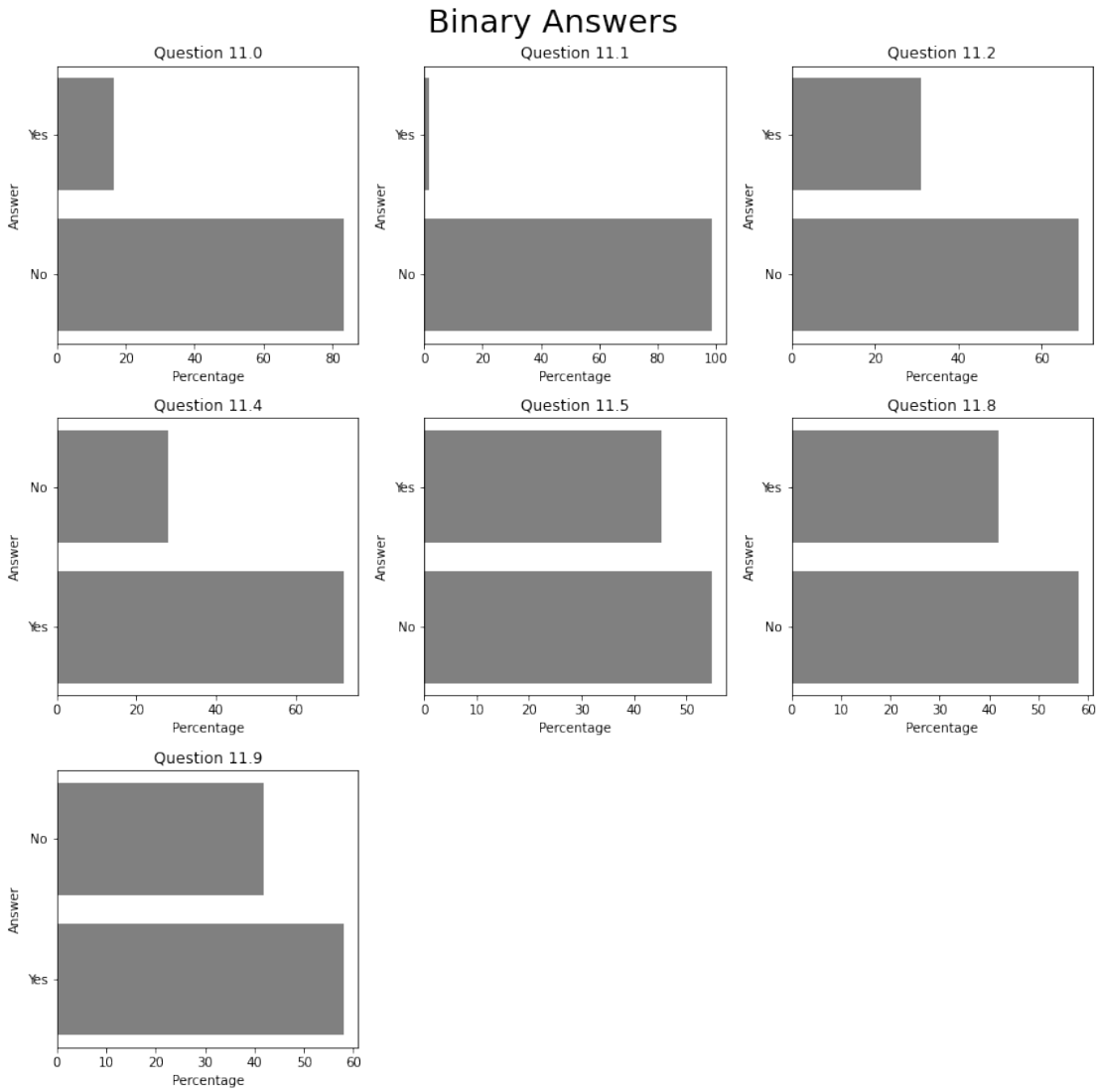

Figure A.20: Answer distribution of questions in the category of Binary Answers (level 3)

A.12.3 Answer distributions of simulation of level *6*

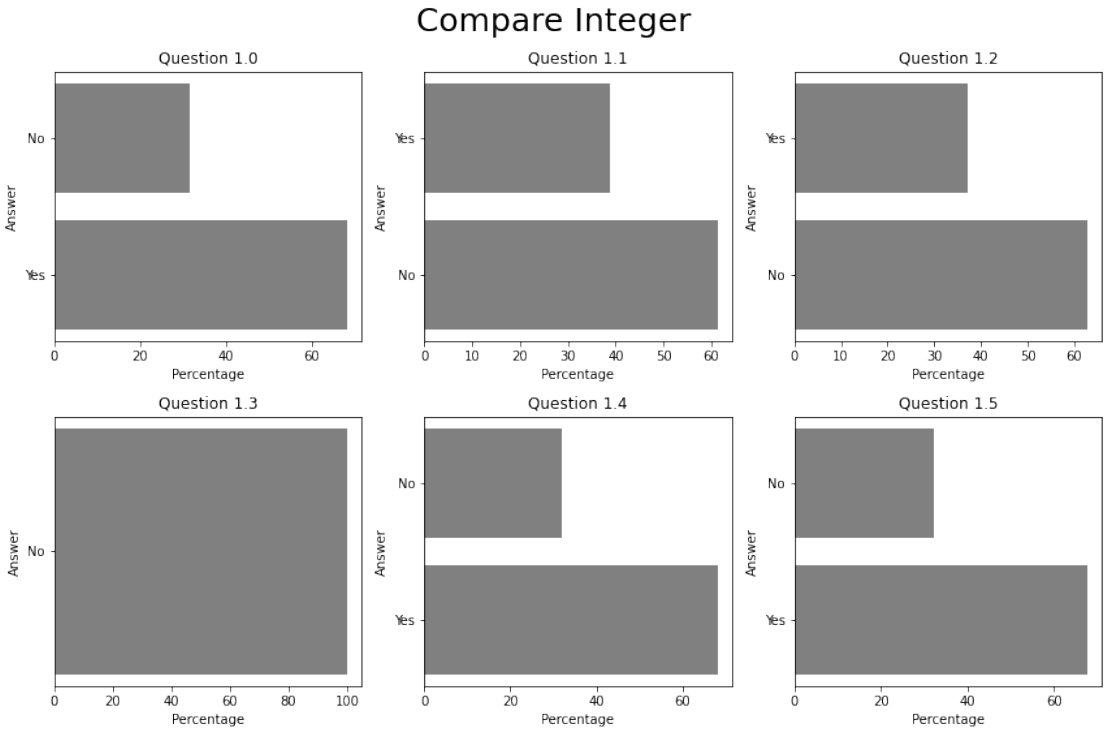

Figure A.21: Answer distribution of questions in the category of Compare Integer (level 6)

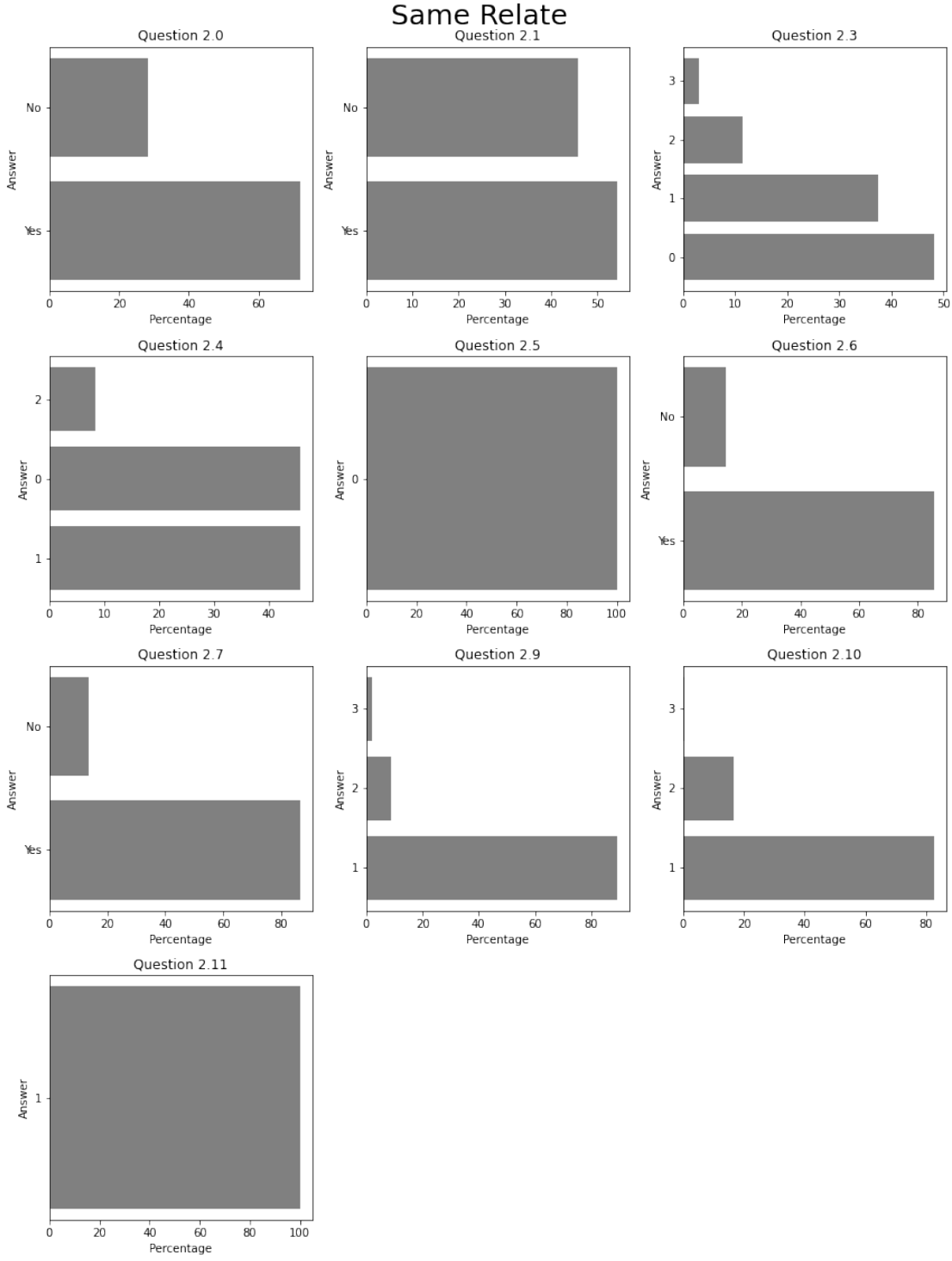

Figure A.22: Answer distribution of questions in the category of Same Relate (level 6)

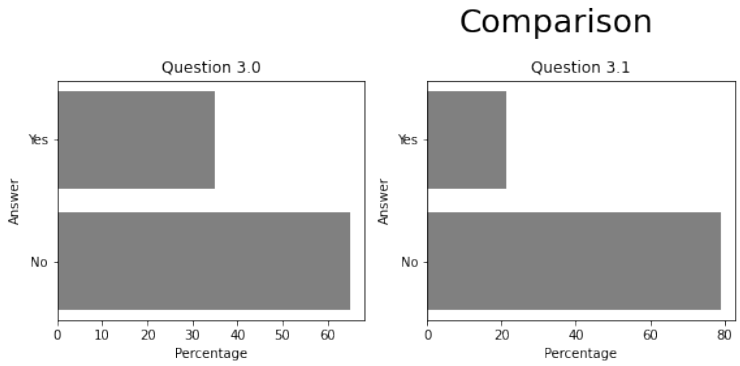

Figure A.23: Answer distribution of questions in the category of Comparison (level 6)

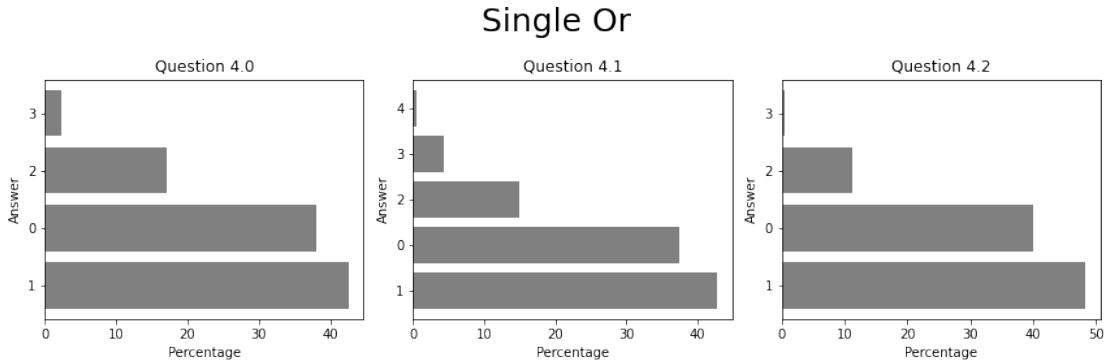

Figure A.24: Answer distribution of questions in the category of Single-Or (level 6)

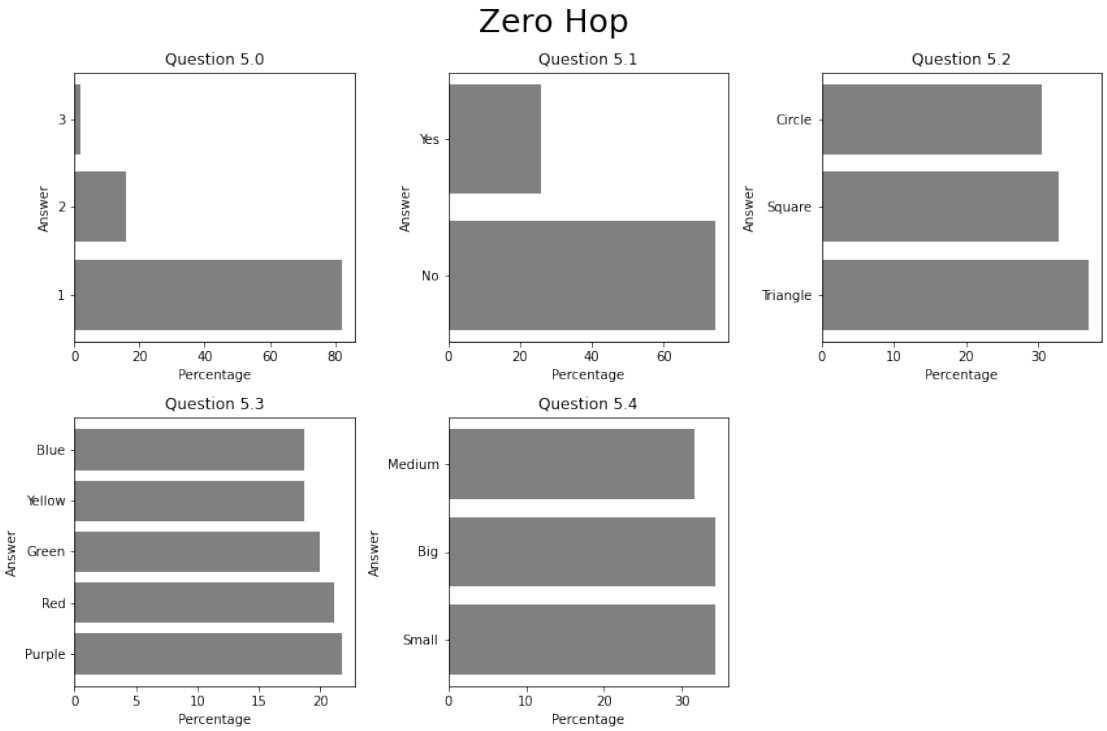

Figure A.25: Answer distribution of questions in the category of Zero Hop (level 6)

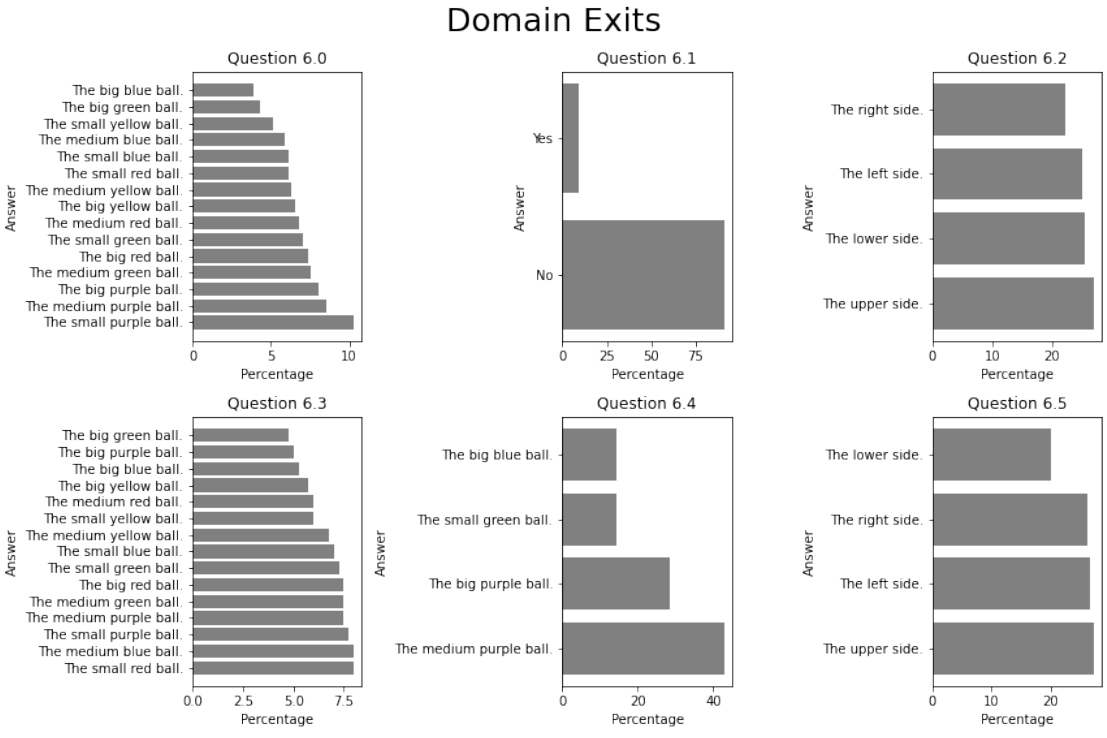

Figure A.26: Answer distribution of questions in the category of Domain Exits (level 6)

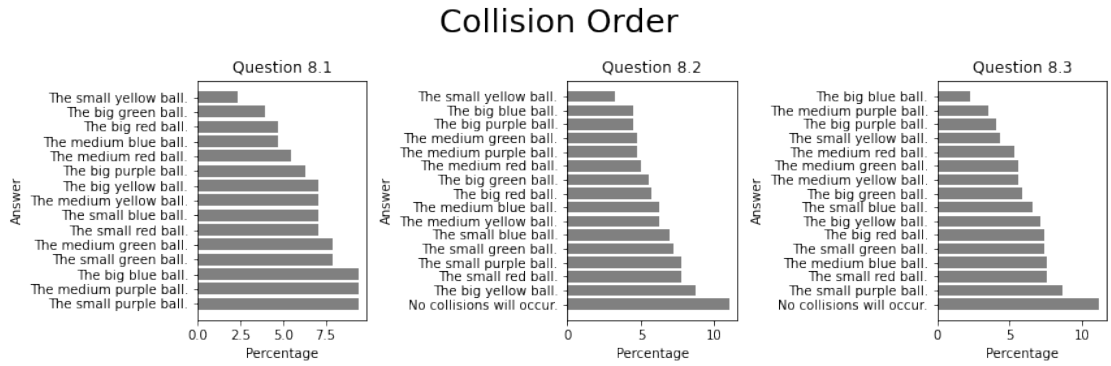

Figure A.27: Answer distribution of questions in the category of Collision Order (level 6)

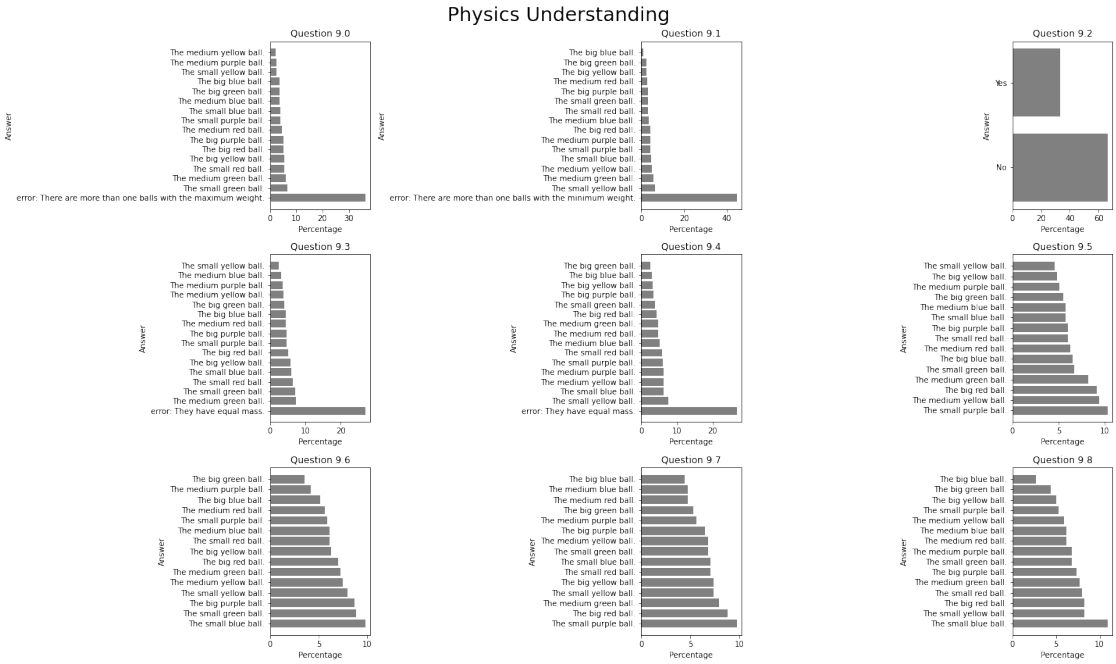

Figure A.28: Answer distribution of questions in the category of Physics Understanding (level 6). Questions 9.9 has equiprobable distribution of the answers, while 9.10 has a 1/3 chance for category 0, and 2/9 for all the others.

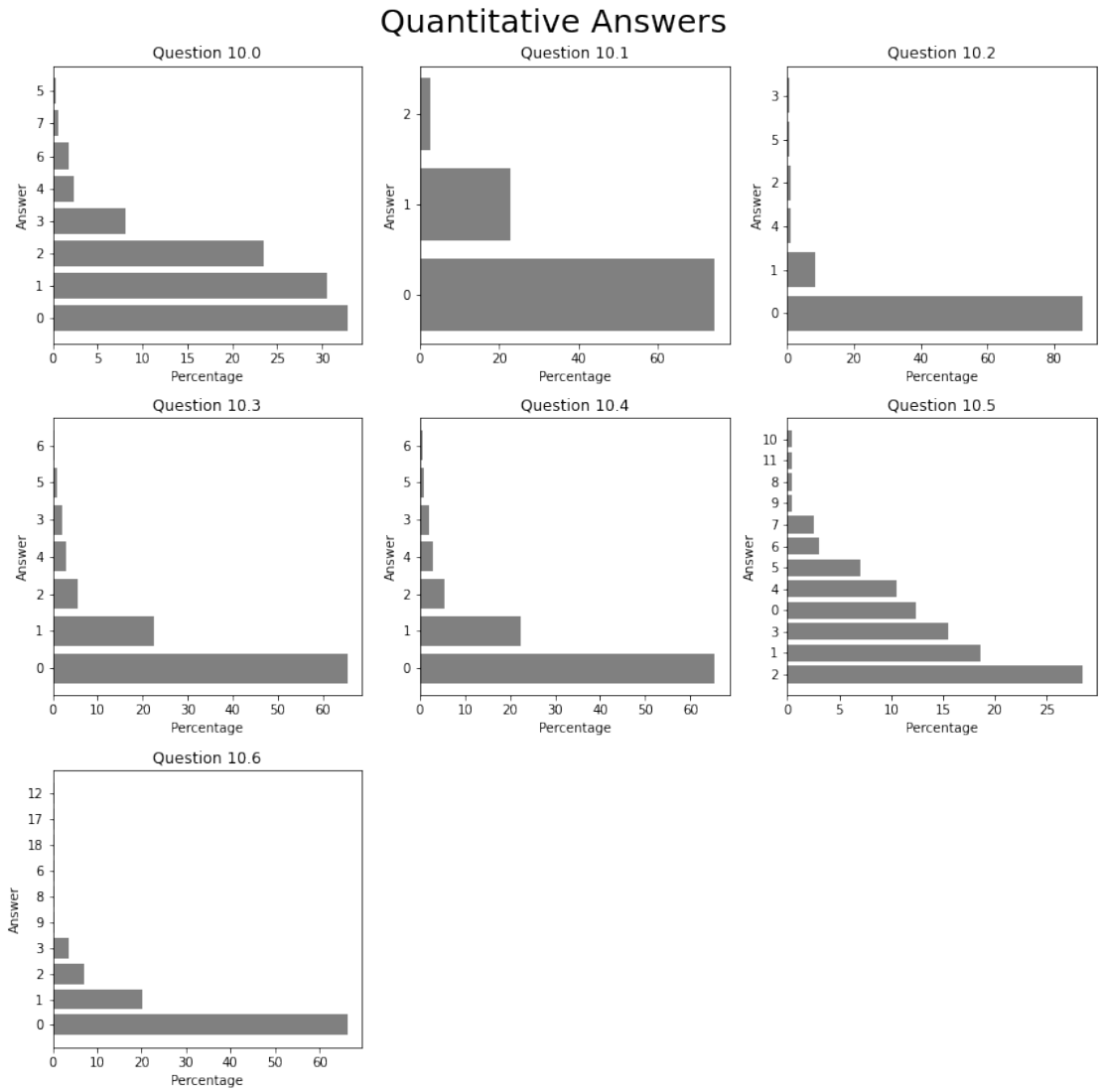

Figure A.29: Answer distribution of questions in the category of Quantitative Answers (level 6)

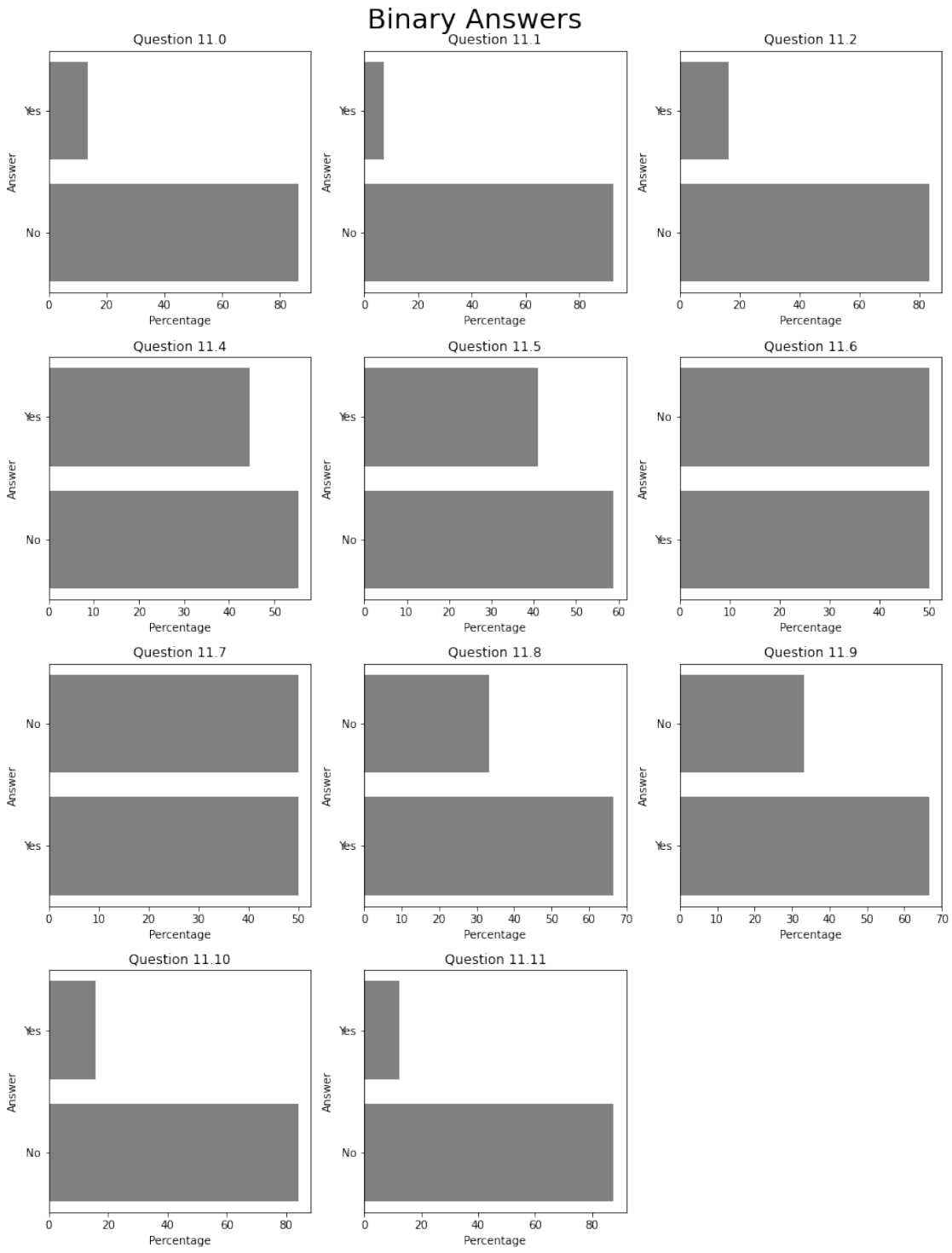

Figure A.30: Answer distribution of questions in the category of Binary Answers (level 6)

A.12.4 Answer distributions of simulation of level *8*

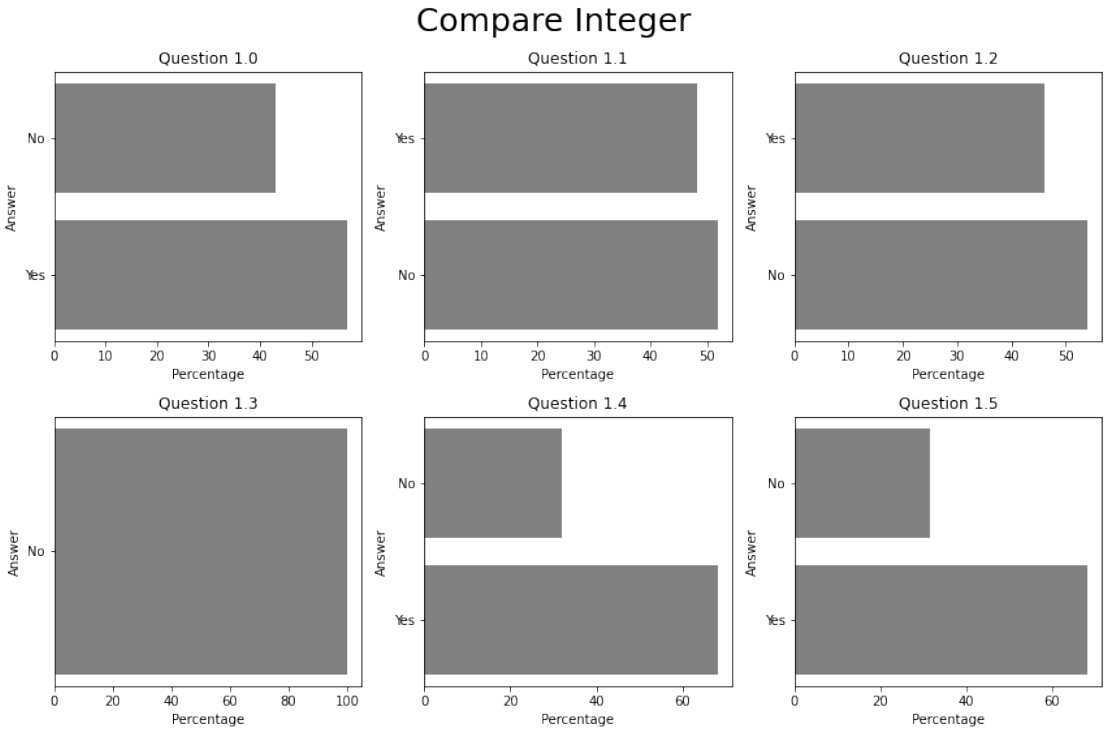

Figure A.31: Answer distribution of questions in the category of Compare Integer (level 8)

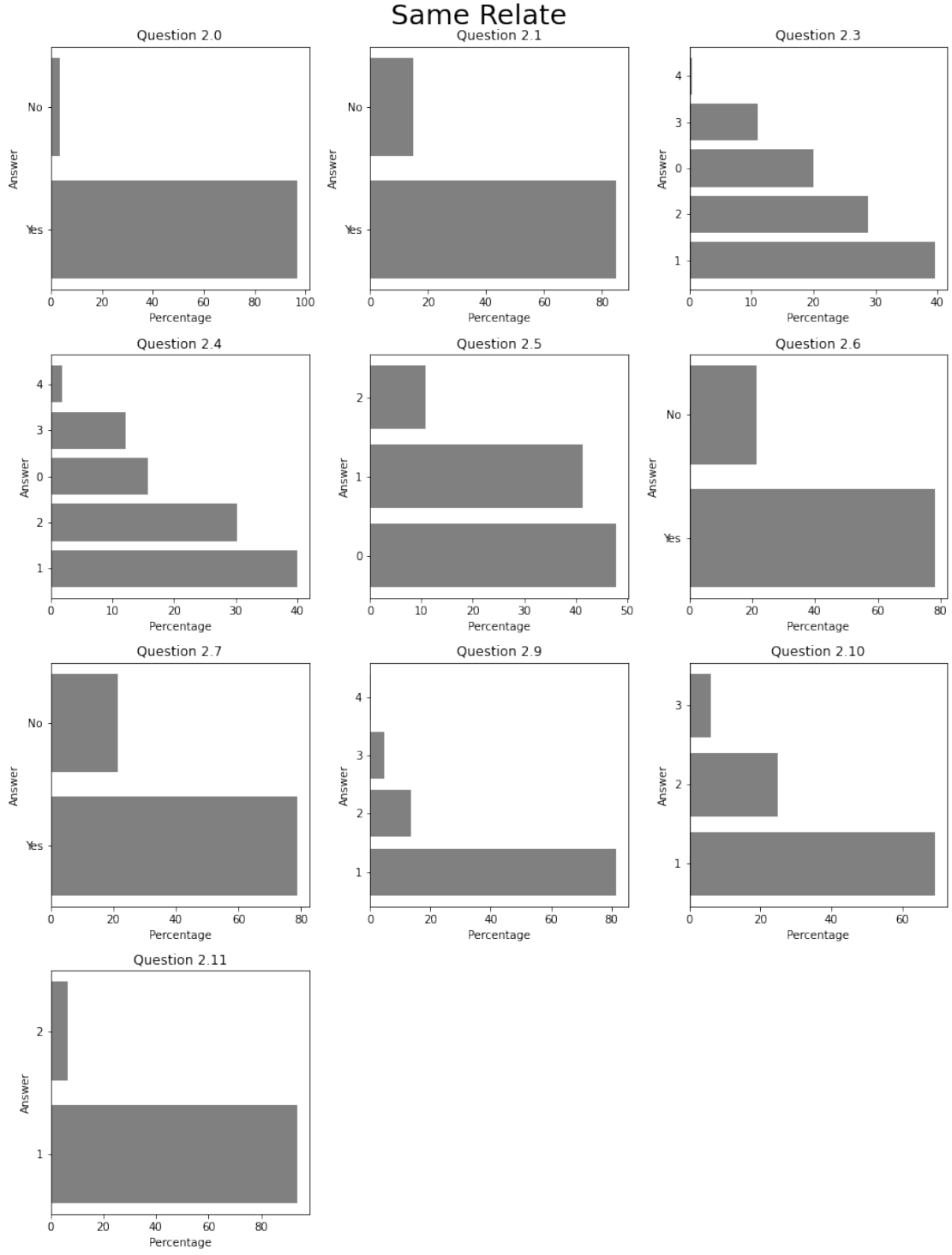

Figure A.32: Answer distribution of questions in the category of Same Relate (level 8)

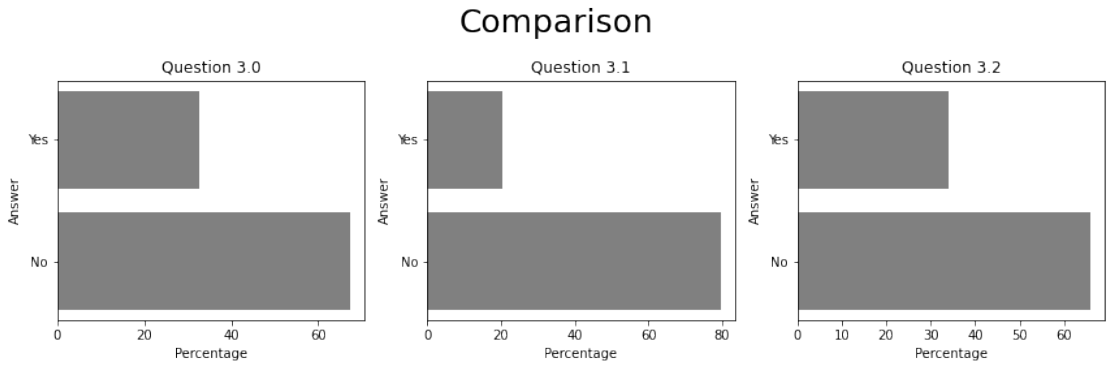

Figure A.33: Answer distribution of questions in the category of Comparison (level 8)

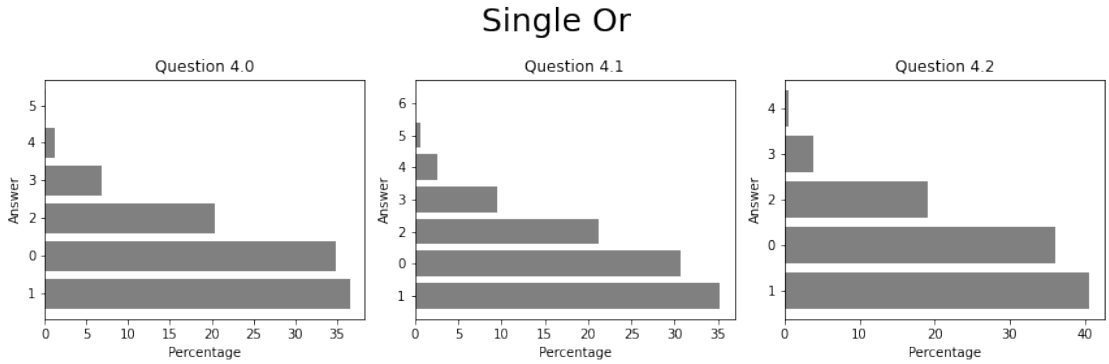

Figure A.34: Answer distribution of questions in the category of Single-Or (level 8)

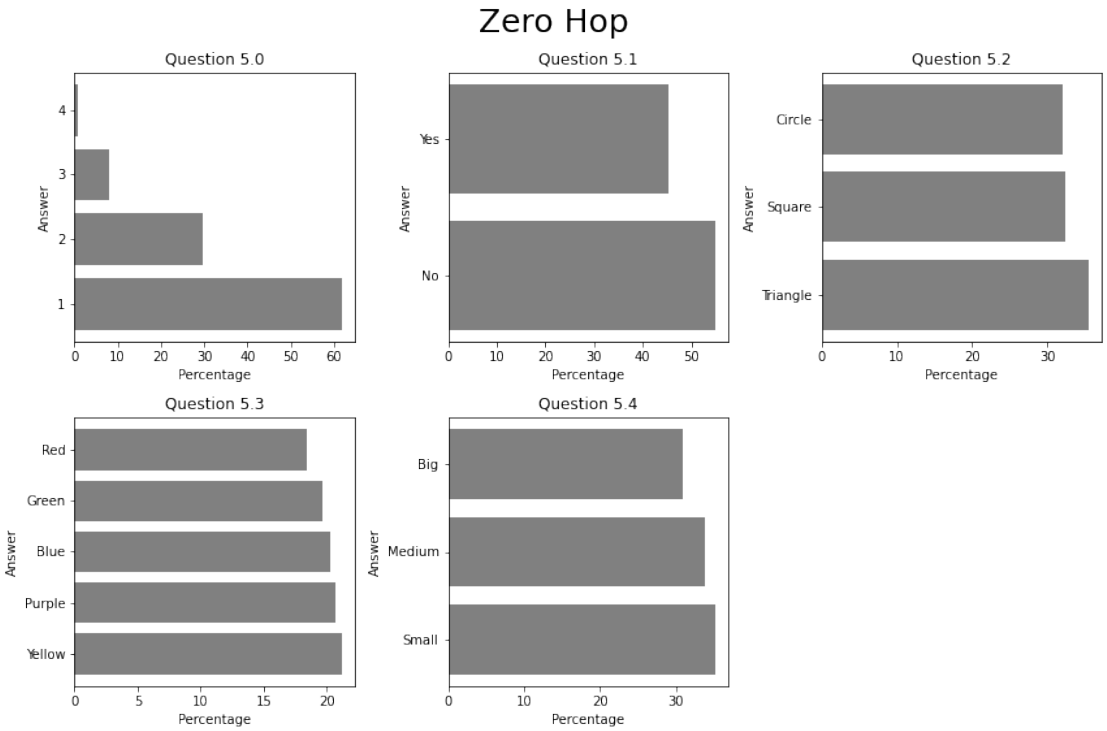

Figure A.35: Answer distribution of questions in the category of Zero Hop (level 8)

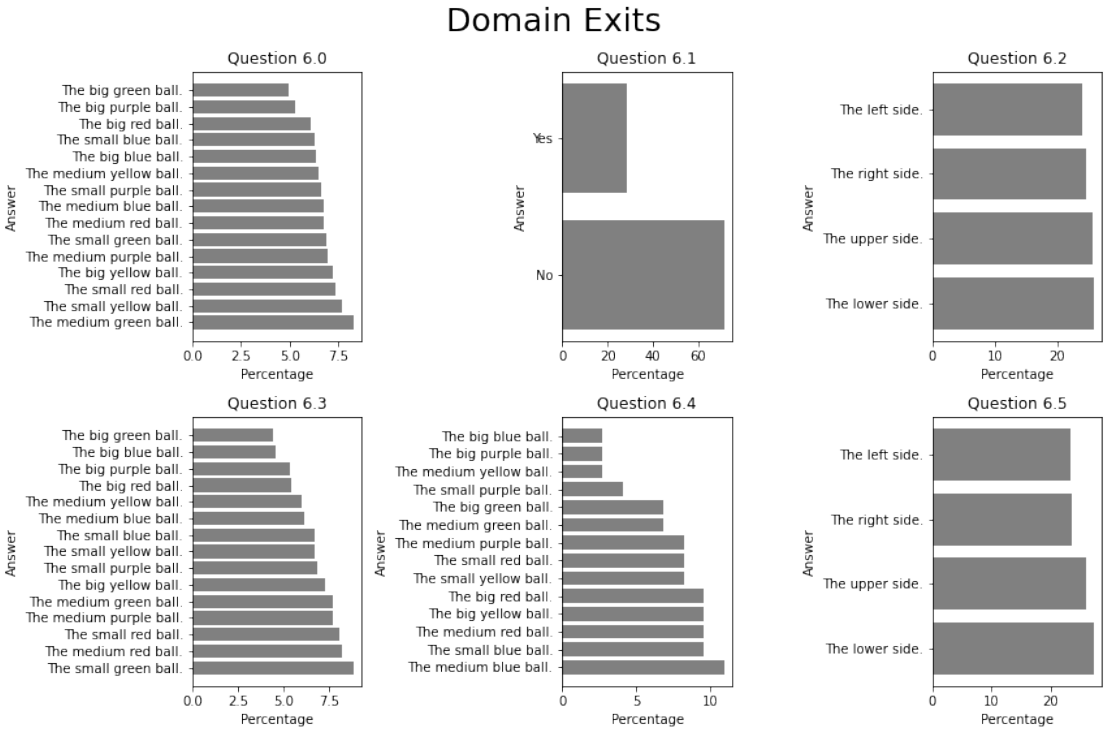

Figure A.36: Answer distribution of questions in the category of Domain Exits (level 8)

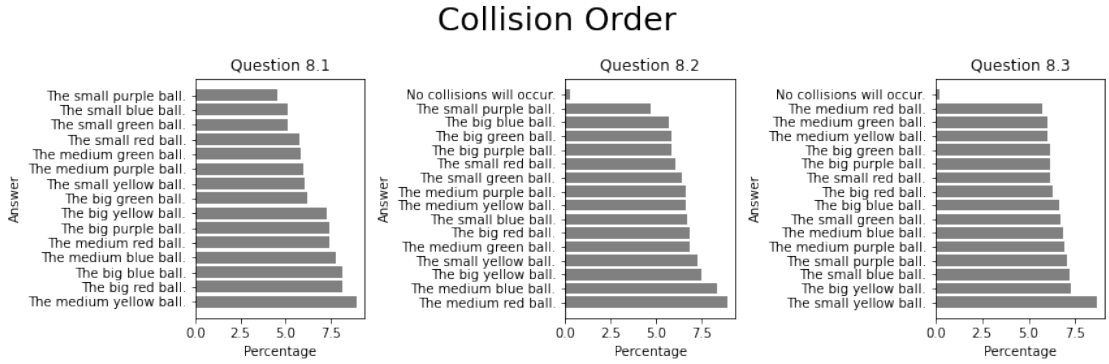

Figure A.37: Answer distribution of questions in the category of Collision Order (level 8)

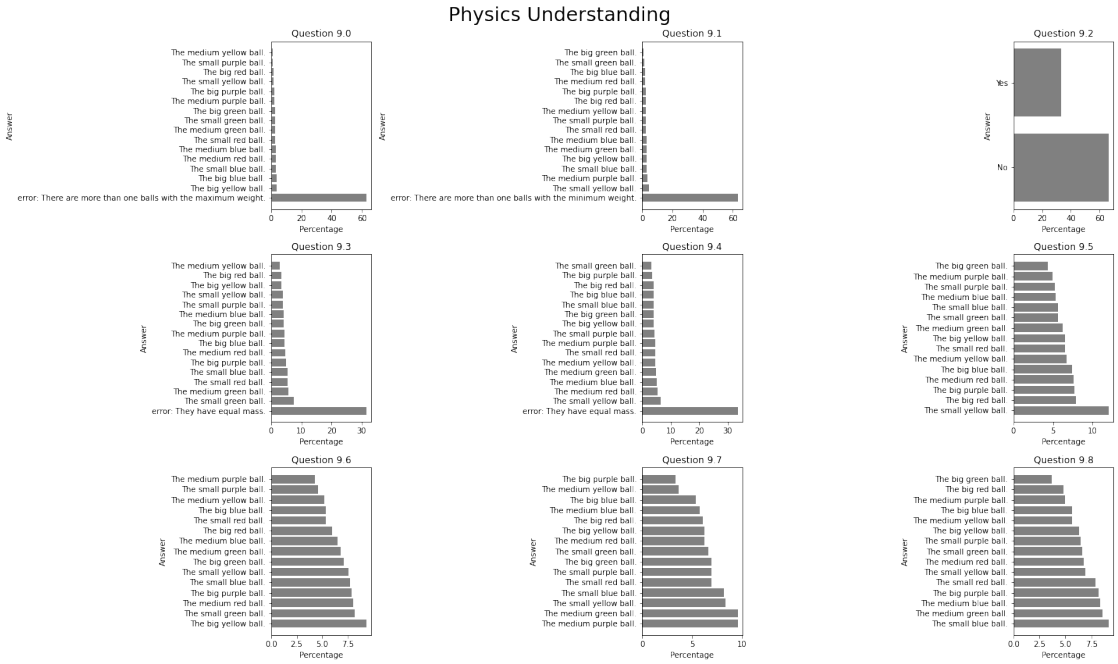

Figure A.38: Answer distribution of questions in the category of Physics Understanding (level 8). Questions 9.9 has equiprobable distribution of the answers, while 9.10 has a 1/3 chance for category 0, and 2/9 for all the others.

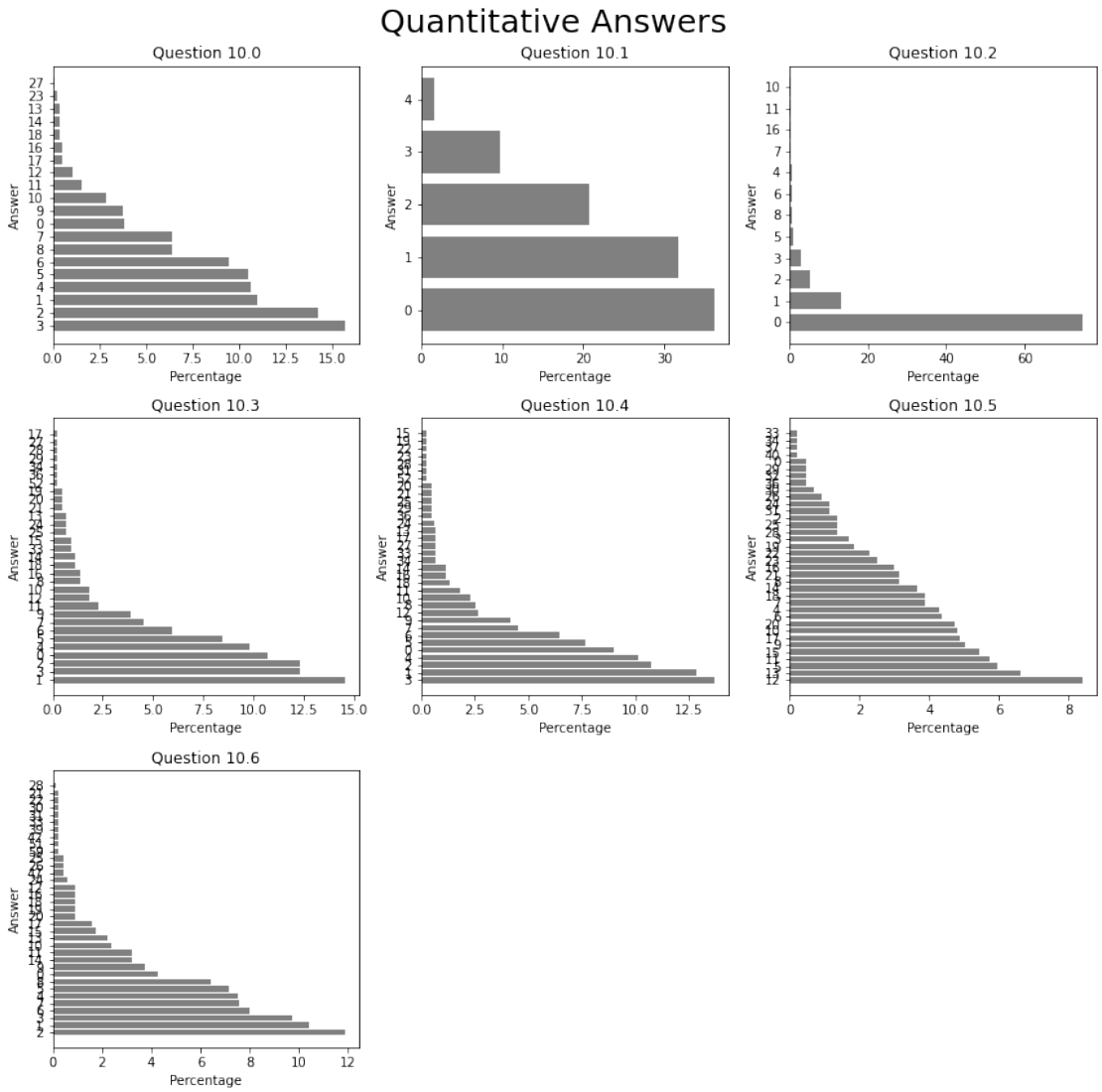

Figure A.39: Answer distribution of questions in the category of Quantitative Answers (level 8)

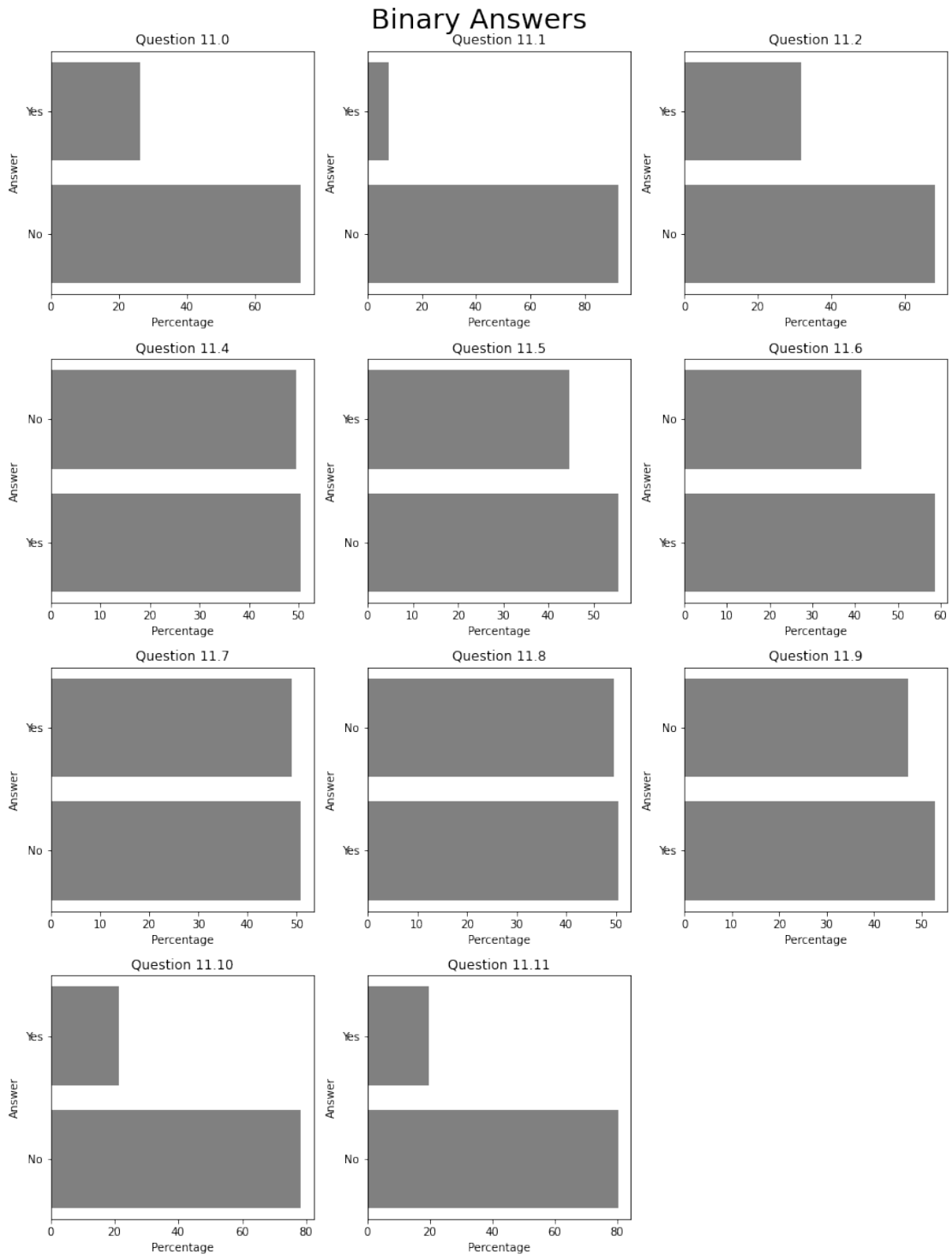

Figure A.40: Answer distribution of questions in the category of Binary Answers (level 8)

A.13 Appendix 3: Propagation Network Fundamentals

Similarly to most graph networks, there are a number of objects (O), and relations (R) that "connect" those objects together:

$$O = o_{i=1,...|O|}, R = r_{k=1,...|R|} \tag{A.1}$$

In most cases, the number of Relations is the square of the number of objects ($|R| = |O|^2$). Each object item, has several attributes:

$$o_i = \langle x_i, a_i^o, p_i \rangle, \text{ where } x_i = \langle q_i, \dot{q}_i \rangle \tag{A.2}$$

where q_i and $\dot{q}_i$ denote the position and its derivative (velocity), a_i^o denotes the attributes of the relation like mass/radius etc., and p_i any number of external forces on the object.

Each relation r_k is compromised of:

$$r_k = \langle u_k, v_k, a_k^r \rangle \tag{A.3}$$

where u_k is a receiver, v_k a sender and a_k^r the type of that relation.

The macroscopic aim of PropNet, is to learn the function ϕ that will take as input the current interaction graph G_t, and output the graph at the next timestep G_{t+1}:

$$G_{t+1} = \phi\left(G_t\right) \tag{A.4}$$

In the case of an Interaction Network, the characteristics of the objects are propagated towards the next timestep in the following form:

$$e_{k,t} = f_R\left(o_{uk,t} o_{vk,t}, a_k^r\right), k = 1...|R| \tag{A.5}$$

$$\hat{o}_{i,t+1} = f_O\left(o_{i,t}, \sum_{k=N_i} e_{k,t}\right), i = 1...|O| \tag{A.6}$$

Despite its flexibilty and efficiency in modeling complex systems, it lacks accuracy in non-local information interpretation, such as the one that takes place in multibody collisions. To address the above issues, the Propagation Network adds an extra *propagation* step in its architecture, where the effects of a single relation are passed to all other relations. The number of these extra propagations L can be tuned, but a good balance between performance and accuracy has been found with around $L = 2 - 4$. These extra propagation steps, as used in PropNet, can be described by the algorithm below:

$$\begin{aligned}
&\text{Step 0 :} && h_{i,t}^0 = 0, i = 1, ...|O| \\
&\text{Step l= 1,...,L :} && e_{k,t}^l = f_R^l\left(o_{uk,t}, o_{vk,t}, a_k^r, h_{uk,t}^{l-1}, h_{vk,t}^{l-1}\right), k = 1...|R| \\
& && h_{i,t}^l = f_O^l\left(o_{i,t}, \sum_{k \in N_i} e_{k,t}^l\right) \\
&\text{Output :} && \hat{o}_{i,t+1} = h_{i,t}^L, i = 1...|O|
\end{aligned} \tag{A.7}$$

Due to the variable number of items and attributes, as well as cases where items are not observable, a latent space is used for the object and relationship representations. Therefore, two encoders A.8, A.9 and two decoders A.10, A.11 are used:

$$c_{i,t}^{o} = f_{O}^{enc}(o_i, t) \tag{A.8}$$

$$c_{k,t}^{r} = f_{R}^{enc}(o_{uk,t}, o_{vk,t} a_k^r) \tag{A.9}$$

$$e_{k,t}^{l} = f_{R}^{l}(c_{k,t}^{r}, h_{uk,t}^{l-1}, h_{vk,t}^{l-1}) \tag{A.10}$$

$$h_{i,t}^{l} = f_{O}^{l}\left(c_{i,t}^{o}, \sum_{k \in N_i} e_{k,t}^{l}, h_{i,t}^{l-1}\right) \tag{A.11}$$

A Propagation Network receives as input a set of attributes and relations regarding some items, and by training a Graph Network Interpreter, it outputs the predicted attributes and relations, for the next timestep. This, compared to a sequential network (like an LSTM) has the added benefit of being able to instantly propagate forces and effects between objects. Such properties, are useful when simulating collisions between many objects e.g. Newton's Cradle.

A Propagation Network receives a number of previous frames (as well as the current), and aims to learn to predict the attributes (positions) of the new objects, as well as the potential events that will take place.

In the pre-formatted form, as presented in CLEVRER [54], it takes 3 subsequent frames, and aims to produce the next one. Instead of using just 2 frames, this gives us two extra properties:

- The simulated first derivative (velocity) is more accurately predicted, since it is now computed based on 2nd degree accuracy, as the first derivative, in first order of accuracy is:

  $$u = \dot{x} = \frac{1}{\Delta t}\left(x_i - x_{i-1}\right) \tag{A.12}$$

  While in second order of accuracy, it is:

  $$u = \dot{x} = \frac{1}{\Delta t}\left(\frac{3}{2}x_i - 2x_{i-1} + \frac{1}{2}x_{i-2}\right) \tag{A.13}$$

- Apart from the first derivative of the position (velocity) we can also determine the second derivative of the position (acceleration), since:

  $$a = \ddot{x} = \frac{1}{\Delta t^2}\left(2x_i - x_{i-1} + 2x_{i-2}\right) \tag{A.14}$$

Ideally, by knowing the acceleration of each object, as well as its velocity, one could estimate its future position with higher accuracy, as well as quantify the forces acted upon the object (as $F = ma$).

B References

[1] Atish Agarwala, Abhimanyu Das, Rina Panigrahy, and Qiuyi Zhang. Learning the gravitational force law and other analytic functions, 2020. arXiv:2005.07724.

[2] Gunjan Aggarwal and Devi Parikh. Neuro-symbolic generative art: A preliminary study, 2020. arXiv:2007.02171.

[3] Anil Ananthaswamy. Ai's next big leap. 10 2020. URL: https://knowablemagazine.org/article/technology/2020/what-is-neurosymbolic-ai.

[4] Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *CoRR*, abs/1612.00222, 2016. URL: http://arxiv.org/abs/1612.00222, arXiv:1612.00222.

[5] Long Chen, Xin Yan, Jun Xiao, Hanwang Zhang, Shiliang Pu, and Yueting Zhuang. Counterfactual samples synthesizing for robust visual question answering, 2020. arXiv:2003.06576.

[6] David Demeter and Doug Downey. Just add functions: A neural-symbolic language model, 2019. arXiv:1912.05421.

[7] Ben Dickson. What happens when you combine neural networks and rule-based ai? 2019. URL: https://bdtechtalks.com/2019/06/05/mit-ibm-hybrid-ai/.

[8] Ben Dickson. What is symbolic artificial intelligence? 2019. URL: https://bdtechtalks.com/2019/11/18/what-is-symbolic-artificial-intelligence/.

[9] Mark D. Feblowitz, O. Hassanzadeh, M. Katz, Shirin Sohrabi, Kavitha Srinivas, and O. Udrea. Ibm scenario planning advisor: A neuro-symbolic erm solution. 2020.

[10] Reuben Feinman and Brenden M. Lake. Learning task-general representations with generative neuro-symbolic modeling, 2021. arXiv:2006.14448.

[11] Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23:298–305, 01 1973. doi:10.21136/CMJ.1973.101168.

[12] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with lstm. 2:850–855 vol.2, 1999. doi:10.1049/cp:19991218.

[13] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. URL: http://arxiv.org/abs/1504.08083, arXiv:1504.08083.

[14] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. URL: http://arxiv.org/abs/1311.2524, arXiv:1311.2524.

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

[16] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. arXiv:1406.2661.

[17] Google. Google colaboratory, 2017. URL: https://colab.research.google.com/notebooks/welcome.ipynb.

[18] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering, 2017. arXiv:1612.00837.

[20] S. Hassantabar. Visual question answering : Datasets , methods , challenges and oppurtunities. 2018.

[21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. URL: http://arxiv.org/abs/1703.06870, arXiv:1703.06870.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL: http://arxiv.org/abs/1512.03385, arXiv:1512.03385.

[23] Drew A. Hudson and Christopher D. Manning. Learning by abstraction: The neural state machine, 2019. arXiv:1907.03950.

[24] Raban Iten, Tony Metger, Henrik Wilming, Lídia del Rio, and Renato Renner. Discovering physical concepts with neural networks. *Physical Review Letters*, 124(1), Jan 2020. URL: http://dx.doi.org/10.1103/PhysRevLett.124.010508, doi:10.1103/physrevlett.124.010508.

[25] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890, 2016. URL: http://arxiv.org/abs/1612.06890, arXiv:1612.06890.

[26] Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *CoRR*, abs/1703.05192, 2017. URL: http://arxiv.org/abs/1703.05192, arXiv:1703.05192.

[27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL: http://arxiv.org/abs/1412.6980.

[28] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[29] Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L. Berg. Tvqa: Localized, compositional video question answering, 2019. arXiv:1809.01696.

[30] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, 10:707–710, 1965.

[31] Qing Li, Siyuan Huang, Yining Hong, Yixin Chen, Ying Nian Wu, and Song-Chun Zhu. Closed loop neural-symbolic learning via integrating neural perception, grammar parsing, and symbolic reasoning, 2020. arXiv:2006.06649.

[32] Yunzhu Li, Jiajun Wu, Jun-Yan Zhu, Joshua B. Tenenbaum, Antonio Torralba, and Russ Tedrake. Propagation networks for model-based control under partial observation. *CoRR*, abs/1809.11169, 2018. URL: http://arxiv.org/abs/1809.11169, arXiv:1809.11169.

[33] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL: http://arxiv.org/abs/1405.0312, arXiv:1405.0312.

[34] Shari Liu, Tomer Ullman, josh tenenbaum, and Elizabeth Spelke. Ten-month-old infants infer the value of goals from the costs of actions, 10 2017. doi:10.31234/osf.io/78qd4.

[35] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *CoRR*, abs/1904.12584, 2019. URL: http://arxiv.org/abs/1904.12584, arXiv:1904.12584.

[36] Michael Sarazen Meghan Han. Mit's josh tenenbaum on intuitive physics psychology in ai. 2018. URL: https://medium.com/syncedreview/mits-josh-tenenbaum-on-intuitive-physics-psychology-in-ai-99690db3480.

[37] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3367–3375, 2015. doi:10.1109/CVPR.2015.7298958.

[38] Emilio Parisotto, Abdel rahman Mohamed, Rishabh Singh, Lihong Li, Dengyong Zhou, and Pushmeet Kohli. Neuro-symbolic program synthesis, 2016. arXiv:1611.01855.

[39] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL: http://arxiv.org/abs/1506.01497, arXiv:1506.01497.

[40] Apollonius Rhodius. The argonautica. 3rd century BC. URL: https://www.gutenberg.org/files/830/830-h/830-h.htm.

[41] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi:10.1007/s11263-015-0816-y.

[42] Tanzila Saba. Recent advancement in cancer detection using machine learning: Systematic survey of decades, comparisons and challenges. *Journal of Infection and Public Health*, 13(9):1274 – 1289, 2020. URL: http://www.sciencedirect.com/science/article/pii/S1876034120305633, doi:https://doi.org/10.1016/j.jiph.2020.06.033.

[43] Theophile Sautory, Nuri Cingillioglu, and Alessandra Russo. Hyster: A hybrid spatio-temporal event reasoner, 2021. arXiv:2101.06644.

[44] Meet Shah, Xinlei Chen, Marcus Rohrbach, and Devi Parikh. Cycle-consistency for robust visual question answering, 2019. arXiv:1902.05660.

[45] Kevin J. Shih, Saurabh Singh, and Derek Hoiem. Where to look: Focus regions for visual question answering, 2016. arXiv:1511.07394.

[46] Robik Shrestha, Kushal Kafle, and Christopher Kanan. Answer them all! toward universal visual question answering models, 2019. arXiv:1903.00366.

[47] Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards VQA models that can read. *CoRR*, abs/1904.08920, 2019. URL: http://arxiv.org/abs/1904.08920, arXiv:1904.08920.

[48] G. G. Stokes. On the effect of internal friction of fluids on the motion of pendulums. 9:8–106 part.2, 1851.

[49] Erno Téglás, Edward Vul, Vittorio Girotto, Michel Gonzalez, Joshua Tenenbaum, and Luca Bonatti. Pure reasoning in 12-month-old infants as probabilistic inference. *Science (New York, N.Y.)*, 332:1054–9, 05 2011. doi:10.1126/science.1196404.

[50] Sjoerd van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions, 2018. arXiv:1802.10353.

[51] Andre Vellino. Artificial intelligence: The very idea: J. haugeland, (mit press, cambridge, ma, 1985); 287 pp. *Artificial Intelligence*, 29:349–353, 09 1986.

[52] Liming Wang, Jianbo Shi, Gang Song, and I-fan Shen. Object detection combining recognition and segmentation. In Yasushi Yagi, Sing Bing Kang, In So Kweon, and Hongbin Zha, editors, *Computer Vision – ACCV 2007*, pages 189–199, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[53] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*, abs/1609.08144, 2016. URL: http://arxiv.org/abs/1609.08144, arXiv:1609.08144.

[54] Kexin Yi, Chuang Gan, Yunzhu Li, Pushmeet Kohli, Jiajun Wu, A. Torralba, and J. Tenenbaum. Clevrer: Collision events for video representation and reasoning. *ArXiv*, abs/1910.01442, 2020.

[55] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B. Tenenbaum. Neural-symbolic VQA: disentangling reasoning from vision and language understanding. *CoRR*, abs/1810.02338, 2018. URL: http://arxiv.org/abs/1810.02338, arXiv:1810.02338.

[56] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *CoRR*, abs/1906.05113, 2019. URL: http://arxiv.org/abs/1906.05113, arXiv:1906.05113.