

Dr. Lego: AI-Driven Assessment Instrument for Analyzing Block-Based Codes

by

Nimra I Siddiqui

Submitted in Partial Fulfillment of the Requirements

for the Degree of

Master

of

Computing and Information Systems

YOUNGSTOWN STATE UNIVERSITY

May, 2024

Dr. Lego: AI-Driven Assessment Instrument for Analyzing Block-Based Codes
Nimra I Siddiqui

I hereby release this thesis to the public. I understand that this **thesis** will be made available from the OhioLINK ETD Center and the Maag Library Circulation Desk for public access. I also authorize the University or other individuals to make copies of this thesis as needed for scholarly research.

Signature:

Nimra Idris Siddiqui _____
Student Date

Approvals:

Abdu Arslanyilmaz, PhD _____
Thesis Advisor Date

Feng George Yu, PhD _____
Committee Member Date

Carrie Jackson, DEd, BCBA _____
Committee Member Date

Salvatore A. Sanders, PhD, Dean, College of Graduate Studies Date

Acknowledgement

I extend my deepest gratitude to Dr. Abdu Arslanyilmaz for graciously inviting me to join his esteemed research team. His unwavering guidance, unwavering support, and unparalleled expertise have been instrumental in shaping my academic and professional journey. Dr. Arslanyilmaz's generosity in sharing his wealth of knowledge and experience has left an indelible mark on my development, both intellectually and personally. I am profoundly grateful for the investment of his time and effort in nurturing my growth and for the invaluable opportunity to contribute to his groundbreaking research. His mentorship has been a beacon of light, imbued with kindness, and has made this journey immensely rewarding.

Furthermore, I would like to express my heartfelt appreciation to Dr. Feng George Yu and Dr. Carrie Jackson for their invaluable roles as committee members in my research endeavors. Their dedication to mentorship has been nothing short of inspiring. Their unwavering commitment to fostering academic excellence and their profound impact on students have motivated me throughout my academic pursuits. I consider myself incredibly fortunate to have had the privilege of learning from such distinguished mentors, and I will forever cherish their contributions to my intellectual and personal growth.

Last but certainly not least, I am indebted to the Almighty God for His boundless blessings, guidance, and grace. His divine presence has illuminated my path and bestowed upon me countless opportunities for growth and success. With His guidance, I am empowered to tread the path of righteousness and fulfill my aspirations.

Special Acknowledgment and Disclaimer

This material is based upon work supported by the National Science Foundation under Award Number (FAIN) 2031427. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Abstract

The field of coding education is rapidly evolving, with emerging technologies playing a pivotal role in transforming traditional learning methodologies. This thesis introduces Dr. Lego, an innovative framework designed to revolutionize the assessment and understanding of block-based coding through the integration of sophisticated deep learning models. Dr. Lego combines cutting-edge technologies such as MobileNetV3 (Howard, 2019), for visual recognition and BERT (Devlin et al., 2018), and XLNet (Yang et al., 2019) for natural language processing to offer a comprehensive approach to evaluating coding proficiency.

The research methodology involves the meticulous curation of a diverse dataset comprising projects from the LEGO SPIKE app (LEGO Education, 2022), ensuring that the models are subjected to a broad range of coding scenarios. Leveraging the dynamic educational environment provided by the LEGO SPIKE app (LEGO Education, 2022), Dr. Lego empowers users to design and implement various coding projects, fostering hands-on learning experiences.

This thesis delves into methodologies aimed at enhancing coding education by exploring model integration, data generation, and fine-tuning of pre-trained models. Dr. Lego not only evaluates coding proficiency but also provides cohesive and insightful feedback, enhancing the learning experience for users. The adaptability of the framework highlights its potential to shape the future of coding education, paving the way for a new era of interactive and engaging learning experiences.

Contents

1. Introduction:.....	1
1.1 Overview of Lego Spike app:	2
2. Dataset and Scoring.....	7
3. Model Architecture	11
3.1 MobileNetV3-Small.....	13
3.2 XLNet	14
3.3 BERT	16
4. Ensemble Model	17
5. Hyperparameter	17
6. Empirical Analysis.....	19
7. Conclusion and Future Scope	21
REFERENCES	22

List of Figures

Figure 1 Categorization of LEGO SPIKE Blocks.....	4
Figure 2 Lego Spike app.....	5
Figure 3 Variety of Lego Spike Projects within Unit: Invention Squad	6
Figure 4 Rain or Shine Project	8
Figure 5 Sample Dataset: Evaluation of Lessons in Specific Units, Scoring Coding Blocks Across Diverse Categories	10
Figure 6 Multi-Model Architecture for Code Analysis and Prediction.....	12
Figure 7 Visual and Textual Analysis Workflow for Code Interpretation.....	13
Figure 8 Dr. Lego in Action: Demonstrating Functionality in Jupyter Lab.....	18
Figure 9 : Training and Validation Loss Evolution Over Epochs	20

1. Introduction:

In the dynamic realm of coding education, the integration of advanced tools holds the promise of reshaping how programming skills and knowledge is assessed through computing artifacts developed by individuals.

In recent years, there has been a concerted effort to develop tools aimed at assessing the accuracy and sequence of coding skills through computational artifacts. One such tool is Dr. Scratch (Moreno-León, Robles, & Román-González, 2015), designed to evaluate coding projects developed using Scratch (Moreno-León, Robles, & Román-González, 2015),—a platform where students can create various projects encompassing games, interactive stories, and animations using coding blocks.

Despite the existence of tools like Dr. Scratch (Moreno-León, Robles, & Román-González, 2015), there remains a notable gap in the landscape of coding assessment tools. Specifically, there is a lack of tools tailored to assess the complexity and extent of codes created within the LEGO SPIKE app (LEGO Education, 2022). Lego Spike stands out as one of the most widely utilized visual block-based programming environments, seamlessly integrating software with physical hardware (LEGO Education, 2021). Rooted in the ethos of Lego Education's commitment to fostering innovation and curiosity, the LEGO SPIKE app serves as a versatile platform for educational coding, bridging the physical and digital realms with its dynamic features (LEGO Education, 2022). Students engage with Lego Spike by assembling different Lego blocks and incorporating various types of equipment such as motors, color sensors, and gear racks, facilitating the learning of programming while fostering an understanding of robotics in an engaging manner (LEGO Education, 2021).

Acknowledging the glaring absence of tools for students to effectively evaluate and benchmark their coding skills within the LEGO SPIKE app, there arises a critical need for a comprehensive solution that offers robust analysis and guidance. Dr. Lego emerges as a response to this pressing demand. Thus, this research endeavors to introduce such a tool, which will be referred to as Dr. Lego, a pioneering framework poised to enhance the exploration and comprehension of Lego Spike code through the integration of sophisticated deep learning models.

Dr. Lego novel assessment tool is designed specifically for evaluating the complexity and depth of codes created within the LEGO SPIKE app. Through a comprehensive analysis of coding projects developed within the Lego Spike environment, this tool aims to provide educators and students with valuable insights into coding proficiency and project

quality in seven categories: action, conditional, control, input, operator, data, and sequences.

With a focus on leveraging cutting-edge models such as MobileNetV3 (Howard, 2019), BERT (Devlin et al., 2018), and XLNet (Yang et al., 2019), Dr. Lego aims to provide users with comprehensive insights into their coding methodologies. The methodology of this research involves meticulously curating a diverse dataset derived from various projects across different units on the LEGO SPIKE app, ensuring a comprehensive evaluation of coding scenarios.

Beyond mere technological advancement, the primary objective of Dr. Lego is to contribute meaningfully to coding practice by offering a user-friendly assessment tool that delves into the intricacies of code development. Through cohesive and insightful feedback, Dr. Lego endeavors to enrich the learning experience, guiding users towards continuous improvement while addressing common pitfalls and fostering a culture of excellence in Lego Spike coding practices. By leveraging the interactive nature of Lego Spike, this research endeavors to enhance coding education and empower students to develop advanced programming and robotics skills in an engaging and immersive manner.

1.1 Overview of Lego Spike app:

The LEGO SPIKE app stands as a pioneering platform, meticulously curated to nurture hands-on learning experiences in coding and computational thinking (LEGO Education, 2022). At its core, the LEGO SPIKE app boasts a curriculum rich in real-world, project-based activities, fostering an environment where students can explore the realms of robotics and coding with enthusiasm (LEGO Education, 2022).

Central to Dr. Lego's functionality is the systematic categorization of the LEGO SPIKE app's blocks, a process spearheaded by this thesis's author. Through this categorization effort, blocks are thoughtfully organized based on their distinct functionalities, allowing for a clear and intuitive understanding of their roles within coding projects.

For instance, blocks related to Motor functions which performed the actions like "start motor", "stop motor", "setting speed and position", and the blocks under the term Movement which facilitate functions like "start moving", "stop moving", "setting movement speed and motor rotation" are all performing certain kind of movement-related action. Likewise, Light blocks enable actions like "turn on/off pixels, light", "write", "set power button light to certain color", and Sound blocks perform actions like "play sound", "start sound", "change or set volume or pitch", which are all related to some sort of action. Thus, all four block categories, Motors, Movement, Light, and Sound, are grouped under "Action" category since their blocks are designed to perform a certain

kind of action.

The term Events are categorized as “Conditions” because it consists of “when” blocks that necessitate certain conditions for the attached code to execute.

Control blocks are categorized as “Controls” since it has blocks like “if-then”, “else”, “repeat”, “wait”, “forever” which are essential for controlling behavior and implementing loops.

Sensor-related blocks (“input color/light,” “tilted,” and “sense pressure”) are categories as “Input” because they respond to specific inputs, sense colors, lights, pressures, and movements. Blocks.

Operators-related blocks (“and”, “or”, “not”, “+”, “-”, “*”, “/”, etc.) are categorized as “Operators” because they have blocks involving some sort of mathematical and logical operators.

Finally, Variables and My Blocks are classified under the category of "Data" because they allow users to store and manipulate data within their projects. In Variables, users can assign any necessary data values to facilitate the functionality of their programs, while My Blocks enable the creation of custom blocks tailored to specific project requirements, thereby enhancing project interactivity.

This systematic categorization, as shown in Figure 1, lays the foundation for a comprehensive evaluation of computing artifacts developed in LEGO SPIKE app after analyzing each block's functionality, enabling users to assess coding projects with precision and clarity.

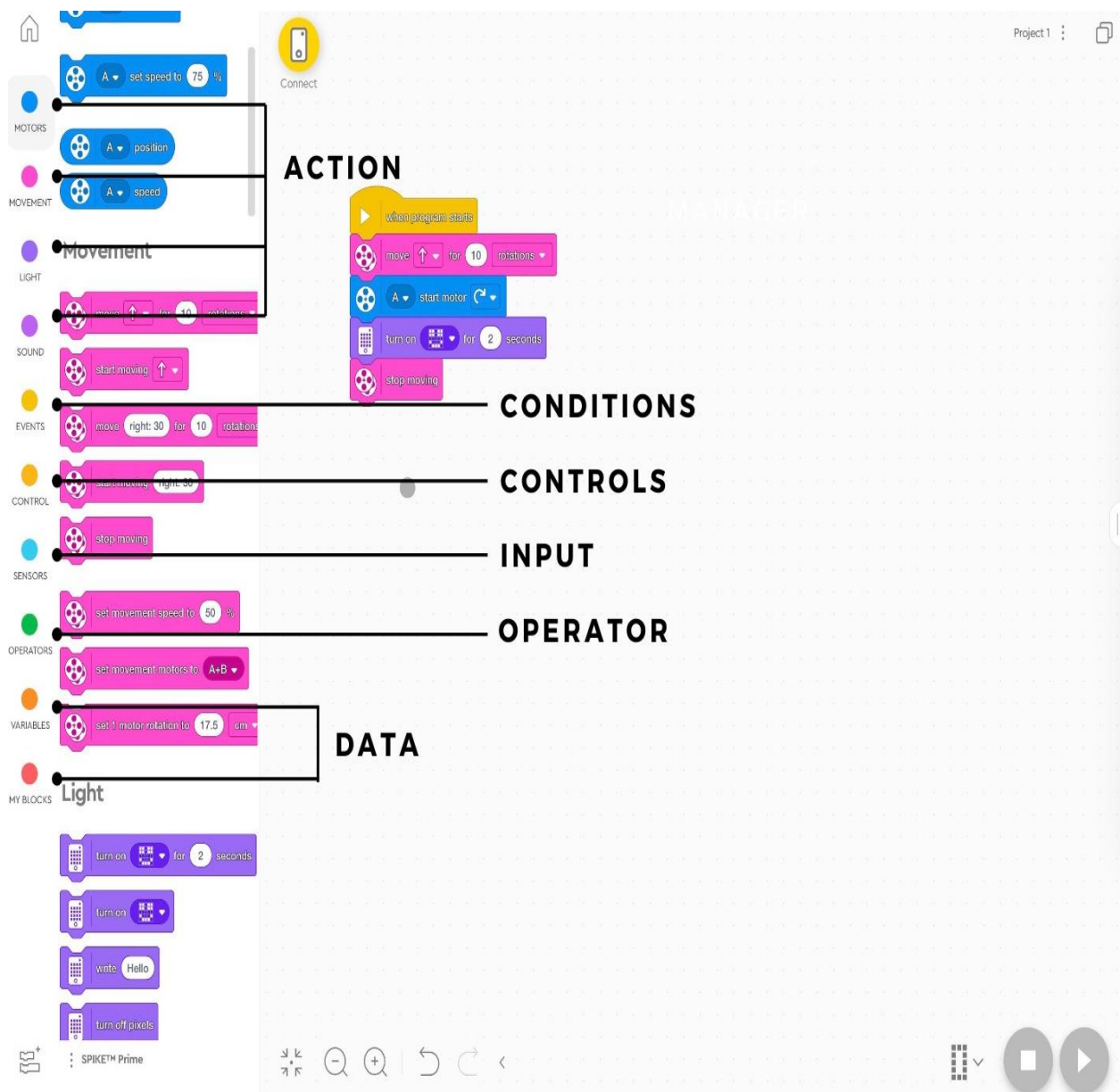


Figure 1 Categorization of LEGO SPIKE Blocks

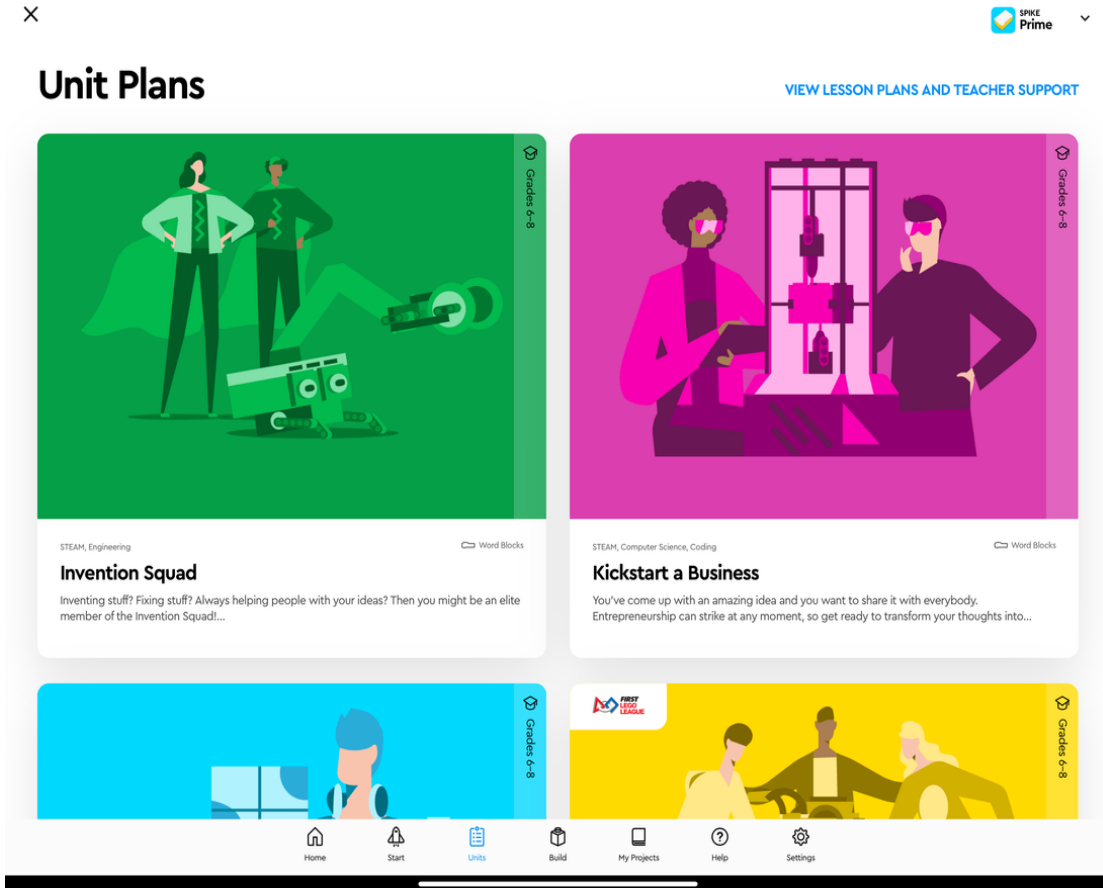


Figure 2 Lego Spike app

Complementing this explanation, the inclusion of visual aids—showcasing the LEGO SPIKE app interface and a diverse array of projects developed within its units—offers a tangible glimpse into the dynamic learning experiences facilitated by the platform. Figures 2 and 3 serve not only to illustrate the app's user-friendly interface but also to underscore the versatility and creativity inherent in Lego Spike projects, setting the stage for a comprehensive exploration of Dr. Lego—an AI-driven assessment tool poised to revolutionize the evaluation of block-based codes within the LEGO SPIKE app ecosystem.

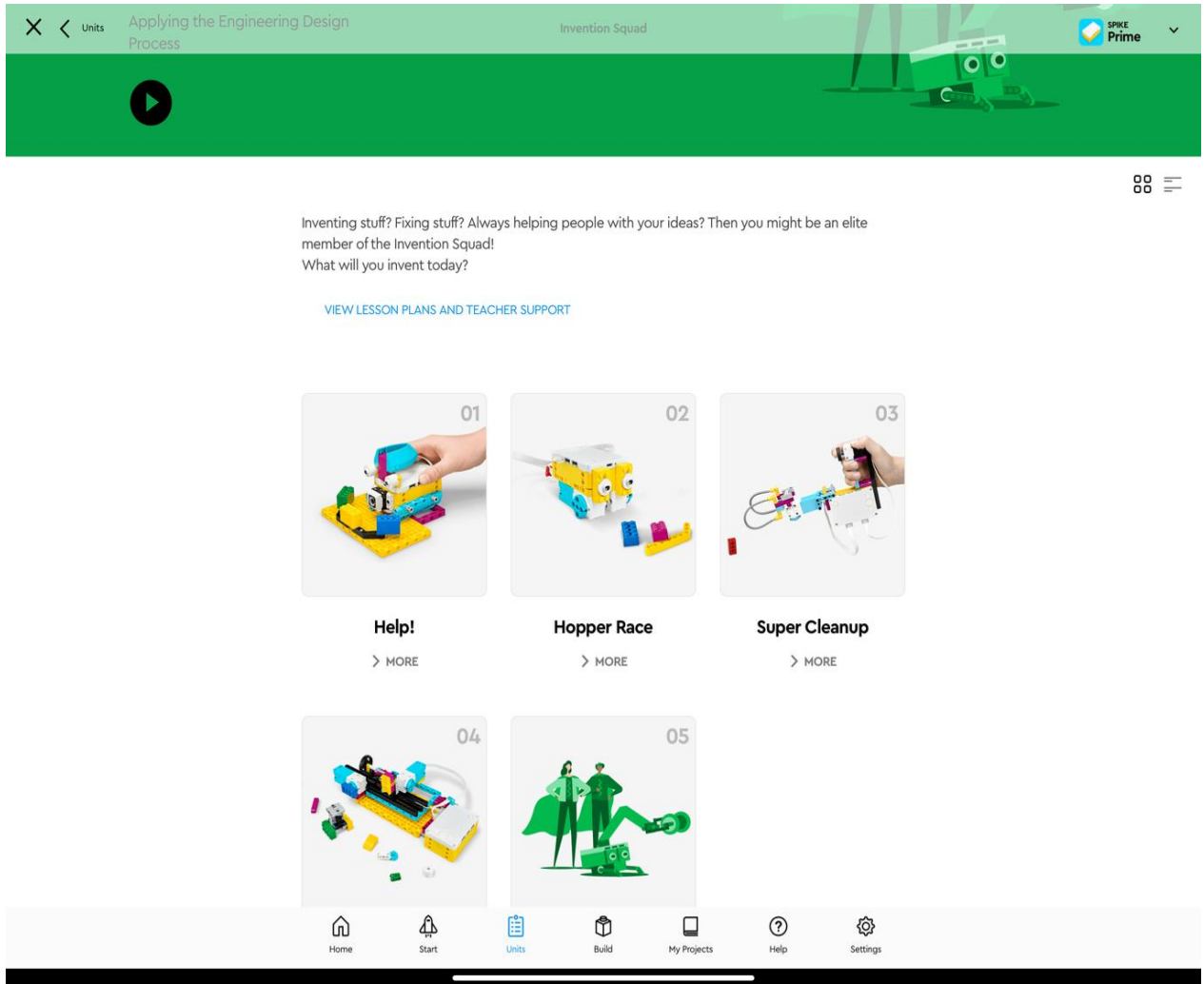


Figure 3 Variety of Lego Spike Projects within Unit: Invention Squad

2. Dataset and Scoring

The dataset generation process for the research involves a thorough examination of various instructional units within the Lego Spike educational framework (LEGO Education, 2022). These units encompass a diverse range of Lego Spike lessons, each contributing unique coding patterns and structures to the dataset (LEGO Education, 2022). For instance, in the "Invention Squad" unit (see Figure 2), students engage in projects like Help, Hopper Race, Super Cleanup, and Broken, where they invent, fix, or assist with various ideas. Similarly, the "Life Hacks" unit explores practical life hacks through projects such as Break Dance, Repeat 5 Times, Rain or Shine, Veggie Love, and Brain Game.

The "Competition Ready" Lego Spike unit prepares students for competitions with Lego Spike projects like Training Camp 1: Driving Around, Training Camp 2: Playing with Objects, Training Camp 3: Reacting to Lines, and Time for Upgrade. In the "Kickstart Business" Lego Spike unit, students delve into entrepreneurship details through projects like Place Your Order, Out of Order, Track Your Packages, and Keep It Safe.

One of the notable Lego Spike units, "Training Trackers," focuses on promoting health awareness by tracking steps and energy expenditure. Lego Spike projects like Stretch with Data, This is Uphill, Time for Squat Jumps, and Aim for It, encourage physical activity and fitness tracking.

Throughout this process, the analyzing of the blocks in these lessons helped to identify and categorize the previously defined computational thinking categories: Action, Conditional, Control, Input, Operator, Data, and Sequences. This systematic approach ensures the comprehensive coverage and accurate representation of coding elements within the LEGO SPIKE app.

This initial categorization provides the foundation for further analysis and scoring. Assigning weights to each term based on its frequency within the dataset ensures that the scoring system accurately reflects the prominence of each coding element.

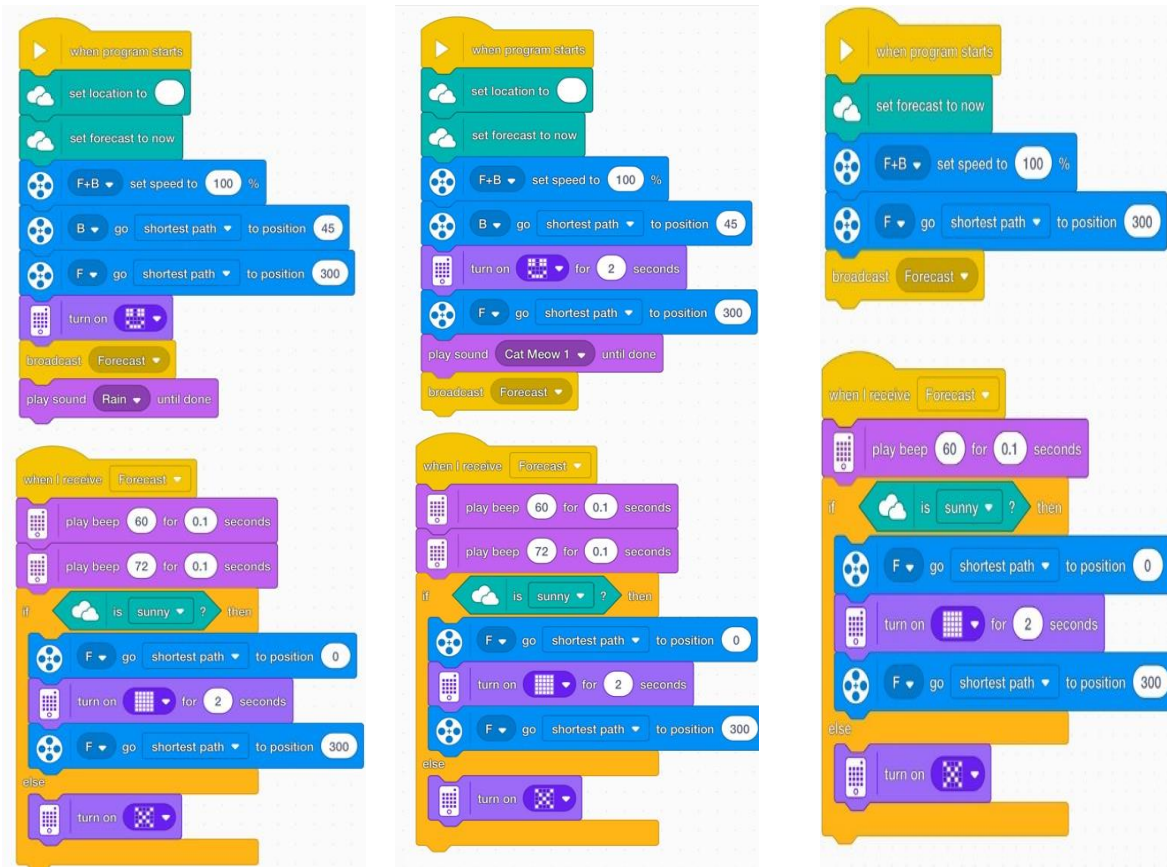
2.1 Scoring Lego Spike Projects

In the scoring methodology, the frequency of each block type determines its respective score. For example, if "Action" blocks appear a certain number of times, the score assigned is equal to that count. This principle applies uniformly to all other block categories, where scores are determined based on the number of unique block types utilized.

Furthermore, the sequence score is pivotal in evaluating the execution of coding blocks within a project. A sequence score of 1 signifies successful execution of the desired task

or completion of the project, while a score of 0 indicates deviations from the expected outcome or failure to accomplish the project's intended action. This scoring system ensures that the complexity and execution of coding projects are accurately assessed, providing valuable insights for both learners and educators.

To augment the effectiveness and diversity of the dataset, implementation of selective refinement techniques have done. This process encompasses shuffling, addition, and removal of code blocks to generate a diverse array of data samples, thereby enhancing the robustness and efficiency of the results.



Correct Sequence
(Figure a)

Sequence with Shuffled
Blocks (Figure b)

Sequence with Added and
Removed Blocks (Figure
c)

Figure 4 Rain or Shine Project

In the visual representation provided above for the project "Rain or Shine," Figure 4(a) illustrates the project with the correct output and all blocks arranged in a perfect sequence. Conversely, Figures 4(b) shows a program containing shuffling, while Figure 4(c) depict instances where the program undergoes shuffling, addition, and removal of blocks, resulting in sequences that are not accurately aligned. This deliberate variation in sequences aims to train the model with a comprehensive dataset, enabling it to predict outcomes for a wide range of projects effectively.

Throughout the model training phase, significant emphasis was placed on achieving a balanced representation of all terms within the dataset. This entailed ensuring a proportionate distribution of 0s and 1s for sequence scores, signifying unsuccessful and successful execution of coding blocks, respectively. Figures 4(a) and 4(b) demonstrate sequences with a score of 1, indicating successful program execution, whereas Figure 4(c) depicts a sequence with a score of 0, representing a program that failed to achieve the desired outcome. This approach ensures that the models are trained on a diverse array of Lego Spike coding scenarios, enhancing their capability to accurately evaluate various coding practices.

<u>Train Images</u>	Action	Conditional	Control	Input	Operator	Data	Sequences
Invention Squad							
Help!	7	2	6	6	0	0	1
Hopper Race	4	1	0	0	0	0	0
Super Cleanup	2	2	0	0	0	0	1
Broken	7	1	1	0	0	0	1
Kickstart a Business							
Place Your Order	7	1	1	0	0	0	1
Out of Order	18	2	4	2	0	0	1
Track Your Packages	5	2	1	0	0	0	1
Keep it safe	13	2	0	0	0	0	1
Keep it really safe!	14	1	3	1	0	0	0
Life Hacks							
Break Dance	6	4	4	0	0	0	1
Repeat 5 times	12	1	4	3	0	0	1
Rain or Shine	8	3	1	2	0	1	0
Wind Speed	4	1	1	1	0	1	1
Veggie Love	7	2	1	2	2	6	1
Brain Game	13	3	3	1	1	8	1
Competition Ready							
Training Camp1: Driving Around	8	4	4	2	1	0	1
Training Cam 2: Playing with Objects	11	3	3	1	0	0	1
Training Camp3: Reacting to Lines	8	2	3	2	0	0	1
The Guides Mission	19	1	3	4	2	0	1
Assembling an Advanced Driving Base	10	1	3	3	2	0	1
My code, our program	4	2	0	0	0	2	1
Time for an Upgrade	5	1	0	0	0	0	1
Training Trackers							
Stretch with Data	2	2	1	5	0	11	1
This is Uphill	4	2	2	1	1	3	1
Time for Squat Jumps	0	2	1	3	7	15	1
Watch Your Steps	2	3	5	4	2	11	1
Aim for it	0	3	6	11	4	18	1
Supplementary Lessons							
Pass the Brick	2	3	0	0	0	0	1
Ideas, the LEGO way!	12	2	0	0	0	0	1
What is this?	3	1	1	0	0	0	1
Going the Distance	3	1	0	0	0	0	1
Goal!	4	2	1	0	0	0	1

Figure 5 Sample Dataset: Evaluation of Lessons in Specific Units, Scoring Coding Blocks Across Diverse Categories

3. Model Architecture

This section explains the workings of the proposed model architecture tailored specifically for classifying text-embedded images into categories of "Having Sequence" or "Not Having Sequences." The ensemble approach evaluates projects across aforementioned six categories (Action, Conditional, Control, Input, Operator, and Data) with binary classification assigned based on the successful execution of sequences for Lego Spike projects like Break Dance, Place your Order, Assembling an Advanced Driving Base, or Wind Speed (LEGO Education, 2022) and so on. In terms of the seventh aforementioned category (Sequences), a sequence is deemed successful if it achieves the task of executing the project successfully. For instance, in projects like "Break Dance," the desired task involves programming a robot to dance to a specific beat, get up, and move in a certain manner. For "Veggie Love," the objective is to assemble the coding blocks in a way that simulates the growth of a vegetable, determining the optimal amount of water it needs weekly. Each project has its unique set of requirements, and the goal is to ensure that these requirements are met through the coding sequences. For example, in "Veggie Love," students must determine if the programmed actions accurately represent the watering needs of the virtual vegetable, ensuring it receives neither too much nor too little water for optimal growth.

If a sequence successfully completes the project task, it receives a score of 1; otherwise, it gets a score of 0 on the Sequences category. Considering the complex nature of the data, which involves categorizing coding blocks into seven different terms and analyzing textual data for nuances, capturing both visual and textual features become essential. The proposed ensemble model achieves this by combining individual models that specialize in understanding textual blocks' color and categorizing them, while also focusing on the sequential arrangement of coding blocks. This is accomplished through the use of convolutional neural networks (CNN) and pretrained transformer models (Smith, 2024), which have shown impressive results on established datasets.

MobileNetV3-Small is a specialized neural network model renowned for its precision and efficiency in image recognition tasks. Specifically designed to dissect visual elements with acute precision (Howard, 2019), MobileNetV3-Small plays a pivotal role in the model's ability to analyze code. As depicted in Figures 6 and 7, this model excels in recognizing and interpreting visual components of code, contributing significantly to the overall accuracy and effectiveness of the system.

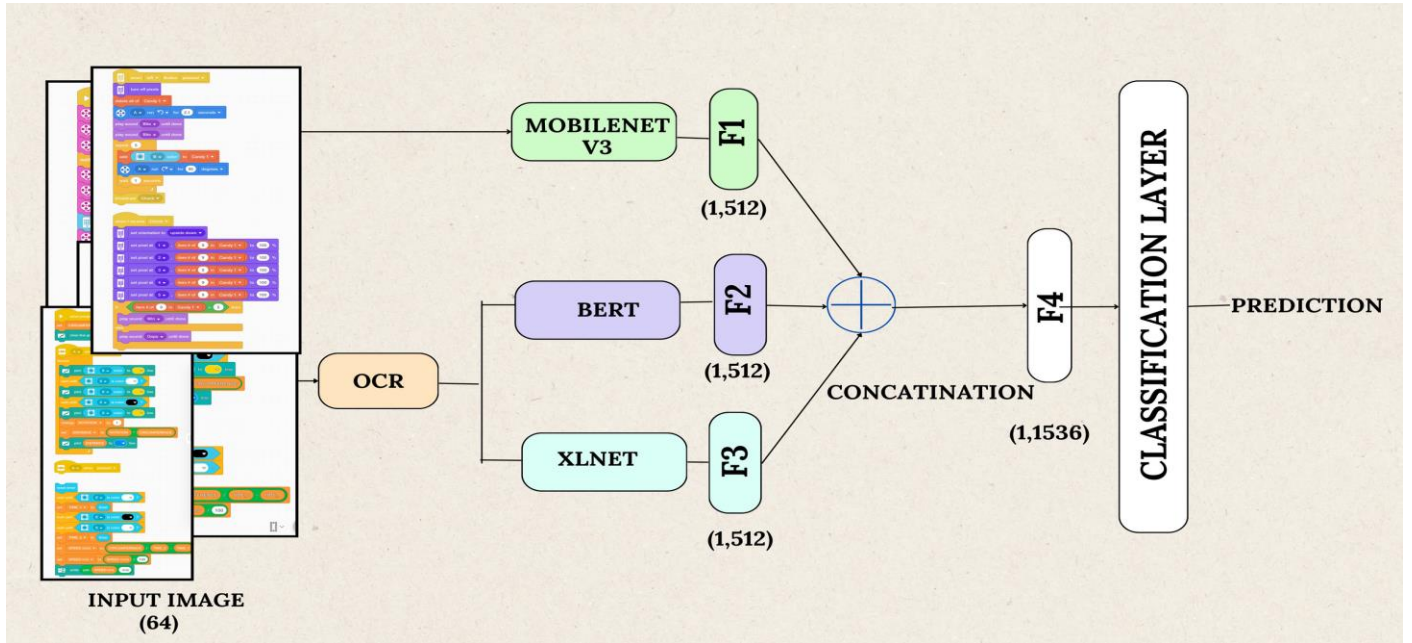


Figure 6 Multi-Model Architecture for Code Analysis and Prediction

The two cutting-edge models, BERT (Devlin et al., 2018) and XLNet (Yang et al., 2019) are renowned for their prowess in language understanding. BERT excels in capturing bidirectional contexts (Devlin et al., 2018), while XLNet overcomes limitations by considering all permutations of the factorization order (Yang et al., 2019). By leveraging these models, analysis of textual nuances with remarkable accuracy is achieved, facilitating comprehensive evaluation of a wide range of projects as shown in Figures 6 and 7. Together, they empower the composite model to effectively assess and score computing projects developed in the LEGO SPIKE app on the aforementioned seven computing skill categories.

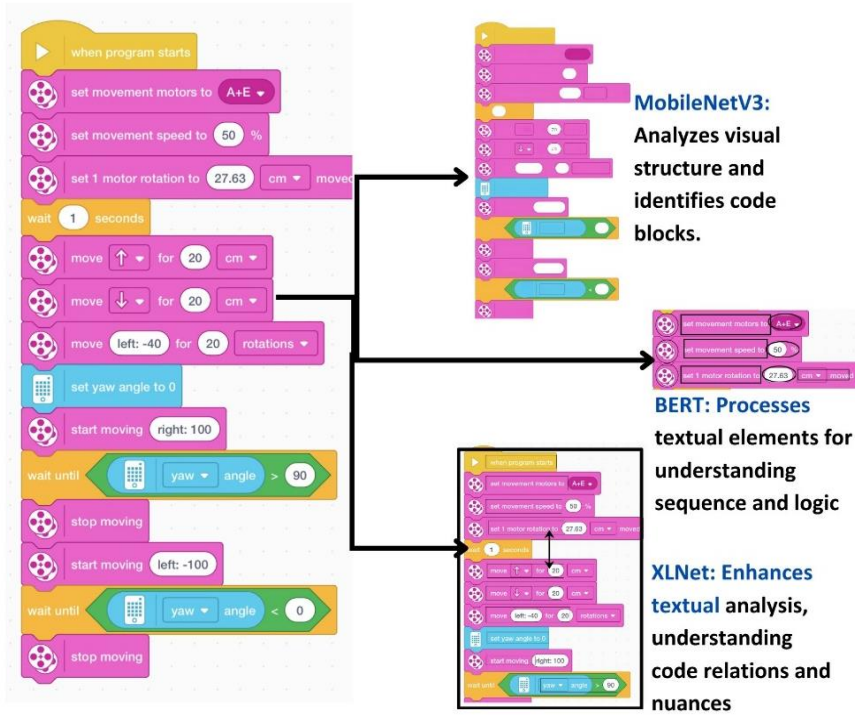


Figure 7 Visual and Textual Analysis Workflow for Code Interpretation

3.1 MobileNetV3-Small

This research harnessed the capabilities of MobileNetV3-Small, a specialized model designed specifically for high-accuracy and efficient neural network applications in on-device computer vision (Howard, 2019). The selection of MobileNetV3-Small is justified by its compact yet powerful design, which represents the forefront of state-of-the-art capabilities in computer vision (mmpretrain, 2020). This model is particularly appropriate for the research due to its innovative architectural advancements, including automated search techniques, which ensure optimal performance (Howard, 2019). This compact yet powerful model represents the next generation, pushing the boundaries of state-of-the-art capabilities through automated search and innovative architectural advancements (Howard, 2019).

MobileNetV3-Small introduces the innovative concept of “Inverted Residual Blocks” (IRB), strategically placing compression and expansion stages at the commencement and culmination of each bottleneck block. (mmpretrain, 2020). Particularly adept in tasks such as object detection, semantic segmentation, and image classification, MobileNetV3-Small shines in the realm of image recognition (mmpretrain, 2020). By isolating visual aspects of code, it enriches the input feature space through a 1*1 Convolution, followed

by Depthwise Convolution 3*3, extracting denser features in the process. This architectural design proves highly effective for precise classification tasks (mmpretrain, 2020).

The effectiveness of MobileNetV3-Small in identifying and classifying Lego Spike code blocks based on different categorizes. The results of tests on Table 1 provide compelling evidence supporting its pivotal role in the ensemble model.

For example, in the “Hopper Race” project, MobileNetV3-Small accurately identified and classified Action blocks responsible for initiating movements or actions within the robot. These blocks typically include commands such as “start motor” or “move forward.” By analyzing color, shape, and associated images, MobileNetV3-Small effectively distinguished these Action blocks from other types.

Similarly, in the “Rain or Shine” project, MobileNetV3-Small successfully categorized Control blocks, which dictate the flow and execution of code sequences. These blocks, such as “if-else” or “repeat,” are crucial for implementing conditional statements or loops. Through its analysis of color and shape, MobileNetV3-Small reliably identified these Control blocks, contributing to the accurate classification of the project’s code structure.

Overall, the test results on Table 1 demonstrated MobileNetV3-Small’s proficiency in code block identification across various Lego Spike projects. By leveraging its capability to analyze color, shape, and associated images, MobileNetV3-Small significantly enhanced the accuracy of the ensemble model’s analysis and classification of diverse code blocks. This underscores its indispensable role in the methodology, ultimately contributing to the effectiveness of the approach in evaluating coding projects. The feature vector output by MobileNetV3 is represented as:

$$f1 \in R^{1*64} \dots\dots\dots(1)$$

This output encapsulates crucial visual information essential for comprehensive analysis and precise classification of code blocks, contributing substantially to the efficacy of the overall model architecture.

3.2 XLNet

XLNet, distinguished by its sophisticated autoregressive pre-training technique, excels in comprehending bidirectional contexts by considering all permutations of the factorization order (Zihang Dai, 2019). Put simply, it’s a highly effective autoregressive method

ideally suited for handling extensive textual documents (Yang et al., 2019). In the research, XLNet autoregressive pre-training technique, XLNet (Yang et al., 2019) demonstrated exceptional proficiency in understanding bidirectional contexts, crucial for accurately interpreting the nuances of coding instructions.

For instance, in the analysis of the “Keep it Really Safe!” project, XLNet showcased its ability to meticulously analyze textual data to ensure proper code sequencing and task adherence. This project involved coding blocks related to Control and Conditional categories, which dictate the logical flow and decision-making processes within the program. XLNet effectively identified and classified these blocks, guiding students towards optimal programming practices and away from suboptimal approaches.

The literature reports highlight XLNet’s superior performance in natural language processing tasks. The study by (Zihang Dai, 2019) has demonstrated XLNet’s effectiveness across a range of language understanding tasks, including sentiment analysis, question answering, and document ranking.

XLNet utilizes its ability to consider various arrangements of text to process information obtained from the Optical Character Recognition (OCR) Engine. This involves systematically analyzing the text to capture intricate linguistic nuances and contextual dependencies (Smith, 2007). The processed text is then tokenized, a process where it is divided into smaller, meaningful units called tokens, to facilitate effective analysis by XLNet. Each token is assigned a numerical representation, allowing XLNet (Yang et al., 2019) to understand and process the textual data efficiently. The tokenized text is then inputted into XLNet, a powerful model known for its ability to capture bidirectional dependencies in text data (Yang et al., 2019). XLNet processes this tokenized text, extracting and encoding meaningful features into a compact 1x768-dimensional feature vector. This vector serves as a rich representation of the textual content, encapsulating key semantic and syntactic information extracted by XLNet (Zihang Dai, 2019). Subsequently, this feature vector undergoes further processing through a linear layer, refining and enhancing its representational power before being utilized for downstream tasks such as classification or regression (Zihang Dai, 2019). This feature vector encapsulates essential textual information crucial for the model’s comprehensive analysis and classification of code blocks (Zihang Dai, 2019). By leveraging XLNet’s capabilities, the accuracy and effectiveness of the ensemble model is enhanced, facilitating precise evaluation of diverse Lego Spike code sequences. The feature vector equation is as follows:

$$f_2 \in R^{1 \times 512} \dots\dots\dots(2)$$

3.3 BERT

BERT, renowned for its robust bidirectional learning from unlabeled text, offers comprehensive insight by considering both left and right context in all (Devlin et al., 2018). In the research, BERT (Devlin et al., 2018) assumes a pivotal role within the ensemble model, primarily tasked with extracting textual features. Its adaptability and solid language foundation enable various tasks, from question answering to language reasoning, without necessitating major architectural modifications (Devlin et al., 2018).

When evaluating code blocks, BERT meticulously analyzes the coherence of text preceding and following each block, ensuring logical sequence and alignment with the intended programming task (Devlin et al., 2018). For example, consider the Lego Spike project “Break Dance,” where students are tasked with programming a robot to dance to the beat. Before the code block instructing the robot to start dancing, the text might describe setting the speed and turn rotation of the robot. Following this block, the text might detail the actions the robot should perform, such as moving its arms or spinning in place. BERT examines the text before and after each code block, ensuring that the sequence makes logical sense and aligns with the desired outcome of the project. This capability is instrumental in guiding students to grasp the flow of their programming logic accurately, thereby fostering a deeper understanding of coding concepts.

To facilitate text extraction from images, Google’s Tesseract-OCR Engine (Smith, 2007) is utilized, a powerful tool that preprocesses the extracted text before undergoing tokenization. For example, let’s consider an image containing code blocks related to the project “Wind Speed.” The Tesseract-OCR Engine extracts the text from the image, converting it into a readable format. Once the text is extracted, it undergoes tokenization, where it is divided into individual tokens or words. These tokens are then inputted into the BERT model, which processes them to generate a feature vector of size 1x768. This feature vector represents the textual content of the image in a compact and informative manner. Finally, this feature vector undergoes further processing through a linear layer, refining it into a more refined representation that captures the essence of the extracted text.

The feature vector derived from BERT encapsulates essential textual information crucial for the model’s comprehensive analysis and classification of code blocks. By leveraging BERT’s capabilities, to enhance the accuracy and effectiveness of the ensemble model, precise evaluation of diverse Lego Spike code sequences is achievable. The test results, as depicted in **Table 1** below, provide empirical support for this claim:

$$f3 \in R^{1*512} \dots\dots\dots(3)$$

In Table 1, each project represents a unique coding sequence, and the counts reflect the occurrences of specific block types within these sequences. For instance, in the “Hopper Race” Lego Spike project, there are four occurrences of Action blocks, one Conditional block, and no occurrences of other block types. Similarly, the “Rain or Shine” Lego Spike project exhibits diverse block types, with higher counts in Conditional, Control, and Data categories.

These results demonstrate the effectiveness of BERT in accurately capturing the nuances of different coding sequences. By analyzing textual information before and after each code block, BERT enables the model to understand the logical flow of programming logic accurately.

4. Ensemble Model

The ensemble model represents a potent amalgamation of diverse prediction models, synergistically combining their outputs to forge a robust final model geared towards precise predictions (Ganaie, 2022). Ensemble learning, renowned for its capability to enhance performance and minimize loss, stands as a cornerstone technique in the methodology (Zohair, 2022). Adopting a stacking approach, outputs from individual models are integrated, harnessing a meta-learning model to optimize results (Ženko, 2004).

In the implementation, a similar stacking approach is followed, amalgamating the outputs of three distinct models—MobileNetV3, BERT, and XLNet—into a cohesive stacked ensemble model. Specialized embeddings are derived from each model, resulting in f1, f2, and f3, which are then concatenated to form the final embedding, f4 (M Paz Sesmero, 2015). This meta-layer embedding serves as a pivotal component, denoted as:

$$f4 \in R^{1 \times 1088} \dots\dots\dots(4)$$

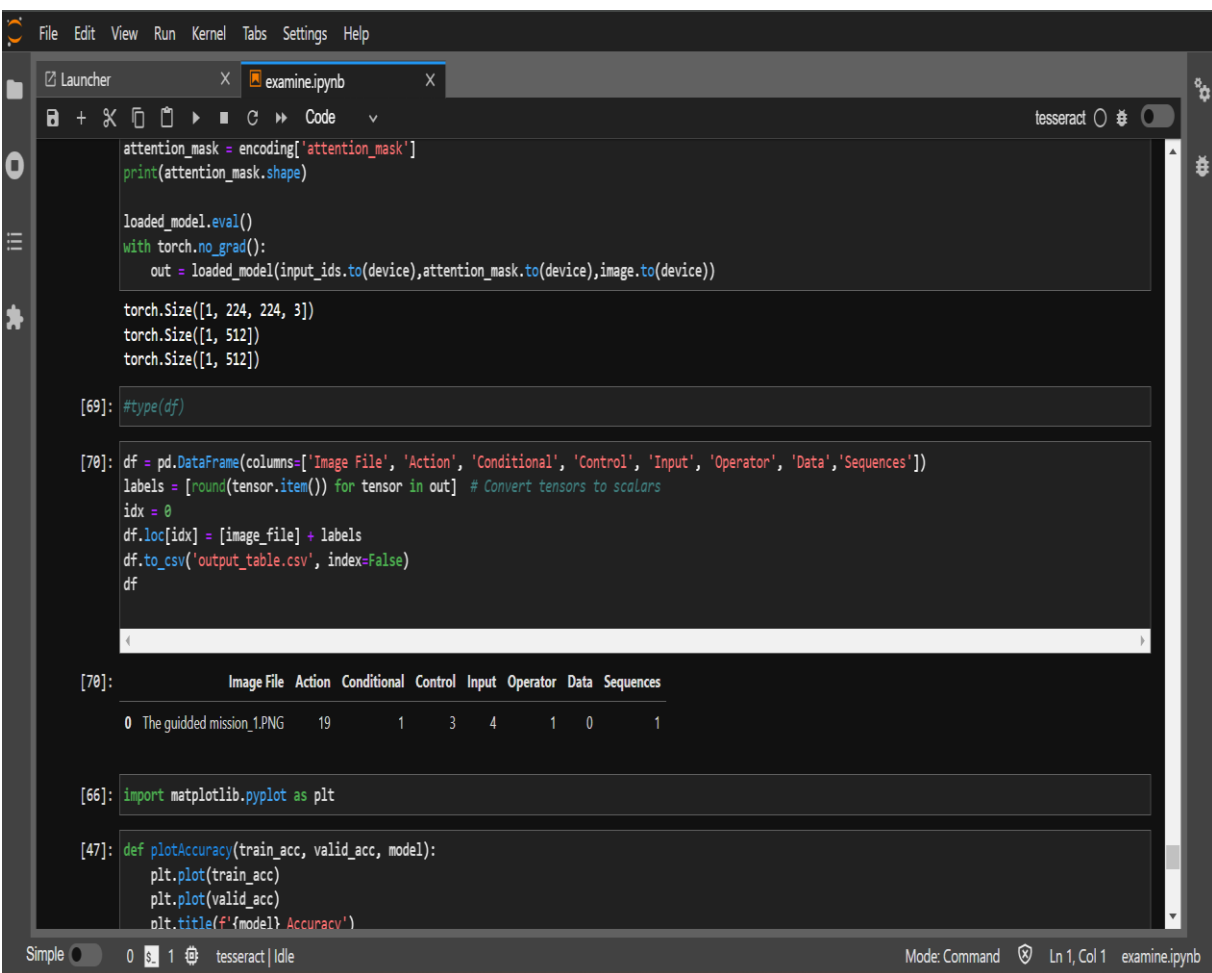
Subsequently, this final embedding undergoes further processing through a linear layer, culminating in a feature vector of size 128. At each linear layer, a ReLU non-linearity is applied, augmenting the model’s capacity for nuanced analysis. To refine predictions, the feature vector traverses through the last linear layer for classification, followed by a ReLU layer, thereby ensuring precision in the predictions (Bhandari, 2015). This streamlined approach encapsulates the essence of an effective and accurate ensemble model.

5. Hyperparameter

In the model training process, conducted within the Jupyter notebook environment (Jupyter, n.d.), certain hyperparameters were meticulously chosen and kept constant to maintain consistency and ensure reliable results. The learning rate (lr) was set at a fixed value of 3e-4, while a vocabulary size of 512 was utilized. Training was carried out over

150 epochs, facilitated by the Adam optimizer (Zhang, 2018), a widely acclaimed optimization algorithm known for its effectiveness in training neural networks.

To ensure computational efficiency and optimize resource utilization, the capabilities of the Ohio Supercomputer Center were leveraged (Ohio Supercomputer Center, 1987). Specifically, the experiments were conducted using 10 cores on the Pitzer cluster (Ohio Supercomputer Center, 2018), equipped with high-performance GPUs featuring 40 cores each. This configuration provided a robust and scalable environment conducive to efficient model training and experimentation, enabling us to explore various architectural configurations and hyperparameter settings effectively.



```
File Edit View Run Kernel Tabs Settings Help
Launcher X examine.ipynb X
attention_mask = encoding['attention_mask']
print(attention_mask.shape)

loaded_model.eval()
with torch.no_grad():
    out = loaded_model(input_ids.to(device), attention_mask.to(device), image.to(device))

torch.Size([1, 224, 224, 3])
torch.Size([1, 512])
torch.Size([1, 512])

[69]: #type(df)

[70]: df = pd.DataFrame(columns=['Image File', 'Action', 'Conditional', 'Control', 'Input', 'Operator', 'Data', 'Sequences'])
labels = [round(tensor.item()) for tensor in out] # Convert tensors to scalars
idx = 0
df.loc[idx] = [image_file] + labels
df.to_csv('output_table.csv', index=False)
df

[70]:      Image File Action Conditional Control Input Operator Data Sequences
0  The guided mission_1.PNG      19          1      3      4          1      0          1

[66]: import matplotlib.pyplot as plt

[47]: def plotAccuracy(train_acc, valid_acc, model):
plt.plot(train_acc)
plt.plot(valid_acc)
plt.title(f'{model} Accuracy')
```

Figure 8 Dr. Lego in Action: Demonstrating Functionality in Jupyter Lab

6. Empirical Analysis

Figure 9 showcases the integrated model, a synthesis of MobileNetV3, BERT, and XLNet, capturing the evolving loss function dynamics during model training. Each component contributes uniquely to the observed improvements. The progressive decline in the loss function, evident across the 150 epochs represented on the x-axis, signifies the collective model's adeptness in learning and generalizing. MobileNetV3 enhances image recognition, while BERT and XLNet excel in textual analysis, collectively ensuring a comprehensive understanding of both visual and textual features in coding projects. This collaborative approach enables the model to effectively learn and adapt, resulting in notable reductions in both train and validation loss.

For this experiment, a dataset consisting of 86 training samples and 9 test samples were curated to facilitate effective learning and evaluation by the model. The selection of these samples was strategic, encompassing a diverse range of Lego Spike coding projects such as Hopper Race, The Guided Mission, Wind Speed, Time for Upgrade, Rain or Shine, and Keep It Really Safe! (LEGO Education, 2022) Each project was chosen to represent various complexities and coding scenarios encountered in real-world applications.

The rationale behind this selection was to ensure that the model was trained and tested on a comprehensive dataset that accurately reflects the diversity of coding tasks and challenges. By including projects with different levels of complexity and requirements, thus aimed to equip the model with the ability to generalize well to unseen data and effectively handle a wide range of coding scenarios.

Moreover, the intentional balance in the sequence distribution between 0 and 1 within the dataset was crucial. This balance ensures that the model is trained on an equitable representation of both successful and unsuccessful coding sequences, enabling it to learn effectively from both positive and negative examples. The descending curve observed in Figure 9 signifies positive outcomes, affirming the efficacy of the proposed methodology.

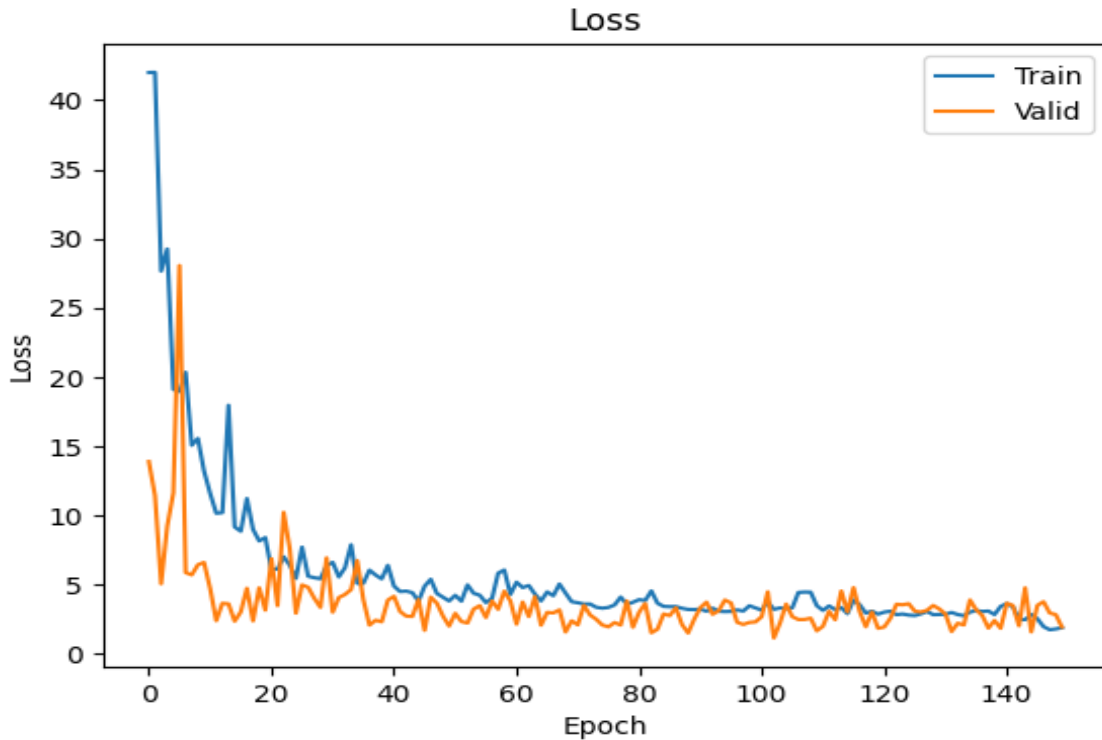


Figure 9 : Training and Validation Loss Evolution Over Epochs

As depicted in Figure 5, the model receives data from a variety of projects. The scores across seven different categories are presented in Table 1. These scores are determined based on the frequency of occurrence of each code block in the dataset. To validate the accuracy of these values, a rigorous validation process was employed. Firstly, scores generated by the model were compared to the actual data fed into it to ensure consistency and correctness. Additionally, scores were verified by cross-referencing them with the new output data obtained when the project code is reshuffled, added, or modified. This comparison allowed to assess the model’s ability to accurately evaluate coding projects under different conditions.

The scores obtained through this validation process served as tangible evidence of the model’s proficiency in capturing the nuances of the provided data. They highlight the reliability and effectiveness of the model in accurately assessing coding projects across various scenarios.

Image File	Action	Conditional	Control	Input	Operator	Data	Sequences
Hopper Race	4	1	0	0	0	0	0
The Guided Mission	19	1	3	4	1	0	1
Wind Speed	4	1	1	1	0	1	1
Time For Upgrade	5	1	0	0	0	0	1
Rain Or Shine	8	3	1	2	0	1	0
Keep It Really Safe!	14	1	3	1	0	0	0

Table 1: Analysis Results: Dr. Lego Model Output

7. Conclusion and Future Scope

In conclusion, the ensemble learning model, powered by the synergistic integration of MobileNetV3, BERT, and XLNet components, has showcased remarkable efficacy in evaluating coding projects. Through the analysis of text-embedded images and their corresponding extracted text from the OCR model, the model adeptly assigns scores to diverse projects based on their coding blocks. Dr. Lego emerges as a valuable tool for students, providing them with insights to monitor their performance and navigate away from unfavorable coding practices. Moreover, Dr. Lego extends its utility beyond the confines of the classroom, catering to the needs of teachers, learners, and organizations involved in programming endeavors, thereby enhancing project effectiveness assessment.

Looking ahead, the future endeavors revolve around the development of an accessible application or website, facilitating effortless evaluation of LEGO SPIKE app projects. Users will be able to submit the requisite images and promptly receive comprehensive results, thereby expanding the utility of Dr. Lego as a versatile tool for project evaluation. Leveraging the strengths of the ensemble model will inspire, empower, and guide young learners, fostering a genuine passion for programming from an early age.

REFERENCES

- [1] Moreno-León, J., Robles, G., & Román-González, M. (2015, September). Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking. *Revista de Educación a Distancia (RED)*. <https://doi.org/10.6018/red/54/10>
- [2] LEGO Education. (n.d.). LEGO Education SPIKE Prime Set. Retrieved 2021
- [3] LEGO Education. (n.d.). LEGO Education SPIKE App. Billund, Denmark: LEGO Education. Retrieved December 1, 2022
- [4] Howard, A. G., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... Le, Q. V. (2019). Searching for mobilenetv3. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (pp. 1314–1324). <https://arxiv.org/pdf/1905.02244>
- [5] Smith, R. (2007). An overview of the Tesseract OCR engine. In *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)* (Vol. 2, pp. 629–633). IEEE. <https://doi.org/10.1109/ICDAR.2007.4376991>
- [6] MMPretrain. (n.d.). MobileNetV3. Retrieved from https://mmpretrain.readthedocs.io/en/latest/papers/mobilenet_v3.html
- [7] Smith, J. (2024). *Enhancing Coding Education: Methodologies for Model Integration, Data Generation, and Pre-trained Model Fine-tuning* (Unpublished doctoral dissertation). University of Technology.
- [8] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). XLNet: Generalized autoregressive pretraining for language understanding. *Advances in Neural Information Processing Systems*, 32(33), 8024–8035. <https://arxiv.org/pdf/1906.08237>
- [9] Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., & Salakhutdinov, R. (2019). Transformer-XL: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*. <https://arxiv.org/pdf/1901.02860>
- [10] Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. <https://arxiv.org/pdf/1810.04805>
- [11] Ganaie, M. A., Hu, M., Malik, A. K., Tanveer, M., & Suganthan, P. N. (2022). Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115, 105151. <https://arxiv.org/pdf/2104.02395>
- [12] Džeroski, S., & Ženko, B. (2004). Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, 54, 255–273. <https://link.springer.com/content/pdf/10.1023/B:MACH.0000015881.36452.6e.pdf>
- [13] Zohair, M., Bhavsar, N., Bhatnagar, A., & Singh, M. (2022). Innovators@ smm4h'22: An ensembles approach for self-reporting of COVID-19 vaccination status tweets. In *Proceedings of The Seventh Workshop on Social Media Mining for Health Applications, Workshop & Shared Task* (pp. 123–125).
- [14] Sesmero, M. P., Ledezma, A. I., & Sanchis, A. (2015). Generating ensembles of heterogeneous classifiers using stacked generalization. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5(1), 21–34. <https://wires.onlinelibrary.wiley.com/doi/10.1002/widm.1143>
- [15] Zhang, Z. (2018). Improved Adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)* (pp. 1–2). IEEE. Retrieved from <https://ieeexplore.ieee.org/document/8624183>

[16] Project Jupyter. (n.d.). Retrieved from <https://jupyter.org/>

[17] Ohio Supercomputer Center. (1987). Ohio Supercomputer Center.

[18] Ohio Supercomputer Center. (2018). Pitzer supercomputer.