SEML: Developing Scratch Programming Extensions for Machine Learning

Mark Kocherovsky
Lawrence Technological University, Department of Mathematics and Computer Science, Southfield, MI 48075

(Dated: 05 Feb 2021)

Machine learning is the study of programming algorithms capable of "learning" from sets of given data and previous experiences. Teachable Machine is an experimental tool created by Google to allow users to apply machine learning to create neural networks (deep learning). Scratch is a programming language and environment created by the Massachussetts Institute of Technology Multimedia Lab to assist in K-12 computer science education. This paper demonstrates that it is possible to import Teachable Machine models into Scratch (both using a hardcoded model location and a model provided dynamically by the user through a URL), and then use that model in a Scratch 3.0 program.

I. BACKGROUND

A. Machine Learning Background

Machine learning is a field of computer science that studies algorithms capable of training predictive models from given sets of data and previous experience [1] [2] [3]. To make machine learning more accessible, Google offers a web-based tool called Teachable Machine to allow users to make their own models using image, pose, and sound-based training [4]. Teachable Machine itself uses TensorFlow.js [5], which uses JavaScript to create a neural network for training and output [6]. It allows users to upload the trained model to the cloud for later usage, which is an integral component in SEML.

B. Scratch Background

Scratch is a programming language/environment created by the Lifelong Kindergarten Group at the Massachusetts Institute of Technology (MIT) Media Lab to teach young people programming concepts. It is a visual language that uses blocks to represent lines of code. The user can directly see and add objects on the output screen (such as Scratch's mascot, a cartoon cat, or the background), the user can import or record sounds, and can provide input through the keyboard and mouse. The system is very versatile, with the inclusion of variables, control features such as loops and conditional statements, logic, and the ability for the user to create their own functions. Using this environment, the user can easily program their own animations and games [7].

C. Introduction to SEML

The SEML Program has two components. First, a test extension was made to detect the location of a user's palm using a Teachable Machine model. However, the main component of SEML is the ability to directly import a Teachable Machine model without the use of any intermediaries.

II. PROGRAM REQUIREMENTS

This section aims to explain what the SEML program is meant to accomplish and what it is meant to output to the user. For a more in-depth explanation of how the program operates, please see Section III. In this paper, the term "Scratch" generally refers to Scratch 3.0, the most recent version.

A. Palm Detection

The palm detection test module will be able to detect the presence of the user's palm using their webcam and return its position, which the user can then export to a variable. This is a test component to gauge Scratch's ability to import Teachable Machine models from the cloud.

B. The Full SEML Extension

The full SEML extension shall allow the user to import any Teachable Machine model from the cloud, provided that they have the URL. SEML will then allow the user to implement the model in the program, which will return output that can be exported into variables.

III. DESIGN

Scratch 3.0 extensions are JavaScript files that contain a class that defines the extension itself, each block, the function corresponding to each individual block, and various helper functions. Blocks can take arguments, include graphics, and contain menus [8]. For example, Figure (1) shows the code corresponding to the `Import Model` block in SEML, and Figure (2) shows the structure of blocks in the Palm Reader model.

```
importModel(args) {
    console.log("import model start");
    //from https://github.com/googlecreativelab/teachablemachine-community/blob/master/libraries/image/README.md
    //also https://medium.com/jspoint/javascript-promises-and-async-await-as-fast-as-possible-d7c8c8ff0abc
    //Example URL: https://teachablemachine.withgoogle.com/models/3RBPI6zKN/
    const url = args.URL;
    const modelUrl = url + "model.json";
    const metadataUrl = url + "metadata.json";
    //https://github.com/LLK/scratch-vm/issues/2914#issuecomment-791514353
    return tmImage.load(modelUrl, metadataUrl)
        .then((tmModel) => {
            model = tmModel;
            maxPredictions = model.getTotalClasses();
            return model;
        })
        .catch((err) => {
            console.log("Something went wrong in Import Model");
            console.log(err);
        });
    return model;
}
```

FIG. 1. The SEML `importModel` function that corresponds to the `Import Model` block as JavaScript code.

A. Palm Detection

The Palm Detection test module begins by requiring the user to import the model from the cloud. Since the Teachable Machine model already exists, its URL is hardcoded in the program's backend. The import command, `Import Block`, then connects the user to the cloud, downloads the palm detection model, and initializes the model.

The user can then select blocks entitled `X Coordinate` and `Y Coordinate`, which will return the x and y coordinates of the user's palm as described in IV A. These blocks function by using the model to ascertain the location of the user's palm.

```
blocks: [
    {
        opcode: 'importModel',
        text: formatMessage({
            id: 'palm.import',
            default: 'Import Model',
            description: 'Import the Palm Reader Model'
        }),
        arguments: {

        },
        blockType: BlockType.COMMAND,
        //filter: [TargetType.SPRITE]
    },
    {
        opcode: 'getXCoord',
        text: formatMessage({
            id: 'palm.getxcoord',
            default: "X Coordinate",
            description: "Get the X quadrant of the user's palm."
        }),
        arguments: {},
        blockType: BlockType.REPORTER,
```

FIG. 2. Palm Reader blocks defined in JavaScript

B. Dynamic Importation

The main SEML program will function nearly identically to the Palm Detection test module. However, there are two key differences. First, the `Import Model` command includes a text input field in which the user can place the URL of a Teachable Machine model. The block, when run, then automatically initializes the given model based on the URL, which uses TensorFLow libraries [9].

Secondly, instead of specific `X Coordinate` and `Y Coordinate` blocks, SEML only provides a single `Return Prediction` block, which allows the user to export the results of their model into a variable, as SEML requires a more general set of commands. The diagram in Figure (3) provides a high-level overview of the SEML module's design.

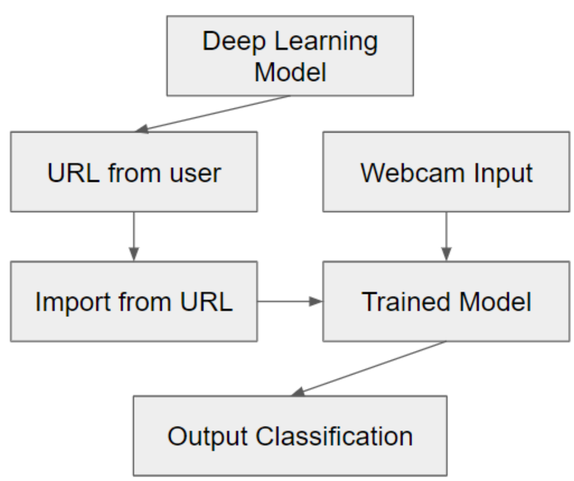

FIG. 3. High Level Diagram of the SEML module

IV. MODELS AND RESULTS

A. Models

The Palm Detection model has five classes. The first four (A1, A2, B1, and B2) indicate the quadrant where the user's palm is located as seen by the webcam. The fifth is a default case that indicates the user is not holding up their palm.

The Palm Detection model was trained using Teachable Machine, where the author held his hand up (palm towards the camera) in each of the four quadrants and took pictures using the webcam. The author took several hundred pictures in each quadrant at various angles to provide the model with a better understanding of what constitutes a palm. The model was then exported as a Tensorflow Javascript model to the cloud.

B. Results

1. *Palm Detection*

A simple Scratch program to determine the location of the user's hand is given in Figure (4). When the program is run, the webcam will turn on, `Import Model` will import the model and assign it as a global variable, and then the x and y coordinates will be evaluated in an indefinite loop. Figure (5) shows the result of this program. This demonstrates that Deep Learning models can be used in Scratch 3.0 programs.

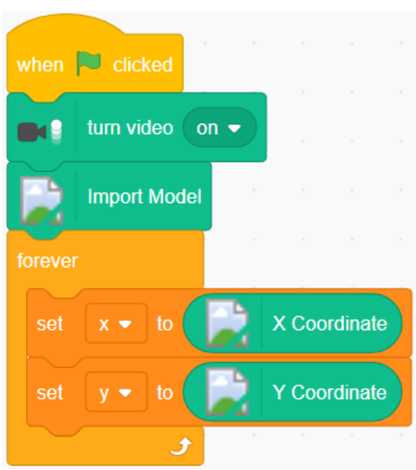

FIG. 4. A simple Palm Reader program.

2. *SEML*

A basic program using the SEML extension is shown in Figure (6). In this example, `Import Model` is given

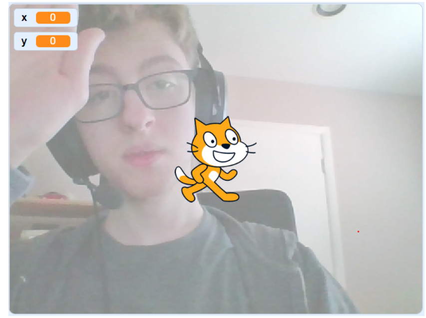

FIG. 5. The result of the Palm Reader Program. Note that the program has a tendency to treat the author's head as his hand, thus confusing the model somewhat. However, it is clear that the Scratch extension itself works.

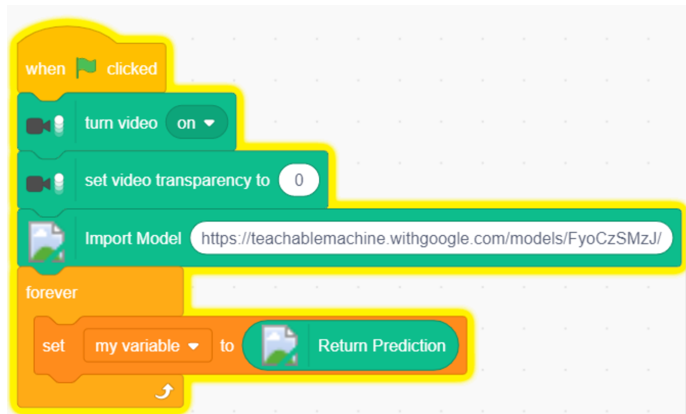

FIG. 6. A simple SEML program. The URL given is the a model to recognize a plush kitten and a puppy.

the a model that recognizes a puppy or a kitten, trained using Teachable Machine with 500-800 images for each class ("Puppy", "Kitten", "None"). The code imports the model , and returns the class as a string. The results are shown in Figure (7).

This proves that it is possible to train a model using Google's Teachable Machine, upload it to the cloud as a JavaScript model, import it into Scratch through an extension, and then use the model in a program.

V. CONCLUSION

SEML proves that it is possible to dynamically import trained JavaScript neural network models into Scratch 3.0 for use in a Scratch Program. As Scratch is meant to be an educational software, SEML thus also has educational applications. SEML can be used to teach students (at the K-12, university, and even professional level) about the basics of neural networks by allowing them to train a model and apply it in a relatable manner without having to go in-depth into the programming background. It has been shown that students may learn better through the use of a kinesthetic (hands-on) component [10] [11],

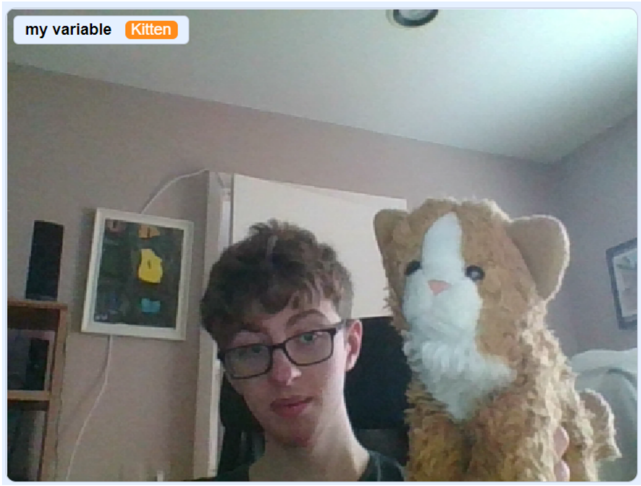

FIG. 7. Output from a SEML program using the a model to determine whether it sees a plush kitten or puppy.

which SEML would provide if used with a Tensorflow JavaScript model.

Future development of SEML should take several factors into account. First, the extension does not come with any graphics, which would need to be developed to make the extension visually appealing and easier to understand. Second, SEML so far can only handle image-based models. A more advanced version of SEML should be able to use Teachable Machine audio and pose models as well. Thirdly, it might be of interest to some to see the numerical value of each class (how probable that what the model sees corresponds to a given classification), so returning these values could be a possible design target for further development.

VI. ACKNOWLEDGEMENTS

Thanks to Dr. Chan-Jin Chung for being my Faculty Advisor for this project.

Thanks to Dr. Mitchel Resnick, Eric Rosenbaum, GitHub users apple502j and adroitwiz, Dale Lane, and the Scratch Multimedia Lab for helping with technical issues.

[1] C. Chung, Neural networks and deep learning with Python (2020), [PowerPoint Slides].
[2] C. Chung, Introduction to AI & ML with Scratch [Pt. 1] (2021), [PowerPoint Slides; Online; accessed 28-February-2021].
[3] C. Chung, Introduction to AI & ML with Scratch [Pt. 2] (2021), [PowerPoint Slides; Online; accessed 28-February-2021].
[4] Alphabet Incorporated, Teachable Machine (2021), [Online; accessed 02-February-2021].
[5] Alphabet Incorporated, Teachable Machine - FAQ (2021), [Online; accessed 02-February-2021].
[6] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems (2015), software available from tensorflow.org.
[7] M. Marji, *Learn to Program with Scratch*, 1st ed. (No Starch Press, 2014) [Online; accessed 06-December-2020].
[8] K. Chadha, C. Willis-Ford, and u9g, Scratch 3.0 Extensions (2020), [Online; accessed 30-March-2021].
[9] I. Alverado and J. Jongejan, Teachable Machine Library - Image (2019), [Online; accessed 28-February-2021].
[10] C. Chung and M. Kocherovsky, CS+PA2: Learning computer science with physical activities and animation — A MathDance experiment, in *2018 IEEE Integrated STEM Education Conference (ISEC)* (2018) pp. 262–267.
[11] M. Shamir, M. Kocherovsky, and C. Chung, A Paradigm for Teaching Math and Computer Science Concepts in K-12 Learning Environment by Integrating Coding, Animation, Dance, Music and Art, in *2019 IEEE Integrated STEM Education Conference (ISEC)* (2019) pp. 62–68.