Augmented Reality Final Project Report

CNN Lego

Xue Yu (Zoe YUU), Trinity College Dublin, MSc. Computer Science – R and VR

Introduction

This project has been inspired by a naïve mistake which I have made in the Deep Learning experiment as an inexperienced researcher. In the model I have designed, the size of the convolutional layer's output did not equal to its input, as a result, as I attempted to build an identity block similar to a Residual Net's identity block, the identical mapping provided by the input tensor cannot be added to the output of the convolutional layer. Naturally, such a simple mistake in design can be quickly corrected once it has been detected as the program ran. However, since the building of a convolutional neural network has been compared with the game of building a Lego model in CS231n's course, if people could play with a model of convolutional neural network like they are playing with a Lego model, they could easily spot the difference in two pieces of layers' size with naked eye, as a result, such a simple mistake in design could be much easier caught, instead of only being detected when the program begins to run.

This incident has then inspired the proposal of this project: to design a system in which the player can create different type of convolutional neural network' layers, can resize the selected layer by adjusting its parameters such as input tensor size, filter number, filter size, stride, can translate and rotate the selected layer so that their size can be easily compared, can create a root of model or root of block and further, can add layers in the model or block. Using this system, the player shall be able to build a convolutional neural network model in the augmented environment in a way that they can play with a Lego model in the real-world.

In my opinion, this program can give a player or a designer inspiration by messing with blocks and layers in augmented reality while they are designing a new convolutional neural network model. Furthermore, this program can show the beginning learners, even children, that how simple and direct a design of new model could be[1].

Theoretical background

In theory, using the HoloLens headset, users can see the virtual objects in the real-world environment. Using gaze, users can select an object in the scene, using gesture or voice control, users can send commands to the selected object to make them complete a certain operation [1,2].

Furthermore, users should be able to run and debug their programs on the HoloLens Emulator without any headset attached.

In theory, using Vuforia, combined with Unity, developers can develop AR programs both with marker and markerless, can display the virtual objects in the real-world environment through a screen of a tablet or computer.

---

[1] The difficult part of conducting experiments with the designed model and finetuning the model's parameters come in only later.

In this project, the completed desktop version of CNN Lego system has been implemented first, then attempts have been made to adapt the desktop version to HoloLens Emulator and the Vuforia engine, so that players of CNN Lego can build and play with virtual models in the augmented environment through a simulated headset's screen[1] and through a screen of a computer or tablet.

Implementation

A fully functional desktop version CNN Lego has been firstly implemented using Unity. The entire system can be described as a GUI system.

Upon the top of the screen, three drop-down menus' buttons have been implemented: "Add Layer", "Adjust Layer", "Add Root" (Figure 1).

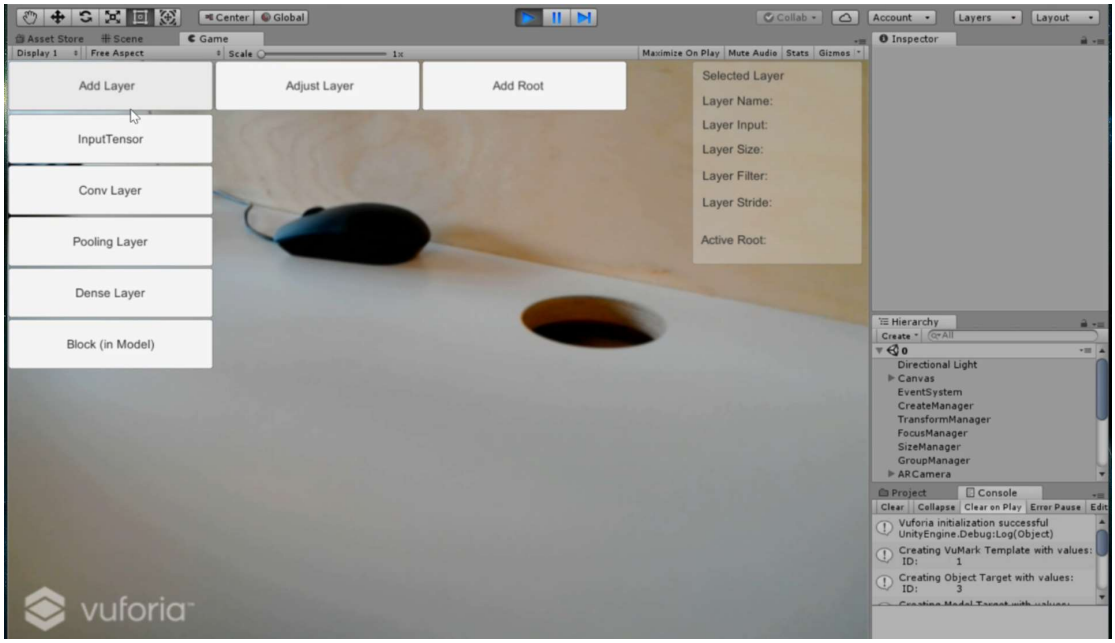


Figure 1. CNN Lego Vuforia version[2]. Menus.

Under "Add Layer", five buttons have been placed in the menu (Figure 1): "Input Tensor", "Conv Layer", "Pooling Layer", "Dense", "Block (in Model)"[3]. On click of each of those buttons, a respective function of class CreateManager[4] will be called, a prefab of a layer model (a cube

[1] The implementation has been tested using the HoloLens Emulator only, instead of on the real headset of HoloLens.

[2] The desk version is almost identical with the Vuforia version from GUI to scripts. Since this project considers the Vuforia version as the final version, the GUI system of the Vuforia version is used to illustrate the functions of the system.

[3] Naturally, the real CNN model may contain various different types of layers other than the four types listed above. Due to the limited time resource, here only four of the most common layers have been listed to test the functions. Block (in Model) means the green colored brick which can be the placeholder of an identity block in a model.

[4] Implemented in script "CreateManager.cs". Added on an empty GameObject called "CreateManager".

of various colors) is to be instantiated and initialized with a certain default size.[1] Class Layer[2] is attached to each of the prefabs, recording the parameters of the layer. On the right top of the screen, a panel is displaying the parameters of the selected layer[3] (Figure 2).

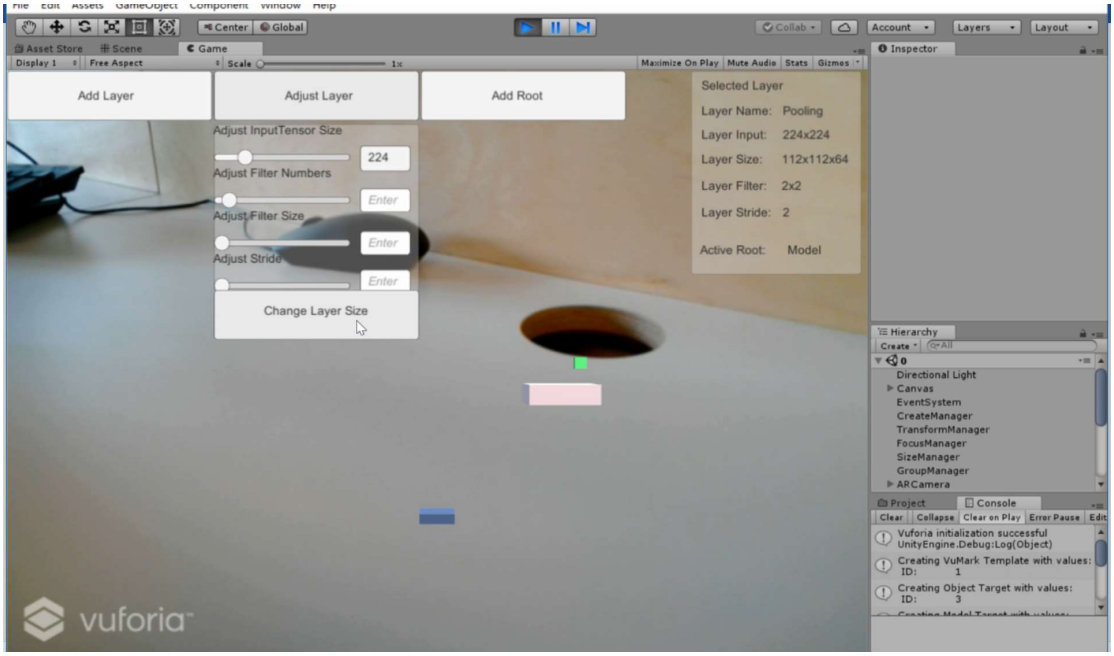


Figure 2. CNN Lego Vuforia version. Menus.

Under "Adjust Layer", four sliders and one button have been placed in the menu (Figure 2). The sliders are: "Adjust InputTensor Size", "Adjust Filter Number", "Adjust Filter Size", "Adjust Stride"[4]. Additionally, there is a text input field on the right side of each slider. Drag a slider's button to the wanted position, or simply input the value of the parameter in the input field, then click the "Change Layer Size" button, the size of the selected layer is to be adjusted according to the input from the text input field or the slider value. The priority of text input field is higher than the slider, since it might be easier for players to simply type in the value. Two classes are associated with functions behind this drop-down menu[5]. The class FocusManager does a raycast based on mouse input, if the ray has hit an object which is not on "root" layer, the previous object with the tag "selected" would be changed to use tag "unselected", and the current object would use the tag "selected"; if the object been hit is on "root" layer, the previous object with tag "selected" keeps its tag, whereas the hit object's tag is to be changed into "activeRoot" whereas the previous object with tag "activeRoot" will

[1] For instance, a convolutional layer is to be created with a default value of 224x224x64, in this game, the size of it is 0.224 x 0224 x 0.064.

[2] Implemented in script "Layer.cs".

[3] The parameters in the display are: input tenor size, layer size, filter number, filter size and stride – the parameters which may affect the size of the layer. Since one of the objectives of this program is to allow players to easily compare the size of two layers.

[4] In the desktop version, there is a button in this menu called "Adjust Scale", which is originally used to adjust the scale of an entire block or model. The experiments' result shows, however, that this function is not as useful as originally designed. Therefore, in the final Vuforia version, this button has been removed.

[5] Implemented in scripts "FocusManager.cs" and "SizeManager.cs". Added on empty GameObjects called "FocusManager" and "SizeManager".

use the tag “inactiveRoot”[1]. On click of the button “Change Layer Size”, every slider and the respective text input field’s value are to be taken as input for class SizeManager. The class SizeManager firstly finds the object with the tag “selected”, then calculate the size of the layer with equation LayerSize = (InputSize - FilterSize)/Stride+1[2], finally, the new values of the layer’s parameters are to be updated on the panel on the right top corner.

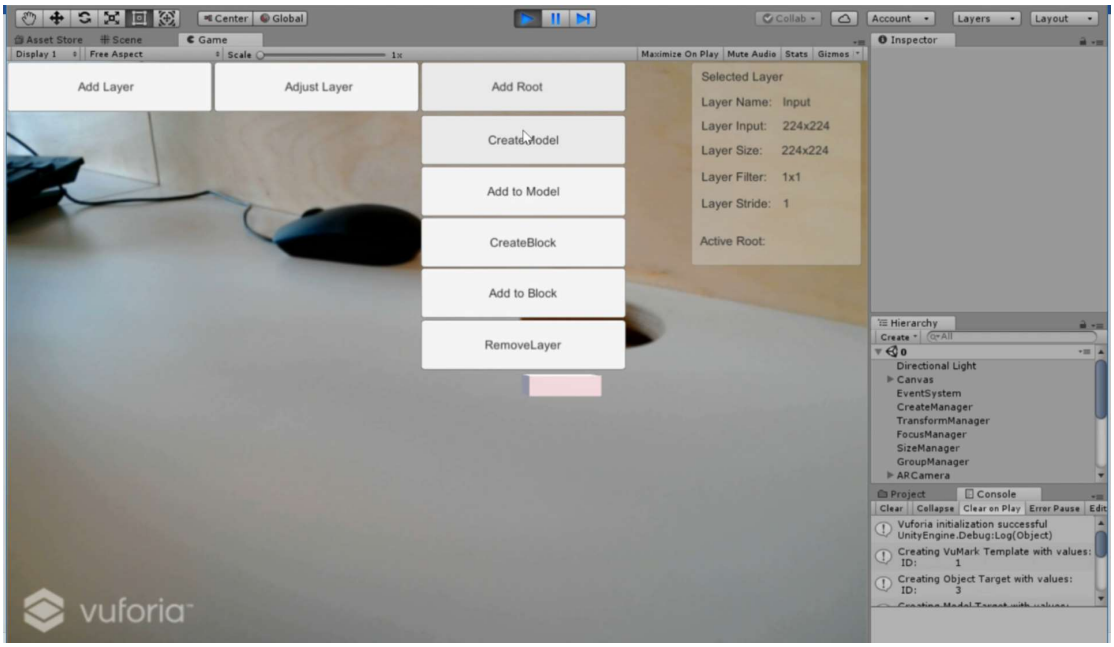


Figure 3. CNN Lego Vuforia version. Menus.

Under “Add Root”, five buttons are placed on this menu: “Create Model”, “Add to Model”, “Create Block”, “Add to Block”, “Remove Layer” (Figure 3). Class GroupManager[3] is associated with this menu’s function. On click of each button, a function in class GroupManager is to be called. Function CreateModel and CreateBlock instantiate a prefab of a root of a model or a block, give it the tag “activeRoot” and the layer “root”, as well as update the text “Active Root” on the panel on the top right corner. Function AddToRoot finds the object with tag “selected” and the object with the tag “activeRoot”, further, set the parent of the current layer to be the current active root[4](Figure 4). Function RemoveLayer finds the object with tag “selected” and set its parent to be null.

---

[1] Root function is to be explained later in this report.

[2] The result is the floored value of the calculated result.

[3] Implemented in script “GroupManager.cs”. Added on an empty GameObject called “GroupManager”.

[4] To keep the scale of the child unchanged, the prefab of a root is different from a prefab of a layer: it consists of an empty GameObject as the real root, which has a scale of [1, 1, 1], and a visible child GameObject, a cube, which is to be displayed on the screen, to be selected, and has the tag of “activeRoot” or “inactiveRoot”. AddToRoot function makes the selected object share the parent with the object with tag “activeRoot”, meaning, they both become the children of this empty GameObject, in this way, the scale of the children can be preserved. The other practical approach to preserve a child’s scale will be explained later in the report.

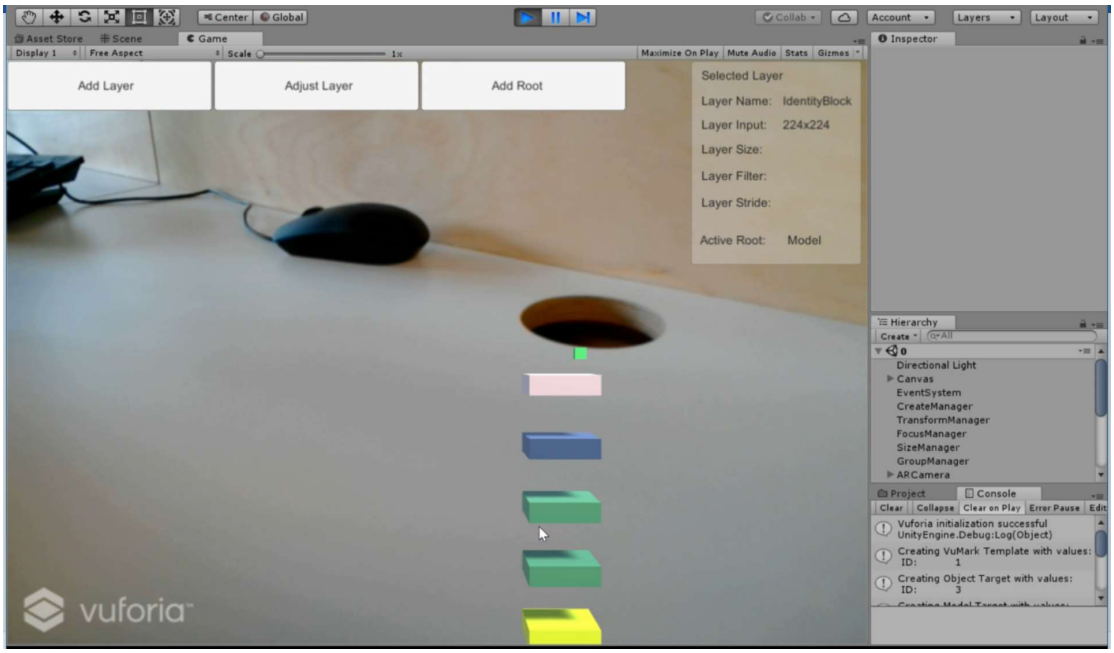


Figure 4. Grouping layer bricks into a model.

Moreover, class TransformManager[1] handles keyboard event. Press key W/A/S/D, the object with tag “selected” is to be translated to up/left/down/right; Press key Z/X/C/V/B/N, the object with tag “”selected” is to be rotated around x/y/z axis or rotated back; Press key arrow head up/ left/ down/ right, the object with tag “activeRoot” is to be translated up/left/down/right.

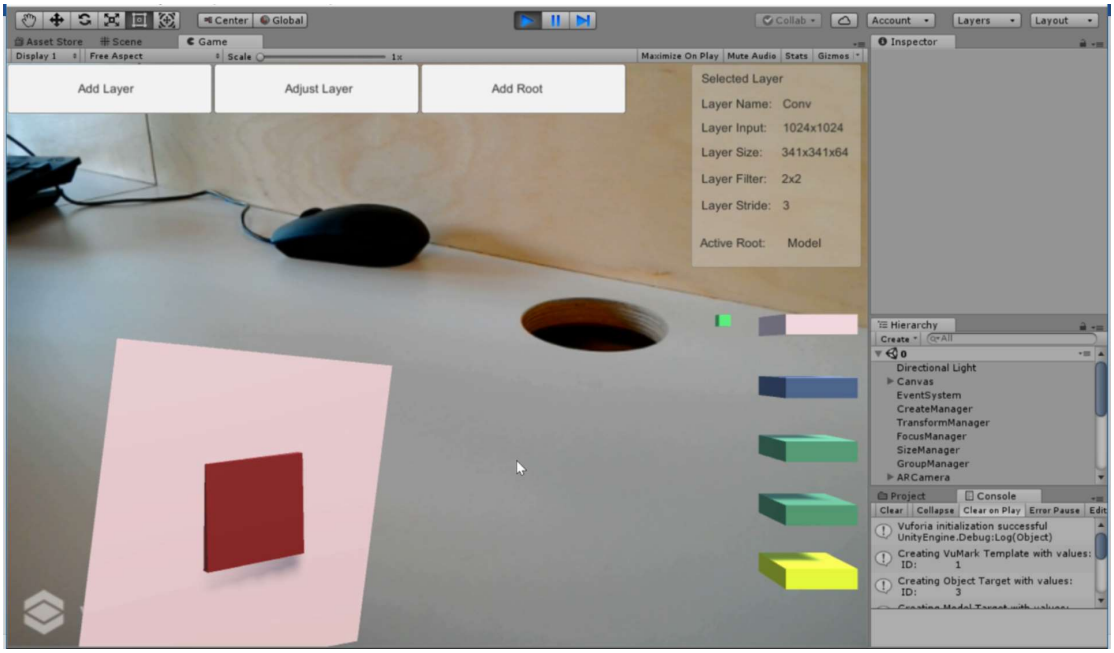


Figure 5. after rotation, translation, compare the size of two layers.[2]

Experiments and analysis of results

[1] Implemented in script “TransformManager.cs”. Added on an empty GameObject called “TransformManager”.

[2]

Attempts have been made to build the desktop version CNN Lego system and run on HoloLens Emulator, however, the attempts failed due to following reasons:

1. Although the functions introduced in the demo which has been provided by the HoloLens team could be duplicated (Figure 6), the built project can only be run once on the computer used for the development[1] - which makes debugging and developing near to impossible.

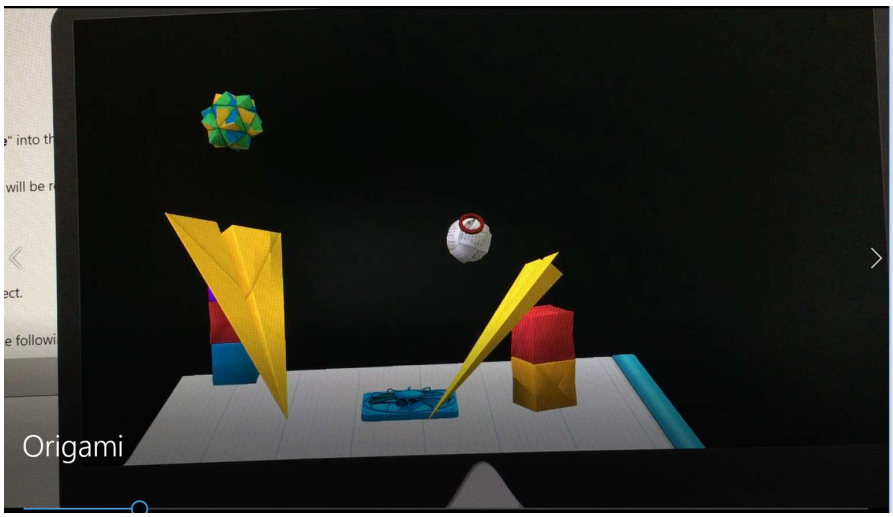
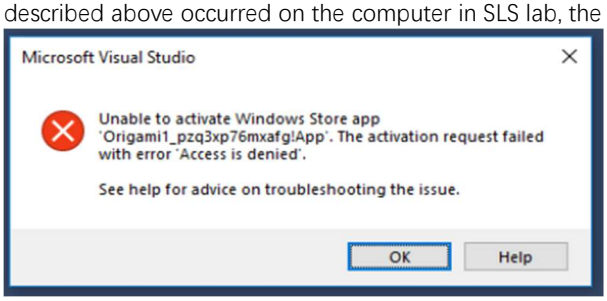


Figure 6. the only runnable Origami example in Emulator. Gaze selection.

2. In the simple Origami example run on Emulator, the flaws are clear to see: firstly, the selection with gaze is too slow and imprecise. Secondly, the display of GUI system is unreliable – without any changes in code or in Unity settings, the GUI system sometimes can be correctly displayed and sometimes not. Thirdly, a short demo containing all key functions of CNN Lego – instantiate layer, group a layer to a root of a model, translate a layer, rotate a layer - has been implemented firstly on the desktop (Figure 7), and then tried to be transplanted in HoloLens Emulator using mouse/keyboard input. However, the program using sole mouse/keyboard input cannot be run on Emulator on the computer which this project has been developed either[2]. Additionally, the gesture control is too

[1] Research has been done to try to solve this problem, which turns out not to be a singular case. However, with four or more solutions suggested in the forum been applied – one suggestion as to change the debugging details, worked for one time; another suggestion as to delete all files generated in building process, worked for three times and then stopped working - the problem remains unsolved. All problems described above occurred on the computer in SLS lab, the error message displayed is usually the following:



Further attempts have been made as to export the GUI system in a Unity package, then create a new project and import this package. However, a project created in this way cannot be run on Emulator either.

[2] The computer in the lab of SLS displayed the following error message:

simple, with only two gestures to use, nothing remotely complicated control system can be applied. To implement the complicated control system of CNN Lego without much aid from GUI system and gesture control, the only alternative is to use voice control – however, since any built projects – even without any bug in – can be run only once on Emulator on the computer used for development, the continuing of the experiment has been deemed to be impractical.

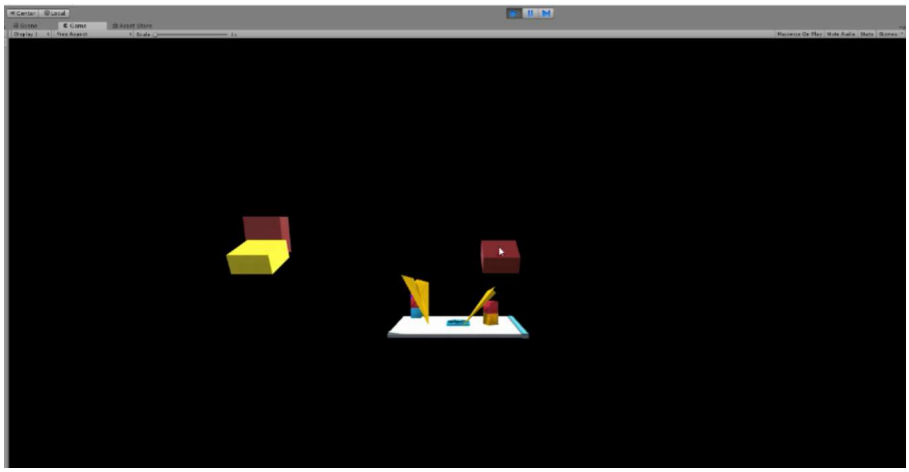
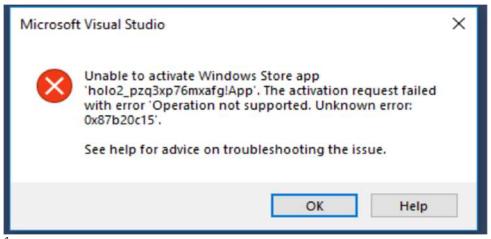


Figure 7. the desktop version of the concise short version of CNN Lego, containing only four key functions[1]. Root and a layer has been created and grouped together, another layer is being rotated. Controlled by mouse and keyboard instead of human input. Cannot be run by Emulator.

Attempts have then been made to add Vuforia ARCamera in the scene and enable the user to play CNN Lego – with virtual layers as bricks – in the augmented environment. Programs have been run on a Surface 3 tablet/laptop, with Win10 as operation system, the Intel(R) Atom(TM) x7-Z8700 as CPU (1.60 GHz), and Intel(R) HD Graphics as GPU, the using of mouse and keyboard have been required.

The results show that using a mark, the GUI system cannot correctly instantiate the prefabs[2]. However, in markerless mode, with ARCamera of Vuforia, using mouse and keyboard control, CNN Lego program can be run smoothly with all functions implemented on the desktop



[1] Detailed explanation of the content of the video and the function of the short demo can be found in the appendix.

[2] With a target marker, all virtual objects shall be added to the scene as the children of the target marker image, which make the instantiate of virtual layer brick inconvenient. Furthermore, the virtual objects generated in scripts cannot be displayed in the screen – they are, however, existed under the list of Inspector and have been generated in the correct position with correct scale. If prefabs have been directly dragged into the scene, in the same position and with the same scale, they can be displayed correctly. A rational explanation for this phenomenon has not been found yet. Since in the game of CNN Lego, a marker image is not even a necessary part, in this project only markerless AR has been used.

version intact. As a result, on a testing laptop/tablet of Surface 3, players can build convolutional neural network model in augmented reality as if playing with Lego bricks. The result can be seen in the video taken.

Conclusions and future work

Conclusion

Inspired by the Deep Learning experiment, in this project, a GUI system implemented so that users can play CNN layers as if they were playing "real" Lego pieces, whereas they can create new layers, models and blocks, resize the selected layer, group the layers in an identity block or in a model, remove a layer from a block or a model, translate and rotate layers, models and blocks, with detailed information of the layer and the root model/block displaying on the panel on the top right corner of the screen.

Further, experiments have been conducted, in which attempts have been made to adapt the desktop version into two different AR developing platforms. Although the attempt of adapting the desktop version of CNN Lego in HoloLens Emulator has failed due to the problem with running the built projects on HoloLens Emulator on the computer in SLS lab, the desktop version CNN Lego can be successfully displayed in the augmented environment using Vuforia engine.

Lessons learnt in this project

Since the performance of HoloLens Emulator is unstable, development on Win10 with HoloLens Emulator is risky, especially when the time resource is extremely limited.

Experience of using two different design patterns has been gained during the implementation of the control systems: the one is the top-down approach, using managers to manipulate the selected object, as has been implemented in the desktop version and the Vuforia version; the other is the more object-oriented approach, sending various messages to objects and then, on each prefab, scripts are added to answer the respective messages and perform their own operations, as has been implemented in the Origami demo and the short demo made in experiments with HoloLens Emulator. Both approaches have benefits and drawbacks and the choice made of choosing from them give me a deeper understanding of the design philosophy behind. In the end, the first approach has been chosen because, in this way, the control system developed can be easily transplanted in another project, regardless the difference between the GameObjects used.

Future work

The following improvements could be made in future to improve the program:

1. The input can be changed to use touch input completely, so that the project could be played using a normal tablet as well, without mouse and keyboard involved.
2. The menu in GUI system could be extended, adding more types of layers, so that the model built could be more precise.
3. Functions should be added to dissolve a group of layers, instead of removing one layer from each root at a time.
4. Furthermore, functions could be added so that Keras code of the design of a model could

be automatically generated, based on the model built inside this program.

References

[1] Holograms 100. https://docs.microsoft.com/en-us/windows/mixed-reality/holograms-100

[2] Holograms 101E. https://docs.microsoft.com/en-us/windows/mixed-reality/holograms-101

Appendix

Video contents:

1. Origami: https://youtu.be/jQAc9cSQs5o

   The video recorded how examples have been run in HoloLens Emulator. When the spheres have been selected using gaze, a red cursor appears, hugging the surface of the sphere[1]. Using "tap" gesture, the first sphere drops down to the desk, whereas the second sphere answers this message sent by the "tap" by rotating slightly.

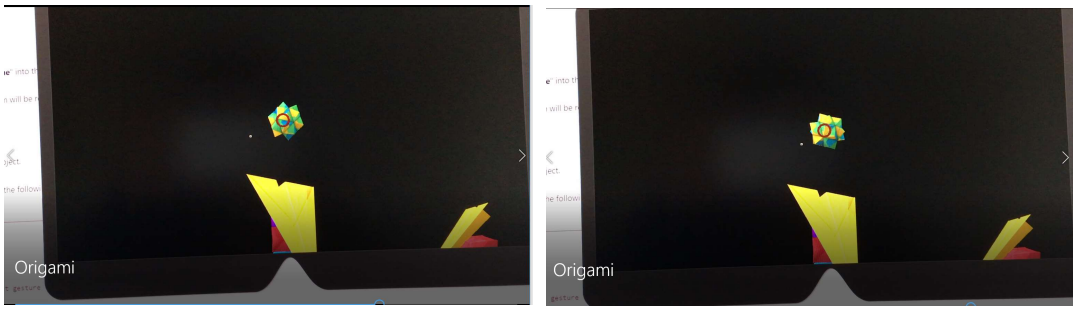


Figure 8. Origami example. The ball answers the gesture control by rotating a unit degree.

2. CNN Lego on desktop: https://youtu.be/ZS5fmjB1Mww

   The video recorded how a user plays the desktop version of CNN Lego:

   a) New layers have been generated: input tensor, pooling layer, identity block, dense layer. Every time a new layer has been instantiated, its information (type of the layer, input tensor size, layer size, filter number, filter size, stride, current active root) are displaying on the panel on the top right corner.
   b) The layer's parameters have been adjusted through text input fields or sliders. The display of the details about the selected layer on the panel top right also changed.
   c) Model is created and displayed on the panel as "current root". Layers are added to the model.
   d) Layers can be translated together with the model.
   e) Layers can be removed from a root (model/block)
   f) Layers (convolutional layer and input tensor) have been rotated and translated, to show clearly that their size/shape does not match. Thus, the output from those two shall not be added together.

   Block is created and displayed on the panel as "Active Root". Layers (convolutional layer and input tensor) have been added to the identity block. Thus, every green brick in the

[1] Implemented by changing the normal of the cursor.

model resembles an identity block.

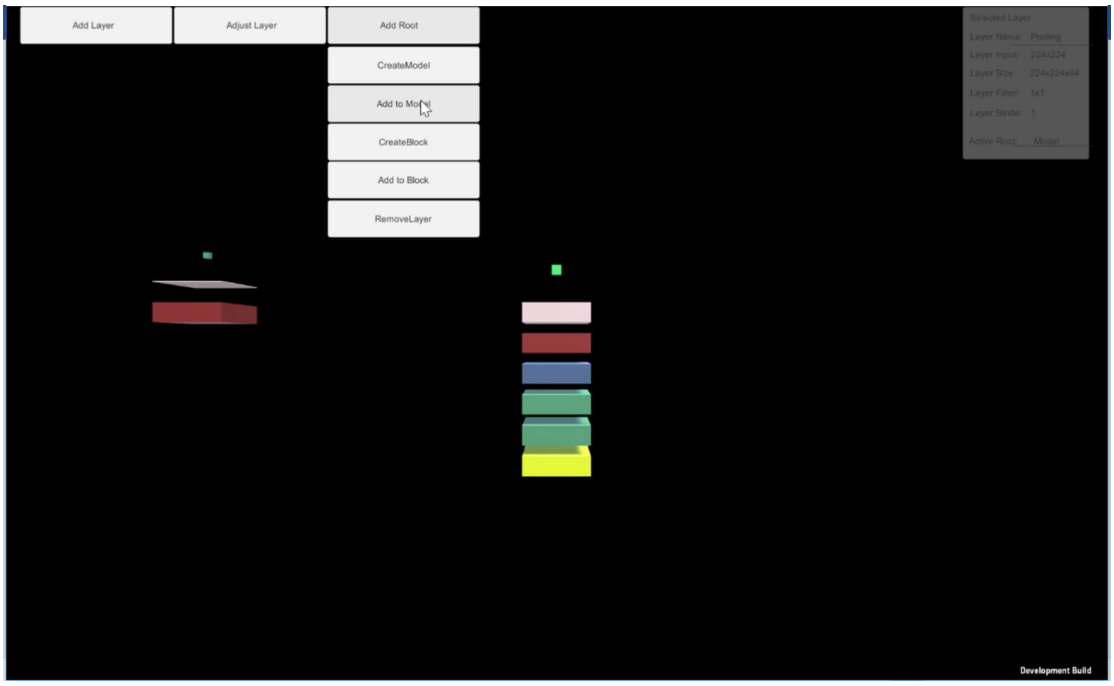


Figure 9. CNN Lego desktop version. Building a block, then use the block in the model.

3. Short demo with key functions (desktop version): https://youtu.be/alxB-uahwXI
   4 key functions have been displayed in the video recorded:
   a) Selection: the yellow brick (resembling a root) can be selected by the mouse, a red cursor is enabled and hugging its surface as the brick has been selected.
   b) Instantiate: a red brick (resembling a normal layer) can be generated, when the cursor hasn't selected any object in the scene and the key c has been pressed.
   c) Transform: when being selected, layer or root can keep rotating (when key r has been pressed) and translate a unit length to right (when key t has been pressed). To stop rotating, select the rotating object and press key r again.
   d) Group: when a layer has been selected and key g has been pressed, it will become the grandchild[1] of the root, and further, can be transformed together with the root.

4. CNN Lego Vuforia AR version: https://youtu.be/xT17Xf4GWds
   The video has been recorded as the Vuforia version CNN Lego run inside Unity, installed on a Surface 3 tablet/laptop. The functions described in video 2 have been repeated here, only in the augmented environment.

[1] An empty GameObject "buffer" of scale [1, 1, 1] has been created firstly and becomes the child of the root, after this, the selected layer is to be set as the child of this "buffer". This approach can guarantee that the original scale of the layer to be preserved regardless the scale of the root. Notice that this approach differs from the approach used in the desktop version of CNN Lego. Both approaches are reliable in keeping the scale of a GameObject unchanged while they are becoming another object's child.