SkyFall: Enabling Vison-based Interactions Using Light-weight AI Models on Resource Constrained Devices.

VICTOR DIBIA, victordibia.com

The application of deep learning has led to advances in the state of the art across multiple fields. One of the most notable examples of this is the performance of convolutional neural networks on the hard tasks of object classification, detection, and segmentation in images. More importantly, advances in the creation of lightweight CNNs such as Mobilenets, enable real time inference on resource constrained devices. In this work, I argue that deep neural network models should be considered as *first class citizens* (same as one would treat other interaction design frameworks) and their easy integration is poised to *enable new forms* of interaction on resource constrained devices. As a concrete implementation of this idea, I present a prototype of a simple physics game – Skyfall - in which users can control an onscreen paddle simply by moving their hands in front of a 2d camera. The structure of the interaction supports multiple players (provided they can be accommodated in the field of view of the camera). The system demonstrates how the integration of a well-trained and light weight hand detection model (treated as a first-class object in the interaction design) is used to robustly track player hands and enable "*body as an input"* interaction in real-time (~30 fps).

KEYWORDS

Convolutional Neural Networks, Interaction Design

1 INTRODUCTION

Imagine an interaction where a user could open an app or browser on a mobile device and immediately play a game using their body parts (recognized via camera stream) as the input method. This mode of interaction could easily extend to include common objects (cups, bottles, cell phones etc) which can also be leveraged as input proxies. All of this achieved without additional hardware (only leverages cameras which can be found on most modern devices), fully ubiquitous, unobtrusive and robust to changing environments. While body as an input device has been explored and shown to generate engagement [4,10], many of the implementations seen today require dedicated sensors (e.g Kinects, Leap Controller, Vive Lighthouse), are sometimes large limiting their ubiquity and in many cases are too complex to integrate with resource constrained devices.

Deep Neural Networks (DNNs) have been shown to achieve results across multiple fields that advance the state of the art. These networks excel in modelling complex functions and non-linearitys and enable tasks previously not possible with machines. Convolutional Neural Networks (CNNs), a type of feed-forward DNNs allow machines learn features of images that were previous manually extracted or engineered. CNNs have been successfully applied to the problem of object detection, allowing the constrcution of bounding boxes around objects for which the CNN has been trainied.

As as a starting point towards realizing the ubiquitous sensor-free class of interactions described above, his work advances the idea of leveraging DNNs and applying them to the domain of creating new types of interactions that run within resource constrained edge devices like the NVIDIA TX2.

Feb 2017.

As a concrete implementation of this idea, I present a prototype of a simple physics game – Skyfall - in which users can control an onscreen paddle simply by moving their hands in front of a 2d camera. The structure of the interaction supports multiple players (provided they can be accommodated in the field of view of the camera). The system demonstrates how the integration of a well-trained and light weight hand detection model (treated as a first-class object in the interaction design) is used to robustly track player hands and enable "*body as an input"* interaction in real-time (~30 fps).

WHY AI (DNNs) FOR Interaction?

There are several existing approaches to tracking hands in the computer vision domain. Incidentally, many of these approaches are rule based (e.g. extracting background based on texture and boundary features, distinguishing between hands and background using color histograms and HOG classifiers,) making them not very robust. For example, they are not robust to adverse conditions like noisy backgrounds, sharp changes in lighting which cause sharp changes in skin color and occlusion [5]. In many cases, they also suffer from drift prompting the need for methods that address these challenges [2,9,11] .

With sufficiently large datasets, DNNs provide opportunity to train models that perform well and address challenges of existing object tracking/detection algorithms - varied/poor lighting, noisy environments, diverse viewpoints and even occlusion. The main drawbacks to usage for real-time tracking/detection is that DNNs can be complex, are relatively slow compared to tracking-only algorithms and it can be quite expensive to assemble a good dataset. But things are changing with advances in fast neural networks. Furthermore, this entire area of work has been made more approachable by deep learning frameworks such as the Tensorflow [1] (object detection api) that simplify the process of training a model for custom object detection. More importantly, the advent of light weight neural network models like ssd [7], faster r-cnn [8], rfcn, YOLO etc. make DNNs an attractive candidate for real-time detection (and tracking) applications.

SYSTEM ARCHITECTURE

This section describes the architecture and components that comprise the Skyfall game – a model curation system and a consumer application.

Model Curation System

The model curation system is built using python and relies on the Tensorflow, OpenCV libraries. It has the following modules.

- Video Stream Processor Module [Python, Open CV]
  A threaded process that captures video frames from the NVIDIA TX2 camera and deposits them in an image input queue.
- Model Loader and Broadcast Module [Python, Tensorflow]
  Spins up multiple threads that load an interaction model into memory. Each thread fetches an image from the image input queue, runs the object detector model against the image and broadcasts the results (class tags and bounding boxes) over a websocket to all connected clients.
- Model Output Manager [Python].
  Given that each frame is processed independently by different threads, the model output manager serves to compute a history of objects seen across frames (e.g. if the hand seen

in frame^{t} is the same as the hand seen in frame^{t-1}). Currently this computation is done using a naïve Euclidean distance approach.

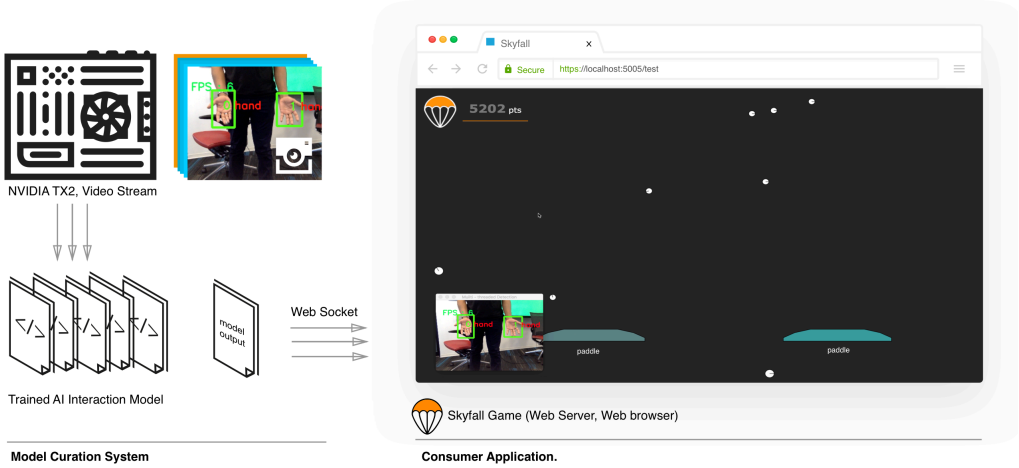

Figure 1.0 Overall System Architecture. Video stream from the camera on the TX2 tracks hand position, identifies them across frames, and streams this information over a websocket. The consumer application maps hand positions to game control paddle positions and is used to control paddles in playing the game.

Figure 2.0 Shows a user playing SkyFall. As they move their hands to the right (inset image shows what the camera sees[1]), the paddle is moved in tandem.

[1] Note: Frames per second is low when the TX2 is not clocked to maximum speed.

Trained AI Interaction Model

In this submission use case (SkyFall), the interaction model used is a hand detection model - a CNN trained on a dataset [3] of hands (4800 images). The model is trained using transfer learning via the tensorflow object detection API. The model is initialized from pre-trained model with the Mobilenet architecture [6]. I have published a longer description[2] (and code) on how this model was trained.

Skyfall Game

System Architecture

Skyfall is a browser based game served from a web application. The web application connects with a socket server on the Model Curation System. To launch the game, web application can be accessed via the webserver url.

- Play Mechanism [Javascript, Planck.js 2D physics engine, JQuery]
- The play mechanism for SkyFall is simple. 3 types of balls fall from the top of the screen in random order – white balls (worth 2 points), green balls (worth 5 points) and red balls (worth -5 points). Players earn points by moving a paddle such that each ball bounces on the paddle.
- WebSocket Module [Javascript]
- Connects to the Model Curation System websocket and listens for output from the trained AI model.
- Model to Controls Map [Javascript]
- Model output is mapped to game controls or used to generate new content within the game. In the SkyFall example, the position of the plyers hands (x-axis) is used to position the play paddle.

NVIDIA TX2 Setup

The TX2 used in testing the application was flashed using the latest Jetpack version 3.1. Next, Tensorflow was built from source on the TX2 and installed. To allow for easy implementation and testing, most of the coding and testing process was done using Tensorflow on a Mac computer and then intermittently tested on the TX2.

Clocking the NVIDIA TX2 for max performance (jetson_clock.sh) provided a significant speed up in performance – hand detection frames per second increased from 4 FPS to 21 FPS.

IMPACT OF THIS WORK

Improved Interaction Design

By advancing the concept of lightweight AI models as first class citizens in interaction design, this work demonstrates avenues for creating user experiences with improved immersion, engagement and enjoyment for resource constrained devices.

---

[2]Victor Dibia | Real-time Hand-Detection using Neural Networks (SSD) on Tensorflow. https://github.com/victordibia/handtracking

A community of continuously optimized AI models

This work also represents a first step towards curating a community of lightweight optimized AI models that specialize in certain interaction tasks. This may include models for tracking body parts, language processing, speech processing etc. As a starting point, I have made the trained AI model for hand tracking available as an open source Tensorflow checkpoint model which other users can integrate into the development of their own applications.

Demonstrating Capabilities on Edge Devices like NVIDIA TX2

This project gives a sense of the extensive processing capabilities of the TX2. My tests surprisingly showed better FPS compared to a MacBook pro (i7, 2.4GHz, Quadcore). This work also can become an initialization point for deploying meaningful compact systems that leverage computer vision for multiple use cases. For example, by tracking hands in real time we can develop a compact system capable of tracking gestures and using that to predict how expressive a presenter is as they navigate the stage during a presentation. Given that hands (and joints) are a critical part of dance expressing, similar models can be learned to score dance performance (rhythm) with a given musical piece.

NEXT STEPS/FUTURE WORK

Future work for this project includes improvements on software, interaction model accuracy and some implementation optimizations.

Software

Improved object identification across scenes.

Currently the algorithm used to identify objects across frames is a naïve algorithm based on Euclidean distance. Further work needs to be done to improve this algorithm and possibly integrate it with other available tracking algorithms.

Improved translation of hand positions and game controls

More work will be done to improve the mechanisms for mapping hand location to game controls. This will ensure smoother controls, better multiplayer experience (current multiplayer is slightly glitchy), etc.

Improved game mechanics, New Games/Interaction Use cases

More work will be done to improve the game play mechanics to improve engagement. Work will also be done to implement this in new games (e.g. platformers where the primary interactions can be hand movements) and interaction use cases .

Model Accuracy

Better Training Dataset for hand tracking model

Qualitative evaluations suggest the hand tracking model excels at tracking hands in multiple lighting and orientation conditions. However, it is still limited and there is opportunity to further improve this accuracy by assembling a larger and more diverse dataset. This will be a critical part

of future work and will amount to benefits that can be shared by others who use the hand tracking model.

Implementation Optimizations

Speed Optimization - Integrating TensorRT

Further work is needed to ensure this application takes advantage of optimizations within TensorRT to improve performance.

CONCLUSION

In this work, I have presented some ideas around integrating AI as a first-class citizen in designing interactions. I present a concrete example of this using a simple physics based game – SkyFall. This approach can be expanded beyond the capabilities shown within SkyFall. First it can be extended to allow integration of other types of models such as light-weight speech, language generation, generative models etc. The approach can leverage benefits from continuous improvements (better data, better optimizations) in each of these models. Finally, the approach can be expanded to allow for simultaneous integration of multiple pre-trained community models – e.g. a model trained to detect hands in addition to a model trained to detect raccoons in implementing a “raccoon catcher game on live video”.

REFERENCES

1. Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mane, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viegas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. Retrieved November 20, 2017 from http://arxiv.org/abs/1603.04467
2. Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. 2011. Robust Object Tracking with Online Multiple Instance Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 8: 1619–1632. https://doi.org/10.1109/TPAMI.2010.226
3. Sven Bambach, Stefan Lee, David J. Crandall, and Chen Yu. 2015. Lending A Hand: Detecting Hands and Recognizing Activities in Complex Egocentric Interactions. In *ICCV*, 1949–1957. Retrieved November 3, 2017 from https://www.cv-foundation.org/openaccess/content_iccv_2015/html/Bambach_Lending_A_Hand_ICCV_2015_paper.html
4. Max Birk and Regan L. Mandryk. 2013. Control your game-self: effects of controller type on enjoyment, motivation, and personality in game. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI '13*: 685–694. https://doi.org/10.1145/2470654.2470752
5. Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, and Xander Twombly. 2007. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding* 108, 1–2: 52–73. https://doi.org/10.1016/j.cviu.2006.10.012
6. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. 2017. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. Retrieved November 3, 2017 from http://arxiv.org/abs/1704.04861
7. Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng Yang Fu, and Alexander C. Berg. 2016. SSD: Single shot multibox detector. In *European conference on computer vision*, 21–37. https://doi.org/10.1007/978-3-319-46448-0_2
8. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 6: 1137–1149. https://doi.org/10.1109/TPAMI.2016.2577031
9. David a. Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. 2007. Incremental Learning for Robust Visual Tracking. *International Journal of Computer Vision* 77, 1–3: 125–141. https://doi.org/10.1007/s11263-007-0075-7
10. Daniel M. Shafer, Corey P. Carbonara, and Lucy Popova. 2011. Spatial presence and perceived reality as predictors of motion-based video game enjoyment. *Presence: Teleoperators and Virtual Environments* 20, 6: 591–619. https://doi.org/10.1162/PRES_a_00084
11. Wei Zhong, Huchuan Lu, and Ming-Hsuan Yang. 2012. Robust object tracking via sparsity-based collaborative model. *Cvpr*: 1838–1845. https://doi.org/10.1109/CVPR.2012.6247882