

# **Virtual Reality Hand Pose Classification with TensorFlow Estimators**

Mike O'Connor

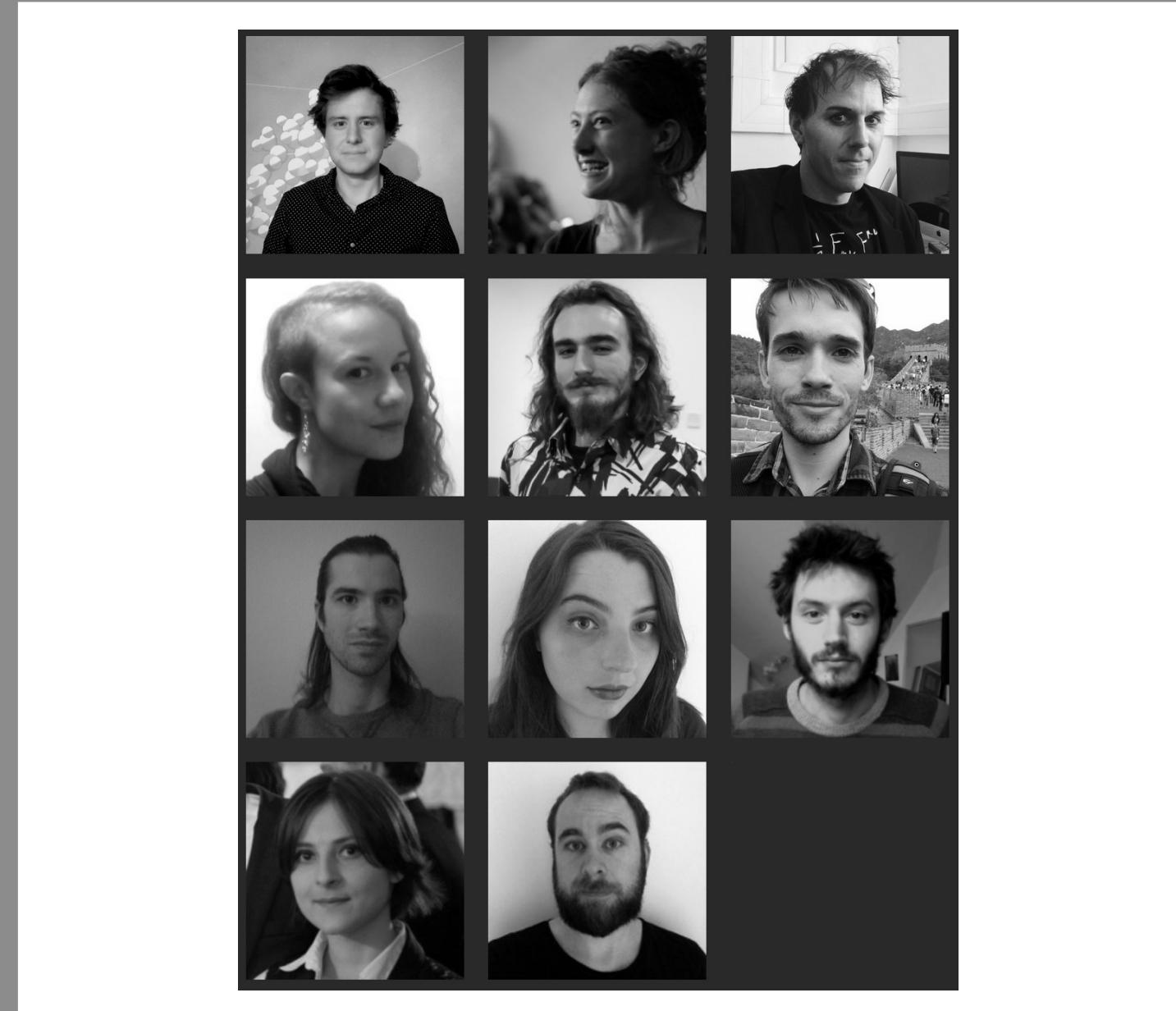
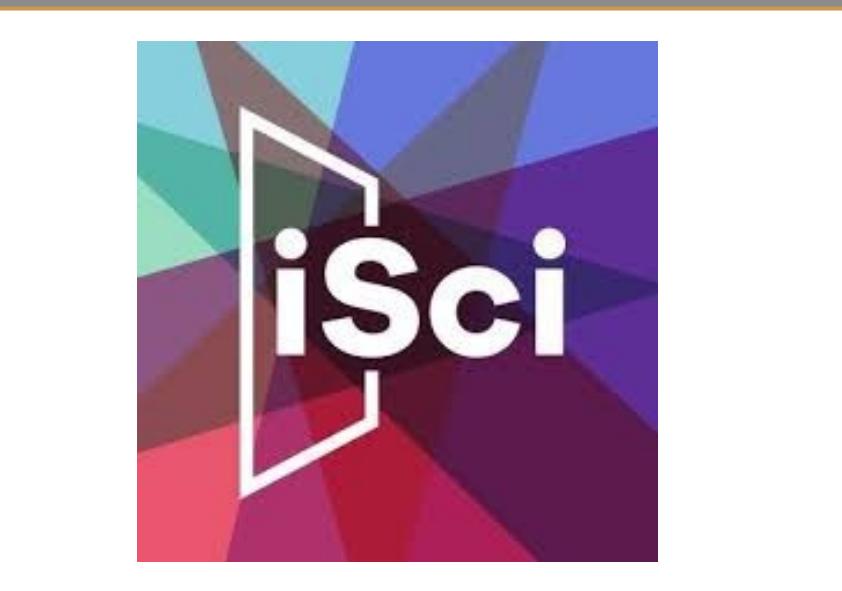
Dept. of Computer Science & School of Chemistry  
[mikeoconn.org](http://mikeoconn.org)



# This talk

- A little bit about molecular simulation in VR
- VR Gloves, and the need for gesture detection.
- Building a classifier with TensorFlow Estimators.
- An end-to-end prototype for a machine learning application.
- **Disclaimer:** I am not a TensorFlow expert by any means!

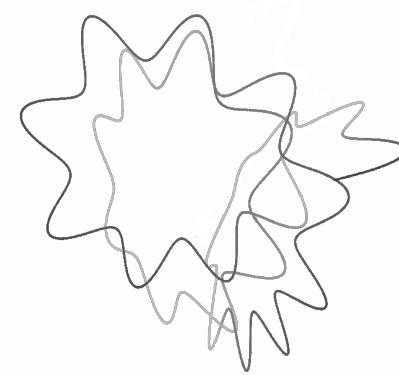
Code: [github.com/mikeoconnor0308/tensorglove](https://github.com/mikeoconnor0308/tensorglove)



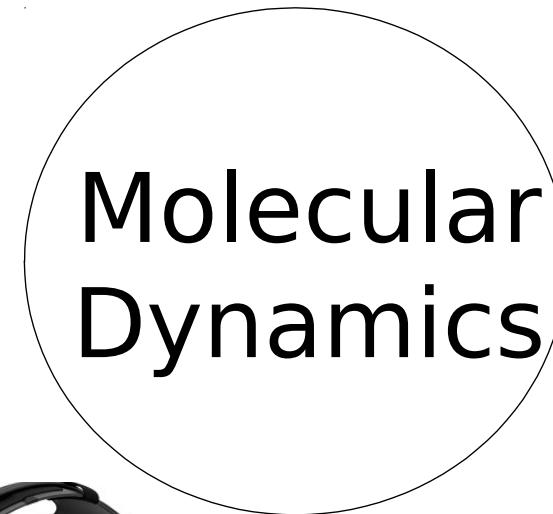
# We span a lot of disciplines...



Super-  
Computing



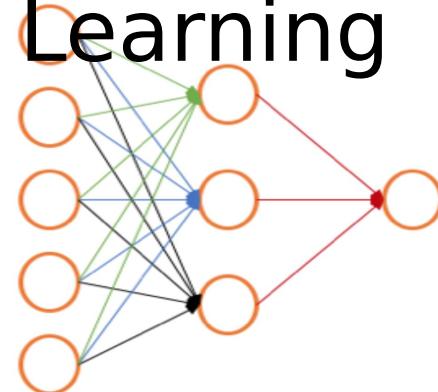
Digital  
Arts

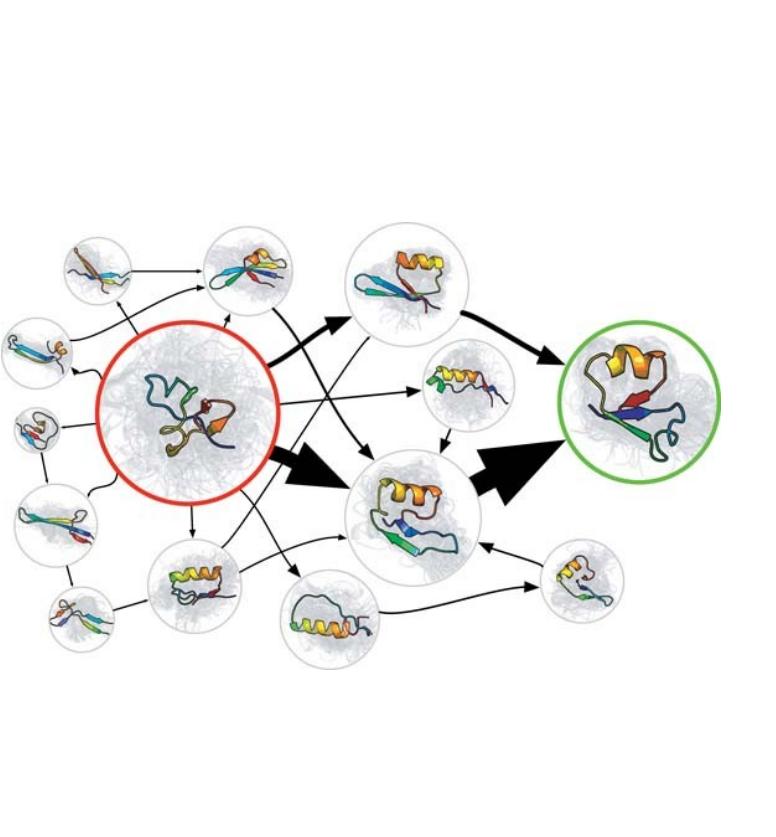


Human  
Computer  
Interaction



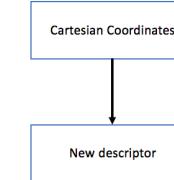
Machine  
Learning



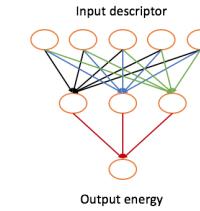


Molecule 1	Energy 1
Molecule 2	Energy 2
Molecule 3	Energy 3
Molecule 4	Energy 4
.	.
.	.
Molecule N	Energy N

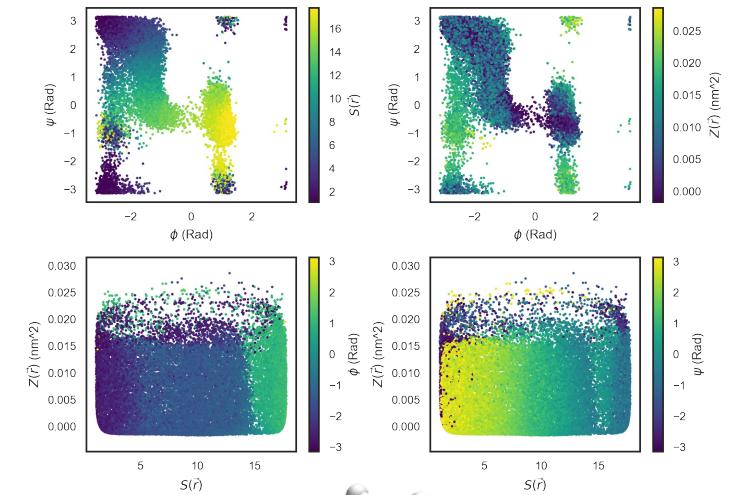
Training data



Convert data to  
suitable input

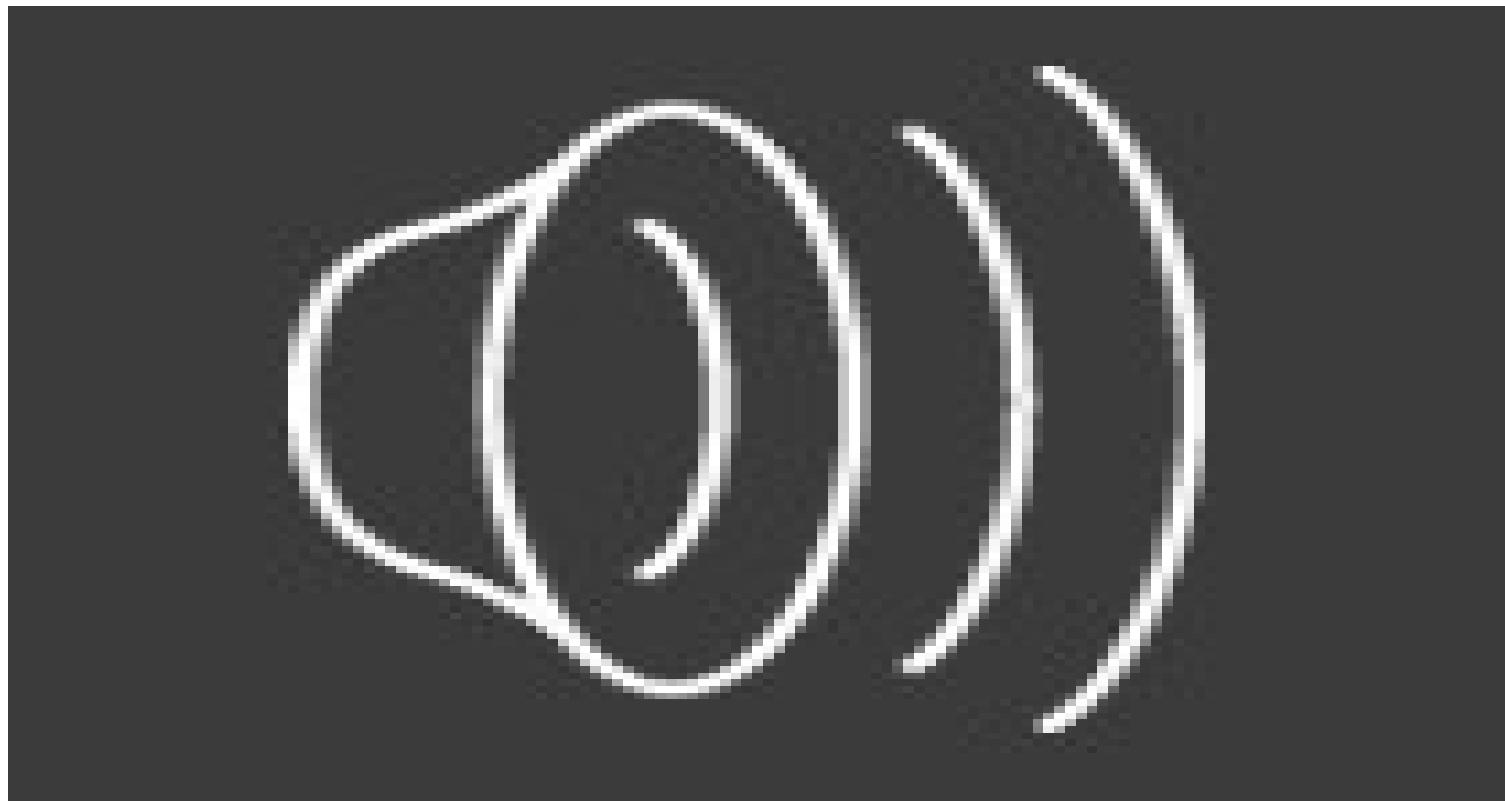


Train neural  
network



# Open source data tools make science happen.

# Interactive Molecular Dynamics in VR



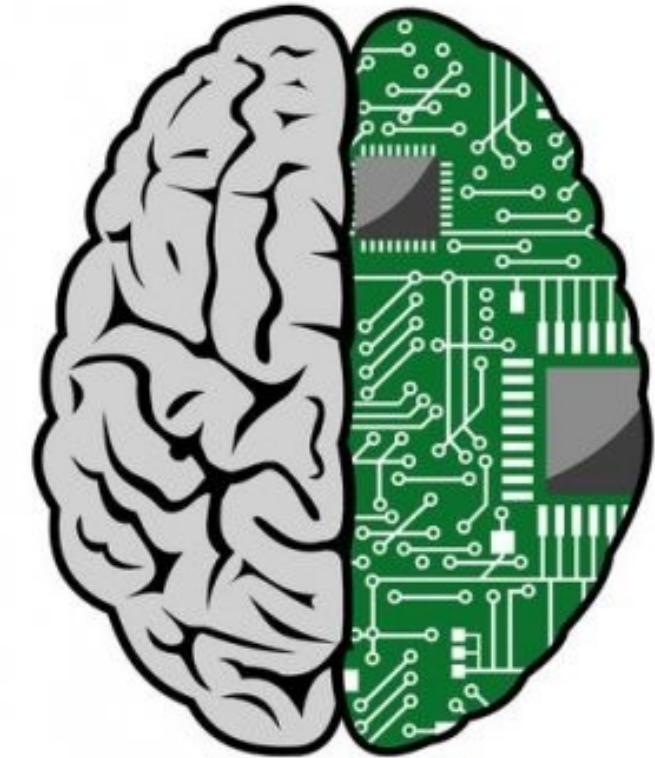
O'Connor, M. et al, *Science Advances*,  
4(6), 2018

# Give it a go!

<https://isci.itch.io/nsb-imd>

# Scientific ‘gamification’

Casting scientific problems as games that scientific players can solve



Which computer is faster?

Can we design algorithms that let machines learn from the experts?



# Wouldn't it be great to use our hands in VR?



# VR gloves are now commercially available



# The Glove Demo Scene



# Tensorflow\* Hackathon

- No gesture controls in the SDK!
- Let's build a classifier that detects hand poses based on the glove data.
- 2 days – aim for end-to-end prototype.
- Use the Premade Estimators to speed things up.

# \*Other Methods Are Available.

Google Scholar search results for "hand gesture recognition".

Search term: hand gesture recognition |

Autocomplete suggestions:

- hand gesture recognition system
- hand gesture recognition using kinect
- hand gesture recognition algorithm
- hand gesture recognition survey
- hand gesture recognition technique
- hand gesture recognition project
- hand gesture recognition using neural networks
- hand gesture recognition using depth data
- hand gesture recognition review
- hand gesture recognition algorithm fast

Result 1: [PDF] ttu.edu.tw

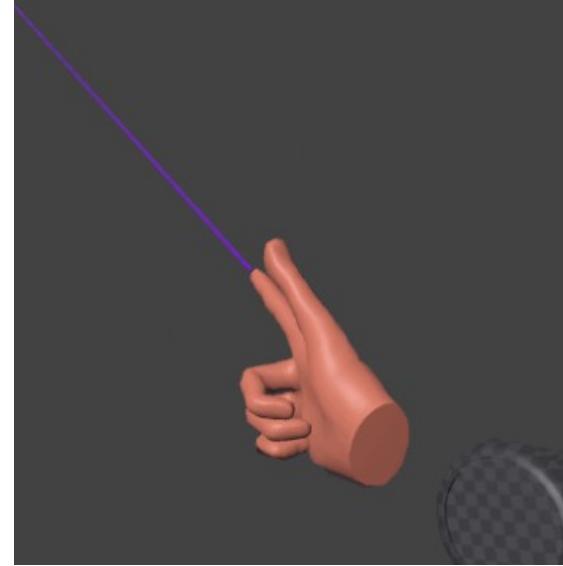
**Robust hand gesture recognition with kinect sensor**  
Z Ren, J Meng, J Yuan, Z Zhang - Proceedings of the 19th ACM ..., 2011 - dl.acm.org  
Abstract Hand gesture based Human-Computer-Interaction (HCI) is one of the most natural and intuitive ways to communicate between people and machines, since it closely mimics how human interact with each other. In this demo, we present a hand gesture recognition ...  
☆ 99 Cited by 818 Related articles All 10 versions »

Get It! @ UoB

Result 2: [HTML] sciencedirect.com

**Hand gesture recognition using a real-time tracking method and hidden Markov models**  
FS Chen, CM Fu, CL Huang - Image and vision computing, 2003 - Elsevier  
In this paper, we introduce a hand gesture recognition system to recognize continuous gesture before stationary background. The system consists of four modules: a real time hand tracking and extraction, feature extraction, hidden Markov model (HMM) training, and ...  
☆ 99 Cited by 569 Related articles All 9 versions Web of Science: 219

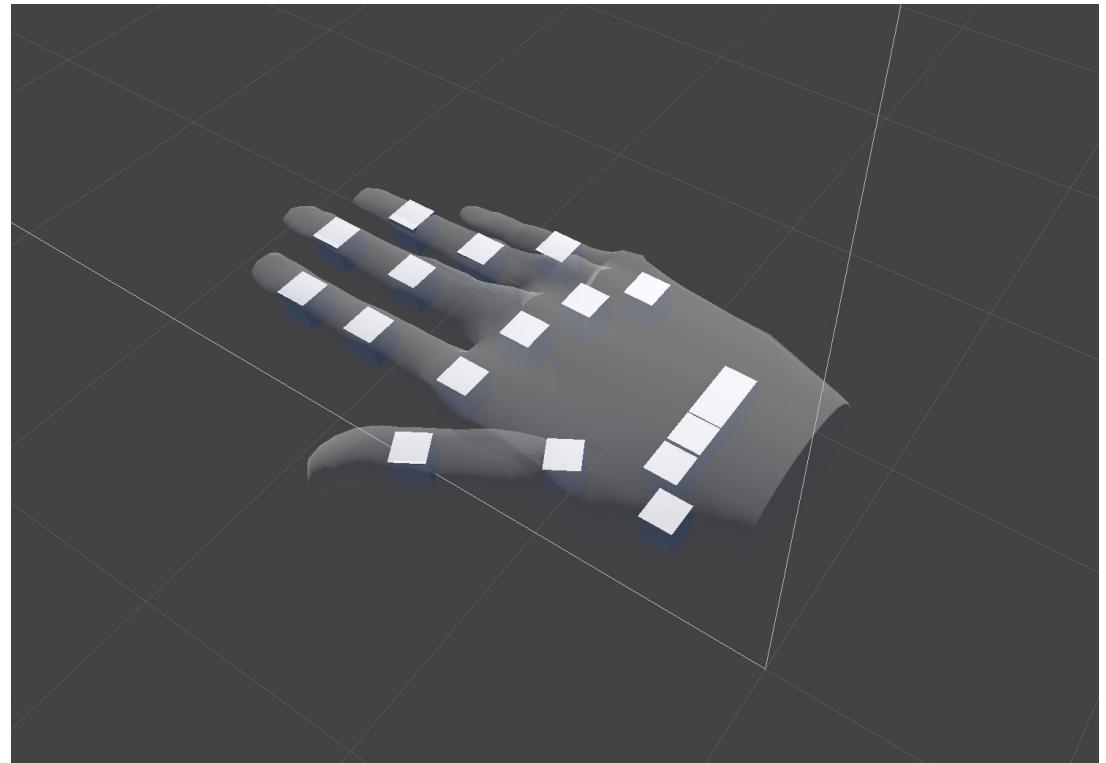
Get It! @ UoB



# Data Collection

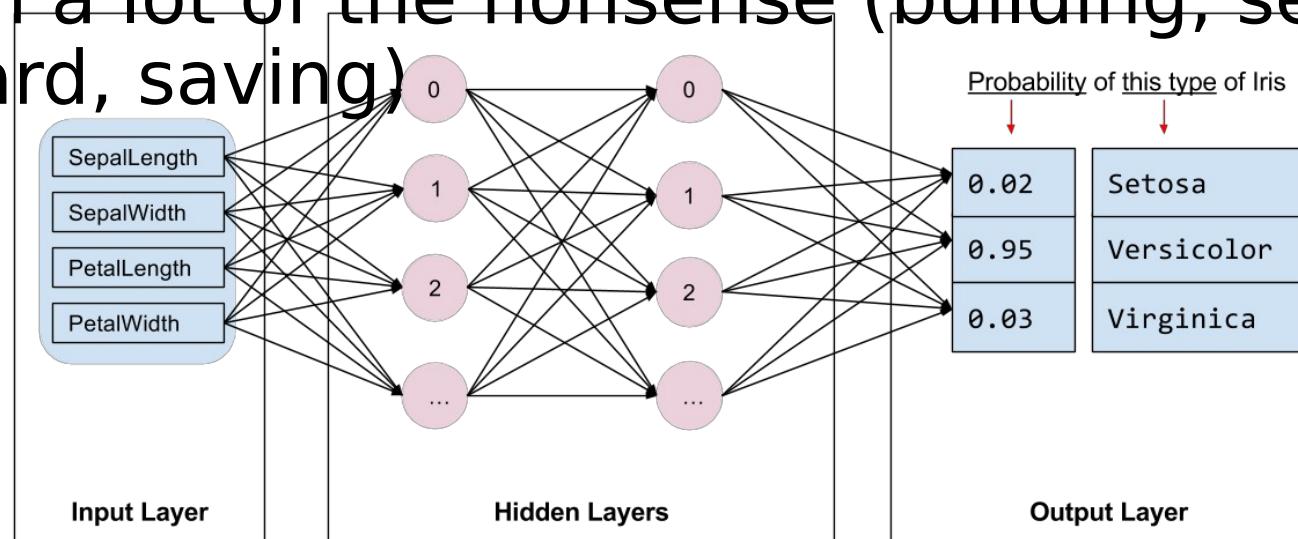
# Feature Selection (the dumb way)

- Gloves only provide rotation data for each joint.
- 21 Quaternions => 84 floats.
- No access to raw sensor data.
- We're building a classifier on top of a model!!



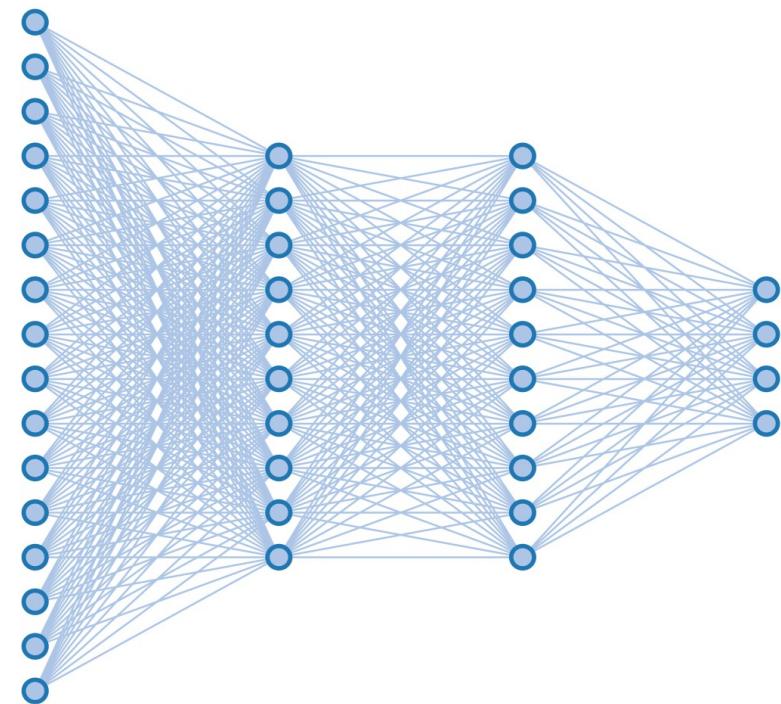
# Tensorflow Estimators

- High-level representation of a model.
- A midway point before high level API such as Keras.
- Pre-made estimators for common tasks: **classification**, regression.
- Deals with a lot of the nonsense (building, session, tensorboard, saving)



# Out-of-the-box model.

```
# Feature columns describe how to use the input.  
my_feature_columns = []  
for key in train_x.keys():  
    my_feature_columns.append(tf.feature_column.numeric_column(key=key))  
  
# Build 2 hidden layer DNN with 10, 10 units respectively.  
classifier = tf.estimator.DNNClassifier(  
    feature_columns=my_feature_columns,  
    # Two hidden layers of 10 nodes each.  
    hidden_units=[10, 10],  
    # The model must choose between 4 classes.  
    n_classes=4,  
    model_dir="model")  
  
# Train the Model.  
classifier.train(  
    input_fn=lambda: glovedata.train_input_fn(train_x, train_y,  
                                              args.batch_size),  
    steps=args.train_steps)  
  
# Evaluate the model.  
eval_result = classifier.evaluate(  
    input_fn=lambda: glovedata.eval_input_fn(test_x, test_y,  
                                              args.batch_size))
```



# Success!!

- 96% Accuracy on test set. (80/20 split).
- Now, we just need to get the predictions to the VR!



# Serving a Tensorflow Model: A Survey

# Option 1) Use the `Estimator.predict` method.

```
predictions = classifier.predict(  
    input_fn=lambda: glovedata.eval_input_fn(predict_x,  
                                              labels=None,  
                                              batch_size=args.batch_size))
```

- Convenient, but...
- By default, reloads model every call.
- How to communicate this to the VR app?

# Option 2) Use TensorFlow serving.

## Serving a TensorFlow Model



This tutorial shows you how to use TensorFlow Serving components to export a trained TensorFlow model and use the standard `tensorflow_model_server` to serve it. If you are already familiar with TensorFlow Serving, and you want to know more about how the server internals work, see the [TensorFlow Serving advanced tutorial](#).

This tutorial uses the simple Softmax Regression model introduced in the TensorFlow tutorial for handwritten image (MNIST data) classification. If you do not know what TensorFlow or MNIST is, see the [MNIST For ML Beginners](#) tutorial.

The code for this tutorial consists of two parts:

- A Python file, [`mnist\_saved\_model.py`](#), that trains and exports the model.
- A ModelServer binary which can be either installed using Apt, or compiled from a C++ file ([`main.cc`](#)). The TensorFlow Serving ModelServer discovers new exported models and runs a [gRPC](#) service for serving them.

# Option 2) Use TensorFlow serving.

- This is the way to do it for **production**.
- Images not available for Windows. I'd have to compile it all.
- Or run a container. Again, not great on Windows (no GPU).
- Requires gRPC communication from the client.
- Seems like overkill for this hack!

# Option 3) “Freeze” the model

The screenshot shows a GitHub repository page for tensorflow/tensorflow. The repository has 8,326 stars and 67,490 forks. The 'Code' tab is selected, showing the file tensorflow/python/tools/freeze\_graph.py. The file has 492 lines (449 sloc) and is 18.2 KB. The commit history shows a recent update by itsmeolivia adding docstrings to the freeze\_graph function. The code itself is a copyright notice for TensorFlow, version 2.0, under the Apache License.

```
1 # Copyright 2015 The TensorFlow Authors. All Rights Reserved.  
2 #  
3 # Licensed under the Apache License, Version 2.0 (the "License");  
4 # you may not use this file except in compliance with the License.  
5 # You may obtain a copy of the License at  
6 #  
7 #     http://www.apache.org/licenses/LICENSE-2.0  
8 #  
9 # Unless required by applicable law or agreed to in writing, software  
10 # distributed under the License is distributed on an "AS IS" BASIS,  
11 # WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
12 # See the License for the specific language governing permissions and
```

# Option 3) “Freeze” the model.

- Freezes a graph by converting all Variables to fixed values.
- Can then be used by C/C++/C# APIs.
- This is how the Unity3D/TensorflowSharp plugins work.
- A good option, but figuring out how to set the inputs/outputs can be fiddly.

# I chose Option 1



# Communication: Open Sound Control (OSC)

/predict, X, Y, Z, W . . .

```
from pythonosc import osc_server  
dispatcher.map("/predict", self.predict)  
server = osc_server.ThreadingOSCUDPServer(  
    ("localhost", 54321), dispatcher)
```

It works!! But it's really slow...



**THREE  
WEEKS  
LATER...**

Physical and Theoretical Chemist

Sign in

Physical and Theoretical Chemistry Laboratory

4.5 ★★★★☆ • 2 reviews

Laboratory

Directions

SAVE NEARBY SEND TO YOUR PHONE SHARE

South Parks Road, Oxford OX1 3QZ

QP5W+JP Oxford

physchem.ox.ac.uk

01865 275400

Opens at 08:00

Claim this business

SUGGEST AN EDIT

Institute of Cognitive and Evolutionary...

Faculty of Theology and Religion

University of Oxford - Social Policy and...

School of Archaeology

Magdalen College

Oxford Castle

History of Art Department

Christ Church Meadow

Magdalen Col

O2 Academ

John Radcliffe Hospital

Oxford Brookes University

South Parks Rd

St Cross Rd

St Catherine's College

River Thames

Banbury Rd

Kingston Rd

A4144

A420

A4144

B4150

B4150

B4495

B4495

B4495

Google

Satellite

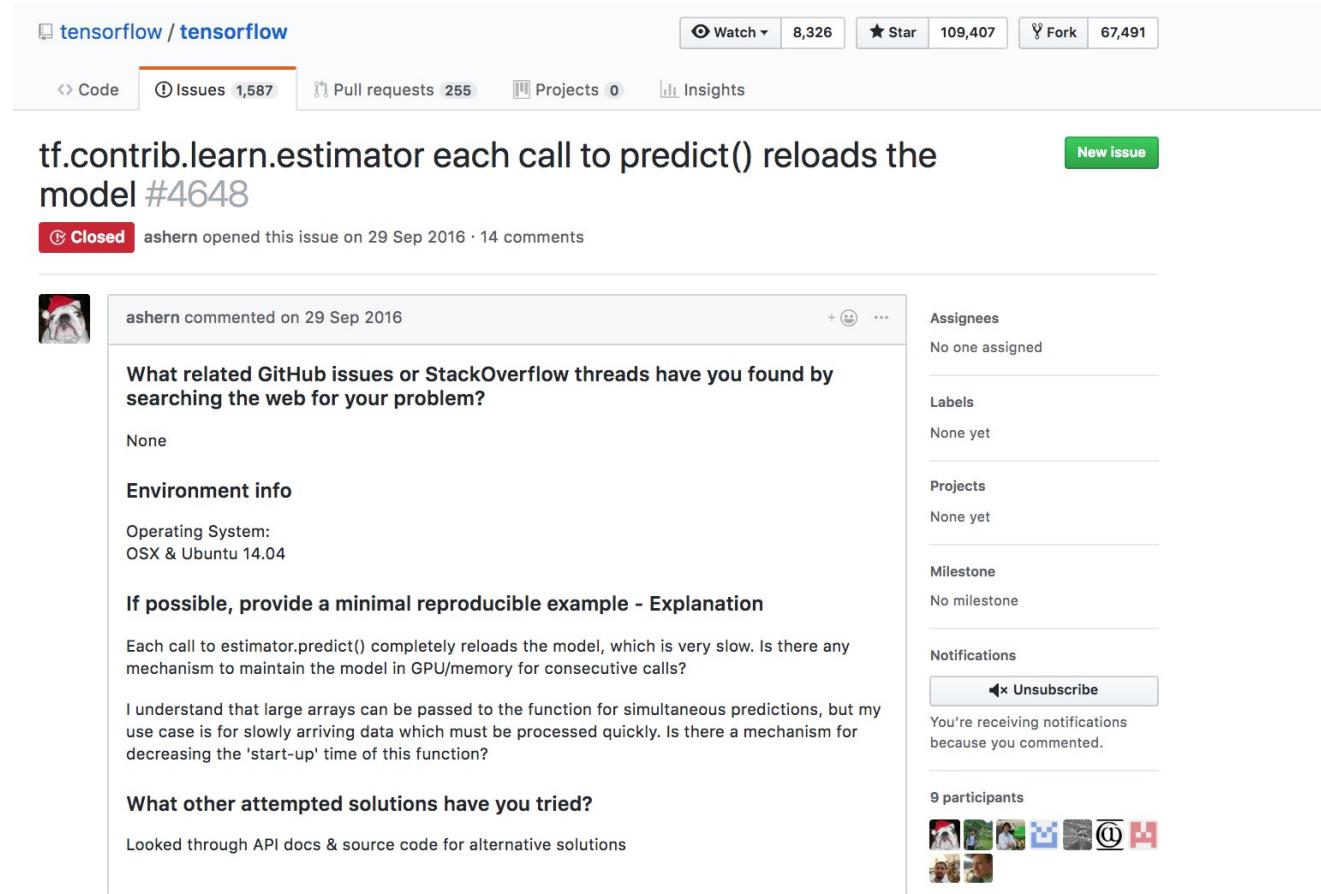
Map data ©2018 Google United Kingdom Ter

TMCS

# 1 more day: Some Hyperparameter Optimisation

- More training data.
- Use 5-fold cross validation.
- Perform a grid search on hidden layer sizes and learning rate.
- Hidden layer sizes now 18 and 20.
- Increased to 99% accuracy. **Overfitted!**
- More sophisticated optimisations out of scope.

# Faster Prediction: Saved by GitHub issue board



**tf.contrib.learn.estimator each call to predict() reloads the model #4648**

**Closed** ashern opened this issue on 29 Sep 2016 · 14 comments

ashern commented on 29 Sep 2016

What related GitHub issues or StackOverflow threads have you found by searching the web for your problem?

None

**Environment info**

Operating System:  
OSX & Ubuntu 14.04

**If possible, provide a minimal reproducible example - Explanation**

Each call to estimator.predict() completely reloads the model, which is very slow. Is there any mechanism to maintain the model in GPU/memory for consecutive calls?

I understand that large arrays can be passed to the function for simultaneous predictions, but my use case is for slowly arriving data which must be processed quickly. Is there a mechanism for decreasing the 'start-up' time of this function?

**What other attempted solutions have you tried?**

Looked through API docs & source code for alternative solutions

Assignees  
No one assigned

Labels  
None yet

Projects  
None yet

Milestone  
No milestone

Notifications  
**Unsubscribe**  
You're receiving notifications because you commented.

9 participants



# Faster Prediction: Saved by GitHub issue board



marcsto commented on 6 Jul 2017



In case someone else runs into this, predict can take a generator as a parameter.

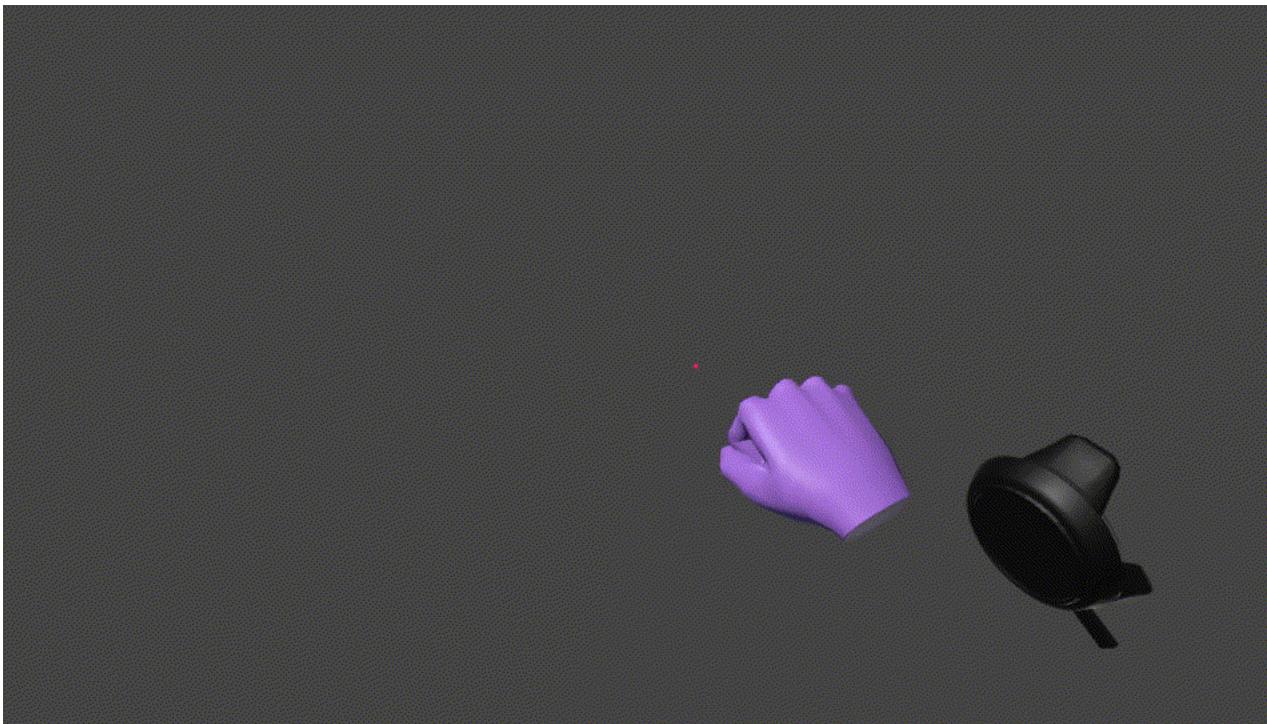
Here's a class that wraps an estimator with a generator. This means you can make multiple calls to predict without reloading the graph.

[https://github.com/marcsto/rl/blob/master/src/fast\\_predict.py](https://github.com/marcsto/rl/blob/master/src/fast_predict.py)



1

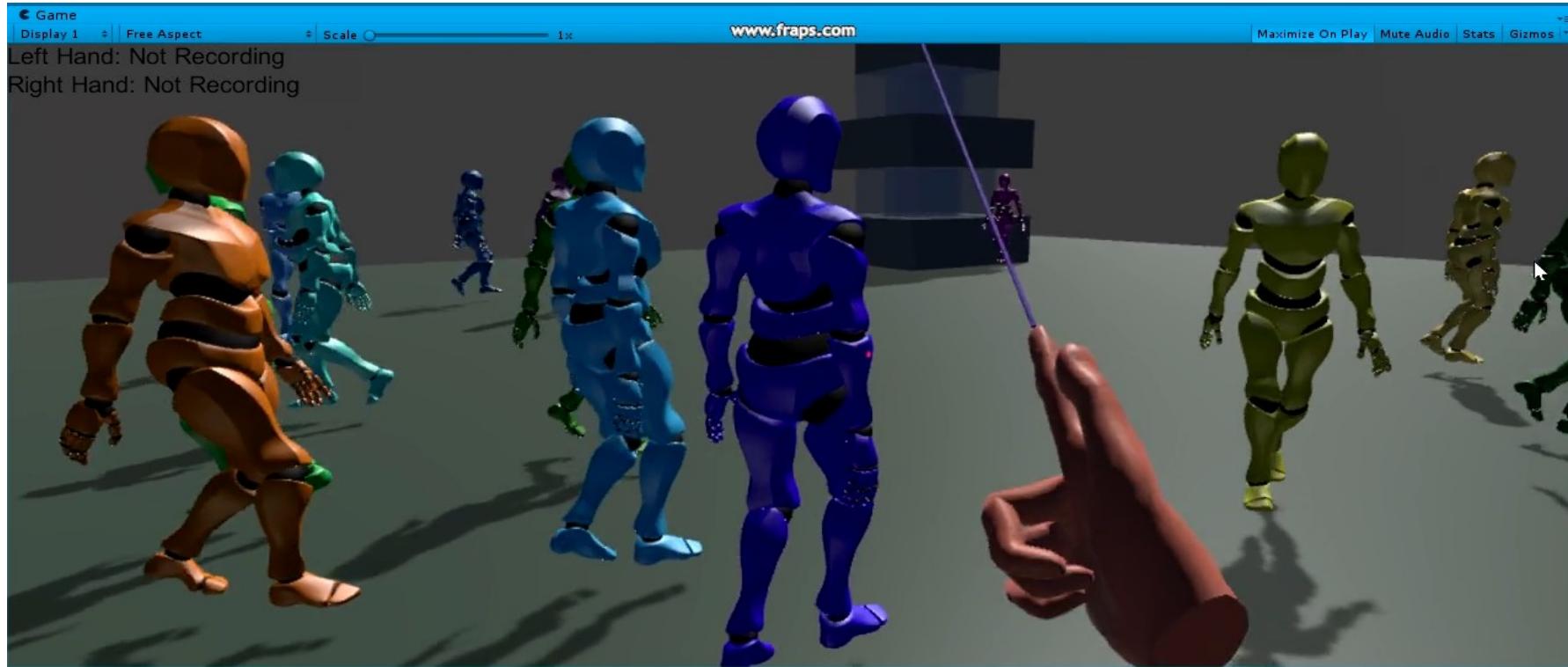
# Putting it all together



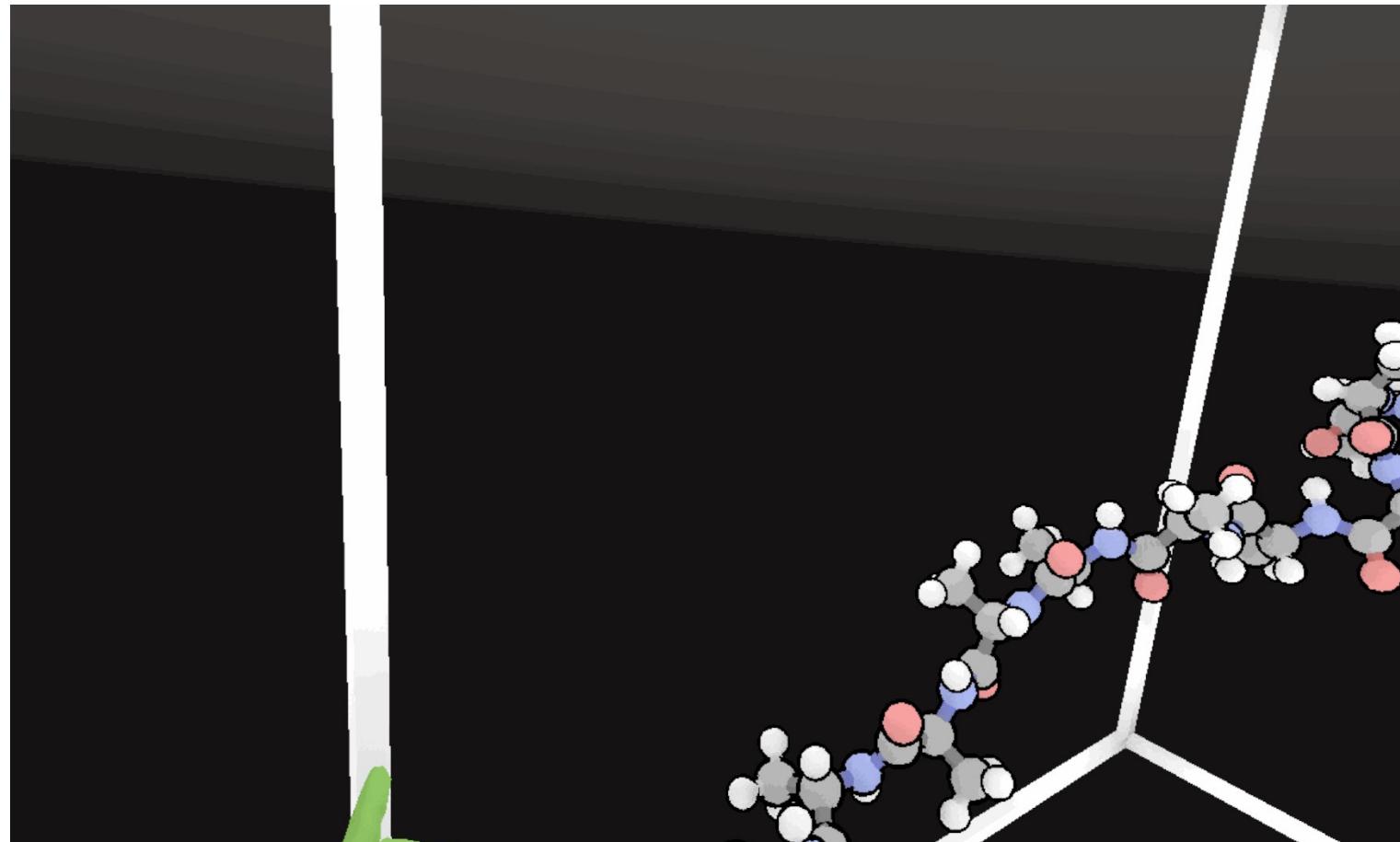
# Garbage in, Garbage out.

- Classification very sensitive to rotations.
- Quaternions not the best features to use.
- Internal coordinates, or raw sensor data would be better.
- The gloves are a pain to calibrate!

# Meanwhile, one student is clearly a Sith...



# Tying a peptide in a knot



# Conclusions

- It sort of works! But is also completely useless.
- TensorFlow estimators make it easy to get going.
- Also a good way to package up a model. Keras -> Estimator script exists.
- Can focus on features, the model, and hyperparameters.
- But Keras/scikit-learn have better support for pipelines/tuning.
- Fast-changing API makes trawling through the docs/issues difficult.
- Serving/exporting to other languages is a tricky maze to navigate.

**Dear Hi5 Customers,**

We have some new updates for UNITY including an interaction SDK. The new UNITY for Hi5 update includes correction of several crash issues with project files, and improvement of lagging issues.



### **UNITY INTERACTION SDK**

The Hi5 UNITY Interaction SDK supports the interactions between hands and objects. Currently the functions include grabbing, pinching, releasing, throwing, lifting, poking, hand pressing and exchanging objects. More functionality will be added in future versions.



# Thanks!

---