Introduction

Background

Gestures, particularly hand gestures, are a versatile and intuitive method for humans to interact. It is also being increasingly looked towards as the next step in human machine interfaces due to its ability to provide more types of input, ease and not necessitating physical contact to function (Brian Dipert, 2013) (Chen, 2013). This is increasingly necessary as devices and systems become increasingly complex. The development of gesture based user interface (UI) can be seen in development of gesture interfaces from vehicles such as Jaguar's vehicle control system (Yanan Xu, 2017) to consumer products such as Microsoft's Kinect and LEAP Motion, Virtual Reality devices and remote triggering of drones. These employ gesture recognition to trigger functions. Gestures have been shown to be less disruptive as an interface and are expected to be used significantly as an interface within the next 5 years (Carl A. Pickering, 2007).

Engineering Challenge

Some of the limiting factors to current usage of gesture UI are reliability when using simple setups and power consumption. These can be seen in the significant reduction in a significant drop in user satisfaction when the system has reliability below a threshold (Liu, 2016). This was found to be around 70% in the study and was attributed to increased effort.

Power consumption has limited use of gesture UI in untethered and handheld applications. These can be seen in examples such as Google's project Soli or Google Glass. In project Soli there was a tradeoff in power consumption and resolution (Jaime Lien, 2016). Resolution was directly correlated with performance in recognition of gestures, meaning that it was still limited in providing a reliable and low power gesture recognition interface. Google glass whose image sensing and computation consumed roughly 50% of the power budget (Robert Likamwa, 2013).

In addition to low power consumption and reliability using simple setups, requirements such as low latency, small size and robustness to different scenarios are necessary for gesture UIs.

Literature Review

Gesture recognition interface hardware

Looking into the literature, some hardware used for gesture recognition include pixel cameras such as normal video web-cameras, radar reflections such as Project Soli (Jaime Lien, 2016), wireless reflections such as AllSee (Gollakota, 2014), structured light cameras such as Microsoft's Kinect (Maruvada, 2017), binary gradient cameras (Jayasuriya, 2016) and neuromorphic vision systems. This paper will focus on the neuromorphic vision system as it is an approach with growing research, accessibility resources and several advantages in the field of gesture UIs.

Neuromorphic Vision System (NVS)

Neuromorphic Vision Systems are also known by the hardware of Dynamic Vision Sensors (DVS), Event-Based Vision or Retina Vision. These devices employ hardware inspired by biological retinas and only produce data when an event occurs. Specifically, it responds to a spike. A spike occurs when there is a change in logarithmic brightness at a pixel above a threshold. The camera responds by providing the pixel position, timestamp and a polarity of the change. Each pixel acts independently and asynchronously (Henri Rebeceq, 2017). This allows microsecond resolution and latency, milliwatt power consumption and redundancy removal. Specifically, removal of static objects such as the background (Christian Brandli, 2014). These features make NVS ideal for use in gesture UIs, since they address the issues of power consumption, latency and simplicity directly. Pre-processing of data through the inherent removal of static objects, detection of change and ability to resolve dynamic changes to greater detail are useful in addressing reliability.

The reduction in power consumption can be attributed to static pixels producing no output and the lack of analogue to digital converters (ADCs) in DVS (Stefanie Anna Baby, 2018). The ADC is a result of brightness levels that the DVS lacks, instead relying on a polarity only. Latency reduction and time resolution are a result of pixels being independent and firing when they detect changes rather than waiting for the frame in frame-based video methods (pixel cameras, binary gradient cameras and structured light cameras).

NVS State of the Art

NVS has been shown to be highly successful in classification, localization and 3D applications.

An successful classification example is IBM research, UC San Diego and ETH Zurich's convolutional neural network (CNN) gesture recognition system which achieved 105ms gesture detection latency, 96.5% accuracy and 200mW power consumption for categorizing 11 hand gestures (Arnon Amir, 2017), demonstrating that NVS could achieve latency, accuracy and power consumption simultaneously even when processing was taken into account. Classification of gestures based on hand tracking with 96.96% recognition rate was demonstrated by Samsung, University of Zurich and ETH Zurich. In addition, implementation of a gesture command mode that controlled a cursor by tracking the user's hand was demonstrated (Lee, et al., 2012).

Similar localization and tracking of fingertip was also demonstrated in a separate study (Lee, et al., 2012). These applications used Leaky Integrate-and-Fire (LIF) neurons which meant they had difficulty handling situations where the hand was not the major moving object. This makes clear the difficulty in localization for NVS now.

NVS has also been demonstrated to be able to extract 3D and texture information with multiple implementations of reconstruction of environments even without stereo vision (Matia Pizzoli, 2014). This understanding of depth could be useful to understand gestures due to the need to differentiate between orientation of a hand and understanding which fingers are in front or behind.

DVS images have also been shown to be robust when operating at various distances (Paul K. J. Park, 2016).

Due to the event-based format of DVS data and being relatively new, NVS applications often employ different architectures and methods.

Convolutional Neural Network (CNN)

One of the architectures used in gesture recognition is the convolutional neural network. This is a multilayer feedforward network composed of convolution layers, pooling layers and fully connected layers.

Convolution layers are where the neurons in a layer are calculated by convolving sections of the previous layer with filters (made of weights that will be learned) of a smaller size than the original image. These convolutions occur in the two-dimensional space that makes up an image and the sections are chosen in a systematic way. The window which the filter convolves is shifted along a set number of pixels (stride) and the original image is often padded with zeros to allow extraction at the edges. An activation function is then applied to the result of the convolution to form a convolution layer in the network. The activation function creates non-linearity and allowing the weights to learn in a meaningful manner, causing the result to become a neuron. The activation function chosen often is the Linear rectifier (ReLU) function. This function sets any negative value to zero and is often used due to being easy to compute, not saturating the output (gradient tending to zero for extreme values) and causes faster convergence toward optimal weights. Temporal convolutions are also used to correlate features in time.

Pooling layers are layers that reduce the size of the representation, speeding up computation and making feature detection more robust. Pooling layers work by applying a filter where the value of the neuron is calculated as a function of the pixel values covered by its window. Maximum (max) pooling is often used. The function for max pooling simply returns the maximum value within its window. This function is computationally cheap and allows checking if a feature was detected within the window.

Fully connected layers are where each neuron in a layer is connected to every neuron in the previous layer and are used to form the output. This is because the weights adapt to learn how the features detected in previous layers relate to form the output.

CNN have proven invaluable in image recognition applications and are the predominant architecture used for gesture recognition tasks as well. This is because they can learn complex non-linear functions that are difficult to define using rules.

Hidden Markov Models and Other Methods and Architectures

In addition to CNN, other methods and architectures have also had success in using gesture recognition, with methods often used in conjunction to each other as stages. These include Hidden Markov Models (HMM), Leaky Integrate-and-Fire neurons (LIF), clustering and hand region extraction algorithms (Eun Yeong Ahn, 2011). HMM used in conjunction with LIF and CNNs are the predominant major architectures used in neuromorphic gesture applications.

Hidden Markov models are where there is a hidden state which impacts a visible output and next state with certain probabilities. Hidden Markov models employs Bayesian probability to predict the hidden states given only the output. The model aims to maximize the probability that the output sequence is observed. It does this by calculating the probability of each state, including dependence on previous states, and the probability of each output corresponding to these. It then chooses the state that maximizes the likelihood of the output occurring as the state predicted to have occurred. The Baum-Welch algorithm is the common method used to find these probabilities.

Leaky integrate-and-fire neurons are neurons that keep track of an internal state (its membrane potential), adding to it when it receives input events such as spikes and gradually decreasing it otherwise. If the potential exceeds a certain level, it will fire a spike event and reduce its membrane potential. By connecting these neurons to an area of pixels in NVS, this method allows spatiotemporal correlation since movement in an area will cause many events to be received in a short time and the LIF neuron to fire. LIF neurons therefore illustrate a clear principle in gesture recognition. The event data produced in NVS needs to be spatially and temporally correlated due to events occurring with a degree of random noise. This can be seen in the image below taken from IBM's DVS128 dataset.

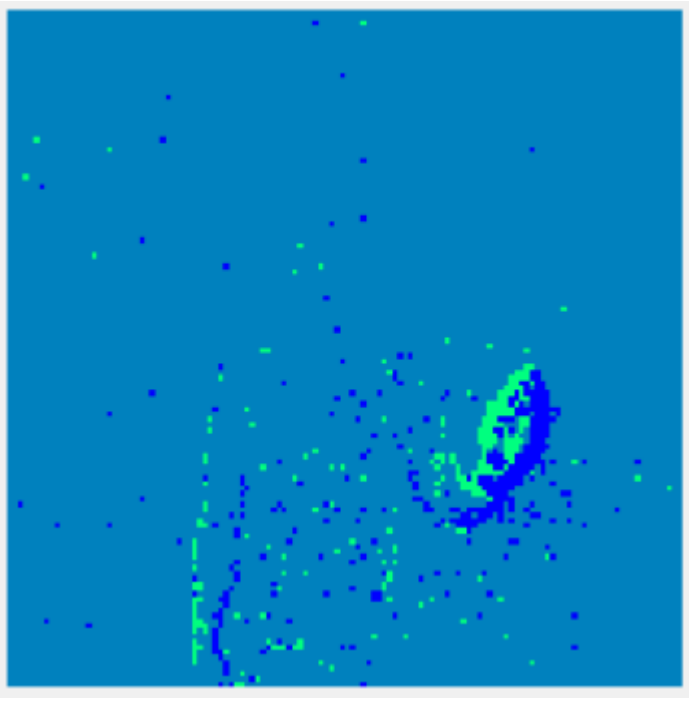


Figure 1. Image illustrating noise in events from DVS128 Dataset

Project Objectives

Much of the literature and applications referenced above are from groups of experienced researchers with greater resources. This project will aim to achieve slightly lower results due to constraints on computation power, time and data.

Problem Statement

This project aims to improve human machine interfaces by producing an implementation of gesture classification using NVS. This will increase the literature and understanding of NVS applications for gesture recognition and allow a point of comparison, facilitating the development of gesture UIs that are capable of low power consumption, high reliability and low latency.

Performance Metric

The performance metric chosen is Top1 accuracy in recognition of 10 gestures. This means that only the top prediction will be counted. The network will be given input data from a clip with one gesture and the prediction will be compared to the ground truth for the clip. Accuracy is given by the equation below.

$$Accuracy = \frac{True\ Positives + True\ Negatives}{True\ Positives + True\ Negatives + False\ Positives + False\ Negatives}$$

This metric was chosen as it is the predominant metric used for gesture recognition tasks and this would allow for effective comparison. Since the datasets have the similar number of training examples in each, training and testing data will not be skewed, and the accuracy metric is acceptable.

The target performance for the project will be to achieve 90% accuracy in classification of gestures. This is the average accuracy of 10 gesture recognition (Zima, 2012).

Benchmark

The network classification capability will be compared against the benchmark of the 20BN-JESTER dataset translated into the DVS domain using pix2NVS (Yiannis Andreopoulos, 2017). The current leading accuracy of the dataset in the pixel domain is 96.77% (Twenty Billion Neurons GmbH, n.d.).

This will be the predominant dataset that will be used. It has already been split into a training, validation and an unlabeled test set. The full 27 classes will be used for fair comparison.

Classification capability will also be compared against current DVS classification using IBM's DVS128 gesture dataset that has classification of 96.49% (Arnon Amir, 2017). This dataset will be split into a training and validation. This dataset will be split 70/30 into training and test sets for comparison.

These benchmarks will allow comparison on how well the implementation performs compared to existing implementations.

Proposed Solution

Architecture

The architectures that will be implemented and tested are the CNN and variations on this. Trinary weight and deep network variations of the CNN will be implemented. CNNs are a versatile architecture used in video recognition due to being able to correlate both spatially and temporally using filters.

The choice to use CNNs and not HMM was that HMM required localization of the hand. Since implementations and localized hand data are few and insufficient to recreate, this was deemed unachievable given the constraints of the project.

Trinary weights are where the values in the neural network are limited to -1, 0 and 1. This was chosen as it allows faster computation. Trinary weights are also used in neuromorphic chips. This information would therefore be useful in informing how well the implementaion can be expected to perform.

Deep network approaches are shown to improve performance when given large datasets. This approach may increase latency but will provide a point of comparison for what should be greater accuracy.

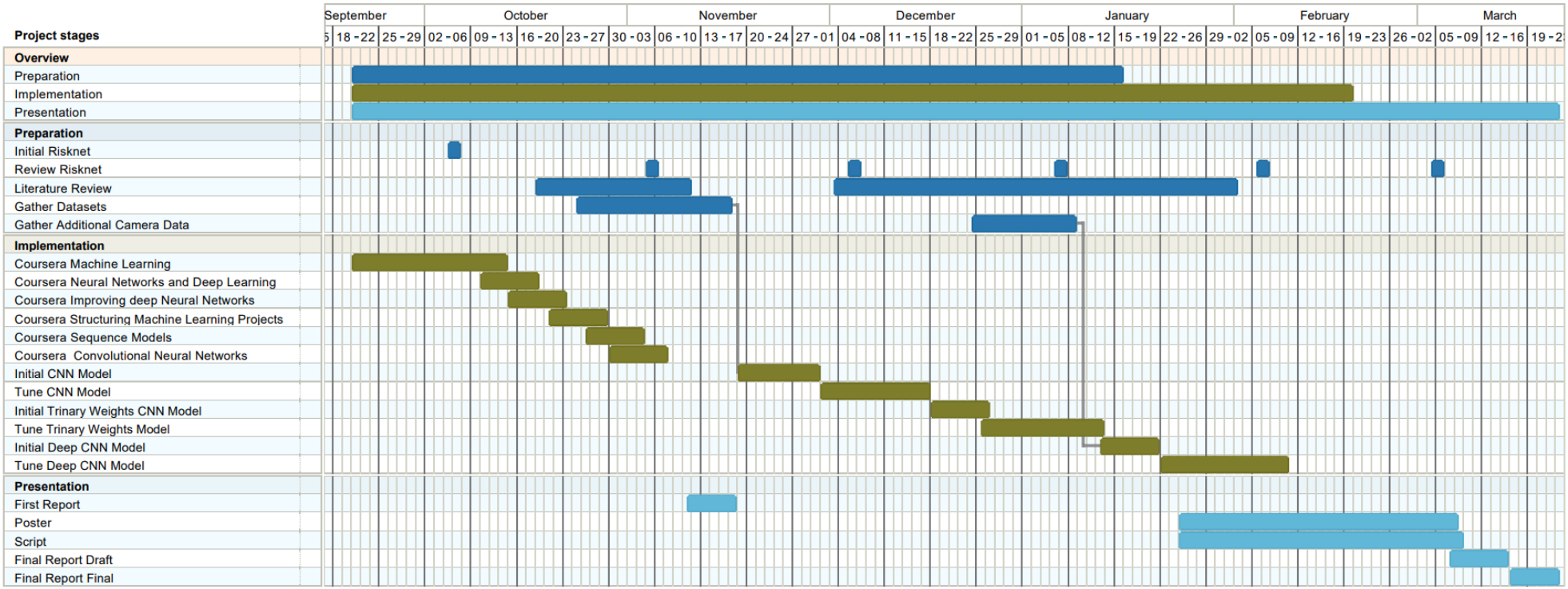
CNN

The CNN architecture planned to be used will be as detailed below. The choice is based on the VGGNet but is an arbitrary choice, due to a lack of existing published architectures for gesture recognition in event space. Temporal filters are used at the start to allow for more events to be recognized. This is an initial design for the CNN and the structure may be altered after observing the workings of the model. The initial version of this CNN will make use of ReLU activation functions and max pool layer.

Input	Conv time	Conv time	Pool	Conv	Conv	Pool	Conv	Conv	Pool	FC	FC	FC	Output
-	3x3	3x3	2x2	3x3	3x3	2x2	3x3	3x3	2x2	-	-	-	-

Plan

The plan is as detailed in the following Gantt Chart. It involves first implementing the simple CNN model described previously. This model will then be observed to see where problems arise and the structure and hyperparameters will be tuned accordingly. A similar process will occur for Trinary weighted and Deep networks. Since the training process involves calculating the validation error, this metric will allow for a estimation of accuracy. Once the model is complete, there will be a single evaluation using the test sets.


September
October
November
December
January
February
March
Project stages
5 | 18 - 22 | 25 - 29 | 02 - 06 | 09 - 13 | 16 - 20 | 23 - 27 | 30 - 03 | 06 - 10 | 13 - 17 | 20 - 24 | 27 - 01 | 04 - 08 | 11 - 15 | 18 - 22 | 25 - 29 | 01 - 05 | 08 - 12 | 15 - 19 | 22 - 26 | 29 - 02 | 05 - 09 | 12 - 16 | 19 - 23 | 26 - 02 | 05 - 09 | 12 - 16 | 19 - 2
Overview
Preparation
Implementation
Presentation
Preparation
Initial Risknet
Review Risknet
Literature Review
Gather Datasets
Gather Additional Camera Data
Implementation
Coursera Machine Learning
Coursera Neural Networks and Deep Learning
Coursera Improving deep Neural Networks
Coursera Structuring Machine Learning Projects
Coursera Sequence Models
Coursera Convolutional Neural Networks
Initial CNN Model
Tune CNN Model
Initial Trinary Weights CNN Model
Tune Trinary Weights Model
Initial Deep CNN Model
Tune Deep CNN Model
Presentation
First Report
Poster
Script
Final Report Draft
Final Report Final

Current Progress

As applications in Machine Learning were beyond the course content, this needed to be learned separately. This involved completing several courses on Coursera. These include Machine Learning, Convolutional Neural Networks, Sequence Models, Structuring Machine Learning Projects, Improving deep Neural Networks: Hyperparameter tuning, Regularization and Optimization, Neural Networks and Deep Learning. This included the creation of a CNN to accomplish image recognition, distinguishing between happy and not happy faces. This achieved an accuracy of 94%. The 20BN-JESTER and DVS128 datasets have also been downloaded and the initial version of the CNN being created.

Bibliography

Arnon Amir, B. T. D. B. T. M. J. M. C. D. N. T. N. A. A. G. G. M. M. J. K. M. D. S. E. T. D. M. F. D. M., 2017. *A Low Power, Fully Event-Based Gesture Recognition System.* Honolulu, IEEE.

Brian Dipert, B. Y. S. C. S. M. C. L. R. e. M. T. G. K. O. I., 2013. *The gesture interface: A compelling competitive advantage In the technology race.* [Online] Available at: https://www.eetimes.com/document.asp?doc_id=1280756

Carl A. Pickering, K. J. B. M. J. R., 2007. *A Research Study of Hand Gesture Recognition Technologies and Applications for Human Vehicle Interaction.* Warwick, IET.

Chen, W., 2013. *Gesture-Based Applications for Elderly People.* Berlin, s.n., pp. 186-195.

Christian Brandli, R. B. M. Y. S.-C. L. T. D., 2014. A 240 × 180 130 dB 3 µs Latency Global Shutter Spatiotemporal Vision Sensor. *IEEE Journal of Solid-State Circuits,* pp. 2333-2341.

Eun Yeong Ahn, J. H. L. T. M. J. Y., 2011. *Dynamic Vision Sensor Camera Based Bare Hand Gesture Recognition.* Paris, IEEE.

Gollakota, B. K. a. V. T. a. S., 2014. *Bringing gesture recognition to all devices.* s.l.:University of Washington.

Henri Rebeceq, G. G. E. M. D. S., 2017. EMVS: Event-Based Multi-View Stereo—3D Reconstruction with an Event Camera in Real-Time. *International Journal of Computer Vision.*

Jaime Lien, N. G. M. E. K. P. A. C. S. E. O. H. R. I. P., 2016. *Soli: Ubiquitous Gesture Sensing with Millimeter Wave Radar.* s.l., s.n., pp. 1-19.

Jayasuriya, S. &. G. O. &. G. J. &. K., 2016. *Deep Learning with Energy-efficient Binary Gradient Cameras.* s.l.:Carnegie Mellon University.

Lee, J. et al., 2012. *Live demonstration: Gesture-based remote control using stereo pair of dynamic vision sensors.* Seoul, s.n.

Lee, J. H. et al., 2012. *Touchless hand gesture UI with instantaneous responses.* Orlando, IEEE.

Liu, X., 2016. *What role does effort play: the effect of effort for gesture interfaces and the effect of pointing on spatial working memory,* s.l.: University of Iowa.

Maruvada, S., 2017. *3-D Hand Gesture Recognition with Different Temporal Behaviors using HMM and Kinect,* s.l.: University of Magedeburg.

Matia Pizzoli, C. F. D. S., 2014. *REMODE: Probabilistic, Monocular Dense Reconstruction in Real Time.* Hong Kong, s.n.

Paul K. J. Park, B. H. C. J. M. P. K. L. H. Y. K. H. A. K. H. G. L. J. W. Y. R. W. J. L. C.-W. S. Q. W. a. H. R., 2016. *Performance improvement of deep learning based gesture recognition using spatiotemporal demosaicing technique.* Phoneix, IEEE.

Robert Likamwa, Z. W. A. C. F. X. L. L. Z., 2013. *Draining our Glass: An Energy and Heat Characterization of Google Glass,* s.l.: Rice University.

Stefanie Anna Baby, B. V. C. C. K. M., 2018. *Dynamic Vision Sensors for Human Activity Recognition.* Nanjing, s.n.

Twenty Billion Neurons GmbH, n.d. *The 20BN-jester Dataset V1.* [Online] Available at: https://20bn.com/datasets/jester

Yanan Xu, Y. D., 2017. Review of Hand Gesture Recognition Study and Application. *Contemporary Engineering Sciences,* pp. 375 - 384.

Yiannis Andreopoulos, Y. B., 2017. *PIX2NVS: Parameterized conversion of pixel-domain video frames to neuromorphic vision streams.* Beijing, IEEE.

Zima, M., 2012. *Hand/Arm Gesture Recognition based on Address-Event-Representation Data.* Vienna: Vienna University of Technology.