Real-time Scalable Video Stream Analysis with Object, Activity and Event Classification.

Joseph Honour

Newcastle University

Author Note

Completed as part of my undergraduate BSc in Computer Science with Industrial Placement (G401), supervised by Matt Collison.

Abstract

Computer vision has been a large area of research in recent years, devising methodologies to understand and act upon events seen within video streams. A major application of computer vision is to detect anomalies autonomously, and alert users to when they occur. Although industrial technologies exist that can do this to a basic standard, they often rely on expensive and exclusive hardware.

This paper proposes an extendable and scalable framework that can provide an end-to-end video processing pipeline capable of accurate event classification, in real-time, without complex hardware requirements. The framework will show how the adoption of distributed computing and machine learning enable real-time event classification, without requiring specialized hardware. My design approach is to allow extensibility at every opportunity, so the framework can be adapted for a multitude of use cases, some of which I propose within this paper. Furthermore, the framework will allow horizontal scaling enabling it to handle large volumes of data, while keeping its real-time requirements intact. Finally, the public hosting of the framework will allow the exploration of new development avenues by the community, with avenues of exploration suggested within this paper.

Keywords: Computer Vision, Distributed Computing, Machine Learning

Real-time Scalable Video Stream Analysis with Object, Activity and Event Classification.

# Introduction and Motivations

In the United Kingdom, an estimated 4.2 million surveillance cameras actively capture video footage every day [1]. CCTV’s deployment to businesses, homes, shops and high streets makes the United Kingdom one of the most monitored nations in the world. Fueling the mass deployment of surveillance equipment is its ability to monitor activities in real-time, while providing traceability if an event of interest occurs. The power of traceability and event monitoring can be seen with CCTV’s 96% availability in homicide investigations [2]. However, if an event of interest does occur that requires viewing, an individual must manually watch all relevant footage to find the event, which is extremely time consuming. A more proactive approach to policing, and general event handling, could evolve if the real-time detection of these points of interest within video-streams became available through automation.

This is the key limitation with widely adopted video surveillance; real-time analysis or understanding of what is happening within the video stream is not available. This means the adoption of surveillance systems, in most cases, is currently in a traceability capacity only, when it has the potential to provide real-time insights into events occurring within the video footage.

With the development of Smart Cities, more information and video streams are becoming available for analyses [3], meaning video stream analytics must be able to scale to huge volumes. Large scale applications bring with them new considerations; minimizing data transfer rates to reduce running costs, distributing computational work to enable greater processing throughput, providing analysis results in a quick enough time to be valuable. As these problems haven’t materialized within computer vision applications before, no current video processing frameworks provide a full solution to this problem (Table 1).

To help enable the demands of intelligent video stream analysis at scale, a base framework is proposed. The framework should provide common functionality for video processing by default, while allowing easy development extension to cater to the broadest range of use cases. Furthermore, horizontally scaling technologies should provide the framework base to meet the demands of even the largest video stream development projects.

# Assumptions

To allow an easy transition to smart video stream analysis from existing video surveillance systems, the framework makes no assumptions about existing video camera hardware deployments, but provides a module for video acquisition [4] and processing. As the framework makes use of distributed computing, we assume that the module interfacing with the raw video feed is connectable to the downstream processing stages.

# Related Work

The providing of an end-to-end video processing framework requires the accommodation and support of all relevant stages within video processing. The exploration of theoretical stages of generic video processing are documented within an industrial case setting (Figure 1) driven with the goal of providing surveillance information to government agencies, giving a strong grounding of fundamental features of video processing. From this, we must then be aware of common techniques adopted at each stage of video processing, to allow a proposed framework to support the current industrial standards.



Figure 1: Adaptation of video processing within a general framework for automated visual surveillance system [5]. Shows common stages involved in providing computer vision applications in the context of automated surveillance.

Object Detection is the base of all computer vision applications, providing the ability to accurately identify objects within a frame. Work in this field has relied upon being able to correctly interpret the combinations of pixels correctly to identify an object [6]. In more recent years, with the improvement of Graphics Processing Units (GPU) hardware, advanced detection techniques have been created upon Deep Learning models [7]. Further to this, object detection techniques often utilize motion and background separation methods, enabling them to provide insight into events and activities occurring [8]. Utilizing the foundation of accurate object detection, objects can be given identity between frames.

Object Tracking enables the observation of object movement vectors, along with the monitoring of object interaction patterns. Work within the object tracking domain build predictive models with online data, meaning they adapt at runtime to provide improved predictions. Algorithms within this domain often model an objects location as a set of positions that each could contain the objects location based on its previous known location [9], [10], [11]. Given successful employment of objects tracking, we can now observe the behaviors of individual objects through a video stream.

Behavior and Activity Analysis allow the assignment of context to object movements, enabling more advanced processing techniques down-stream in the video processing pipeline, such as event classification. Within computer surveillance it is often desirable to identify endangering or suspicious activities [5]. Active work within this field is varied, with successful modelling of behaviors being produced using Markov Models [12], [13], along with the more computationally intense techniques architected with Neural Networks [5]. Markov Model techniques have identified behaviors to precision rates above the 90th percentile [14], however they struggle with noise in the data which can cause accuracy to drop. The understanding of activities within a video stream provides a deep insight into what is transpiring, allowing the modelling of the video context through time.

Event Classification builds upon activity analysis, giving a method for identifying unusual data points within the context of the video stream. Work in this field focuses on building a model to represent the current state of the data, and then compares new data points to the model calculating how far the point deviates from the existing data set. Multiple models are able to provide this, frequently built around clustering techniques [15]. With the wide adoption of Neural Networks, many methods have emerged built around this core architecture that can provide high accuracy detections, at the cost of requiring high performance hardware [16].

Existing Technologies that aim to provide a basis for computer vision applications within the realm of the established processing pipeline (Figure 1), are often only designed for a single use case or are closed for development and therefore cannot be built upon. Table 1, shows an overview of recent video analysis frameworks and their limitations.

|  |  |  |  |
| --- | --- | --- | --- |
| Product | Overview | Functionality | Limitations |
| Nest [17] | Alerts users in real-time, via an app, to event and motion detection events seen on camera. | Provides event and anomaly detection and real-time alerts when these occur. | Requires specialized hardware, is not extendable for development. |
| A Video Analysis Framework for Surveillance System [18] | Gives a novel framework approach to online video analysis using .NET 2.0. | Provides object detection, event detection with extensibility to insert new functionality within the application. | Does not make use of distributed computing so will be unable to scale to all user requirements. Further to this, the product is developed in .NET, requiring a Microsoft workstation to run. |
| DiVA [19] | Gives a distributed video processing framework that uses a database as a message source, allowing components to communicate agnostic of technologies adopted. | Provides object detection with the goal of detecting object abandonment and removal. Extensibility is provided through the communication of modules/algorithms occurring at a database level. | Uses a singular database to communicate, rather than a distributed messaging system. It also requires fixed hardware meaning it currently does not make use of Cloud Computing. |

Table 1: Current technologies in the market that aim to provide computer vision applications.

Although currently available frameworks provide a good grounding for computer vision applications, it is apparent that work is required to allow for these systems to scale to the level required of modern application, for instance Smart Cities.

# Architecture

We propose a scalable framework, that can support common video processing techniques by default, while being open for extension to enable domain specific modifications (Figure 2). It adopts a streaming architecture to provide real-time analytics on data generated from the raw video input. This enables decisions to be made as video footage is processed, while still supporting traditional batch processing applications.

Figure 2: The proposed video processing framework architecture, showing the avenues of communication through the framework.

Pre-Processing happens as close as possible to the raw video input, reducing the latency between the video source and the processing applied to it. Using a technique known as Edge Computing [20], applying filtering and detection on the video feed to control the amount of data entering the downstream processing steps within the pipeline, the networks capacity is maximized.

The sub systems of the framework communicate through a distributed messaging layer, decoupling systems and maintaining a flexible and extensible application. Communication can then be achieved by using a common Application Program Interface (API) between systems and the messaging layer. The decoupled nature of the sub systems means they can be deployed independently, allowing the most appropriate tool to be used for each area of processing, with individual resource allocations for different parts of the system. This enables users to deploy infrastructure based on individual services, giving a fine-grained level of control to avoid over or under allocation of resources to a task. A messaging layer adds complexity to the framework, as systems have the added overhead of indirectly communicating with each other, which is a noticeable cost when considering this approach.

The framework is intended to be deployed to a Cloud environment, a service that allows clients to rent hardware rather than buying it upfront, as this gives a flexible way of managing infrastructure depending on specific performance requirements. However, the deployment can be made to a local environment, and considerations can be made as to the most appropriate production infrastructure on a per use case basis [21]. Once deployed, the framework makes use of distributed technologies to manage throughput and latency during processing.

Taking successful aspects from previously seen work [18] [19], giving a modular approach to design, while extending the ability to distribute work between machines. The framework improves upon these designs by making use of Cloud Computing infrastructure, while enabling clusters of machines to distribute work for a single task, rather than just distributing work between different stages of the processing pipeline. This approach hopes to overcome the challenges of scaling, while maintaining the success of modular design seen in previous work, enabling large scale computer vision applications to become possible.

This approach has its limitations, the network latency of the infrastructure heavily affects distributed processing, as each node in a cluster must communicate with its counterparts to organize and distribute work. This can drastically reduce the performance of the proposed framework if deployed onto a degraded network environment. Mitigating this, the frameworks deployment to Cloud infrastructure should allow for dynamic network configuration to meet individual requirements, coupled with intense processing happening at the edge of the Cloud. Accompanying this feature, Cloud deployments should be made smooth and approachable enabling users with little infrastructure experience to deploy applications to production.

Further to this, Cloud infrastructure can become expensive as network usage is charged to the user along with the rented computing power of the machines. To combat this, the framework is not linked to a single Cloud provider, or to the Cloud at all, giving the freedom of choice to the adopting user.

# Implementation

The proposed framework architecture is implemented as shown in Figure 3, producing the data processing pipeline in Figure 4. The pre-processing stage (Figure 2) is intended to happen as close to the camera as possible, with support for directly connected cameras. The data produced from the pre-processing stages is then sent to Apache Kafka, the distributed messaging broker.



Figure 3: The proposed video processing framework implementation, distributed computing technologies (Apache Kafka, Apache Flink, Apache Spark) are adopted to provide high throughput, low latency, processing within a Cloud environment.



Figure 4: The data flow through the application, showing how each processing stage makes use of Apache Kafka to read input and produce outputs.

Apache Kafka [22] is chosen due to its ability to scale to support millions of messages per second, along with its low latency [23]. This enables projects of any size to adopt the framework and allows the framework room to scale as demand increases (Figure 5, Figure 6). This data displays how even Apache Kafka configurations that sacrifice some asynchronous performance for reliability of message storage, still support hundreds of thousands of messages per second. This enables data integrity guarantees to be met while continuing to meet real-time requirements of video processing systems. Further to this, Apache Kafka can act as an efficient buffer between systems allowing for asynchronous communication, reducing time spent waiting for message responses. With Apache Kafka at the heart of the video processing framework, the individual video processing components are able to communicate with efficient buffering.

Figure 5: Apache Kafka throughput in messages per second [24].

Figure 6: Apache Kafka latency in milliseconds [24].

Enabling the activity analysis phase of the framework is Apache Flink [25], a real-time stream processing framework. Apache Flink can process messages with low latency in parallel, meaning expensive operations on the raw video data can be done without falling behind the high frequency of raw video data being produced.

Finally, Apache Spark [26] is used for running the event classification machine learning models. Apache Spark streams data by processing it in incremental batches which, although add a larger latency overhead, enable it to process at a greater throughput [27]. This means, when attempting to build a model from numerous records, we can build detections at a much faster rate. The base framework uses both Apache Flink and Apache Spark to show its amongst technologies while proving the most appropriate technology can be adopted to provide the best final product.

The framework can then store data in any database, and currently Neo4J [28] is supported. This allows querying and interfacing with the data, enabling exploration and understanding of event causality as data is streamed within the pipeline.

Within the implemented framework there are one click deployment scripts that allow each individual sub system to be deployed to an Amazon Web Services (AWS) Cloud environment. This, by default, deploys a minimal number of machines to run each service, but can be adapted to deploy a range of multi-node clusters with little modification.

# Use Case Evaluation

The proposed framework delivers core video processing functionality that enables extension and flexibility to meet a broad range of client use cases. Presented below is a practical use case showing the frameworks successful adoption to a specific domain, providing core insights and analytics into the video footage in real-time. It aims to show the initial steps in enabling computer vision, while showing how further extension is made possible.

## Scenario

The Abbey Road in London, pictured on the front of the famous Beetles album “Abbey Road”, is a popular destination for tourists and locals attempting to recreate the album cover for themselves. Due to this, it is under live surveillance by a multitude of video cameras at all times. This presents an opportunity to gain real-time analytics of the activities occurring at the crossing in real-time. This task, although simple, shows how a deployment of computer vision to a cities CCTV infrastructure can allow autonomous learning.

To achieve this the proposed framework is deployed to detect cars and people within the live video stream, extrapolate activities to track walking, standing and people attempting to recreate the album cover photo. It will analyze the activity log for anomalies, alerting the user upon discovering an anomalous event.

## Use Case Implementation

As the proposed architecture diagram shows (Figure 1), Edge Computing is adopted in order to maintain real-time requirements and limit the stream of data from the video input source. This enables the detection and tracking of people and vehicles to occur at, or near, the camera source, drastically reducing the data load sent over the network.

When identifying cars and people within the video feed, the base framework allows the use of Haar Cascades [6]. This is a common, high-performant, detection methodology that provides an acceptable accuracy rate in identifying objects (Figure 7). This was deployed as a Python package, streaming the live video and providing pre-processing in order to capture objects locations.

Between frames, tracking needs to occur in order to give people and vehicles a persistent identity, enabling the computation of further server-side analytics in identifying object movement patterns. Providing this is a Kernalized Correlation Filters (KCF) tracker [29], which tracks an object by treating its location as a set of positions (‘bags’) that each could contain the objects location based on its previous location. This was deployed alongside object tracking, with the objects location and identity being sent to server-side processes through Apache Kafka.



Figure 7: The deployed video processing framework identifying people and cars within the “Abbey Road” real-time video stream.

Apache Kafka was deployed to a cluster on AWS, with the video stream processing application publishing messages containing objects locations, which can then be consumed by server-side services to provide more insightful analytics.

In order to gain a deeper awareness into the video stream and enable the objective of identifying people performing the “Abbey Road” album cover, an activity analysis service was needed. The base framework uses Apache Flink for this capability, and has built in behavior for detecting walking, standing and running activities (Figure 8). The framework can then be extended to not only detect people standing, but to identify four people standing, with the correct distribution between them, that would be considered an impersonation of the album cover. As Apache Flink comes with advanced pattern matching, this is a simple extension to make to the framework. Configuration files are used to provide fine grained tuning of the activity models, letting the user easily tweak the deployed activity identification models to their specific object tracking data feed. This service was also deployed to AWS, with identified activities being sent to Apache Kafka for downstream analysis.



Figure 8: The proposed video processing framework detecting activities within the given video stream.

Given the knowledge of the activities being performed, and their locations, an event classification service can be utilized to identify unusual activities within the video feed. Providing this service, the base framework makes use of Apache Spark to deploy a distributed K-Means unsupervised [30] machine learning model. This service can run on AWS in its own multi-node cluster. The model is fed the activity locations, along with the activity type, and is then able to calculate how far an activity deviates from the center of common activities (Figure 9). This permits the detection of not only activities performed outside of their normal locations within the video stream, but also activities of an unusual type. This offers the flexibility to identify the album cover activity as an event if it is not being performed often, however if there is a series of video that has frequent similar activities, the model can adjust itself to detect that as the normal, giving it the ability to adapt to changing behaviors.

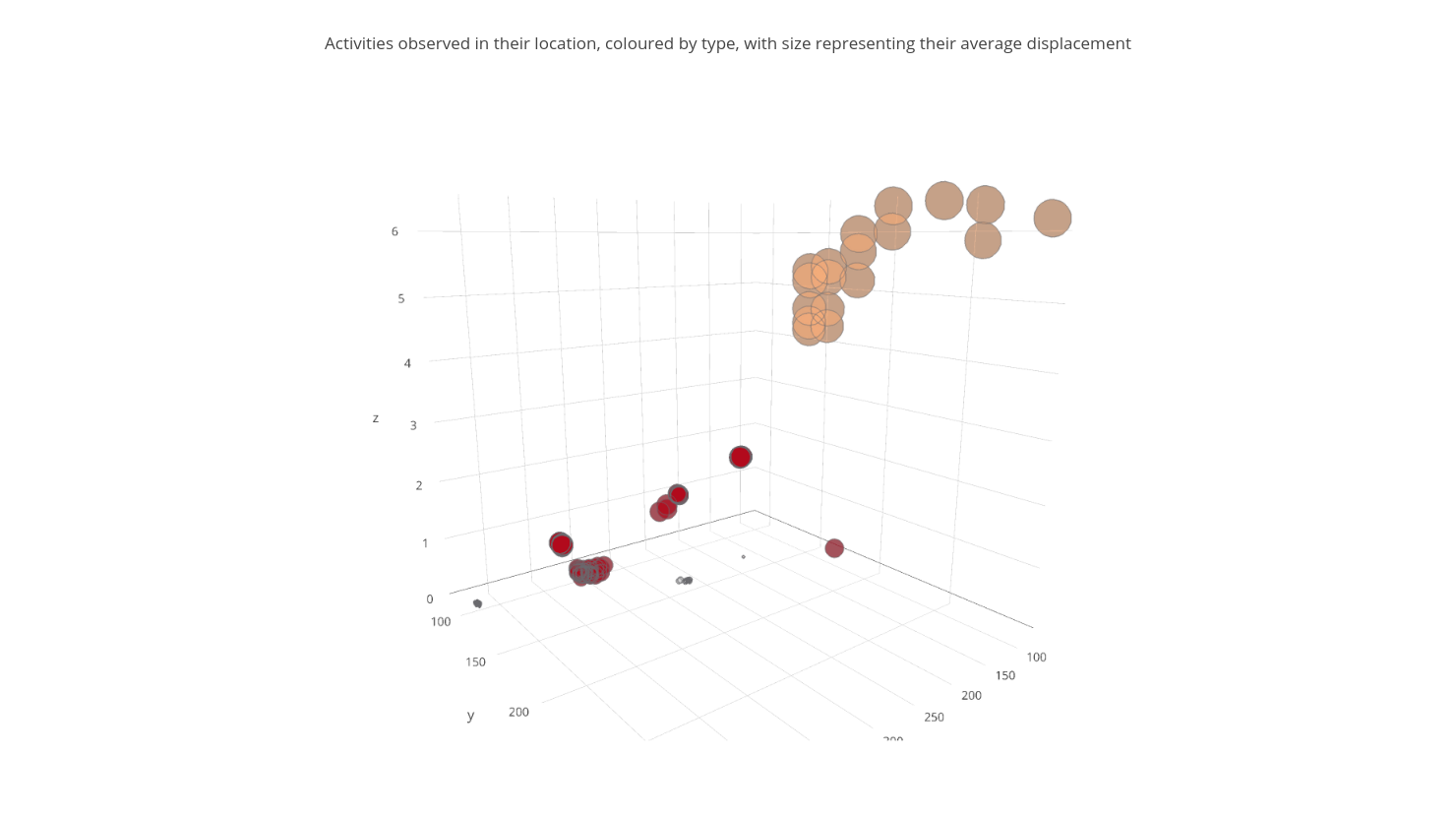


Figure 9: A chart showing the clustering of activities based on the objects average displacement during identification, the activity type and location.

The service then outputs the deviation of a given activity from the normal to Apache Kafka, which can then be consumed by a simple service that sends an alert to the owner of the camera that either an anomalous event has been performed or that an event of a specific type has been seen, allowing the captured video to be investigated. If the event classification service is configured to weight the “Abbey Road” activity as an event of interest, then this can aid in the aggregation of all video footage of people performing the action.

As the base framework comes with deployment scripts written in Terraform for all the discussed technologies; Apache Flink, Apache Spark, Apache Kafka, the deployment of the entire application to AWS is simple. The scripts require minimum setup, and one click deployments of individual services, meaning we could include new services within the processing pipeline as and when they are configured.

# Discussion

With the proposed framework we aim to achieve the scalability and ease of deployment in order to enable more computer vision applications to become a possibility. Through the use case presented, we have shown the adoption process in using the framework for a bespoke application. This is limited and, although the technologies adopted are able to scale to millions of messages per second, further work needs to be completed showing the framework working at scale. Smart cities are in their infancy at present, and the full requirements required to provide real-time analytics should not be overlooked. However, the frameworks proof of work with a single camera, shows how custom analytics are easily enabled, and that the system can provide an end-to-end solution for video processing.

References

[1] C. Norris and M. McCahill, “CCTV: Beyond penal modernism?,” *Br. J. Criminol.*, vol. 46, no. 1, pp. 97–118, 2006.

[2] S. Yard, “CCTV in Homicide Investigations,” 2010. [Online]. Available: https://goo.gl/oS5Tgn. [Accessed: 15-Nov-2017].

[3] G. P. Hancke, B. de C. de Silva, and G. P. Hancke, “The role of advanced sensing in smart cities,” *Sensors (Switzerland)*, vol. 13, no. 1. pp. 393–425, 2013.

[4] D. Thiel, M. Goulart, S. Botelho, and R. De Bem, “Hardware and Software Infrastructure to Image Acquisition using Multiple Cameras,” *ucsp.edu.pe*, 1998.

[5] T. Ko, “A survey on behavior analysis in video surveillance for homeland security applications,” *Appl. Imag. Pattern Recognit. Work. 2008. AIPR ’08. 37th IEEE*, pp. 1–8, 2008.

[6] P. Viola and M. Jones, “Robust real-time object detection,” *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2001.

[7] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553. pp. 436–444, 2015.

[8] Y. Zhu, N. M. Nayak, and a K. Roy-Chowdhury, “Context-Aware Activity Recognition and Anomaly Detection in Video,” *Sel. Top. Signal Process. IEEE J.*, vol. 7, no. 1, pp. 91–101, 2013.

[9] B. Baben and S. Belongie, “Visual tracking with online Multiple Instance Learning,” *2009 IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 983–990, 2009.

[10] Z. Kalal, K. Mikolajczyk, and J. Matas, “Forward-backward error: Automatic detection of tracking failures,” in *Proceedings - International Conference on Pattern Recognition*, 2010, pp. 2756–2759.

[11] Z. Kalal, K. Mikolajczyk, and J. Matas, “Tracking-learning-detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1409–1422, 2012.

[12] J. Kröckel and F. Bodendorf, “Intelligent Processing of Video Streams for Visual Customer Behavior Analysis,” *ICONS 2012, Seventh Int. Conf. Syst.*, no. c, pp. 163–168, 2012.

[13] T. T. Z. T. T. Zin, P. T. P. Tin, T. Toriu, and H. Hama, “A Markov Random Walk Model for Loitering People Detection,” *Intell. Inf. Hiding Multimed. Signal Process. (IIH-MSP), 2010 Sixth Int. Conf.*, 2010.

[14] P. Antonakaki, D. Kosmopoulos, and S. J. Perantonis, “Detecting abnormal human behaviour using multiple cameras,” *Signal Processing*, vol. 89, no. 9, pp. 1723–1738, 2009.

[15] M. Goldstein and S. Uchida, “A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data,” *PLoS One*, vol. 11, no. 4, 2016.

[16] M. Längkvist, L. Karlsson, and A. Loutfi, “A review of unsupervised feature learning and deep learning for time-series modeling,” *Pattern Recognit. Lett.*, vol. 42, no. 1, pp. 11–24, 2014.

[17] Nest, “No Title,” 2017. [Online]. Available: https://nest.com/uk/cameras/nest-cam-indoor/overview/. [Accessed: 24-Nov-2017].

[18] N. Suvonvorn, “A video analysis framework for surveillance system,” *2008 IEEE 10th Work. Multimed. Signal Process.*, pp. 867–871, 2008.

[19] J. C. SanMiguel, J. Bescós, J. M. Martínez, and Á. García, “DiVA: A Distributed Video Analysis Framework Applied to Video-Surveillance Systems,” in *International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, 2008, pp. 207–210.

[20] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge Computing: Vision and Challenges,” *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, 2016.

[21] M. Armbrust, A. Fox, R. Griffith, A. Joseph, and RH, “Above the clouds: A Berkeley view of cloud computing,” *Univ. California, Berkeley, Tech. Rep. UCB* , pp. 07–013, 2009.

[22] Apache, “Apache Kafka,” 2018.

[23] J. Kreps, N. Narkhede, and J. Rao, “Kafka: a Distributed Messaging System for Log Processing,” 2011.

[24] J. Kreps, “Apache Kafka Performance Benchmark,” 2014. .

[25] Apache, “Apache Flink,” 2018. [Online]. Available: https://flink.apache.org/. [Accessed: 19-Feb-2018].

[26] Apache, “Apache Spark,” 2018. .

[27] S. Chintapalli *et al.*, “Benchmarking streaming computation engines: Storm, flink and spark streaming,” in *Proceedings - 2016 IEEE 30th International Parallel and Distributed Processing Symposium, IPDPS 2016*, 2016, pp. 1789–1792.

[28] Neo4j, “Neo4j: The World Leading Graph Database,” *Neo4J.Org*, 2016. [Online]. Available: https://neo4j.com/.

[29] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, “High-speed tracking with kernelized correlation filters,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 3, pp. 583–596, 2015.

[30] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, “Scalable K-Means++,” Stanford, 2012.