Real-time Scalable Video Stream Analysis with Object, Event and Anomaly Detection.

Joseph Honour - 130291538

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 on 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 are able to do this to a basic standard, they often rely on expensive and exclusive hardware.

This paper proposes an extendable and scalable framework that is able to provide accurate anomaly detection, in real-time, without complex hardware requirements. The framework will show how the adoption of distributed computing and machine learning enable real-time anomaly detections, 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 framework will be hosted publicly allowing new avenues to be explored by the community, with avenues of exploration suggested at the end of this paper.

Declaration

“I declare that this dissertation represents my own work, except where otherwise stated.”

Acknowledgments

This is my acknowledgments.

Table of Contents

[Abstract 2](#_Toc506809353)

[Declaration 3](#_Toc506809354)

[Acknowledgments 4](#_Toc506809355)

[Introduction 7](#_Toc506809356)

[Motivation 7](#_Toc506809357)

[Aim 8](#_Toc506809358)

[Objectives 8](#_Toc506809359)

[Paper Structure 8](#_Toc506809360)

[Background and Literature Review 9](#_Toc506809361)

[Video Processing Methodologies and their Adoption 9](#_Toc506809362)

[Object Detection Techniques 10](#_Toc506809363)

[Object Tracking Techniques 11](#_Toc506809364)

[Behavior and Activity Analysis 12](#_Toc506809365)

[Anomaly Detection with Machine Learning 13](#_Toc506809366)

[Anomaly Detection Models 13](#_Toc506809367)

[The Impact of Human Behavior 13](#_Toc506809368)

[Distributed Computing and the Cloud 13](#_Toc506809369)

[Parallel Computing 13](#_Toc506809370)

[Distributed Messaging 14](#_Toc506809371)

[Cloud Providers 14](#_Toc506809372)

[Existing Technologies and Approaches 14](#_Toc506809373)

[References 15](#_Toc506809374)

[Footnotes 17](#_Toc506809375)

[Tables 18](#_Toc506809376)

[Figures title: 19](#_Toc506809377)

# Introduction

## Motivation

In the United Kingdom, an estimated 4.2 million surveillance cameras watch us every day (Norris and McCahill, 2006). CCTV is deployed to businesses, homes, shops and on high streets making us one of the most watched nations in the world. This mass deployment of surveillance equipment is meant to deter criminals and, if a crime is committed, track the perpetrator. This can be seen through its 96% availability in homicide investigations (Yard, 2010). However, for it to be used someone must watch all relevant footage highlighting points of interest, which can be considered as anomalies in the footage. If these anomalies could be detected in real-time a more pro-active approach to policing could be evolved.

This is the key problem with current CCTV, it does not provide real-time analysis or understanding of what is happening within the video stream. This means CCTV can only be adopted in a post-event capacity, with a large time input from users. It also means, due to its simple nature, all video footage must be stored for long periods of time, when a user may only be interested in the anomalous periods within the video stream. To solve this a system needs to be developed that can work on existing CCTV systems and is able to understand events occurring within a video, detecting and storing anomalies.

Systems in the market that could contribute to providing this functionality, either only provide sub-parts of the desired system, or require specialized hardware to be used which means current deployed CCTV cameras would have to be replaced during adoption (Table 2).

To enable the demands of intelligent video, a framework should be developed that can be used as a base for complex development projects, adapting to individual use cases and providing common functionality by default. A proposed framework should be applicable to many use cases through configuration and extendibility. Furthermore, it must be easy to scale, allowing it to meet the demands of any user.

## Aim

Propose an extendable framework for object detection and tracking, behavior and activity analysis, and anomaly detection, making use of distributed computing and machine learning.

## Objectives

1. To research existing video processing techniques and available software packages in order to detect objects and events within a video stream.
2. To research machine learning techniques for detecting anomalies in time series data produced from objective one.
3. Develop testing scenarios that will allow the evaluation of machine learning models in their ability to detect anomalies in real-time.
4. Develop a framework that provides a minimum viable product of object, event and anomaly detection, while being scalable and extensible.
5. Using the test scenarios defined in objective three, evaluate the applications ability to detect anomalies and alert users in real-time.
6. Compare and contrast the performance and storage requirements of the proposed framework against existing CCTV technologies and approaches.

## Paper Structure

I will describe my paper structure here.

# Background and Literature Review

When developing an extendable framework for video processing, we must consider both software and hardware designs that will allow the framework to be suitable for largest variety of video stream analysis scenarios. To do this, we must first look at common video processing methodologies and the software techniques adopted in providing them. Further to this, we need to decide how best to scale and extend the framework, making it a contender for high frequency scenarios. This will involve the investigation and adoption of Cloud Computing. The combination of suitable software and hardware decisions will provide the base for implementing a successful framework.

## Video Processing Methodologies and their Adoption

In order to develop an effective video processing framework that enables accurate autonomous decisions, we must be able to understand what is occurring within a series of sequential video frames. Existing Architecture designs presented (Figure 1) show theoretically the different stages you must build upon to enable successful video processing. These stages build upon each other, until we reach a point of desired intelligent understanding of the processed video stream.

The base of these stages is the ability to detect objects within a frame that are of interest to us. From this, tracking can then occur from frame to frame, giving objects persistent identity. Detection and tracking gives our first intelligent understanding of what is occurring in the video, and provides the basis for more advanced techniques. Advanced techniques often include facial recognition and person identification, along with behavior and activity analysis, all of which require the knowledge of objects location and previous movements of the object through the video. Given we are able to provide this intelligence over the video stream we can then build an autonomous way of providing anomaly detection on the seen behaviors, allowing real-time alerts to be sent to users.



Figure 1: Adaptation of video processing within a general framework for automated visual surveillance system (Ko, 2008).

### Object Detection Techniques

The first stage of the video processing pipeline gives the ability to detect objects accurately. Detecting objects in images relies on being able to interpret the combinations of pixels correctly to identify the particular object you are looking for. The most common method of doing this is using a Haar feature-based cascade classifier (P. Viola and Jones, 2001). This machine learning approach works by showing a classifier a multitude of images, with some containing the object you wish to detect. The classifier then attempts to apply features to the image that allow it to accurately detect the desired object. Once the classifier is trained, we extract the Haar features from the images, where each feature is a single value obtained by subtracting the sum of pixels under the white rectangle from the sum of pixels under the black rectangle (Figure 2).

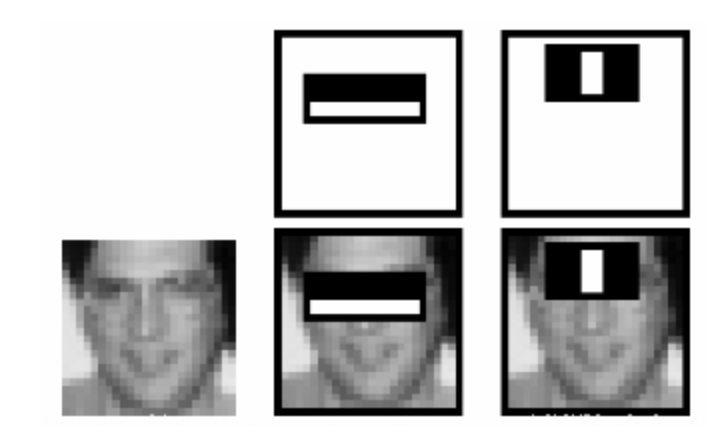
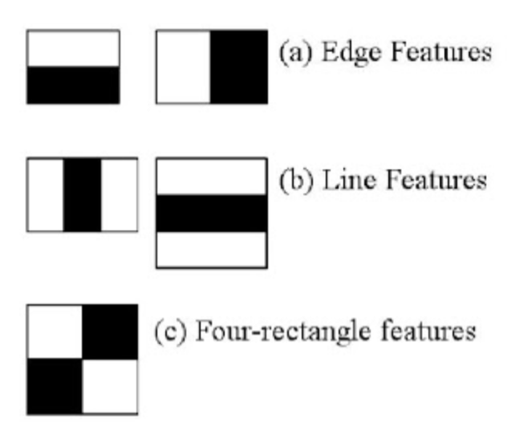


Figure 2: Left, an example of Haar cascade features. Right, the adoption of Haar cascade features in detecting a face (OpenCV, 2018).

From this point we are able to improve the performance of detections using an algorithm called Adaboost (P. a Viola and Jones, 2001), which allows ranking of features based on their error rate when attempting to successfully identify the object.

Further work has been proposed in this field using a deep learning based approach (Lecun, Bengio and Hinton, 2015). However, to make this approach feasible within real-time constraints specialized GPU (Graphics Processing Unit) hardware must be used, which would limit the extensibility of any proposed framework making use of these techniques.

### Object Tracking Techniques

Once we have accurate object detection we then need to be able to identify the same object seen through multiple frames of the video stream, employing tracking techniques. Tracking works on an online basis, meaning as new sequential data becomes available to the model, it can adapt at runtime to provide improved predictions on future data. Usually, this incorporates identifying an objects appearance and motion pattern allowing the accurate predictions of an objects location as it moves. When providing object tracking capabilities a variety of algorithms have been developed and made available (Table 1).

|  |  |  |  |
| --- | --- | --- | --- |
| Approach | Overview | Pros | Cons |
| Multiple Instance Learning | 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 gives the learning algorithm the responsibility of removing the ambiguity of the exact object location and predicting which instance in each bag is most correct (Baben and Belongie, 2009). | Works well when object partially occluded due to its bag representation. | The tracker cannot handle full occlusion of objects well. |
| Median Flow | Tracks an object by selecting a variety of points within the object space and then computes the trajectory for this object, both forwards and backwards in time. This means the tracker is able to compare both trajectories and make accurate predictions of the objects final location (Kalal, Mikolajczyk and Matas, 2010). | Works well when an objects trajectory is predictable. | The tracker becomes less reliable under fast or unpredictable motion. |
| Tracking-Learning-Detection | Breaks the tracking problem up into: Tracking, learning and detection. The tracker then follows an object between frames with the detector correcting the tracker based on all observed appearances. The learning estimates the error of the detector and updates it to avoid these errors in future (Kalal, Mikolajczyk and Matas, 2012). | Works well under long periods of occlusion. | The tracker does not perform well under large full rotations and can create lots of false positives. |

Table 1: A comparison of three common approaches to object tracking.

### Behavior and Activity Analysis

Detection techniques.

## Anomaly Detection with Machine Learning

This will be a talk on anomaly detection and machine learning.

### Anomaly Detection Models

Talk about the models specifically and research done into them.

### The Impact of Human Behavior

Talk about how humans may affect ability to detect anomalies.

## Distributed Computing and the Cloud

In order to provide video stream analysis at scale a large amount of computing power is required. Processing video with real-time requirements requires the latency of any system to be kept under a desired threshold, even when presented with high volumes of data. In recent years the availability of processing power has become more widely available, and cheap to access, through Cloud computing suppliers. This has enabled the adoption of distributed computing technologies that allow the parallel completion of tasks, enabling video processing to be possible at any scale. Tools developed in recent years utilize distributed computing, and in this section, we will discuss their use and application in the field of computer vision.

### Parallel Computing

Within the realm of video processing large amounts of calculations are performed on individual frames, ranging from initial object detection to tracking and analyzing. For these computationally expensive operations to scale as more video streams are added to an application, we must be able perform these calculations in parallel. Two of the most popular frameworks that supply the capability of horizontally scalable distributed computing are Apache Storm (Apache, 2017) and Apache Flink (Apache, 2018).

Apache Storm is a real-time distributed stream data processing engine that, when given a source of data events known as a spout, is able to perform guaranteed parallel computing as events are created. Apache Storm runs on a distributed cluster of machines, where a data processing pipeline is defined as a topology. A topology can be viewed as an execution plan for an individual event, with each task within a topology called a bolt. This execution happens within a worker process, with its state managed by a Zookeeper service. As events flow through a topology they are tracked as they leave each bolt, allowing Apache Storm to provide ‘at least once’ processing of events. Apache Storm then uses service known as Nimbus to schedule and maintain the flow of data through a topology. This architecture gives Apache Storm power to process and produce billions of events in a single day (Toshniwal *et al.*, 2014).

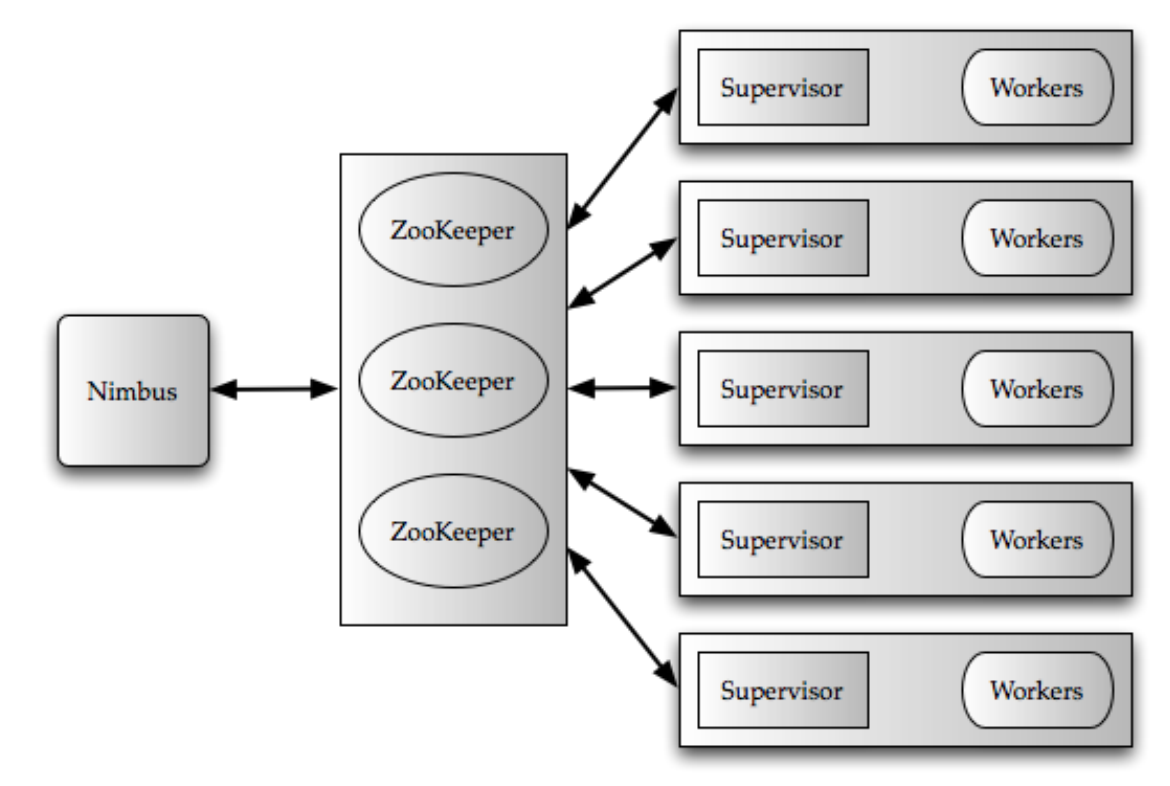


Figure 3: A high level architecture overview of Apache Storm (Toshniwal *et al.*, 2014).

A competitor to Apache Storm is the Apache Flink framework. Apache Flink is system for processing streaming and batch data. Similar to Apache Storm, it works by expressing data processing as a pipeline of tasks that need to be completed on some event source. An Apache Flink architecture is built upon the core components of; a client, a Job Manager and at least one Task Manager. The client takes user code and transforms it into a dataflow graph, similar to Apache Storm. The Job Manager then coordinates the distributed execution of the dataflow, tracking state and progress of each operation and stream. This then leaves the actual data processing to be executed within the Task Managers, with the result of any operation being reported to the Job Manager. Differing from Apache Storm, messages are sent between stages in the pipeline using buffers, using the back pressure from these buffers to control the throughput of the pipeline. This allows Apache Flink to provide throughputs of up to 80 million events per second (Carbone *et al.*, 2015).

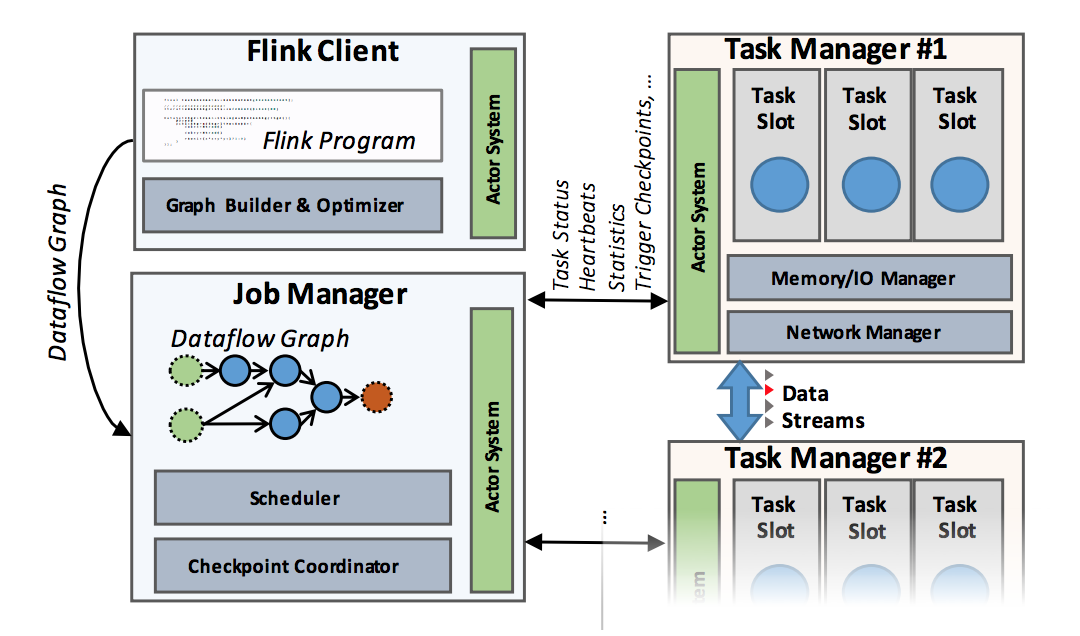


Figure 4: The Apache Flink processing model, showing the architecture adopted from an initial program definition to execution within an Apache Flink cluster (Carbone *et al.*, 2015).

As Apache Storm and Apache Flink offer the same stream processing service, performance comparisons can be made between them. Under high workloads, it has been found that Apache Flink out performs Apache Storm marginally with regards to latency, with both systems producing similar throughputs (Chintapalli *et al.*, 2016). However, all proposed systems are in their infancy of development, therefore no full conclusion can be made about the final performance of each system.

### Distributed Messaging

Apache Kafka, talk about the key technologies.

### Cloud Providers

Talk about cloud providers and their benefit.

## Existing Technologies and Approaches

|  |  |  |  |
| --- | --- | --- | --- |
| Product | Overview | Functionality | Limitations |
| Nest (Nest, 2017) | Gives event and motion detection allowing you to be alerted through an app in real-time if something happens within your home out of the ordinary. | Provides event and anomaly detection and real-time alerts when these occur. | Requires specialized hardware, is not extendable for development and is marketed for home use only. |
| A Video Analysis Framework for Surveillance System (Suvonvorn, 2008) | Gives a novel framework approach to online video analysis using .NET 2.0. | Provides object detection, event detection and extendibility to insert new functionality within the application. | Does not make use of distributed computing so will be unable to scale to all user requirements. It is also written in a language that requires a Microsoft workstation to run. |
| DiVA (SanMiguel *et al.*, 2008) | 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. This can be extended by implementing modules/algorithms that communicate through the integrated database. | 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 2: A list of current technologies in the market that aim to provide smart CCTV.

References

Apache (2017) *Apache Storm*. Available at: http://storm.apache.org/ (Accessed: 19 February 2018).

Apache (2018) *Apache Flink*. Available at: https://flink.apache.org/ (Accessed: 19 February 2018).

Baben, B. and Belongie, S. (2009) ‘Visual tracking with online Multiple Instance Learning’, *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 983–990. doi: 10.1109/CVPR.2009.5206737.

Carbone, P. *et al.* (2015) ‘Apache Flink: Unified Stream and Batch Processing in a Single Engine’, *Data Engineering*, 36, pp. 28–38. doi: 10.1109/IC2EW.2016.56.

Chintapalli, S. *et al.* (2016) ‘Benchmarking streaming computation engines: Storm, flink and spark streaming’, in *Proceedings - 2016 IEEE 30th International Parallel and Distributed Processing Symposium, IPDPS 2016*, pp. 1789–1792. doi: 10.1109/IPDPSW.2016.138.

Kalal, Z., Mikolajczyk, K. and Matas, J. (2010) ‘Forward-backward error: Automatic detection of tracking failures’, in *Proceedings - International Conference on Pattern Recognition*, pp. 2756–2759. doi: 10.1109/ICPR.2010.675.

Kalal, Z., Mikolajczyk, K. and Matas, J. (2012) ‘Tracking-learning-detection’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7), pp. 1409–1422. doi: 10.1109/TPAMI.2011.239.

Ko, T. (2008) ‘A survey on behavior analysis in video surveillance for homeland security applications’, *Applied Imagery Pattern Recognition Workshop, 2008. AIPR ’08. 37th IEEE*, pp. 1–8. doi: 10.1109/AIPR.2008.4906450.

Lecun, Y., Bengio, Y. and Hinton, G. (2015) ‘Deep learning’, *Nature*, pp. 436–444. doi: 10.1038/nature14539.

Nest (2017) *No Title*. Available at: https://nest.com/uk/cameras/nest-cam-indoor/overview/ (Accessed: 24 November 2017).

Norris, C. and McCahill, M. (2006) ‘CCTV: Beyond penal modernism?’, *British Journal of Criminology*, 46(1), pp. 97–118. doi: 10.1093/bjc/azi047.

OpenCV (2018) *Facial Detection with Haar Cascades in OpenCV*. Available at: https://docs.opencv.org/3.3.0/d7/d8b/tutorial\_py\_face\_detection.html (Accessed: 30 January 2018).

SanMiguel, J. C. *et al.* (2008) ‘DiVA: A Distributed Video Analysis Framework Applied to Video-Surveillance Systems’, in *International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS)*, pp. 207–210. doi: 10.1109/WIAMIS.2008.29.

Suvonvorn, N. (2008) ‘A video analysis framework for surveillance system’, *2008 IEEE 10th Workshop on Multimedia Signal Processing*, pp. 867–871. doi: 10.1109/MMSP.2008.4665195.

Toshniwal, A. *et al.* (2014) ‘Storm@twitter’, in *Proceedings of the 2014 ACM SIGMOD international conference on Management of data - SIGMOD ’14*, pp. 147–156. doi: 10.1145/2588555.2595641.

Viola, P. a and Jones, M. J. (2001) ‘Fast and Robust Classification using Asymmetric AdaBoost and a Detector Cascade’, *Advances in Neural Information Processing System*, (December), pp. 1311–1318. Available at: https://papers.nips.cc/paper/2091-fast-and-robust-classification-using-asymmetric-adaboost-and-a-detector-cascade.pdf.

Viola, P. and Jones, M. (2001) ‘Robust real-time object detection’, *International Journal of Computer Vision*, 57(2), pp. 137–154. doi: http://dx.doi.org/10.1023/B:VISI.0000013087.49260.fb.

Yard, S. (2010) *CCTV in Homicide Investigations*. Available at: https://goo.gl/oS5Tgn (Accessed: 15 November 2017).

Footnotes

1[Add footnotes, if any, on their own page following references. For APA formatting requirements, it’s easy to just type your own footnote references and notes. To format a footnote reference, select the number and then, on the Home tab, in the Styles gallery, click Footnote Reference. The body of a footnote, such as this example, uses the Normal text style. (Note: If you delete this sample footnote, don’t forget to delete its in-text reference as well. That’s at the end of the sample Heading 2 paragraph on the first page of body content in this template.)]

Tables

Table 1

[Table Title]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Column Head | Column Head | Column Head | Column Head | Column Head |
| Row Head | 123 | 123 | 123 | 123 |
| Row Head | 456 | 456 | 456 | 456 |
| Row Head | 789 | 789 | 789 | 789 |
| Row Head | 123 | 123 | 123 | 123 |
| Row Head | 456 | 456 | 456 | 456 |
| Row Head | 789 | 789 | 789 | 789 |

Note: [Place all tables for your paper in a tables section, following references (and, if applicable, footnotes). Start a new page for each table, include a table number and table title for each, as shown on this page. All explanatory text appears in a table note that follows the table, such as this one. Use the Table/Figure style, available on the Home tab, in the Styles gallery, to get the spacing between table and note. Tables in APA format can use single or 1.5 line spacing. Include a heading for every row and column, even if the content seems obvious. A default table style has been setup for this template that fits APA guidelines. To insert a table, on the Insert tab, click Table.]

Figures title:

Figure 1. [Include all figures in their own section, following references (and footnotes and tables, if applicable). Include a numbered caption for each figure. Use the Table/Figure style for easy spacing between figure and caption.]

For more information about all elements of APA formatting, please consult the APA Style Manual, 6th Edition.