This section details the techniques investigated for optimising communication between the client and server. An established limitation of HE is the size of ciphertexts [MAKKAOUI]. Consequently, the \textit{transmission time} of data is significantly impaired. The \textit{transmission time} can be defined by Equation \ref{eq:transmission}.

EQUATION

Transferring large volumes of data to the cloud is a critical component of the MLaaS model, so a substantial portion of the investigation was dedicated to reducing transmission time. The problem was considered from two angles: reducing the \textit{video size} and increasing the \textit{transmission rate}.

# Video Size

## Seam Carving

Developed by Avidan and Shamir in 2007, \textit{seam carving} describes a method of resizing images using \textit{geometric constraints} while also considering \textit{image content} [SEAMCARVING]. Consequently, an image can be resized while preserving important features, such as people or buildings. There are two categories of techniques for distinguishing these features. \textit{Top-down} methods use tools such as \textit{face detectors} to highlight features in the image [VIOLA]. \textit{Bottom-up} approaches use saliency maps\footnote{a representation highlighting the regions of an image where a person’s eyes are first drawn [SALIENCY].} to locate important areas [ITTI].

The details of the original seam carving paper, and a depiction of the algorithm, are included in Appendix \ref{app:seamCarving}. However, a more advanced algorithm was implemented to investigate more optimal results.

To quantify the importance of a pixel, seam carving defines an \textit{energy function}. Rubinstein et al.\ [RUBINSTEIN] proposed the \textit{forward energy} function using dynamic programming. This method calculates the energy of a pixel by accounting for the impact on future energies if it is removed. To achieve this, the \textit{energy difference} function is defined by Equation \ref{eq:energyDiff}. The cost of removing pixels, $C$, is measured as the forward differences between the pixels that would become neighbours after deletion. There are three cases for this: diagonally adjacent in each direction and orthogonally adjacent, depicted by Figure \ref{fig:adjacency} and defined by Equation \ref{eq:adjacency}.

EQUATIONS

Once the energy has been calculated, the image can be split into \textit{seams}. A \textit{vertical seam} is a path of pixels connecting the top of an image to the bottom, only including a single pixel in each row. Likewise, a \textit{horizontal seam} connects the left of an image to the right, only including a single pixel from each column. These is defined by Equation \ref{eq:vSeam} and Equation \ref{eq:hSeam} respectively.

EQUATIONS

for an $n \times m$ image, \vec{I}, where $x : [1, \ldots, n] \rightarrow [1, \ldots, m]$ and $y : [1, \ldots, m] \rightarrow [1, \ldots, n]$.

From this, the \textit{optimal seam} can be found. That is, the seam that minimises the \textit{seam cost} – the sum of all pixel costs in the path. Some implementations will use variants of Dijkstra’s algorithm for this. Alternatively, Equation \ref{eq:forwardEnergy} defines a dynamic programming approach.

It is important to note that there have been several extensions to seam carving that may apply to this project. Particularly, optimisations for videos by introducing two-dimensional seams to allow time to be accounted for, and implementations using GPUs to reduce execution time [RUBINSTEIN, DUARTE].

## Graph Representations

Representing images using graphs has several advantages. Firstly, graphs are discrete, mathematically simple objects with an established set of provably correct algorithms. More pertinently, graphs provide flexible representations that can be used to tune image size.

Graph-based image processing methods operate on \textit{pixel adjacency graphs} - graphs whose vertex set is the set of image pixels and edge set defines adjacency of pixels. An example of some pixel adjacency graphs is given by Figure \ref{fig:pixelAdjacency}. Three-dimensional pixel adjacency graphs can account for relationships between video frames. Examples of are depicted by Figure \ref{fig:3dAdjacency}.

However, to improve the video size, pixel adjacency graphs must be extended to \textit{region adjacency graphs}. Rather than representing each pixel with a node, pixels are amalgamated into regions represented by a single node. Figure \ref{fig:pixelToRegion} provides a pictorial example of this.

To achieve this, similarity between pixels must be quantified. This constitutes another image segmentation problem. Consequently, established algorithms producing valid solutions exist. Unsupervised clustering algorithms such as \textit{the watershed transform} [WATERSHED] or \textit{k-means clustering} [KMEANS] are two methods that have proved useful in existing works.

Importantly for this investigation, the number of regions in the image will directly impact the transmission time. Reducing the number of nodes in the graph is advantageous because it reduces the amount of data transmitted. However, this also reduces image resolution. Consequently, removing too many nodes from the graph will remove clarity, making inference worthless. Figure \ref{fig:regions} depicts this. Therefore, a balance must be struck heuristically to maximise video size reduction while minimising impact on video quality. This optimal point is likely to be different for every image, adding a further layer of complexity.

Using similar techniques to seam carving, it is possible to make this trade-off less severe. For example, \textit{Foveal sampling}, depicted in Figure \ref{fig:foveal}, utilises the visual activity of the eye to determine regions [FOVEAL]. The \textit{Fovea centralis} is a region of the retina responsible for the sharp central vision used by mammals to focus on particular objects. Consequently, its shape can be used to bias the placement of nodes of a graph to prioritise more critical areas. This allows the region budget to be used more efficiently to reduce noticeable quality reduction. Other techniques have been developed using saliency maps or similar.

# Transmission Rate

This section targets the bottlenecks limiting the transmission rate of the system by investigating the application of \textit{parallel computing}.

Figure \ref{fig:parallelStack} depicts the abstract layers of the application’s networking processes. The components are split into two categories: the \textit{client} and \textit{server} components handling communication, and the \textit{packing} and \textit{unpacking} components manipulating data before and after transmission. Parallelisation was selected for investigation because of the growing support in computer architecture. Traditionally, computer design has focussed on \textit{sequential computing} to improve performance. However, factors such as Dennard scaling [DENNARD] mean the improvements predicted by Moore’s law [MOORE] may not continue indefinitely. Therefore, architects are utilising multiprocessing to achieve similar gains. Consequently, this seemed like a viable opportunity for the investigation to consider future iterations of surveillance technology.

## Communication

Parallelisation already exists in networking protocols. The \textit{Transmission Control Protocol} uses a \textit{sliding window protocol} to send groups of data packets concurrently, ensuring they are reordered correctly when received. Figure \ref{fig:slidingWindow} depicts this. These protocols exist in the \textit{data-link} layer of the \textit{OSI network model} [OSI]. This section of the investigation aimed to evaluate the result of moving parallelisation higher up this abstract stack.

Taking inspiration from sliding windows, instead of using a single network socket to send data in a single stream, videos are split into frames, and each frame is divided into packets. Meanwhile, a pool of threads is created, analogous to the window, and mapped to packets, enabling multiple sockets to be opened in parallel, increasing the amount of data sent simultaneously.

However, there are limitations to this technique. Firstly, more data must be transmitted than during sequential communication. The algorithm is non-deterministic, so the order of packet delivery cannot be guaranteed. Consequently, further information must be attached to packets to ensure videos are reassembled correctly. While this is worth noting, the size of this additional data is negligible compared to HE data, so it is not a critical issue.

A more pressing concern is the overhead of creating threads and establishing connections. This cost means creating too many threads will countermand parallelisation benefits and make transmission slower. Therefore, an optimal balance between the cost of parallelisation and the volume of data to send must be achieved to maximise improvements.

## Data Manipulation

Splitting videos into small packets has further advantages when preparing data for transmission. Before data can be transmitted, it must be prepared, or \textit{packed}. When data arrives at its destination, it must be \textit{unpacked}. There are three distinct stages to packing: \textit{encryption}, \textit{compression}, and \textit{serialisation}, which are reversed during unpacking. The encryption stage is missing from the server-side pipeline.

In a naïve implementation, each video pixel might be packed individually. However, this can be improved. The CKKS scheme operates on vectors of real values. Therefore, decomposing a frame into rows provides the opportunity for \textit{vectorising} the application by encrypting each row as a single ciphertext object. Consequently, the number of ciphertexts needed is reduced - for an $n \times m$ pixel frame, the number of objects is reduced from $nm$ to $n$ - so the time and space complexity of encryption is reduced from quadratic to linear complexity.

Moreover, decomposing a video into independent packets allows the packing and unpacking pipelines to be executed in parallel. Therefore, the cost of encryption, compression, and serialisation can be amortised by preparing multiple packets concurrently. This process can take advantage of the multithreading described in §\ref{sec:communication} so that no further overhead of thread creation is required. Consequently, potential improvement is only limited by the need for each stage to terminate entirely before the next begins, so reducing the size of packets allows pipelines to terminate faster. However, smaller quanta require more communication threads, so this cannot be indefinitely exploited.