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: attempting to reduce the \textit{video size} and attempting to increase 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 for distinguishing these features. Firstly, \textit{top-down} methods use tools such as \textit{face detectors} to highlight were the features appear in the image [VIOLA]. Whereas a \textit{bottom-up} approach uses saliency maps\footnote{a representation highlighting the regions of an image where a person’s eyes are first drawn, see [SALIENCY].} to locate the most important [ITTI].

The details of the original seam carving paper, as well as a depiction of the algorithm, are included in Appendix \ref{app:seamCarving}. However, a more advanced algorithm was implemented for this investigation to provide 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, such that there is only a single pixel from each row in the path. Likewise, a \textit{horizontal seam} connects the left of an image to the right, such that only a single pixel from each column is included. Formally, this 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 $y : [1, \ldots, m] \rightarrow [1, \ldots, n]$.

Using these definitions and the energy function, 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 pertinent to this investigation, 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 account for relationships between video frames when handling video files. An example of these is depicted by Figure \ref{fig:3dAdjacency}.

However, to improve the video size, pixel adjacency graphs must be extended to \textit{region adjacency graphs}. In this case, 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, the notion of similarity between pixels must be quantified. Perhaps surprisingly, this is 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 such methods that have proved useful in existing works.

The concerns surrounding weighting edges indicate that this problem may be more complicated than it first appears. In fact, grouping the nodes is a form of low-level image segmentation. While this makes the problem more computationally complex, it has the advantage that there exist well-established algorithms providing reasonable solutions to it. Unsupervised clustering algorithms such as \textit{the watershed transform} [WATERSHED] or \textit{k-means clustering} [KMEANS] are two such methods that have been applied to this problem previously.

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, in doing so, image resolution is also decreased. Consequently, removing too many nodes from the graph will remove any 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 to 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} is a method of recreating the visual activity of the eye when determining 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

Where the previous sections aimed to improve video transmission time by reducing the size of video files, this section targets the bottlenecks limiting the transmission rate of the system. To do this, the project investigates the application of \textit{parallel} and \textit{concurrent} computing.

Parallel computing is often conflated with \textit{concurrent computing}. However, the terms are distinct. Parallel computing means a task is broken down into numerous similar sub-tasks that can be completed independently [PARALLELISMVSCONCURRENCY]. Concurrent computing means each sub-task will address unrelated processes, often requiring inter-task communication [PARALLELISMVSCONCURRENCY]. In Figure \ref{fig:parallelStack}, the abstract layers of the networking processes have been coloured to indicate whether concurrent or parallel computing is used.

These techniques were selected for investigation because of the growing trend of support in computer architecture. Traditionally, computer design has focussed on \textit{sequential computation} 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 gain similar gains. Consequently, this seemed like a viable opportunity for the investigation to consider future iterations of surveillance technology.

## Communication

Parallelisation already exists in some communication protocols. The \textit{transmission control protocol} (TCP) uses a \textit{sliding window protocol} to send a group of data packets concurrently, ensuring they are ordered correctly at the receiving end. Figure \ref{fig:slidingWindow} depicts this. This and similar protocols exist in the \textit{data-link} layer of the \textit{OSI network model}. The goal of this section of the investigation was to attempt to move the parallelisation higher up the abstract stack.

Taking inspiration from sliding windows, instead of sending all video data in a single stream, videos are split into frames, and each frame is divided into packets. Meanwhile, a pool of threads can be created to represent the size of the window. When a packet is ready to be sent, a thread is assigned and establishes a connection with the server. Consequently, multiple connections will be open in parallel, allowing more data to be sent.

However, there are limitations to this technique. Firstly, more data will have to be transmitted than in sequential communication. The algorithm is non-deterministic, so there can be no guarantees about the order in which the packets will arrive after transmission. Consequently, further information must be provided 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. The cost is such that creating too many threads will remove parallelisation benefits or make transmission slower. Consequently, an optimal balance between the cost of parallelisation and the amount of data to send must be found to maximise gains from this approach.

## Data Manipulation

Splitting videos into small packets has further advantages. Before data can be transmitted, it must be prepared, or \textit{packed}. Equivalently, when data arrives at its destination, data must be \textit{unpacked}.

Depicted by Figure \ref{fig:packingAndUnpacking}, there are three distinct stages of packing in the client \textit{encryption}, \textit{compression}, and \textit{serialisation}. The unpacking process will reverse these stages in order. In the server-side of the project, the encryption and decryption operations are unsurprisingly missing from these pipelines.

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\footnote{This would fall under concurrency rather than parallelism because objects such as encoders, encryptors, and keys must be shared between processes.}.

Similarly, compressing and serialising subsections of video can be parallelised to improve time and space complexity. Therefore, once encryption has completed, these operations can be performed by the same thread. No further overhead of thread creation is required, so this improvement is only limited by the need for each stage to terminate entirely before the next begins. Consequently, the smaller the quantum of data they operate on, the quicker they will complete. However, smaller quanta requires more threads, so a similar balance to that described in §\ref{sec:communication} must be achieved.