MLPEG: Using Machine Learning for Data Compression

Anthony Valencia & Jason Wiemels

2022-2023

Science Research

Abstract

Video data is the most transmitted form of data over the internet. Although streaming is quick, it can be quicker. Making streaming quicker and more efficient will allow for devices like phones, tablets, and televisions to wait less when rendering 4K images. For long term archive storage of video footage can be improved to decrease storage size and overhead cost. MLPEG aims to make video footage have a smaller footprint and is quicker than H.264 (MPEG-4), which is the most common form of video compression currently.

MLPEG uses FILM, ESPCN, Python, and an AMD Ryzen 5 3600 processor to compress video files. Videos are compressed by selecting keyframes in a video through selecting data at a certain interval, downscaling the resolution on those images, and then saving those images into a file. To decompress, MLPEG will read the compressed file, run FILM–a video interpolator–to make up the frames that were lost in the compression process, and then upscale the images to the original resolution the video was at.

Using MLPEG, file sizes became increasingly smaller at an exponential rate by a factor of two. When compared with the file sizes of MPEG-4 videos MLPEG videos were relatively equal to or less than MPEG-4. Despite some discrepancies in decompressed videos, MLPEG was also able to maintain visual quality and resolution.

MLPEG shows that machine learning models may be a viable alternative or addition to traditional forms of data compression.

Introduction

In the early years of computers, there was a need for saving space on a hard drive. Limited hard drive space meant that programs had to be small. Eventually, data compression became a method of reducing file sizes and saving precious hard drive space. The advent of data compression meant that there were many different algorithms going about and subsequently there was a need for standardization of data compression (Le Gall, 1991). MPEG (Moving Picture Expert Group) was founded on the basis of standardizing data compression for audio, video, and images. They created the file we are familiar with today: MPEG-3 (audio), MPEG-4 (video), and JPEG (image).

Today, data compression is used everywhere. New methods of compression have been a pivotal reason for the developments in the Internet, digital TV, mobile communication, and video communication (Pu, 2004). Since video data is the most transmitted form of data in the world and was projected to be two-thirds of traffic in 2017 (Pepper, 2013), there is a demand for smaller data sizes of videos so that video data can be transferred at a smaller, and therefore, more efficient speed on the internet.

Data compression can be broken into two different forms: lossy and lossless. Lossy compression means that the data that was compressed and decompressed is not exactly the same as the original data such as a discrepancy in a pixel's alpha value for an image. Lossless compression means that when compressed and decompressed, the resultant data is exactly the same as the input data (Pu, 2004). Video compression codecs typically come in the form of lossy because minute differences in an image are not necessarily perceived by the human eye. They rely on I, P, and B frames in a video to remove spatial and temporal redundancies in the frames of a video (Rippel et al, 2019). I-frames (“intra-coded”) are compressed using an image codec like JPEG, P-frames (“predicted”) are extrapolated from frames in the past, and B-frames (“bi-

directional") are interpolated from previously transmitted frames in the past and future (Rippel et al, 2019; Le Gall, 1991).

New machine learning model architectures in image manipulation have made it quicker and more accurate to upscale images and interpolate between frames in a video. Efficient Sub-Pixel Convolution Nueral Network (ESPCN) has made image upscaling quick and accurate. Given a 1080 HD (1920x1080 resolution) video image, ESPCN was able to outperform existing upscaling methods by more than ten times with speeds of 0.029 seconds per frame on some videos (Shi et al, 2016). For image interpolation, Google's Frame Interpolation for Large Motion (FILM) is able to interpolate–create a new frame given two input frames–a video. FILM was able to create frames that were more accurate in terms of color (Reda et al, 2022)

These new developments in machine learning can improve video compression because of the speed and accuracy of these models. Designed for addressing the demand for smaller video sizes, MLPEG combines the power of machine learning with video compression for compression sizes that are smaller than MPEG-4 and compress/decompress times that are better or relative to MPEG-4.

Method

Materials

- Python
- Tensorflow, Numpy, OpenCV, Pandas
- YouTube
- Inter4K: A Dataset for Video Interpolation and Super-Resolution in 4K
- AMD Ryzen 5 3600 CPU

Methods

First, the method of compression was built. The design of the compression algorithm was created to remove a pattern of frames at a sample interval. For instance, if the sample interval was 2, every other frame of the video would be saved. The purpose of this is to remove data to reduce the file size. The compression step also includes the downscaling of images using OpenCV. The image is downscaled by a factor of 4 because the upscale algorithm we are using (ESPCN) can only upscale by a maximum of 4. Because videos are a string of images, each frame (image) is saved into a ".mlpg" file which will be read from and decoded by the compression algorithm.

The decompression component was built second. This includes reading files and normalizing data to feed to FILM and our upscale algorithms. Reading files was important at first because reading and writing image files helped with seeing how data flowed throughout the program. Especially when failed runs of programs would lose the data. Using Jupyter Notebooks also maintained context of variable values so that data flow could be addressed sequentially throughout the program without loss of data. Normalizing of data was necessary when transferring between libraries that were being used. For instance, when an image was read in

from OpenCV it was in type “uint8” while the FILM tensorflow model required an image in type “float32”. Normalizing, or converting to the data type that was being used at the time, ensures that images look how they are supposed to and are not corrupted.

Lastly, was testing which included running the compress-decompress pipeline on the Inter4K video dataset. The script “test.py” first decimates the image and saves the separated images into a folder and an “.mlpg” file then takes the time for that whole process while testing at different sample intervals. Then the script decompresses the video and times how long it takes. The test calculates the ratio in size between the original image and MLPEG compressed video as well as the ratio in size between the MLPEG video and an MPEG-4 video.

Results

Quantitative Results

At the seven different sample intervals (2, 4, 8, 16, 32, 64, 128), MLPEG video sizes became increasingly smaller. For the 10 videos tested at sample interval 2, the average file size was 27.31 megabytes which is almost two times larger than the average file size of the input MPEG-4 videos. As the sample interval increased exponentially by a factor of 2, the file sizes got smaller exponentially by a factor of 2. For instance, sample interval 4 had an average file size of 13.66 MB and sample interval 8 had an average size of 6.91 MB. This also means the compression ratio between MLPEG and MPEG-4 decreased exponentially by a factor of two because the MPEG-4 video sizes remained constant.

Timing, on the other hand, was slower than MPEG-4 when compressing and decompressing. Compressing followed the trend of being an exponential decay by a factor of two while decompressing was less predictable in how long it took to decompress. MPEG-4 on average would take 30 seconds to compress a video while MLPEG would take a minute to three seconds (depending on sample interval). Where MPEG-4 encoding is nearly instantaneous, MLPEG on average took more than 20 mins to completely decompress a video.

Qualitative Results

As sample intervals got higher, more video data disappeared which resulted in the video looking less coherent. Additionally, for large and sudden movements, the frame interpolation cannot keep up and will misplace a group of pixels which may make edges look frayed or in worse cases misplace entire objects. During cut transitions in videos, MLPEG will interpolate the cut which adds unnecessary motion and frames to the video.

Discussion

The ratio of the MLPEG files to MPEG-4 files indicates that MLPEG is able to compress videos to a smaller size than MPEG-4. At the sample interval 4, MLPEG typically created files that were 1 MB to 2 MB smaller than an MPEG-4 encoded video. This means that MLPEG accomplished its goal of compressing video data smaller than MPEG-4 encoding. The trade-off trend between quality and file size was consistent with findings in image compression. The Identity Preserving Reconstruction (IPR) model lost image quality in favor of smaller compressions ratios (Xiao, 2022)

MLPEG's time to compress is equal to MPEG-4 and its time to decompress is not as fast as MPEG-4. MPEG-4 take is instantaneous in decompressing the videos while MLPEG will take at least 20 minutes to decompress a video. This is due to the extra calculations that MLPEG must take to run. MLPEG runs two different machine learning models and runs on the Python programming language.

Given the sizes of the MLPEG files, ratio of MLPEG to MPEG-4, compress/decompress times, and visual qualities, the most optimal sample intervals to compress videos is going to be within the 4-8 range of sample intervals. The file sizes in that sample interval are small enough to be better than MPEG-4, while preserving image quality and remaining fast enough when compared to other sample intervals.

Future research includes more testing in the 4-8 sample interval range to find a better medium between file sizes, speed, and image quality. Additionally, further time spent on optimizing MLPEG to run at a faster speed such as utilizing other machine learning algorithms for image compression and using a lower level programming language will boost the speed performance of MLPEG. For more accurate testing results, different computer processors should

also run MLPEG to see performance on other devices to see if a device with less resources will produce similar results.

References

Pu, I. (2004). Data compression. Undergraduate Study in Computing and Related Programmes. University of London.

Rippel, Oren, et al. "Learned video compression." Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019.

Pepper, Robert. "Cisco visual networking index (VNI) global mobile data traffic forecast update." Technical report, Cisco, Tech. Rep. (2013).

Le Gall, Didier. "MPEG: A video compression standard for multimedia applications." Communications of the ACM 34.4 (1991): 46-58.

Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., ... & Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1874-1883).

Reda, F., Kontkanen, J., Tabellion, E., Sun, D., Pantofaru, C., & Curless, B. (2022, November). Film: Frame interpolation for large motion. In Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part VII (pp. 250-266). Cham: Springer Nature Switzerland.

Xiao, J., Aggarwal, L., Banerjee, P., Aggarwal, M., & Medioni, G. (2022). Identity Preserving Loss for Learned Image Compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 517-526).

Appendix

Figure 1.1 – MLPEG's Compression Method

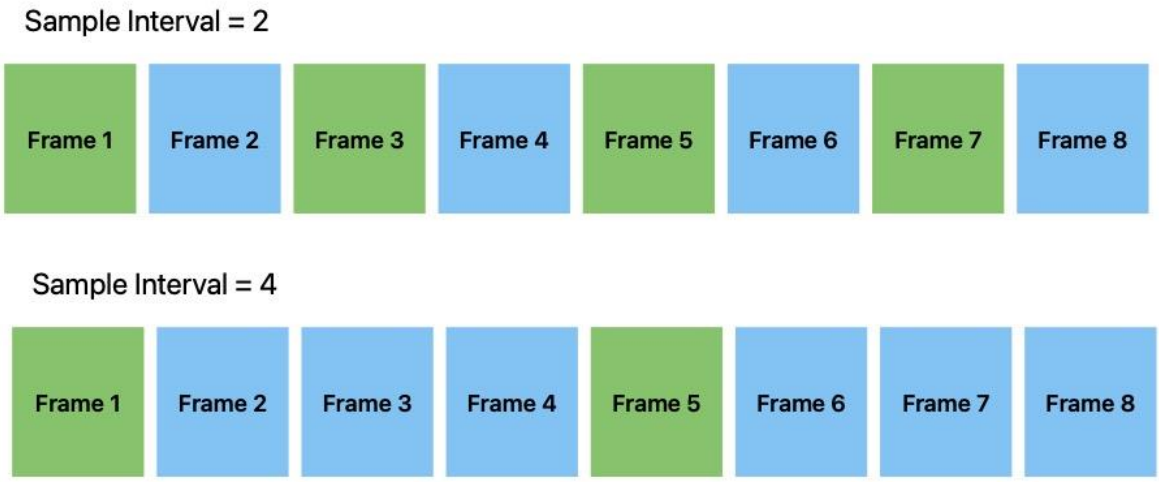


Figure 1.2 – MLPEG's Decompression Method

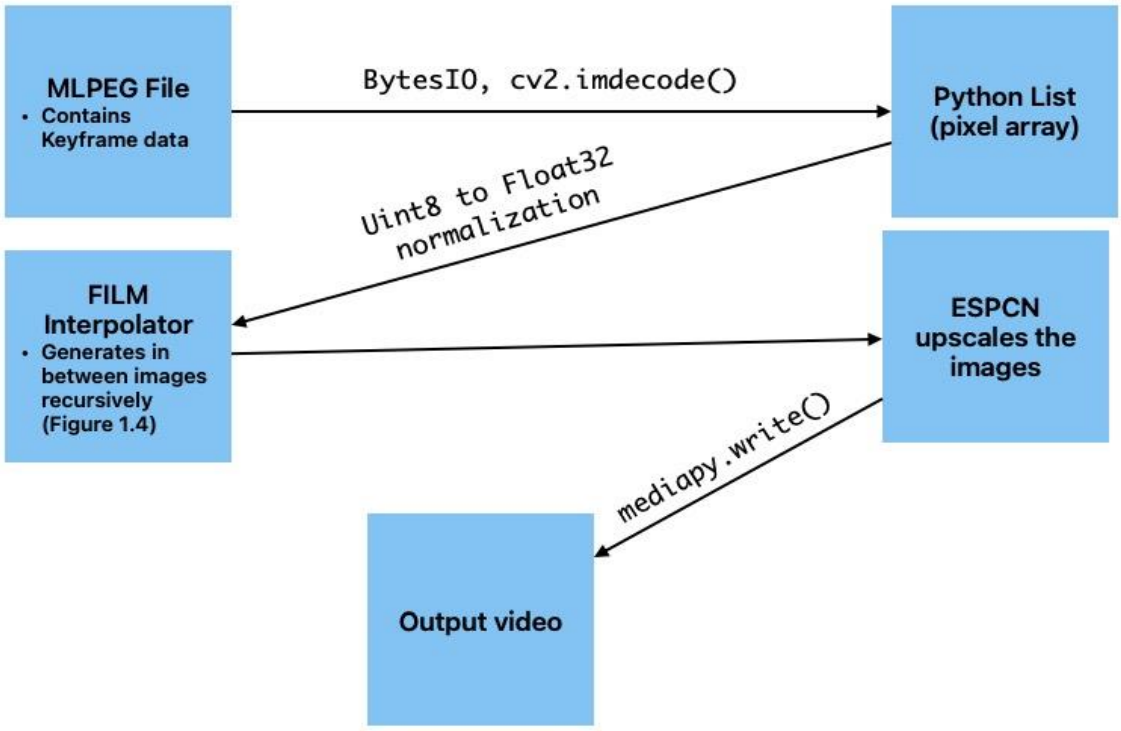


Figure 1.3 – FILM Interpolation: Given two input images one is generated for in between

Figure 1.4 – FILM Recursive Interpolation: Recursive representation of interpolation above. First generated then creates second generated with input images to interpolate.

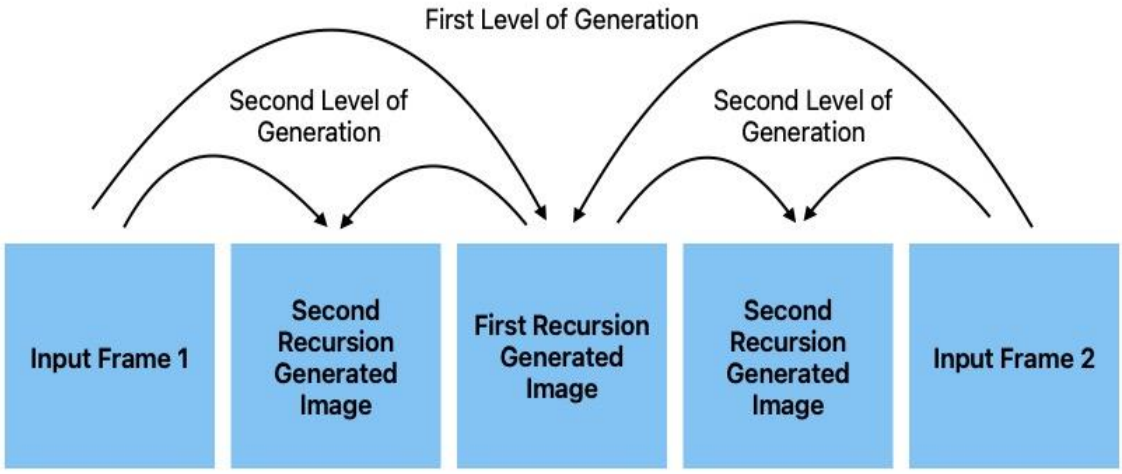
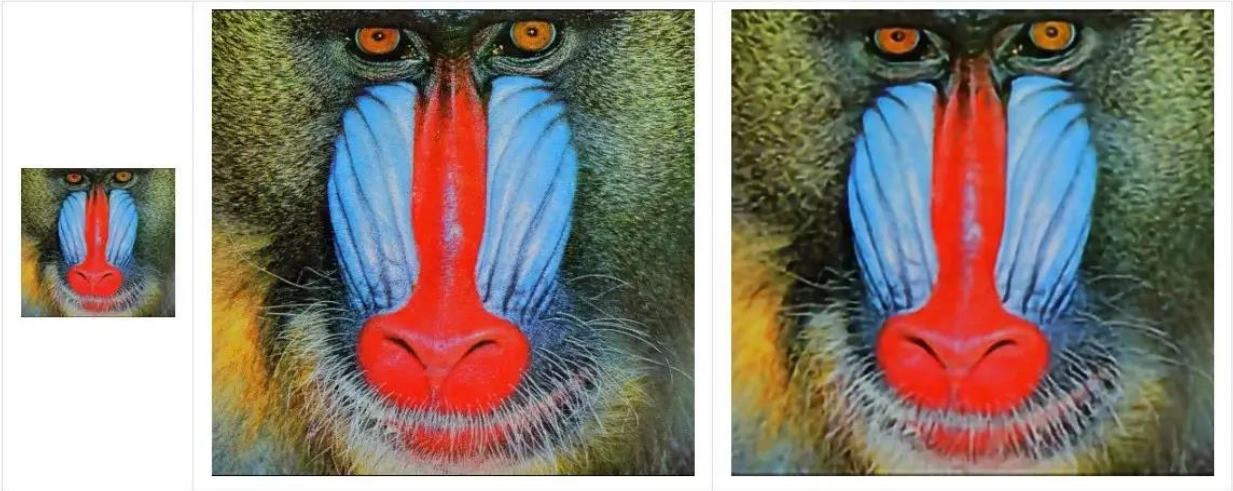


Figure 1.5 – A super-resolution example of ESPCN; Left: low-resolution image, Middle: original Image, Right: super-resolution result, Upscaling factor: 3



Source: zhuo Cen, Medium

Figure 1.6 – Testing Suite Results

Sample Interval 2

Inter4K Filename	MPEG-4 sizes (Bytes)	MLPEG sizes (Bytes)	Ratio (mlpg/mp4)	Compression Time (Seconds)	Decompression Time (Seconds)
1.mp4	11304067	18272736	1.616474495	52.80124235	1119.838737
156.mp4	12653142	22441103	1.773559721	63.27473068	1123.948545
2.mp4	25177768	41673450	1.65516856	86.14250231	1155.605619
3.mp4	29912201	38795568	1.296981389	110.6449261	1272.727749
4.mp4	21036459	32338075	1.537239466	112.3509462	1221.378205
43.mp4	5059672	14767686	2.918704216	37.42716289	1149.110417
5.mp4	14187840	33367193	2.351816274	101.0973794	1257.535869
69.mp4	14012037	21777400	1.554192299	48.54501271	1133.287172
7.mp4	13748691	27033195	1.966237731	72.59654951	1138.33119
80.mp4	7921244	22591209	2.851977417	65.39847541	1594.746594

Sample Interval 4

Inter4K	MPEG-4 sizes	MLPEG sizes	Ratio	Compression Time	Decompression

Filename	(Bytes)	(Bytes)	(mlpg/mp4)	(Seconds)	Time (Seconds)
1.mp4	11304067	9150861	0.8095193526	25.77977324	1598.692418
156.mp4	12653142	11213818	0.8862477004	30.97028399	1696.187432
2.mp4	25177768	20842880	0.8278287416	48.849962	1648.664836
3.mp4	29912201	19422324	0.6493110955	57.79728889	1684.817638
4.mp4	21036459	16168874	0.7686119608	44.46138644	1656.97424
43.mp4	5059672	7388685	1.460309087	24.48572636	1780.066223
5.mp4	14187840	16711523	1.177876477	60.329319	1667.237134
69.mp4	14012037	10891515	0.7772970482	33.7744689	1690.864428
7.mp4	13748691	13511330	0.9827357383	50.26615262	1659.083535
80.mp4	7921244	11272153	1.423028125	37.37516308	1611.87781

Sample Interval 8

Inter4K Filename	MPEG-4 sizes (Bytes)	MLPEG sizes (Bytes)	Ratio (mlpg/mp4)	Compression Time (Seconds)	Decompression Time (Seconds)
1.mp4	11304067	4638235	0.410315597	22.74543738	1897.045268
156.mp4	12653142	5730672	0.4529050571	21.7568047	1917.079088
2.mp4	25177768	10565159	0.4196225416	31.38230133	2090.567316
3.mp4	29912201	9810660	0.3279818827	41.53620362	1959.859966
4.mp4	21036459	8198864	0.3897454415	46.88282108	1968.989771
43.mp4	5059672	3733430	0.7378798468	23.56463861	1924.099245
5.mp4	14187840	8469598	0.5969617644	30.57977748	1967.750871
69.mp4	14012037	5489792	0.3917911436	18.3438015	2810.727045
7.mp4	13748691	6840509	0.4975389293	34.23227477	1951.566646
80.mp4	7921244	5657757	0.7142510697	27.92808318	1831.836093

Sample Interval 16

Inter4K Filename	MPEG-4 sizes (Bytes)	MLPEG sizes (Bytes)	Ratio (mlpg/mp4)	Compression Time (Seconds)	Decompression Time (Seconds)
1.mp4	11304067	2349266	0.2078248475	16.78882337	1949.856193
156.mp4	12653142	2935573	0.2320034818	12.74310851	2085.505846
2.mp4	25177768	5280906	0.2097448034	23.98728704	2534.698445
3.mp4	29912201	4911380	0.1641932	38.04984879	2008.721777
4.mp4	21036459	4116854	0.1957009019	22.74711227	2083.414353
43.mp4	5059672	1867039	0.3690039591	13.61734343	2041.406591
5.mp4	14187840	4234807	0.2984814461	24.46915412	1960.752388

69.mp4	14012037	2709353	0.193358967	6.895041227	2159.43291
7.mp4	13748691	3415440	0.2484192859	19.52371645	1957.030949
80.mp4	7921244	2926664	0.3694702499	10.45878935	1957.521864

Sample Interval 32

Inter4K Filename	MPEG-4 sizes (Bytes)	MLPEG sizes (Bytes)	Ratio (mlpg/mp4)	Compression Time (Seconds)	Decompression Time (Seconds)
1.mp4	11304067	1250116	0.1105899319	7.785771847	2018.354043
156.mp4	12653142	1529141	0.1208506946	10.89265275	2007.187786
2.mp4	25177768	2780733	0.1104439838	12.24899817	2056.050641
3.mp4	29912201	2586144	0.08645783037	11.9379797	2083.63454
4.mp4	21036459	2168422	0.1030792302	11.46385217	2129.408181
43.mp4	5059672	1013204	0.2002509254	8.674286842	2122.677311
5.mp4	14187840	2236118	0.1576080644	16.32043123	2121.079389
69.mp4	14012037	1489518	0.1063027453	13.17772675	2129.188908
7.mp4	13748691	1800111	0.1309296281	35.438236	2051.507511
80.mp4	7921244	1559738	0.196905688	22.44906282	2030.195457

Sample Interval 64

Inter4K Filename	MPEG-4 sizes (Bytes)	MLPEG sizes (Bytes)	Ratio (mlpg/mp4)	Compression Time (Seconds)	Decompression Time (Seconds)
1.mp4	11304067	656228	0.05805238062	5.803548813	1830.179873
156.mp4	12653142	835009	0.06599222549	8.897085667	1813.932294
2.mp4	25177768	1396344	0.05545940371	6.280879259	1798.154296
3.mp4	29912201	1331258	0.044505518	4.361325979	1811.618325
4.mp4	21036459	1094538	0.05203052472	3.704227209	1846.366643
43.mp4	5059672	504035	0.09961811754	9.271451473	1806.855965
5.mp4	14187840	1113154	0.0784583136	6.921127319	1805.501269
69.mp4	14012037	807040	0.0575961939	2.312932014	1803.598094
7.mp4	13748691	899052	0.0653918253	5.848042727	1812.851403
80.mp4	7921244	801255	0.1011526725	5.174343348	1801.65989

Sample Interval 128

Inter4K	MPEG-4 sizes	MLPEG sizes	Ratio	Compression Time	Decompression

Filename	(Bytes)	(Bytes)	(mlpg/mp4)	(Seconds)	Time (Seconds)
1.mp4	11304067	431512	0.03817316369	2.420104027	1815.231803
156.mp4	12653142	542584	0.04288136496	2.902349949	1836.313236
2.mp4	25177768	837842	0.03327705617	6.966519833	1815.334652
3.mp4	29912201	801023	0.02677913939	3.952387094	1831.549708
4.mp4	21036459	664554	0.03159058281	3.446572781	1823.731307
43.mp4	5059672	301167	0.05952302837	3.125184774	1856.272758
5.mp4	14187840	669816	0.04721056905	10.1213212	1839.162736
69.mp4	14012037	525509	0.03750411164	1.663832426	1814.52226
7.mp4	13748691	539495	0.03923973562	9.484451532	1829.337549
80.mp4	7921244	471604	0.05953660814	3.734176397	1817.829919

Averaging for each Sample Interval

Sample Interval	MPEG-4 size (Megabytes)	MLPEG size (Megabytes)	Ratio (mlpg/mp4)	Compression Time (minutes)	Decompression Time (minutes)
2	15.50	27.31	1.952235157	1.250464879	20.27751683
4	15.50	13.66	0.9762765327	0.6901492075	27.82410949
8	15.50	6.91	0.4938993274	0.4982535728	33.86586885
16	15.50	3.47	0.2488201143	0.3154670409	34.56390219
32	15.50	1.84	0.1323418722	0.2506483305	34.58213961
64	15.50	0.94	0.06782571754	0.09762493968	30.21786342
128	15.50	0.58	0.04157153598	0.07969483336	30.46547655

Figure 1.7 – Average MLPEG File Size

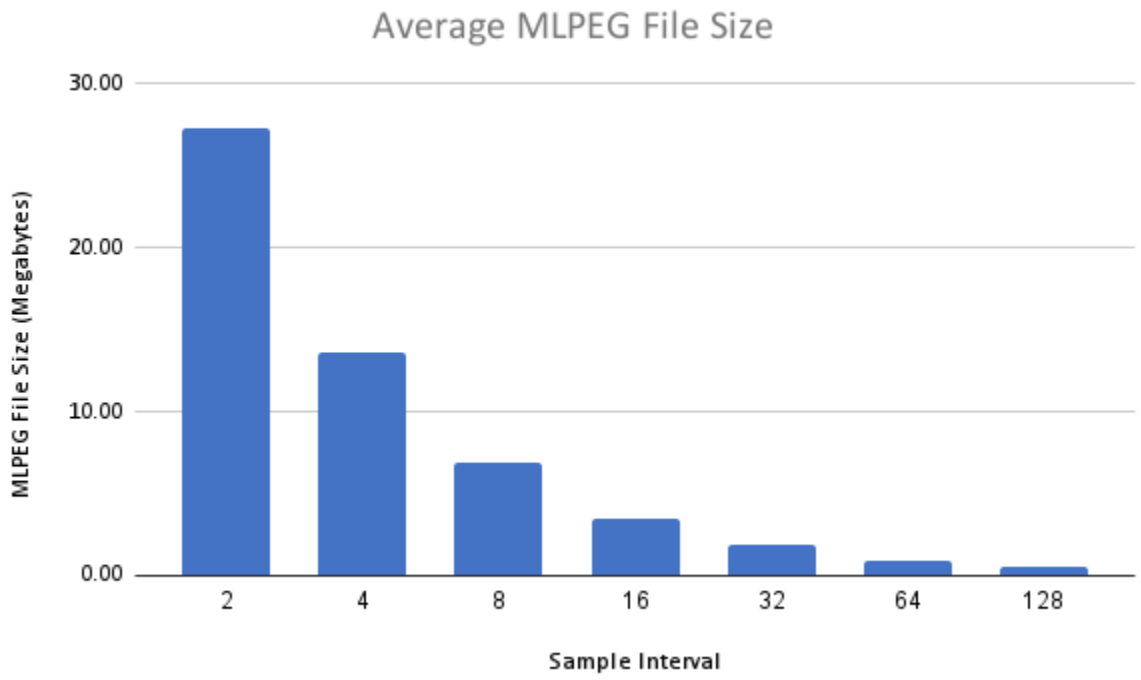


Figure 1.8 – Average Compression Ratio

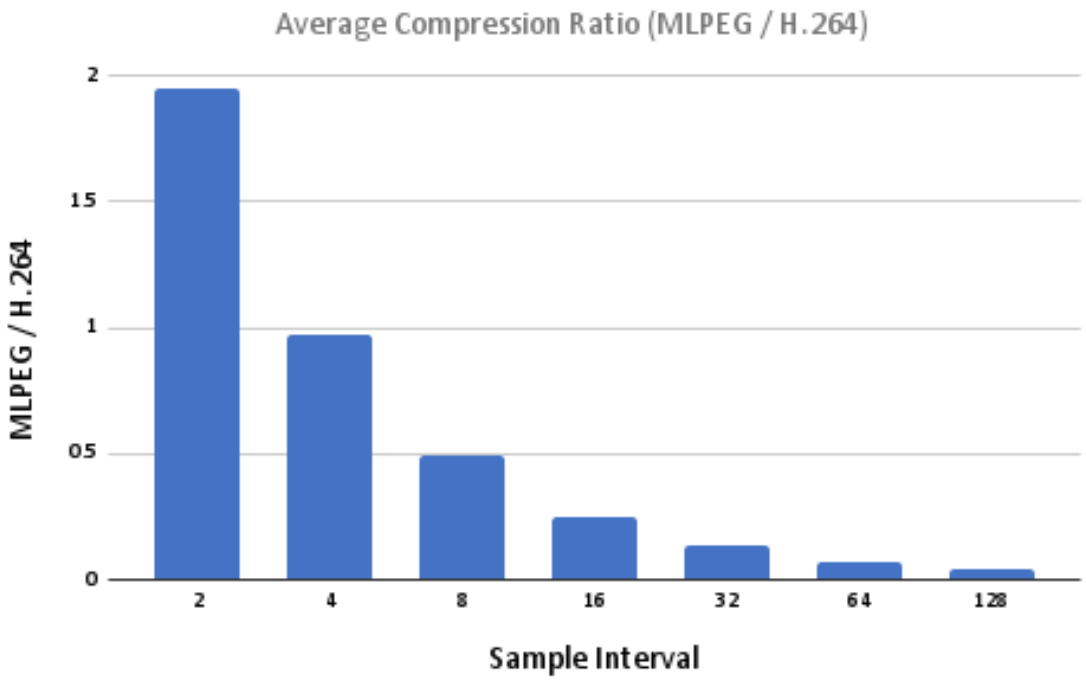


Figure 1.9 – Compression Time

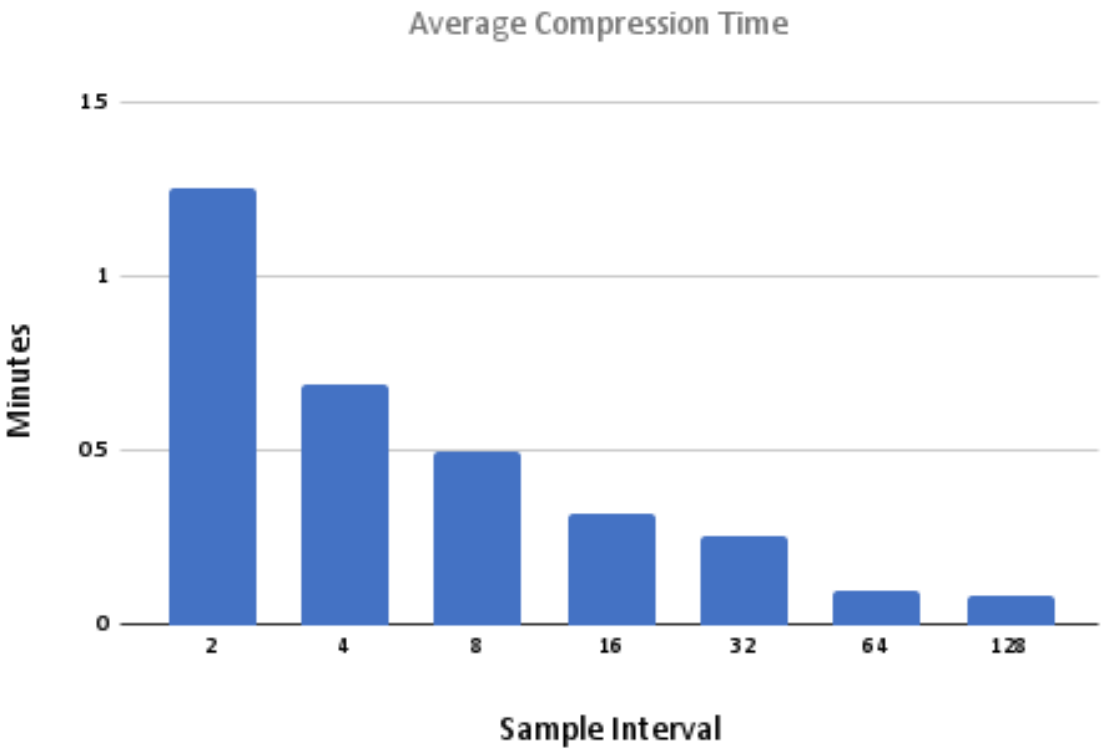


Figure 1.10 – Decompression Time