OVAL: Online Video Compression using Autoencoder Learning

Project Final Report

Project Group 27

Word Count: 2411

Isidor Kaplan (1005904005)
Adam Glustein (1006068425)
Ryan Ghosh (1006418627)
Khantil Desai (1006155161)

Apr. 13, 2022

APS360 - Applied Fundamentals of Machine Learning, Winter 2022

1 Introduction

Real-time video conferencing has become increasingly important with the catalyst of the COVID-19 pandemic. As video data is continuously generated, it must be compressed and decompressed with lossy reconstruction. Finding efficient compressions is a task well-suited to artificial neural networks, especially autoencoders, since they reduce inputs into smaller latent-space representations. Existing video compression methods are either algorithmic or based on offline training, and therefore cannot adapt to individual video streams. OVAL is an online learning model for video compression which finds efficient video encodings in real-time. The goal of the project is to encode video input with a high compression ratio and strong input/output similarity after decoding. By reducing the size of in-conference transmissions, OVAL can mitigate lag and other undesirable qualities.

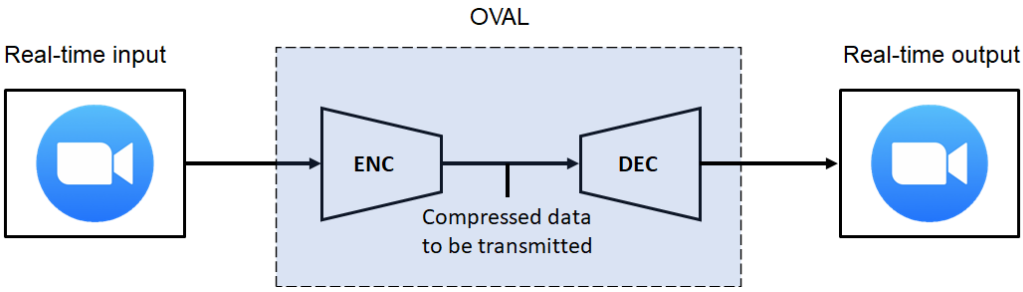


Figure 1: OVAL is an autoencoder for video data, primarily for use in video conferencing.

Our online learning approach is a novel method in the machine learning community. We expected strong results from online learning due to the nature of videoconferencing; individual video streams are largely static, which means that their encodings can be learned quickly.

2 Visualization of OVAL

In the *critical path* (**Figure 2**), video frames are encoded and sent to the receiver. The receiver decodes the data using parameters which are periodically updated by the sender. As the sender encodes frames, the model performs unsupervised learning using the input video data. In an isolated pipeline, data moves through a FIFO buffer to a training-stage autoencoder. The training error is compared to the live error and OVAL decides whether to update the receiver with new model parameters. The dynamic update policy allows large live errors to be corrected promptly while small errors need to be corrected infrequently.

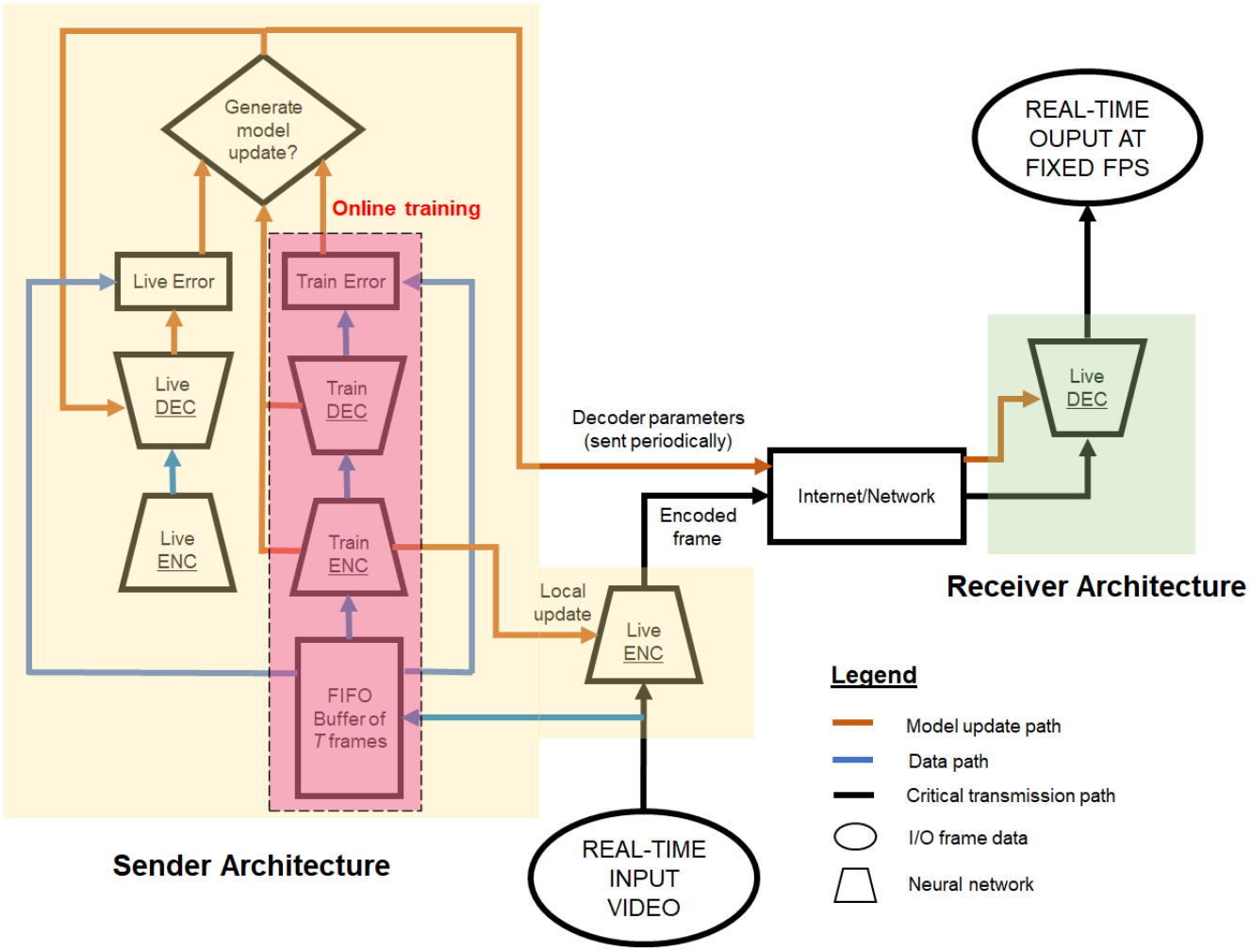


Figure 2: The high-level architecture of the OVAL model, allowing for concurrent online learning and encoding.

The sender's neural network autoencoders (**Figure 6**) are used separately for training and live encoding, which is necessary for online learning. The receiver's decoder does not need to be identical to the sender's if a small live error is preserved during transmission. Thus, having two separate live decoders reduces the need for transferring decoder parameters.

3 Background and Related Work

Autoencoders have previously been used for video compression and summarization, with most approaches using offline learning and offline encoding (offline-offline). A recent paper [1] used autoencoders for video compression with offline training but online encoding. Using a fixed recurrent neural network, the pre-trained model was able to encode videos in real-time. Upon testing, the model achieved a high input/output similarity index and outperformed existing offline-offline models.

Another recent paper [2] used online training with offline encoding. The model learned frame-by-frame representations for video summarization. However, when tested on new data, the model required the entire new video before it could encode a summarization. Their model achieved state-of-the-art performance for summarizing object motion in video. The encoder network architecture consisted of initial convolution layers for feature extraction and three series LSTM encoding layers.

There are no published online-online video compression implementations, which is the novelty of OVAL.

4 Data Processing

The OVAL autoencoder was pre-trained in an offline learning phase and continuously improves during the online learning phase.

4.1 Data Collection

Our training data was representative of video conferencing applications, which focus on a single or small set of speakers. Since no existing datasets met this criteria, we created our own dataset from YouTube. Training videos met the following criteria:

1. **Format** Videos focused on a set of speakers in front of a stable background. Examples include newscasters, political speeches, and video podcasts.
2. **Length** Videos were 2-5 minutes long. Longer videos would be too large to pre-load

as tensors in RAM memory, and shorter videos would provide insufficient frames for learning.

3. **Quality** Videos were selected in 360p and 720p. These are small enough to pre-load into RAM and also trained the model with different resolutions.

4. **Diversity** Videos featured a diverse set of speakers across race, gender, language and age. This ensured fairness in training.

Examples from the dataset include a press conference by Jagmeet Singh, a statement in German by Angela Merkel, and Greta Thunberg's speech at the UN. In total, 60 videos were pulled from YouTube with 40 for training and 10 for each of validation/testing.

4.2 Data Cleaning

We converted the raw `.mp4` files to PyTorch tensors for training. Gaussian noise was added to training frames to add resiliency to the autoencoder (*denoising* [3]). OVAL also dynamically downsamples training frames for the online learning component if necessary to ensure the online training stays up-to-date with live video. **Figure 3** shows the cleaning pipeline and **Figure 4** shows video frames with added Gaussian noise.

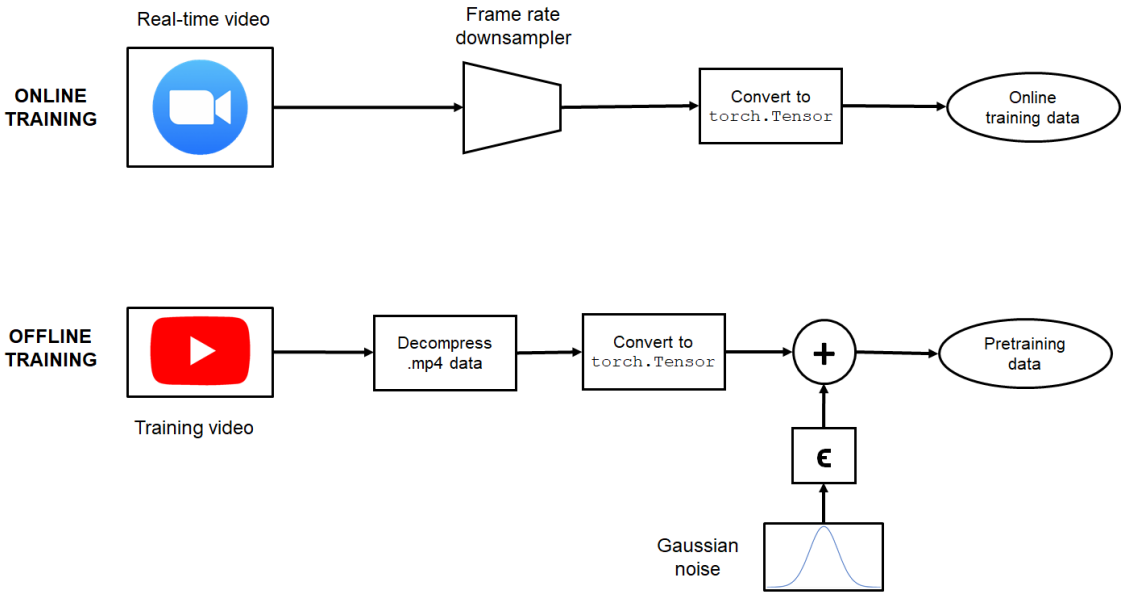


Figure 3: The data cleaning process for online/offline training.

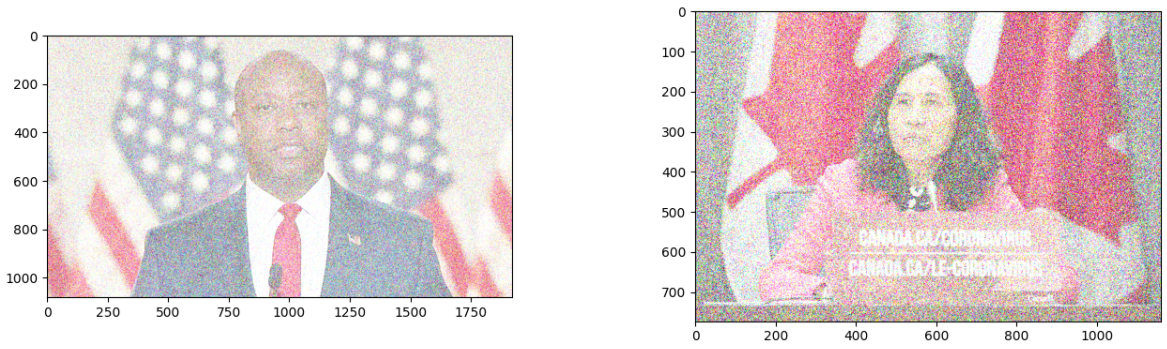


Figure 4: Scaled frames from training examples with added Gaussian noise.

4.3 Fairness and Diversity

We ensured a wide range of demographics were represented in our training data. **Figure 5** shows key diversity statistics of our dataset.

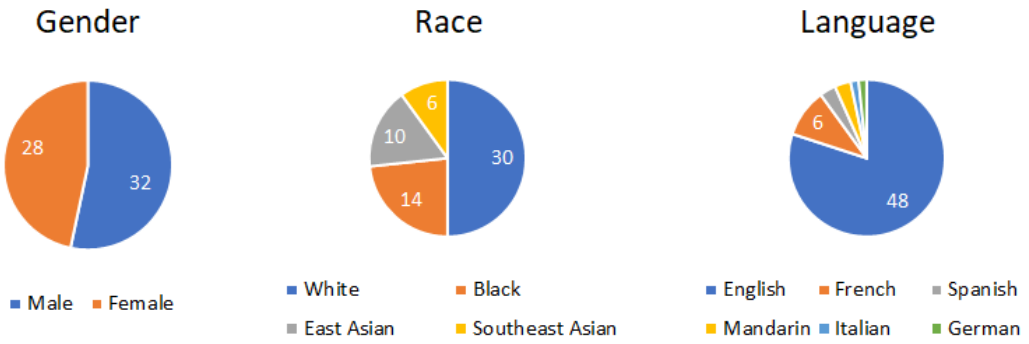


Figure 5: Diversity of our dataset across gender, race and language. All values are respective of the primary speaker of the video.

5 Model Architecture

The final encoder model is a 5-layer convolutional neural network (CNN) which is mirrored by the decoder. **Figure 6** shows the autoencoder architecture and **Table 1** shows the convolution parameters for each layer. Each convolution layer is followed by a ReLU activation except for the final layer.

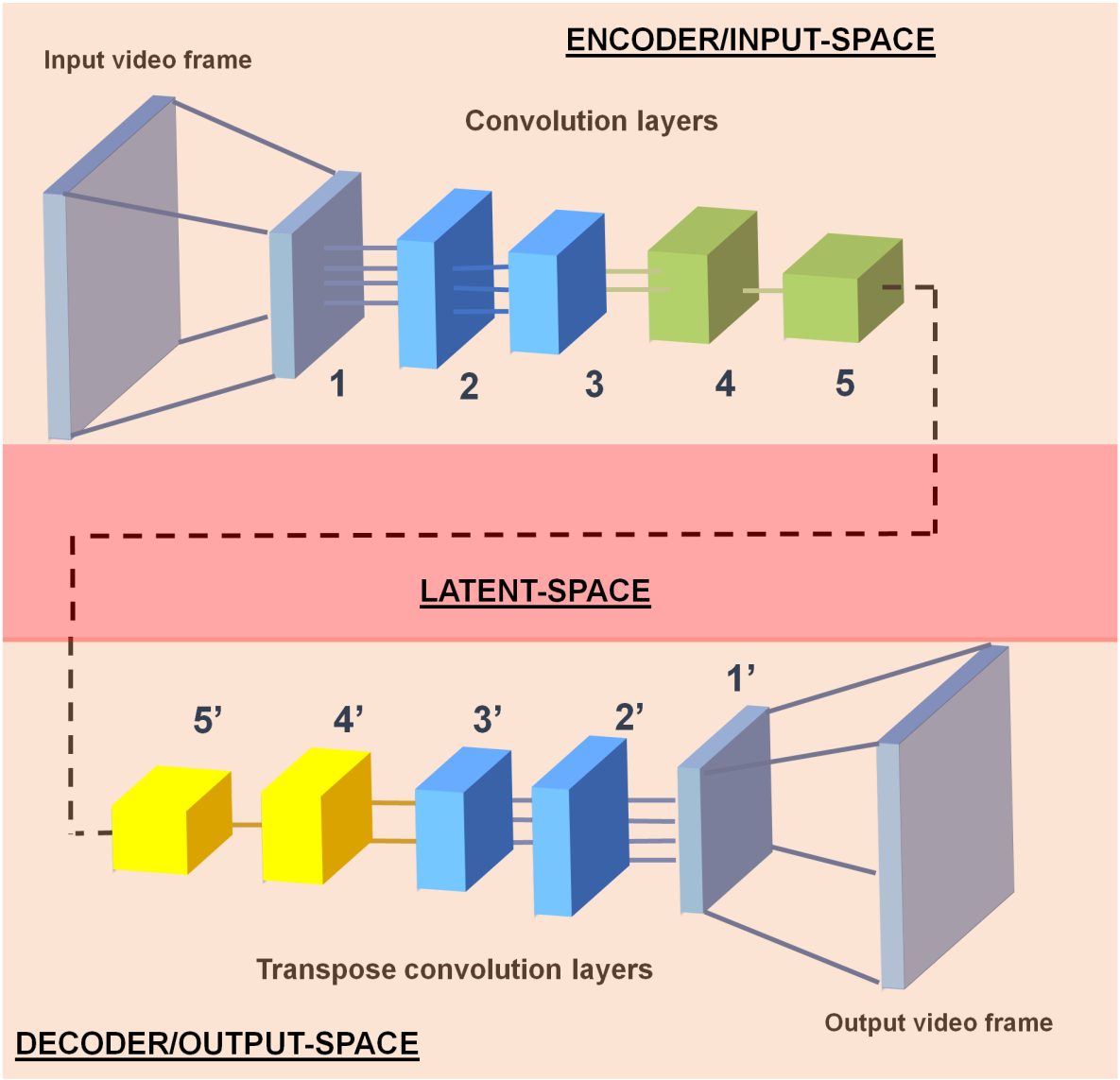


Figure 6: The final OVAL autoencoder architecture, consisting of a 5-layer CNN encoder and a symmetric 5-layer decoder.

Layer	Input Channels	Output Channels	Kernel Size	Stride
1	3 (RGB)	6	3	1
2	6	8	3	2
3	8	9	4	1
4	9	10	3	3
5	10	10	3	2

Table 1: Convolution parameters for each layer in the encoder CNN. The decoder CNN mirrors the encoder with transpose convolutions.

5.1 Baseline Model

We created two viable baseline models to compare results from our autoencoder.

5.1.1 k-MSB Compressor

This model compresses the input video frames by only keeping the k most significant bits for each color pixel. We tested this baseline with 2x compression (cutting 4 bits) and 4x compression (cutting 6 bits), with example frames shown in **Figure 7**.

Figure 7: k-MSB compression with compression ratios of 2x (left) and 4x (right).

5.1.2 Image Interpolation Compressor

Using standard image interpolation contained in the `cv2` library, this encoder resizes the frames by a set compression ratio to be scaled back by the decoder. **Figure 8** shows the baseline performance with compression ratios of 5x and 20x.

6 Quantitative Results

6.1 Offline Training

The model was pre-trained for 25 epochs over 18 hours on a CUDA-compatible device. The loss function was binary cross entropy (BCE) between the original image frames and

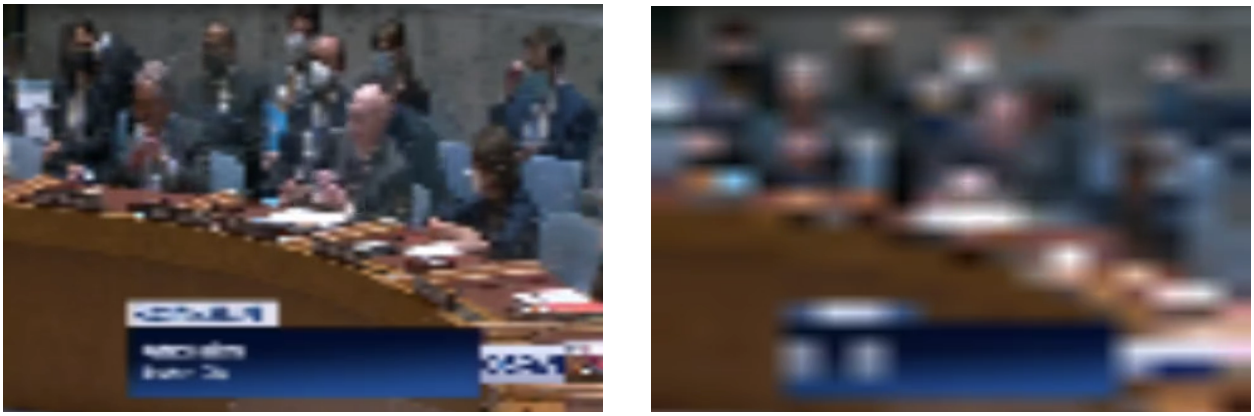


Figure 8: Image interpolation compression with compression ratios of 5x (left) and 20x (right).

the decoded frames. BCE loss is a well-documented loss function for autoencoders with normalized images [4]. Additionally, BCE loss showed superior learning behaviour to both MSE loss and MAE loss. **Figure 9** shows the model learning curves at a 23.3x compression ratio.

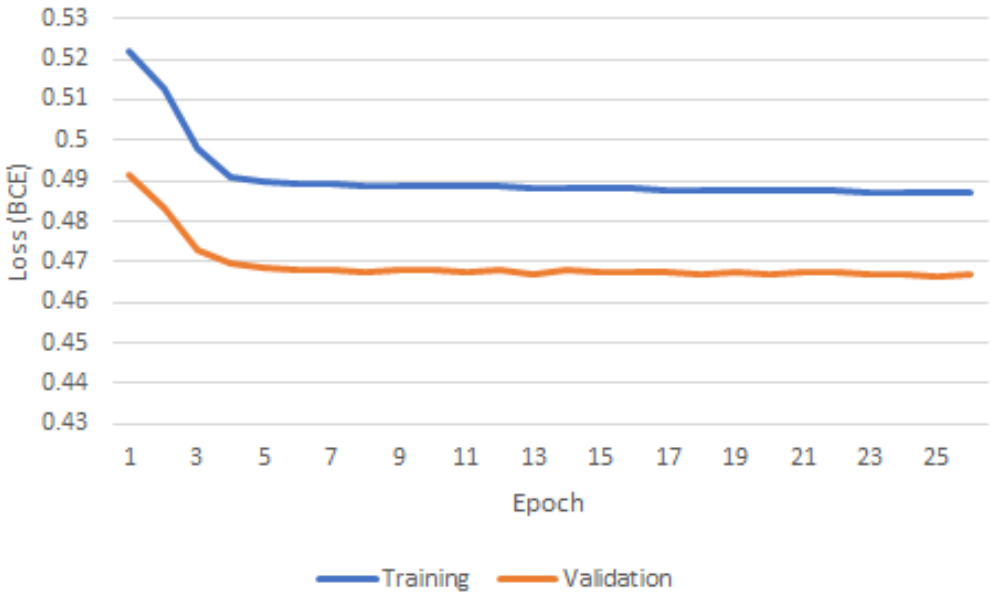


Figure 9: Training and validation loss over 25 epochs of autoencoder training.

We verified strong results by observing a decrease in both MSE and MAE loss (**Figures 10** and **11**) even though BCE was being used. **Table 2** contains the hyperparameters we tuned during training that were used on the final model.

Hyperparameter	Value
Learning Rate	1e-3
Frame Batch Size	16
Number of Epochs	25
Loss Function	BCE
Optimizer	Adam

Table 2: Training parameters used for the final 18 hour training run.

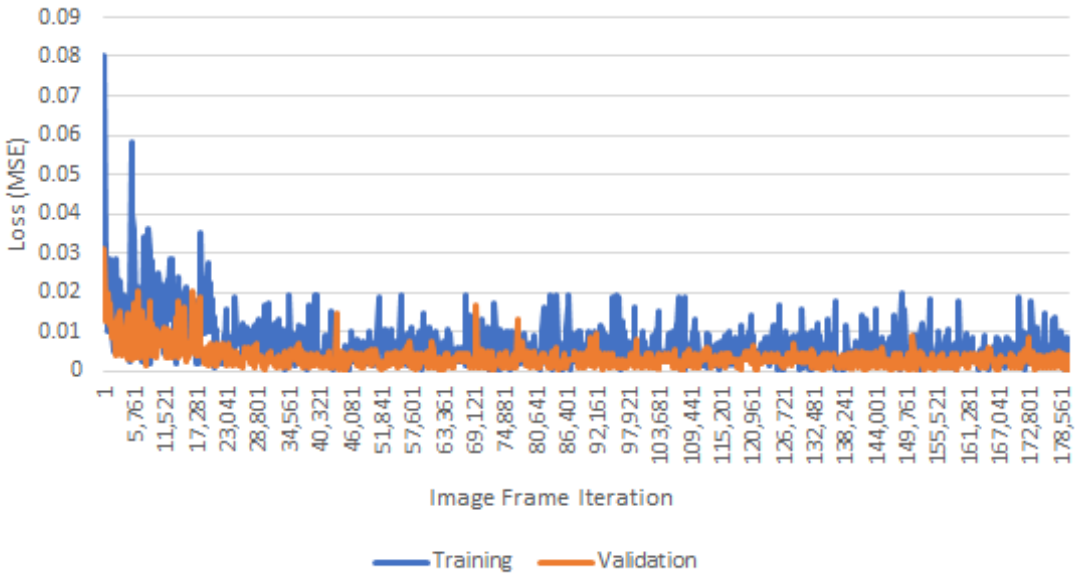


Figure 10: MSE training and validation loss over 180,000 image frames.

6.2 Online Learning

After pre-training the offline model, we deployed it with test videos for online learning. **Figure 12** shows the real-time error when the model was used with three sample test videos. MAE loss was used to evaluate online learning since it represents the mean absolute difference between video frames, which is an appropriate measure of video quality. We observe that the loss is strongly reduced as each video progresses compared to the purely offline model. The loss starts equal for both compressors, since at the start of each video OVAL is using the pre-trained model. However, online learning is able to specially adapt to each new video, resulting in better performance. This is a major result, as online learning for video compression has never been implemented before in the ML community. **Table 3** shows the final tuned hyperparameters used in online learning.

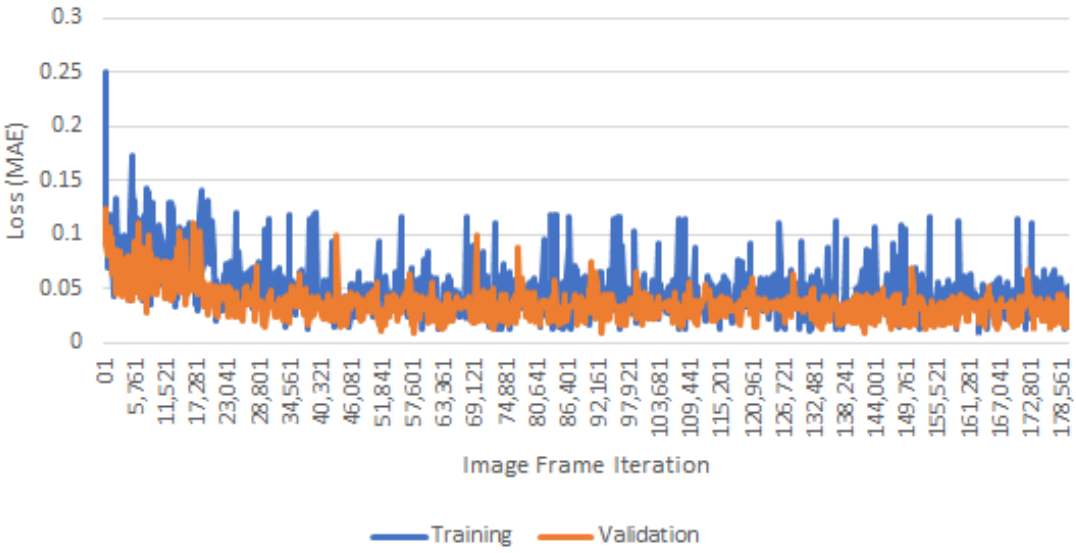


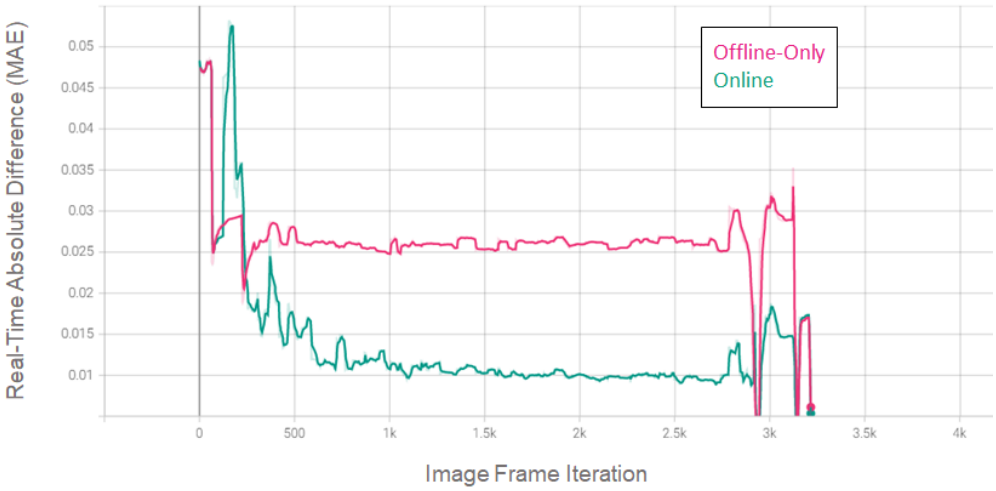
Figure 11: MAE training and validation loss over 180,000 image frames.

Hyperparameter	**Value**
Learning Rate	2.5e-3
Frame Batch Size	5
Input Buffer Size	20
Loss Function	BCE
Optimizer	Adam

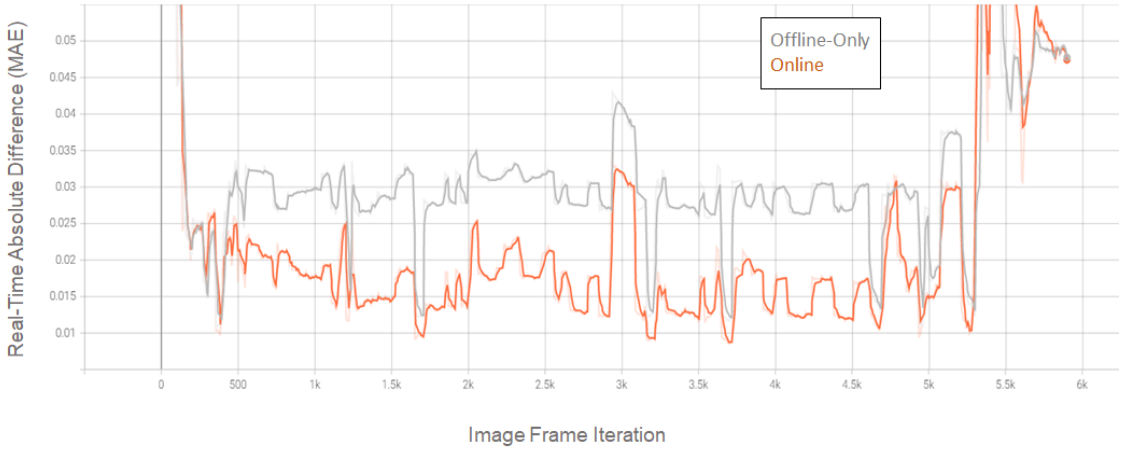
Table 3: Training parameters used during the online learning phase.

6.3 Comparison to Baseline Models

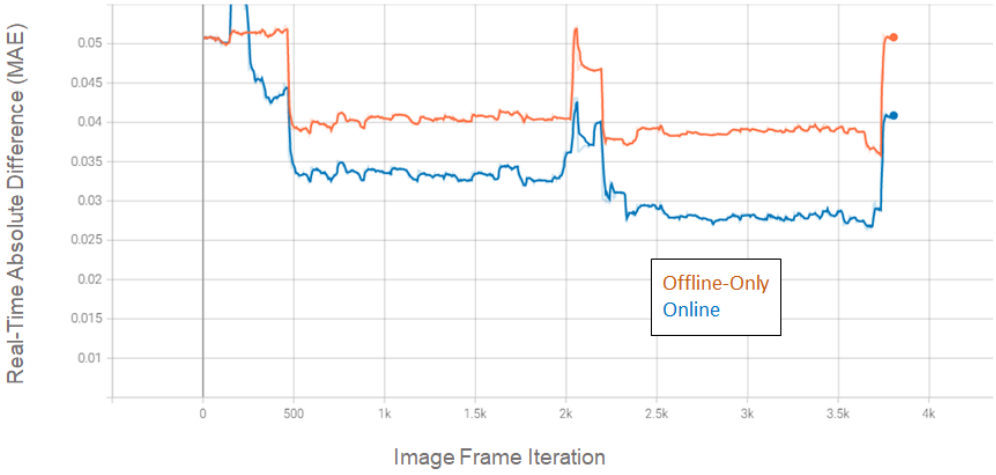
Our quantitative results show strong performance in comparison to the baseline models. The average *mean absolute error (MAE)* on the test set and compression ratios of all models are shown in **Table 4**. MAE was used as a test similarity metric as we observed it to best reflect qualitative video caliber.



(a) Online learning during test video 1.



(b) Online learning during test video 2.



(c) Online learning during test video 3.

Figure 12: Real-time absolute difference (MAE) for sample test videos. For all samples, OVAL with online learning outperforms the pre-trained model as each video progresses.

Model	Compression Ratio	Mean Absolute Error
OVAL (Offline only)	23.3x	0.0346
k-MSB Baseline	2x	0.0285
k-MSB Baseline	4x	0.1162
Interpolation Baseline	5x	0.0268
Interpolation Baseline	20x	0.0645

Table 4: Average mean absolute error on the entire test set with OVAL compared to the baseline models. OVAL has almost half the MAE as the interpolation baseline at a slightly better compression ratio, and one-third the MAE as the k-MSB baseline at a 5x better compression ratio.

7 Qualitative Results

7.1 Examples of Compressed Video

An example of a full transmitted test video is available here. OVAL treated this video as a live feed and encoded/decoded video frames in real-time between a sender and receiver.

7.2 Online Learning

The quantitative improvement in loss during online learning is reflected in real-time improvements to video quality. **Figure 13** shows in-video improvements with three test videos. As a stable video feed holds (such as an individual speaker) the quality of the decoded frames improves.

7.3 Comparison to Baseline Models

When used to compress the same live video feed, OVAL produces clearer results for the receiver. **Figure 14** shows the qualitative differences between OVAL and both baselines when applied to the same test set videos.

(a)

(b)

(c)

(d)

(e)

(f)

Figure 13: Improvements in video quality during online learning at 23x compression with our test set. Frames on the left are from early in the video, whereas frames on the right are from later in the video. Improvements are particularly noticeable with respect to color saturation and blur.

(a) OVAL with online learning at 23.3x

(b) OVAL without online learning at 23.3x

(c) k-MSB at 4x

(d) interpolation at 5x

(e) interpolation at 20x

Figure 14: Frames from the same compressed video using OVAL compared to baseline models.

7.4 Input Dependence for Online Learning

There are factors in the input video that influence the performance of online learning. Online learning fits specific features in the real-time video feed, so it is logical that stable videos (i.e. a single speaker) can be compressed better than more dynamic videos. We observe this behaviour in tests between two static/dynamic videos in **Figure 15**.

Since the original application of OVAL was for video conferencing, which normally has unchanging video feeds, its limitation for dynamic video does not affect its goal.

8 Evaluation on New Data

8.1 Test Video Dataset

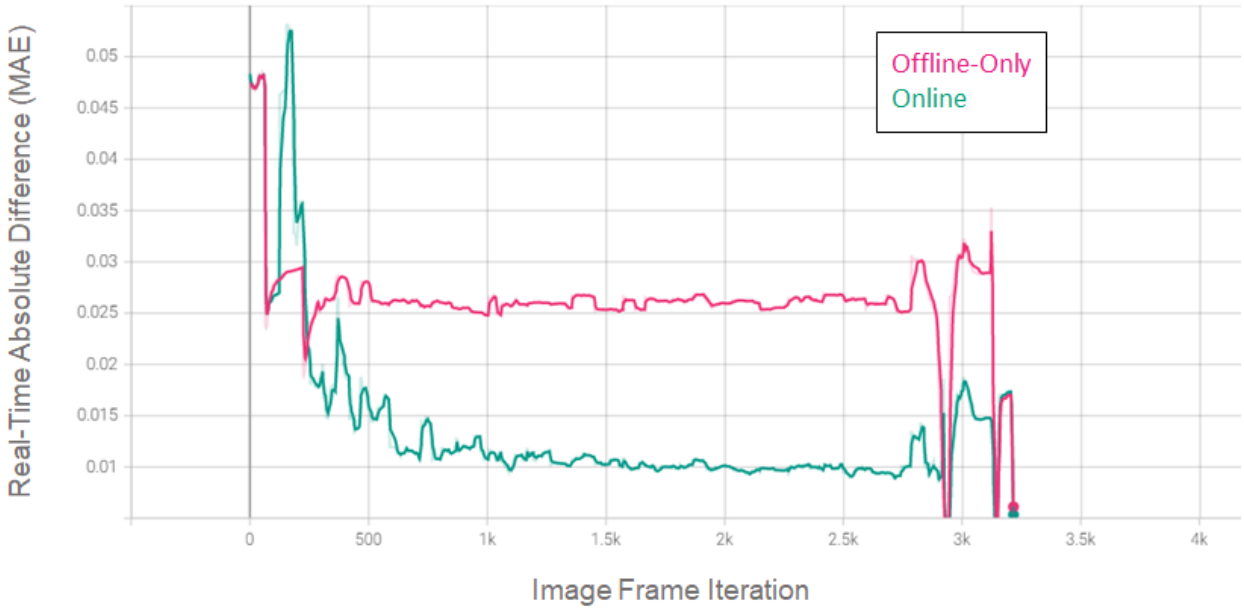
Upon creating our dataset (Section 4) we allocated 10 videos for a test set which was only accessed after hyperparameter tuning. These samples were assigned randomly from the collected YouTube videos, all of which focused on a small number of individual speakers. These test samples were used to:

1. Verify the pre-trained model's generalization
2. Evaluate online learning performance
3. Test real-time encoding ability in a live video setting

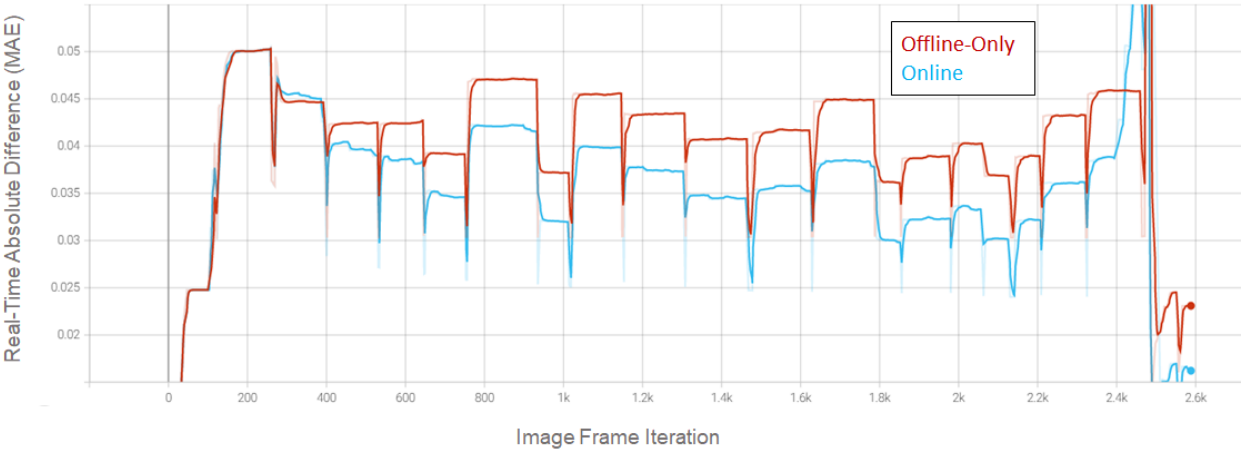
Results shown in Sections 6 and 7 verify strong generalization and robust online performance.

8.2 Use in a Video Conference

The final use of OVAL is for video conferencing. Therefore, our final tests on the model were on live video. We used OVAL to encode/decode each speaker's video for our final presentation. Frames were encoded, transmitted and decoded by a local "receiver" in real-time. The resulting output video is shown in the bottom-right corner here. **Figure 16** shows decoded frames from each speaker.



(a) Online learning for a video with a single seated speaker.

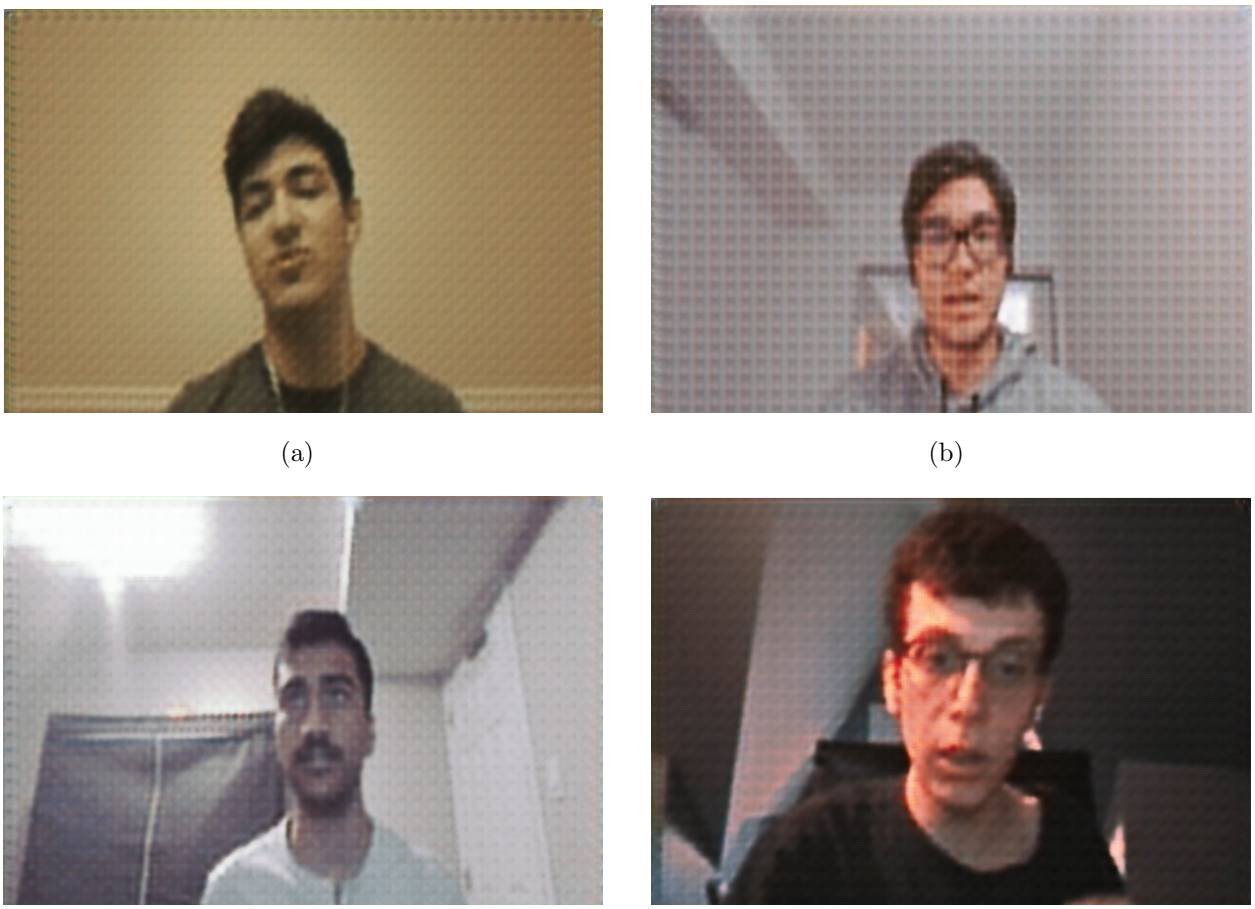


(b) Online learning for a video with multiple speakers and camera angles.

Figure 15: Online learning performance for a mainly static video compared to a rapidly changing video. For the static video, online learning is able to achieve a 75% decrease in loss, whereas for the dynamic video online learning only achieves a 20% decrease in loss.

Figure 17 shows the improvement in video quality with online learning for a single speaker and **Figure 18** shows proper behaviour from the dynamic update policy.

These results confirm the practical viability of OVAL. Firstly, the receiver frame rate matches the sender frame rate, resulting in smooth real-time video. To ensure online training stays



(a) (b)

(c) (d)

Figure 16: OVAL decoded frames for all group members from our final presentation.

up-to-date with encoding, OVAL downsamples training frames (Section 4.2) exhibited in **Figure 19** from our live test.

Secondly, the live test verifies OVAL's compatibility with a camera input. Whereas YouTube videos were stored as `.mp4` files and converted to PyTorch tensors (**Figure 3**), the successful live test shows that camera input can be loaded to tensors as well without slowing down OVAL's functionality.

Figure 17: Real-time absolute error between decoded frames and the actual frames at 23.3x compression during the webcam test.

9 Discussion of Results

9.1 Offline Learning (Pre-Training)

We continue the analysis from Section 6.1 with key insights from the pre-training process.

Firstly, although we trained the model using a BCE loss function we observe (**Figures 10-11**) that both MSE and MAE loss also decrease throughout training. These results provide evidence that the model learned how to encode features of the input video rather than simply gaming the loss function. Additionally, the results verify the appropriateness of the BCE loss function for training. While BCE training loss only decreased by 7% from epoch 1 to 25 (**Figure 9**), the MSE and MAE loss decreased by 75% and 50% respectively. Thus, the BCE loss properly drove the model to find robust encodings of video data.

Secondly, we notice in **Figure 9** that validation loss is consistently lower than training loss. While normally unusual, this observation can be explained by natural differences in the training and validation datasets. As videos were randomly assigned to each set with only 10 videos in validation, the validation videos were likely more static and therefore easier to encode (Section 7.4). Thus, the actual loss value for each set is unimportant compared to their relative improvement in training.

Lastly, we note that at epoch 25 the training and validation loss were still slightly decreasing.

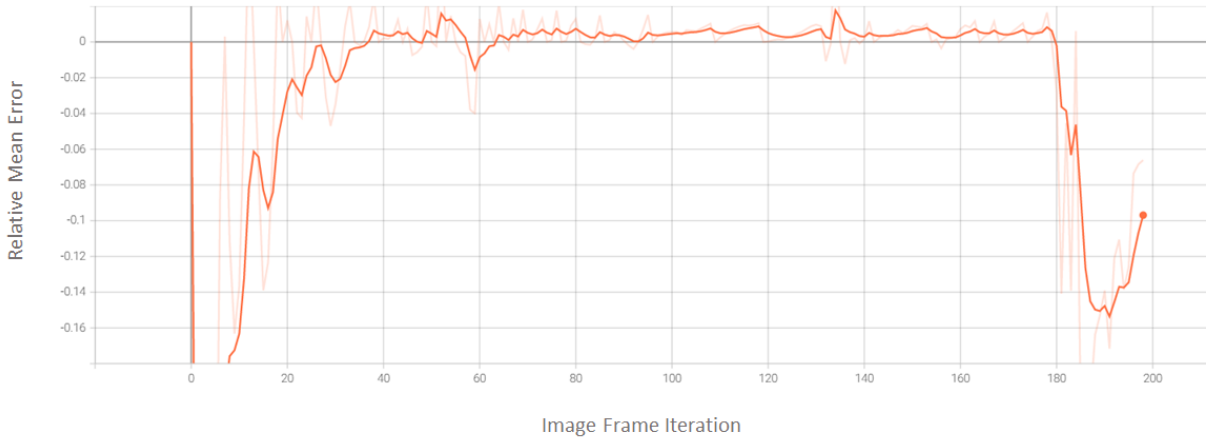


Figure 18: Relative pixelwise difference between the sender's training encoder and live encoder during the webcam test. A low error is preserved due to dynamic updates between the sender and receiver.

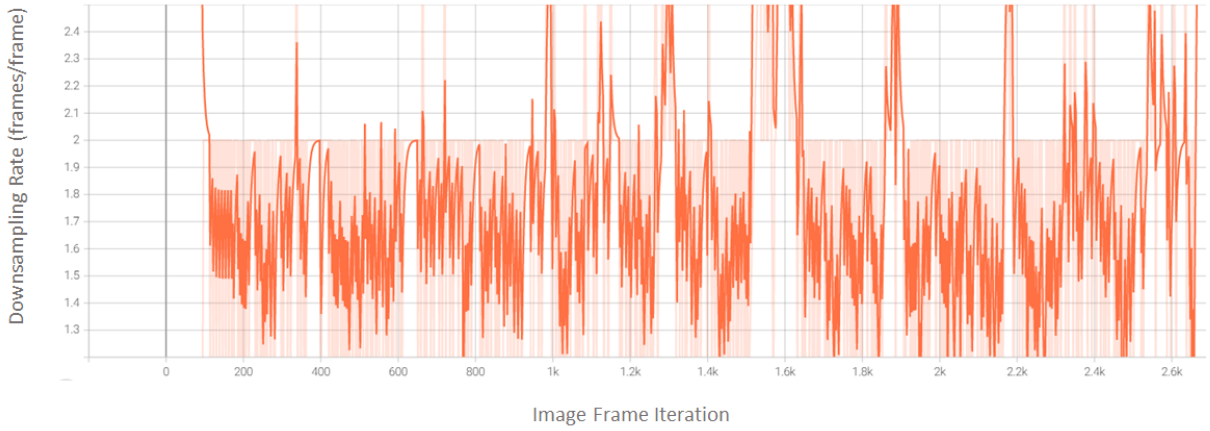


Figure 19: Frame downsampling from the FIFO buffer (**Figure 2**) during the live test. The downsampling rate is the ratio of total video frames to frames used for training. For example, a downsampling rate of 2 means that only half the input frames are used for training in order to keep up with the input frame rate.

Training for 25 epochs took over 18 hours on a GPU. Therefore, we did not train for more epochs to attain a marginal increase in performance.

9.2 Online Learning

The primary goal of OVAL was to use online learning for effective video compression. Results show that online learning was highly effective in fitting individual video streams (Section 6.2).

Largely, OVAL performs as expected when used in an online context. Real-time error is reduced throughout the video and changes are adapted to quickly. We observe that even when a rapidly changing video is encoded, online learning continuously optimizes the encodings. After loss spikes, online loss slowly decreases whereas the pre-trained model remains flat. Examples of this constant learning behaviour can be seen between frames 400-1600 in **Figure 15 (b)**.

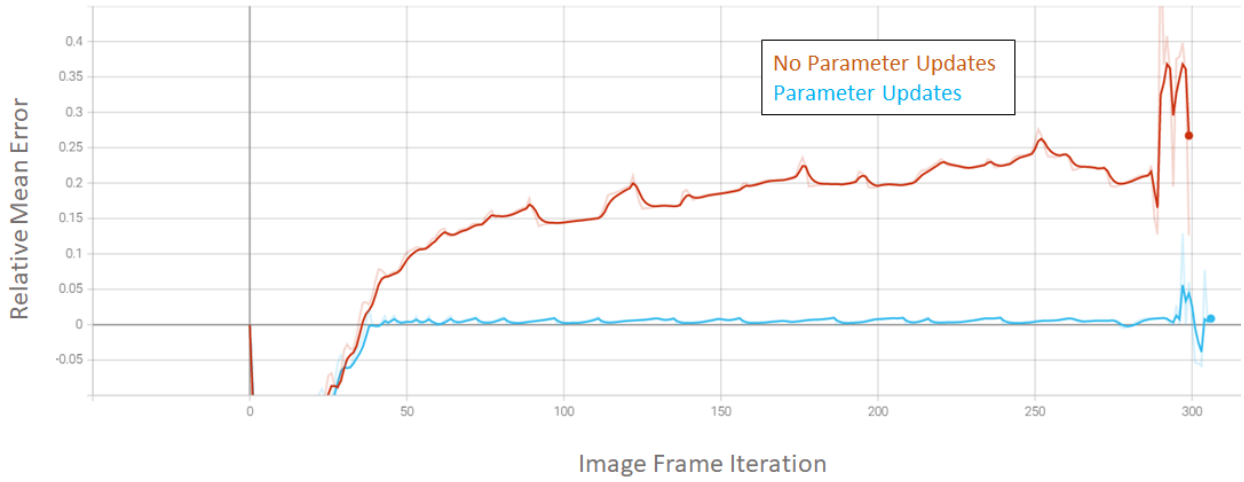
An interesting observation is the speed at which online learning fits new video feeds. In **Figure 15 (a)** the online loss stabilizes after 1,000 frames are transmitted. In comparison, when pre-training the offline model, it took around 40,000 frames before loss stabilized (**Figure 11**). This is likely indicative of strong pre-training which learned a wide range of common features, such as faces and backgrounds. Thus, when a new video feed is fit online, the relevant features are already known and just need to be identified for encoding.

9.3 Dynamic Update Policy

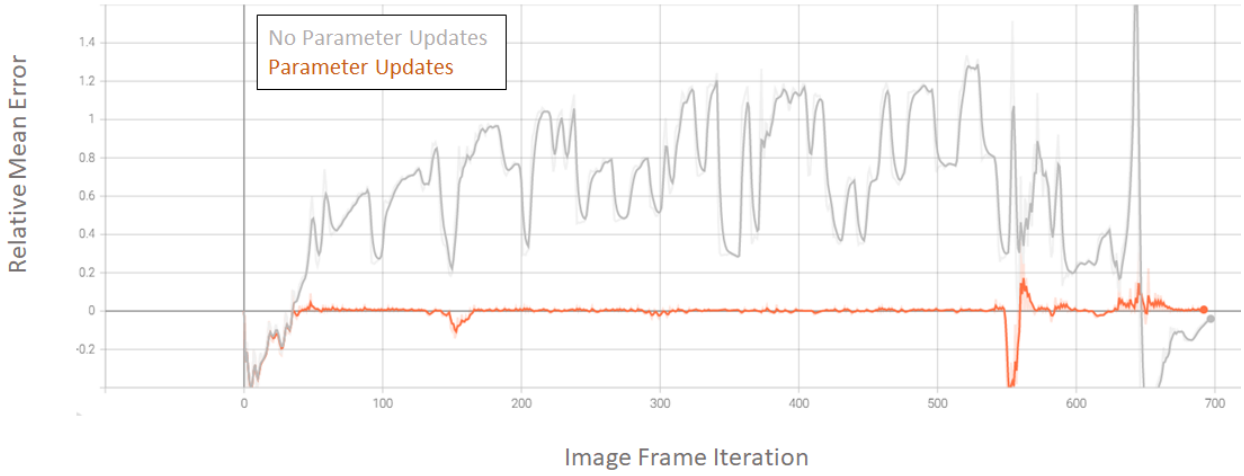
The sender/receiver architecture of OVAL uses a dynamic update policy where new live model parameters are used if the error between the training/live autoencoders reaches a threshold value. **Figure 20** demonstrates that this update policy can maintain an arbitrarily small relative error between the sender's autoencoders. Clearly, the relative error increases without periodic updates, since the sender's training model is improving but is not being used. With the update policy, at specified relative errors the live model is updated to incorporate the improvement from online training. In practice, there is a trade-off between relative error and performance since updating autoencoder parameters incurs a computational cost on the sender, receiver and connecting network.

9.4 LSTM Networks and Transfer Learning

Since LSTM networks were commonly used in past papers on video compression, we attempted to incorporate an LSTM layer into the OVAL architecture. However, the LSTM



(a)



(b)

Figure 20: OVAL's dynamic update policy applied to two test videos with online learning. The relative mean error (pixelwise difference) between the sender's training model (**Figure 2**) and the sender's live model is plotted for both videos *with* and *without* periodic updates.

layer slowed training down too significantly to be used in practice. Since the concurrent online training needed to stay up-to-date with real-time video, the LSTM was not feasible. We also tried a ConvLSTM, a special RNN that uses convolutions for both the input-state and state-state transitions. Similarly, this architecture was too slow to be used in OVAL.

We also considered using transfer learning with a simple, pre-trained CNN model such as AlexNet. The pre-trained model would act as part of the encoder. This extension was not explored due to time constraints.

10 Ethical Considerations

An ethical issue in the implementation of OVAL is data privacy. During a video conference, a malicious agent could intercept the OVAL encodings as well as their associated decoder parameters. Then, they could access private video feeds. When using OVAL in practice, the decoder parameters *must* be encrypted during transmission. Additionally, the compressed data should be encrypted.

Since our model's primary use is video conferencing, it must not be biased towards specific users based on their race or gender. For example, a biased model may create stronger encodings for male facial features which result in better video quality. To ensure our model was unbiased, it was trained on a diverse range of speakers (**Figure 5**). In **Figure 13**, we see that online learning improves skin tone accuracy across different races and improves facial feature definition across different genders. These results verify our fairness goal.

11 GitHub Repository

OVAL GitHub repository.

References

[1] A. Golinski, R. Pourreza, Y. Yang, S. Guillaume and S. Taco. "Feedback Recurrent Autoencoder for Video Compression," Qualcomm AI Research, 2020.

[2] Y. Zhang, X. Liang, D. Zhang, M. Tan, and E. P. Xing. "Unsupervised object-level video summarization with online motion auto-encoder," Pattern Recognition Letters, vol. 130, pp. 376–385, 2020.

[3] P. Vincent , H. Larochelle , Y. Bengio , P.A. Manzagol. "Extracting and composing robust features with denoising autoencoders," ICML '08: Proceedings of the 25th international conference on Machine learning, pp. 1096–1103, 2008.

[4] A. Creswell, K. Arulkumaran, and A.A. Bharath. "On denoising autoencoders trained to minimise binary cross-entropy,", submitted to Pattern Recognition Letters, 2017.