Deception classification from video input

Lillian Ma
Stanford University
353 Jane Stanford Way, Stanford, CA 94305
ylma@stanford.edu

Stephanie Vezich Tamayo
Stanford University
353 Jane Stanford Way, Stanford, CA 94305
isvezich@stanford.edu

Abstract

Reliable deception detection is important across a number of real-life contexts from criminal law to political negotiations. However, human performance is notoriously poor, and past machine learning work in the domain has largely leveraged traditional techniques that either rely on laborious handcrafted features or do not leverage the spatial structure of video inputs. In this research, we explore the use of image and video Vision Transformers for supervised deception classification. We also explore the benefit of video compression techniques to improve memory efficiency. This resulted in four models: 1) ViT without video compression, 2) ViT with video compression, 3) MViT without video compression, 4) MViT with video compression. All four models demonstrate superior accuracy to human baselines and we see modest improvement from both video compression and video models relative to their uncompressed and image model counterparts. These results suggest that video models may encode better spatiotemporal structure and that compression may have a regularizing effect.

1. Introduction

Deception detection is an important task across a variety of domains, from forensics to everyday interactions. However, humans are notoriously poor at assessing truthfulness in interactions, with past studies showing accuracy rates ranging from 60% to less than 50% [4]. Unlike tasks like visual perception that have a very high human accuracy baseline, social perception tasks like this are difficult in part because it is impossible for humans to focus entirely on subtle cues (both verbal and nonverbal) that may indicate whether deception is occurring [5]. Rather, we are currently multitasking in these interactions–trying to decipher the content of the speech, intuit the speaker's intention, generate a response of our own, etc.–which leaves limited cognitive capacity to attend to deception cues even if they exist.

Computer vision models, however, have no such competing sources of cognitive load. Hence, it is possible that they may be able to exceed the low baseline of human performance. In line with this hypothesis, there is a robust body of existing literature on automated deception detection, categorized by the classes of features they use and their applications in Figure 1 [15].

However, to our knowledge, the techniques used largely vary from statistical analysis to some traditional ML (e.g., decision trees, SVM) and deep learning (e.g., CNN, LSTM) classifiers, and often rely on hand-crafted features [15]. Given recent advances in data-driven learned feature representations and the shift away from manual feature engineering, we would like to explore whether we can classify deception based on video inputs alone.

Specifically, the data we use is a labeled set of video clips. The input to each model is sampled video frames of faces in which the subject is telling a truth or a lie, and the output is a binary classification score corresponding to the probability that the clip contains a lie. We use CNNs as a feature tokenizer. We then feed the output as a sequence of tokens to an image model backbone (ViT-B16) [8] in the baseline model and a video model (MViTv2) [16] in the candidate model, both with classification heads. We leverage adapter layers between blocks in each of the pretrained models for parameter-efficient fine tuning. Finally, we explore video compression as a preprocessing step for better model efficiency in memory constrained environments.

2. Related Work

There has been a wealth of research on deception detection tasks in the machine learning literature over the past decade. Here we discuss high-level trends in: a) the type of data used, b) preprocessing and feature extraction, c) model algorithms and architectures. In each section, we discuss how these trends informed our study design.

2.1. Data

Publicly available video datasets on deception detection are generally categorized by whether they involve high-stakes (e.g., criminal records, political negotiations) or low-

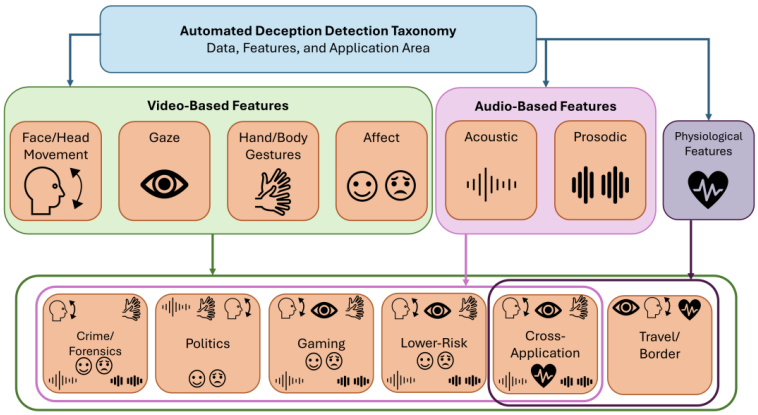

Figure 1. Taxonomy of deception detection features, from King and Neal [15]

stakes (e.g., interpersonal conversations, games) scenarios [15]. High-stakes datasets such as the Real-Life Trial Dataset [18] or the Mock Theft Experiment [1] tend to utilize either professional actors in scripted recordings or opportunistic data from real-life sources such as criminal proceedings. Low-stakes datasets such as Bag of Lies [12] and Werewolf [13] skew towards convenience sampling methods such as recording university student study participants in a lab setting.

In the current study, we utilized the DOLOS dataset, which is composed of YouTube clips from a game show that involves telling truths and lies [11]. This dataset had several desirable properties for training a model that can generalize well: 1) it is intentionally balanced on class labels and gender, 2) it is from a more naturalistic setting (e.g., includes multiple angles and scenes, subjects vary in age) than lab-recorded data, and 3) the deception content covers a broad range of topics. However, it is limited to the game show context and does not include high-stakes scenarios; it would be useful to construct a dataset that varies in this dimension as well.

2.2. Preprocessing and feature extraction

Video data for deception detection is often split into visual and auditory modalities. For visual data, researchers have typically cropped frames to the face of the speaker using standard libraries such as OpenCV [19] or MTCNN [24]. They have also applied normalization techniques to make aspects such as lighting and image resolution more consistent across examples [20]. In addition to the video input itself, many studies hand annotate eye gaze, body language, etc. to use as additional features. Some of these annotation schemes such as MUMIN [2] have been codified for consistent use across studies.

We followed these common cropping and normalization approaches to ensure that the frame input passed to each model focused primarily on facial expressions (rather than attending to irrelevant scene input or variance in image facets like lighting). Although audio waveforms have been shown to include valuable signal for this task both in unimodal and multimodal settings, we decided to focus on visual input only for the scope of the project; however, adapting the current work to a multimodal setting would be a natural extension. In addition, though the DOLOS dataset included annotations of body language, eye gaze, etc., we also excluded this data from our analysis to keep the modeling focused on visual input. It would be interesting to explore the incremental value of this metadata.

One area of video preprocessing that has been a notable gap in the deception detection literature is video compression. Given the subtlety of the task and the temporal structure over which facial cues may operate to indicate deception (e.g., change in eye gaze), longer sequences of frames may be valuable for this task. However, there is a clear tension between higher frame sampling rates and memory constraints. To better balance this tradeoff, we explored compression as a preprocessing step, which has shown promising results in other video understanding tasks [23].

2.3. Model algorithms and architectures

Past literature largely treats deception detection as a fully-supervised binary classification problem, with each example labeled as deceptive or truthful [4]. More recently, there has been work to explore unsupervised methods such as Deep Belief Networks [17] and semi-supervised methods [10] in order to address the data bottleneck of hand labeling, but these approaches are still less common.

Historically, traditional ML algorithms for classification have been used, including decision trees [14], SVM [12], and logistic regression [7]. More recently, deception researchers have explored deep learning models well suited for sequences and images such as LSTMs [9] and CNNs

[3]. With the surge in popularity of Transformer-based architectures in the last 5-10 years, some deception researchers have explored these methods as well, though they have largely been scoped to ViTs with less exploration of video native models [11].

Given the strong empirical performance of Transformer-based architectures across a variety of image and video understanding tasks, along with the availability of reliable pretrained models, we chose to use Vision Transformers as our backbone models. However, we added several enhancements, described in more detail in the Methods section. Namely, instead of using raw image patches as input, we first passed videos through convolutional layers, treating the CNN as a learnable tokenizer. This was to allow the training procedure to learn rich spatial structure-aware feature maps despite the small sample size. We also inserted learnable adapter layers between Transformer blocks to enable parameter-efficient fine tuning for this task. Lastly, we explored a compression algorithm using VAE to preprocess the input data. This not only dramatically reduced the memory and computation required to train, but also slightly improved model performance.

3. Methods

3.1. Image model

For the baseline model, we leveraged source code from Guo et al. that treats the data as a sequence of images, generates an embedding for each image, then passes the embeddings as a sequence of tokens to a pretrained image model [11]. Specifically, it first extracts features from the frame images by using a series of Conv2d blocks. Each block consists of a Conv2D layer, BatchNorm2d and ReLU. The architecture is structured with three main Conv2D blocks, each followed by two additional Conv2D blocks with residual connections. Average pooling is applied at the end for global feature extraction, resulting in a feature map of size 64 x 256.

This sequence is then passed to the pretrained model ViT-B16 (Vision Transformer) [8]. Vision Transformers adapt the Transformer block architecture to image tasks by dividing images into a sequence of patches and treating each patch as a token input. This specific version of Vision Transformer is a mid-size base model (B) that operates on 16 x 16 pixel patches (16). Though ViT-B16 typically expects raw image patches as input, in this case the CNN works as a learnable tokenizer. In other words, we pass a sequence of 64 tokens, each of which is a 256-dimension embedding. Using a CNN as a tokenizer could be beneficial in this case where the dataset is relatively small and the domain of face classification is fairly specialized because it introduces inductive bias about image locality.

A positional embedding is added and then passed to ViT-B16 with a classification head and 4 adapters. Each adapter is inserted between the attention and feed-forward layer of the transformer encoder, and consists of dropout (p=0.1), linear, ReLU, linear, and layernorm layers. This enables parameter-efficient fine tuning because only the adapter layers need to be trained while the rest of the transformer encoder (mostly) can remain frozen.

While the baseline model showed promising performance in Guo et al. [11], it has two key limitations. First, while the CNN features encode some spatial structure internally, that resolution is weakened once we pass a sequence of 64 embeddings to the ViT. Second, ViT-B16 was trained on images and is not particularly tailored for video tasks.

3.2. Video Model

To address these limitations, we kept a similar overall procedure but substituted MViTv2 (Multiscale Vision Transformer) for the backbone model [16]. MViTv2 is a variant of Vision Transformers that leverages a unified architecture for image and video data. It has been shown to have strong performance on a variety of visual recognition tasks from image classification to video understanding. This flexibility is achieved primarily through multiscale attention, which processes inputs at multiple scales rather than using fixed-length tokens as in ViT and thus enables the model to capture representations of both coarse- and fine-grained details.

We adopted an analogous feature extraction pipeline in our video model. Video frames were first passed through 3 sets of 3 3DConv blocks, followed by 3D average pooling. Each set of the 3DConv block had a residual connection for the last two blocks, similar to its 2D counterpart. This configuration explicitly convolved across the temporal dimension, enhancing the model's capacity to learn spatiotemporal features. The resulting feature map had shape (N, 256, 4, 4, 4), where N is the batch size, 256 is the channel dimension, and the final three dimensions correspond to time, height, and width. We then flattened the spatial-temporal dimensions and projected them to a 96-dimensional space using a linear layer followed by ReLU activation, matching the expected input size of MViT. Next, MViT-style positional encodings were added, and the sequence was passed through four pre-trained MultiscaleBlocks. Each block was followed by the same adapter layer used in the ViT-based model, and the output was fed into the same classification head (Figure 2).

Several additional modifications distinguish this model from the original. Although we initialized the model with default pre-trained weights, we chose to unfreeze the MultiscaleBlocks during training. This decision was informed by early signs of overfitting when the blocks were frozen, likely due to the limited adaptability of the fixed adapters. Moreover, by using MViT's built-in positional encoding

scheme instead of fixed embeddings, we were able to better preserve the spatiotemporal structure of the video features.

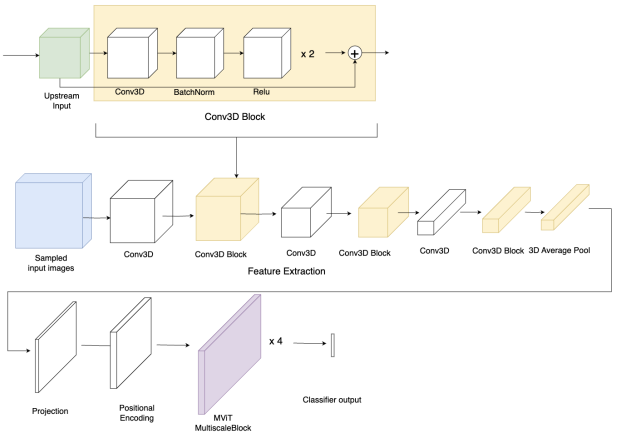

Figure 2. Video model architecture.

3.3. Video Compression

In addition to using the original 64-length sampled image sequences as CNN inputs, we also experimented with compressing the video input to enable larger batch sizes. Using the original cropped jpegs as inputs, the batch size was set to 4 due to memory limits. This de-stabilized our training and made tuning more difficult. Additionally, training speed was slow (over 2 hours for 120 epochs). This prompted us to explore techniques to reduce this bottleneck.

Namely, we followed the neural compression approach from Wiles et al. [23], which demonstrated strong performance on downstream classification tasks. The neural compressor consists of a standard VQ-VAE (Vector Quantized Variational Autoencoder) [22], which encodes input data into a discrete latent space using a learned codebook of embedding vectors. This discrete representation can improve interpretability and is often more efficient or better suited for downstream tasks like compression or generative modeling compared to traditional VAEs. The decoder can then take this quantized representation and reconstruct the original input.

For purposes of compression, we only used the encoder portion of the compressor. The encoder maps images to a spatial tensor, where each vector in the tensor is then compared with a sequence of embedding using nearest neighbors. We modified the code base provided by Wiles et al., such that the existing encoder-decoder only produces quantized embeddings for each frame. We then stored each clip as a collection of these chosen embeddings for each image frame on disk as a .npy file. At training time for video classification, these .npy files were loaded from disk and fed directly as tensor inputs to the downstream neural net (see Figure 4). They were uniformly sampled down to 64 embedding vectors per clip. An example of an original frame from our sample dataset, as well as its de-compressed frame using the VQ-VAE decoder, is shown in Figure 3.

Figure 3. On the left is a frame of the original cropped video clip used in uncompressed models. On the right is the decompressed image produced by decoding the VQ-VAE embeddings tensor.

We chose to use the pre-trained encoder with a compression rate of 786, which produces generated tenors of shape (N, 64, 512 14, 14), where N is the batch size, 64 is the number of frames, 512 is the embedding size of the VQ-VAE encoder, 14 x 14 is the new compressed height and width dimensions. Compared to the size of the previous jpeg inputs (N, 64, 3, 244, 244), the compressed image representations are 56.2% of the original size. Due to its modified shape, we also modified our downstream baseline and video's convolution blocks. The new models each only have a single group of 3 Conv2D/Conv3D blocks where the last two blocks have a skip connection. The convolution layers now have a kernel size of 1 and output channel size of 256, while preserving the other dimensions. We also apply the same average pool at the end prior to passing the input to the ViT/MViT layers/blocks (Figure 4).

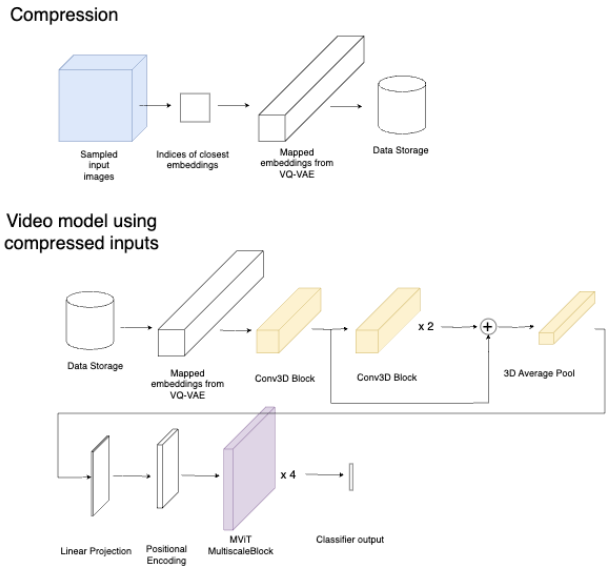

Figure 4. Compression model architecture.

The CNN architectural reduction was based on the hypothesis that the VQ-VAE encoder had already captured

rich high-level features, eliminating the need for additional deep convolutional layers. By simplifying the network, we significantly lowered memory and computational costs, which in turn enabled more flexible training and using additional model layers. Crucially, this compression strategy did not result in noticeable performance drop compared to baseline models trained on the original, uncompressed inputs (see Results section).

4. Dataset and Features

We used the DOLOS Audio-Visual Deception Detection dataset from Guo et al. published at ICCV 2023 [11]. The authors sourced clips from a British reality comedy game show available for download on YouTube that involves telling lies and truths. They edited the clips to satisfy two requirements: 1) the clip includes only relevant truth/lie content in a clear voice free of background noise, 2) the speaker's face is visible with no occlusion.

Over 84 episodes, they produced 1427 clips from 213 individuals (141 male, 72 female). The min clip duration is 2 seconds and the max is 19 seconds. The class labels are fairly balanced with 46% truths and 54% lies. This dataset was randomly split into 965 train examples (452 truth, 513 lie) 463 validation examples (214 truth, 249 lie).

Each clip was hand annotated with features such as the gender of the speaker, hand gestures, eye movement, verbal fluency, etc. However, we did not include these features in the current analysis. Each clip also included audio wav files and jpeg image files as the original paper focused on multimodal understanding–they built audio only, visual only, and fusion models–however, we focused on visual input only so discarded the audio data.

The data repository included image preprocessing scripts which we used directly. This process uniformly samples 64 images from each video clip. It then crops the images to faces with the MTCNN face detector [24]. Finally, they are resized to 224 x 224 pixels and normalized using mean ([0.485, 0.456, 0.406]) and standard deviation ([0.229, 0.224, 0.225]) values from ImageNet [6]. No data augmentation was applied.

5. Experiments/Results/Discussion

We trained all models for 120 epochs using binary cross entropy loss (Equation 1) and the Adam optimizer, with the exception of the baseline model at 100 epochs for early stopping.

$$L = -\frac{1}{N}\sum_{i=1}^{N}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right] \quad (1)$$

We experimented with learning rates between 1e-3 and 1e-7, and chose 3e-7 for the findings we reported as it yielded the best results. We also used 0.8 for exponential decay rate for momentum, and 0.99 for velocity. This helped stabilize our training and we saw a more steady decrease of our loss over time. We explored using L2 regularization to reduce overfitting, but had little success. Instead, we found that it was more effective to unfreeze the pre-trained layers of the respective downstream transformer models. However, due to memory limitations, this was not feasible for the baseline model and was only possible using compressed input. This was also possible in the video transformer model without uncompressed data due to 3DConv blocks using less memory for the same input size compared to its 2D counterparts. We kept a relatively low batch size of 4 for baseline and 8 for our video model. For the uncompressed input models, we used a batch size of 16. In an effort to keep the training procedure as fast as possible, we did not do cross validation but simply maintained a 30% validation set for evaluation. The results our models are summarized in Table 1.

Given relatively balanced classes, we focused on accuracy (Equation 2) and F1-score (harmonic mean of precision and recall; Equation 3) as our primary evaluation metrics.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} = \frac{\sum_{i=1}^{N} \mathbb{1}(\hat{y}_i = y_i)}{N} \quad (2)$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (3)$$

Model type	Accuracy	F1 Score	Loss	Train time
Baseline (ViT)	61.26%	0.6324	0.796	2.156 hr
Baseline (ViT) with compressed inputs	63.64%	0.6945	0.6554	39.65 min
Video (MViT)	66.67%	0.6957	0.6215	2.076 hr
Video (MViT) with compressed inputs	69.05%	0.7023	0.6213	40.6 min

Table 1. Summary results of model performances and training times

5.0.1 Baseline image model - uncompressed inputs

This model simply replicates the results found in Guo et al. [11] with slightly different hyperparameter values (100 epochs vs. 20 to enable better learning, LR of 3e-7 vs. 3e-4, batch size of 4 vs. 16 to address OOM). The results were quite similar; we observe 80.71% train accuracy and 61.26% validation accuracy (vs. 61.44% validation accuracy in the original paper), and a train F1-score of 0.82 and a validation F1-score of 0.63 (vs. 0.69 validation F-1 in the

original paper). We see clear evidence of overfitting, as validation loss starts to steadily increase and we see a moderate train-validation gap (Figure 5).

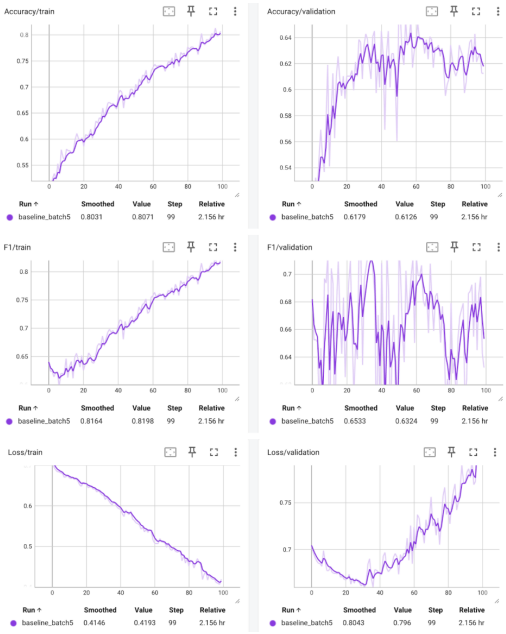

Figure 5. Image model without compression.

5.1. Baseline image model - compressed inputs

This model retains the image-pretrained backbone and the same adapter layers but introduces a compression step in image preprocessing. This allows for a batch size of 16 and enables experimentation with additional encoder layers. Fewer convolution blocks were also needed, likely contributing to reduced training time and memory usage. Importantly, we saw that the validation accuracy did not suffer but increased from 61.26% to 63.64%. This is significant as the training time also decreased substantially from 2.156 hours to just 39.8 minutes. This hints that the VQ-VAE encoder is a rich video feature extractor, better than that of the series of Conv2D blocks used in the baseline model.

We were also able to unfreeze all ViT encoder layers during training, which could have contributed to reducing overfitting: training accuracy dropped from 80.71% to 67.43%, closer matching the validation accuracy. Similarly, the training F1-score decreased from 0.82 to 0.71, but the validation F1-score rose from 0.63 to 0.69. The validation loss no longer increased, as it did in the baseline model without compression. Most notably, these results highlight the benefits of memory-efficient techniques in improving model performance while reducing computational cost (Figure 6).

5.2. Video model - uncompressed inputs

In this model, we do not compress the video inputs but replace the ViT pretrained model with MViT v2 and correspondingly replace the preceding 2D CNN tokenizer with 3D CNN. Relative to the image model with compression, we see improvement in validation accuracy, increasing from 63.64% to 66.67%. Validation F1-score is similar at 0.70, vs. 0.69 in the image model without video compression. Similar to the baseline model without video compression, we see evidence of overfitting (82.16% train accuracy vs. 66.67% validation accuracy, and 0.83 train F1-score vs. 0.70 validation F1-score). This led us to wonder whether the compression technique has a reliable regularizing effect. Furthermore, the limited performance gain over the baseline suggests that the 2D Conv blocks and ViT transformer were relatively effective at extracting features relevant for lie detection from frame collections, without explicit temporal modeling. This may indicate that collections of static facial features alone are reasonably informative, potentially rivaling short video clips in detecting deception (Figure 7).

Figure 6. Image model with compression.

5.3. Video model - compressed inputs

In the final model, we combine the use of video compression and the MViT pretrained model with a 3D CNN tokenizer. Like the compressed baseline model, we see a slight improvement in validation accuracy (68.61% vs. 66.67%) despite the runtime difference of 2.076 hr vs 40.5 min. We also see comparable validation F1-score (0.70 vs. 0.70) relative to the video model without compression. Interestingly, we do not observe as strong of a regularizing effect from compression as we saw in the image model. There is still moderate overfitting (84.54% train accuracy vs. 68.61% validation accuracy, 0.86 train F1-score vs. 0.70 validation

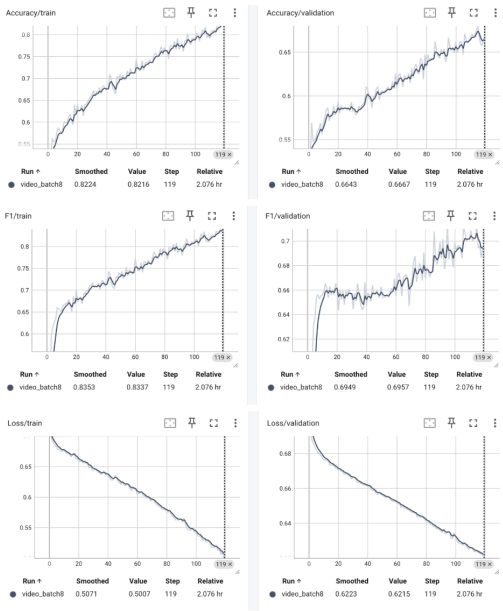

Figure 7. Image model with compression.

F1-score), which is comparable to what we observe in the video model without compression but less severe than the image model without compression. This suggests that video compression may have a slight regularizing effect, but more research is warranted on the topic. This also adds stronger evidence of the hypothesis proposed before–that using VQ-VAE encoder as a pre-processing technique for videos may be more effective in classification tasks for short clips with lower frame resolution, while reducing compute resources (Figure 8).

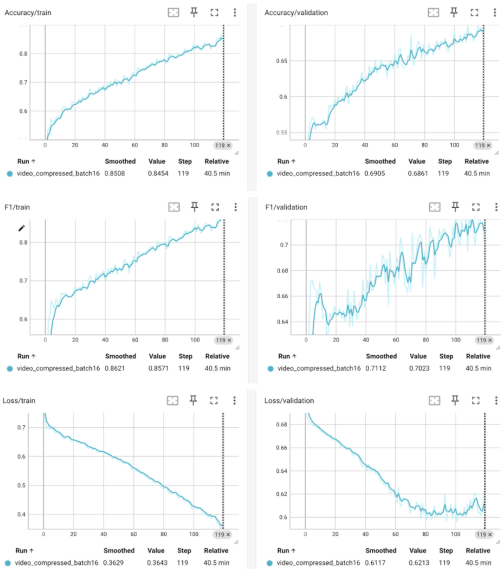

Figure 8. Image model with compression.

5.4. Qualitative analysis

We examined video clips where the models made incorrect classifications. Across all 4 models, 43 clips were consistently misclassified. This is approximately 26% to 32% of each model's total false predictions. We did not observe a significant increase in shared misclassifications between compressed vs. their non-compressed counterparts. Looking at the data itself, there were no clear visual patterns distinguishing correctly detected lies from false predictions. This is not surprising, as it is quite difficult for a human to detect lies by only looking at the video without sound (Figure 9). Given this challenge, the compressed video model's 69% accuracy is impressive. However, we did identify a potential area for improvement: in some of the falsely predicted lie videos, the beginning of the video was not cleanly cut - the clip contained reaction shots of other faces, as shown in Figure 10. These unintended frames, often showing genuine emotional responses, may have contributed to misclassification during both training and validation.

Figure 9. Example of truth telling in top row vs. example of lying in the bottom row.

Figure 10. Example of a wrongly categorized truth clip where the beginning of the clip is polluted by a reaction shot.

6. Conclusion/Future Work

This research explored the use of image and video Vision Transformer models (using preceding CNNs as tokenizers and adapter layers for parameter-efficient fine tuning) to classify deception in video clips. To address memory bottlenecks and enable higher frame sampling rates, we also explored the use of video compression.

Consistent with past ML research on deception detection, we find that these models perform significantly above chance and human benchmarks. We replicated Guo et al.'s [11] past results using ViT and no video compression, achieving 61.26% validation accuracy. We im-

proved on these results by adding a compression step before the image model, which increased validation accuracy to 63.64%. Using a video model further improved performance (66.67% validation accuracy without compression, 68.61% with compression).

These results suggest that both video models and video compression have a positive impact relative to image models and uncompressed inputs. Video models are likely beneficial because they can encode both spatial and temporal structure, whereas the CNN tokenizer process loses representation of spatial structure as it outputs a single embedding for each frame. The benefit of video compression is likely due to the VQ-VAE encoder being able to extract richer temporal features better than CNN tokenizers, at least for small clips of human faces. We also see some evidence that it could have a regularizing effect, though more investigation is warranted.

There are several future directions that would be interesting to explore with more time. First, it may prove valuable to extend this approach to a multimodal setting by including audio waveform inputs, the human annotation data, or transcripts of the spoken content. In addition, the compression technique we explored could be adapted to the audio modality to explore whether it further improves performance in a multimodal setting.

For improving compression classification performance, one could use the augmentation framework provided by compressed vision neural net, which provides horizontally flipped versions of the compressed video inputs. It would also be worthwhile to experiment with different variational autoencoders such as Neural Variational Autoencoder [21], which is a more advanced VAE that can provide high-fidelity image synthesis, to compare performance. Finally, it would be valuable to investigate the effectiveness of using VQ-VAE as a pre-processing step for image and video data across a range of classification tasks. Variational autoencoders, particularly in their vector-quantized form, may serve as a general-purpose pre-processing technique to reduce memory usage and training time, while preserving task-relevant information for downstream classification.

While there is still significant room for improvement before it can be used comfortably in high-stakes scenarios such as court proceedings, the current work illustrates that Vision Transformers can perform deception detection tasks reliably better than human baselines and provides several fruitful directions for future research.

7. Contributions

Lillian worked on the compression model and the video model tuning. Stephanie worked on the video model. Both worked on setting up the baseline model, generating the input data, and writing the paper.

Links to public repos that we adapted:
`https://github.com/google-deepmind/compressed_vision` - neural compressor VQ-VAE encoder used to generate compressed video input
`https://github.com/NMS05/AV-Data-Processing` - video input data
`https://github.com/NMS05/Audio-Visual-Deception-Detection-DOLOS-Dataset-and-Parameter-Efficient-Crossmodal-Learning` - ViT baseline model

References

[1] M. Abouelenien, V. Pérez-Rosas, R. Mihalcea, and M. Burzo. Detecting deceptive behavior via integration of discriminative features from multiple modalities. *Trans. Info. For. Sec.*, 12(5):1042–1055, May 2017.

[2] J. Allwood, L. Cerrato, K. Jokinen, C. Navarretta, and P. Paggio. The mumin coding scheme for the annotation of feedback, turn management and sequencing phenomena. *Language Resources and Evaluation*, 41:273–287, 12 2007.

[3] V. Belavadi, Y. Zhou, J. Z. Bakdash, M. Kantarcioglu, D. C. Krawczyk, L. Nguyen, J. Rakic, and B. Thuriasingham. Multimodal deception detection: Accuracy, applicability and generalizability. In *2020 Second IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, pages 99–106, 2020.

[4] A. S. Constâncio, D. F. Tsunoda, H. d. F. N. Silva, J. M. d. Silveira, and D. R. Carvalho. Deception detection with machine learning: A systematic review and statistical analysis. *Plos one*, 18(2):e0281323, 2023.

[5] H. Delmas, V. Denault, J. K. Burgoon, and N. E. Dunbar. A review of automatic lie detection from facial features. *Journal of Nonverbal Behavior*, 48(1):93–136, 2024.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[7] M. Ding, A. Zhao, Z. Lu, T. Xiang, and J.-R. Wen. Face-focused cross-stream network for deception detection in videos. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7794–7803, 2019.

[8] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

[9] S. Fernandes and M. S. Ullah. Use of machine learning for deception detection from spectral and cepstral features of speech signals. *IEEE Access*, PP:1–1, 05 2021.

[10] H. Fu, P. Lei, H. Tao, L. Zhao, and J. Yang. Improved semi-supervised autoencoder for deception detection. *PLOS ONE*, 14(10):1–13, 10 2019.

[11] X. Guo, N. M. Selvaraj, Z. Yu, A. W.-K. Kong, B. Shen, and A. Kot. Audio-visual deception detection: Dolos dataset and parameter-efficient crossmodal learning, 2023.

[12] V. Gupta, M. Agarwal, M. Arora, T. Chakraborty, R. Singh, and M. Vatsa. Bag-of-lies: A multimodal dataset for deception detection. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 83–90, 2019.

[13] H. Hung and G. Chittaranjan. The idiap wolf corpus: exploring group behaviour in a competitive role-playing game. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, page 879–882, New York, NY, USA, 2010. Association for Computing Machinery.

[14] M. Kamboj, C. Hessler, P. Asnani, K. Riani, and M. Abouelenien. Multimodal political deception detection. *IEEE MultiMedia*, 28(1):94–102, 2021.

[15] S. L. King and T. Neal. Applications of ai-enabled deception detection using video, audio, and physiological data: A systematic review. *IEEE Access*, 12:135207–135240, 2024.

[16] Y. Li, C.-Y. Wu, H. Fan, K. Mangalam, B. Xiong, J. Malik, and C. Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection, 2022.

[17] L. Mathur and M. J. Matarić. Affect-aware deep belief network representations for multimodal unsupervised deception detection. In *2021 16th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2021)*, pages 1–8, 2021.

[18] V. Pérez-Rosas, M. Abouelenien, R. Mihalcea, and M. Burzo. Deception detection using real-life trial data. In *Proceedings of the 2015 ACM on International Conference on Multimodal Interaction*, ICMI '15, page 59–66, New York, NY, USA, 2015. Association for Computing Machinery.

[19] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov. Real-time computer vision with opencv. *Commun. ACM*, 55(6):61–69, June 2012.

[20] L. Sun, Y. Wang, F. Wu, X. Li, W. Dong, and G. Shi. Deep unfolding network for efficient mixed video noise removal. *IEEE Trans. Cir. and Sys. for Video Technol.*, 33(9):4715–4727, Sept. 2023.

[21] A. Vahdat and J. Kautz. Nvae: A deep hierarchical variational autoencoder, 2021.

[22] A. van den Oord, O. Vinyals, and K. Kavukcuoglu. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6309–6318, Red Hook, NY, USA, 2017. Curran Associates Inc.

[23] O. Wiles, J. Carreira, I. Barr, A. Zisserman, and M. Malinowski. Compressed vision for efficient video understanding, 2022.

[24] J. Xiang and G. Zhu. Joint face detection and facial expression recognition with mtcnn. In *2017 4th International Conference on Information Science and Control Engineering (ICISCE)*, pages 424–427, 2017.