Truth or Lie - ResNet for Video-Based Lie Detection

Yuchen Huang

December 6, 2024

Abstract

Deception detection is a critical problem requiring reliable and objective solutions. This project, "Truth or Lie," employs deep learning to analyze videos and identify deceptive behavior. A major challenge was handling variability in video length and resolution, addressed by converting videos into frame sequences and standardizing their size for analysis. CNNs and ResNets were tested, with ResNet selected for its superior performance. The results are promising, showcasing the potential of deep learning in improving deception detection, though further refinement is necessary for enhanced accuracy.

Introduction

Detecting deception is a critical challenge in fields such as law enforcement, security, and psychology, where the ability to accurately identify lies can have significant consequences. Traditional methods of lie detection often rely on human observation of behavioral cues like body language, facial expressions, and speech patterns. However, these methods are inherently subjective and can be unreliable due to individual differences and the potential for practiced deceit.

The advent of deep learning and computer vision technologies offers new possibilities for enhancing deception detection by uncovering subtle patterns and inconsistencies that may elude human perception. This project, titled "Truth or Lie," seeks to develop a deep learning model capable of analyzing video data to detect deception more accurately and objectively.

One of the primary challenges in this endeavor is the preprocessing of video data. Videos vary widely in length, resolution, and content, introducing significant variability that can affect model performance. To address this issue, the project involved converting each video into a sequence of frames and resizing these frames to a consistent dimension. This process ensures uniformity across the dataset, allowing the model to focus on learning relevant features rather than adjusting to inconsistencies in the input data.

During the model development phase, both CNNs and Residual Networks ResNets were explored as potential architectures. The ResNet model was ultimately selected due to its superior training performance, likely attributed to its ability to train deeper networks without suffering from the vanishing gradient problem. The preliminary results obtained from the ResNet model are promising, indicating its potential effectiveness in deception detection tasks. However, there remains considerable room for improvement, and further research is necessary to refine the model and enhance its accuracy.

This project contributes to the growing body of research on automated deception detection by demonstrating the applicability of deep learning models to video analysis in this context. By addressing the challenges associated with data preprocessing and model selection, it lays the groundwork for future advancements that could have meaningful impacts on security practices and the understanding of human behavior.

You should also include a link to your github repository and have enough documentation there to make your work reproducible.

Introduction text here.

Related Work

Given that the dataset for this project originates from Kaggle, several contributors, including the dataset owner, have developed models for it. Among the publicly available models is a CNN model created by the dataset owner, *"kambing bersayap hitam,"* specifically designed for the *"Truth or Lie"* dataset. This model achieved an accuracy of 50%, serving as a baseline for comparison in this project.

Another noteworthy model is a pretrained Vision Transformer developed by Rishith Mangatt. This model features an effective data preprocessing pipeline using an open-source computer vision library, with a method for extracting frames at a consistent frame rate. However, their video extraction method has certain limitations. Building on their approach, I implemented improvements to address these shortcomings, aiming to optimize the data preprocessing step for this project.

Approach (or Methodology)

The methodology for this project involves several stages: feature extraction, dataset preparation, model design, and training. Each stage is described mathematically below.

0.1 Feature Extraction

Feature extraction was performed using a pretrained ResNet-18 model. The original classification layer (f_c) was removed to use ResNet as a feature extractor. Let $\mathbf{X} \in \mathbb{R}^{H \times W \times 3}$

represent a video frame, where H and W are the height and width of the frame. Each frame was resized to 224×224 pixels and normalized as follows:

$$\mathbf{X}_{\text{normalized}} = \frac{\mathbf{X} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}, \tag{1}$$

where $\boldsymbol{\mu} = [0.485, 0.456, 0.406]$ and $\boldsymbol{\sigma} = [0.229, 0.224, 0.225]$.

The normalized frames were passed through the ResNet-18 model, represented as a function $\text{ResNet}(\cdot)$, to extract features:

$$\mathbf{f} = \text{ResNet}(\mathbf{X}_{\text{normalized}}), \quad \mathbf{f} \in \mathbb{R}^d, \tag{2}$$

where $d = 512$ is the size of the ResNet output feature vector.

For a video with T frames, features were extracted at a fixed frame rate f_r, with an interval:

$$I = \left\lfloor \frac{\text{FPS}}{f_r} \right\rfloor, \tag{3}$$

where FPS is the video's frame rate. If the number of extracted frames T' was less than the desired sequence length T_{desired}, zero-padding was applied:

$$\mathbf{F} = \begin{cases} [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_{T'}], & \text{if } T' \geq T_{\text{desired}}, \\ [\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_{T'}, \mathbf{0}, \ldots, \mathbf{0}], & \text{if } T' < T_{\text{desired}}, \end{cases} \tag{4}$$

where $\mathbf{F} \in \mathbb{R}^{T_{\text{desired}} \times d}$ represents the sequence of frame features.

0.2 Dataset Preparation

The dataset consists of truth and lie videos, represented as:

$$\mathcal{D} = \{(\mathbf{F}_i, y_i) \mid i = 1, \ldots, N\}, \tag{5}$$

where $\mathbf{F}_i$ is the feature sequence for video i, and $y_i \in \{0, 1\}$ is the corresponding label (0: lie, 1: truth). The dataset was split into training ($\mathcal{D}_{\text{train}}$) and validation ($\mathcal{D}_{\text{val}}$) sets with a ratio of 70:30.

0.3 Model Design

The classifier predicts the label y from the feature sequence $\mathbf{F}$. Features were aggregated across the temporal dimension using mean pooling:

$$\mathbf{f}_{\text{pooled}} = \frac{1}{T_{\text{desired}}} \sum_{t=1}^{T_{\text{desired}}} \mathbf{f}_t, \quad \mathbf{f}_{\text{pooled}} \in \mathbb{R}^d. \tag{6}$$

The pooled feature vector was passed through a dropout layer ($p = 0.4$) and a fully connected layer for classification:

$$\mathbf{z} = \mathbf{W}\mathbf{f}_{\text{pooled}} + \mathbf{b}, \quad \mathbf{z} \in \mathbb{R}^2, \tag{7}$$

where $\mathbf{W} \in \mathbb{R}^{2\times d}$ and $\mathbf{b} \in \mathbb{R}^2$ are learnable parameters.

The output logits $\mathbf{z}$ were converted to probabilities using the softmax function:

$$\hat{y}_j = \frac{\exp(z_j)}{\sum_{k=1}^{2} \exp(z_k)}, \quad j \in \{0, 1\}. \tag{8}$$

The predicted label is given by:

$$\hat{y} = \arg\max_{j\in\{0,1\}} \hat{y}_j. \tag{9}$$

0.4 Training Configuration

The loss function used was cross-entropy loss:

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=0}^{1} y_{i,j} \log(\hat{y}_{i,j}), \tag{10}$$

where $y_{i,j}$ is a one-hot encoded label for video i.

The model parameters were optimized using the Adam optimizer:

$$\theta = \theta - \eta \left(\frac{\partial \mathcal{L}}{\partial \theta} + \lambda\theta \right), \tag{11}$$

where $\eta = 1 \times 10^{-4}$ is the learning rate and $\lambda = 5 \times 10^{-3}$ is the weight decay parameter.

0.5 Early Stopping

Early stopping was implemented to prevent overfitting. If the validation loss did not improve for 5 consecutive epochs, training was stopped. Validation accuracy was calculated as:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of samples}} \times 100. \tag{12}$$

—

This methodology effectively leverages pretrained networks for feature extraction, custom temporal pooling for sequence aggregation, and structured model training to tackle the deception detection problem. "'

You can copy and paste this into a LaTeX editor to render the methodology section with equations. Let me know if you need further refinements!

Datasets

The dataset consists of approximately 850 videos, evenly balanced between truth and lie labels. Each video varies in length, ranging from 20 to 80 seconds, and features a single main character. Notably, individuals may appear in multiple videos but consistently exhibit either truthful or deceptive behavior across all their clips.

To prepare the dataset for analysis, video frames were extracted at a specific frame rate and resized to a fixed dimension of 224x224 pixels. These frames were then normalized for consistency. Each video was represented as a sequence of frames, and for videos with fewer frames than the desired sequence length, blank frames were added to ensure uniform input sizes.

For dataset splitting, care was taken to ensure that individuals appearing in the dataset were not shared between the training and validation sets, maintaining the integrity of the evaluation process. This ensures that the model evaluates unseen individuals, providing a more robust measure of its ability to generalize to new data.

Evaluation Results

The model's performance was evaluated using validation accuracy and validation loss. Validation accuracy, which measures the proportion of correctly classified samples, increased steadily from 54.25% in the first epoch to 86.64% in the 20th epoch, while validation loss decreased from 0.6669 to 0.4018, indicating improved alignment between the model's predictions and the ground truth labels. These results demonstrate that the model effectively learned meaningful features for deception detection while avoiding overfitting, as evidenced by consistent performance improvements. Compared to the baseline CNN model by the dataset owner, *"kambing bersayap hitam"*, which achieved an accuracy of 50%, the ResNet-based model significantly outperformed, achieving a margin of over 36 percentage points. This substantial improvement highlights the effectiveness of using ResNet for feature extraction and mean pooling for temporal aggregation. While the results are promising, further refinements could enhance performance, including fine-tuning the pretrained ResNet, exploring advanced architectures such as Vision Transformers, and implementing additional data augmentation techniques to improve generalization. Overall, the evaluation results validate the methodology and highlight the potential of deep learning models for automated deception detection.

Conclusion

The model demonstrates high accuracy, indicating its strong effectiveness in detecting deception. However, its reliability may be constrained by the dataset's limitations. Despite including 850 videos, the dataset comprises short clips derived from fewer than 50 longer videos, which reduces diversity and may limit the generalizability of the model. Additionally, if a person tells the truth in some clips and lies in others, the model's ability to identify consistent distinguishing features remains uncertain.

For future work, efforts can focus on improving the data preprocessing pipeline. Currently, the model might incorrectly capture the number of blank frames in videos (reflecting their length) as a feature, which could bias predictions. A more randomized approach to splitting videos into training and validation sets is also needed to prevent unintended correlations based on video length or other metadata. By addressing these limitations, future studies can further enhance the model's reliability and robustness.

References

[1] Kambing Bersayap Hitam. *Truth or Lie Dataset.* Kaggle, 2022. `https://www.kaggle.com/datasets/kambingbersayaphitam/truthorlie`

[2] Kambing Bersayap Hitam. *CNN - Truth or Lie.* Kaggle, 2022. `https://www.kaggle.com/code/kambingbersayaphitam/cnn-truth-or-lie`

[3] OpenAI. *ChatGPT: Optimizing Language Models for Dialogue.* OpenAI, 2023. Available: `https://openai.com/blog/chatgpt`.

[4] Rishith Mangatt. *truth_or_lie_v18.2.* Kaggle, 2024. Available: `https://www.kaggle.com/code/rishithmangatt/truth-or-lie-v18-2`.

[5] Juan C. Olamendy. *Real World ML: Early Stopping in Deep Learning - A Comprehensive Guide*. Medium, 2024. Available: `https://medium.com/@juanc.olamendy/real-world-ml-early-stopping-in-deep-learning-a-comprehensive-guide-fabb1e69f8cc`.