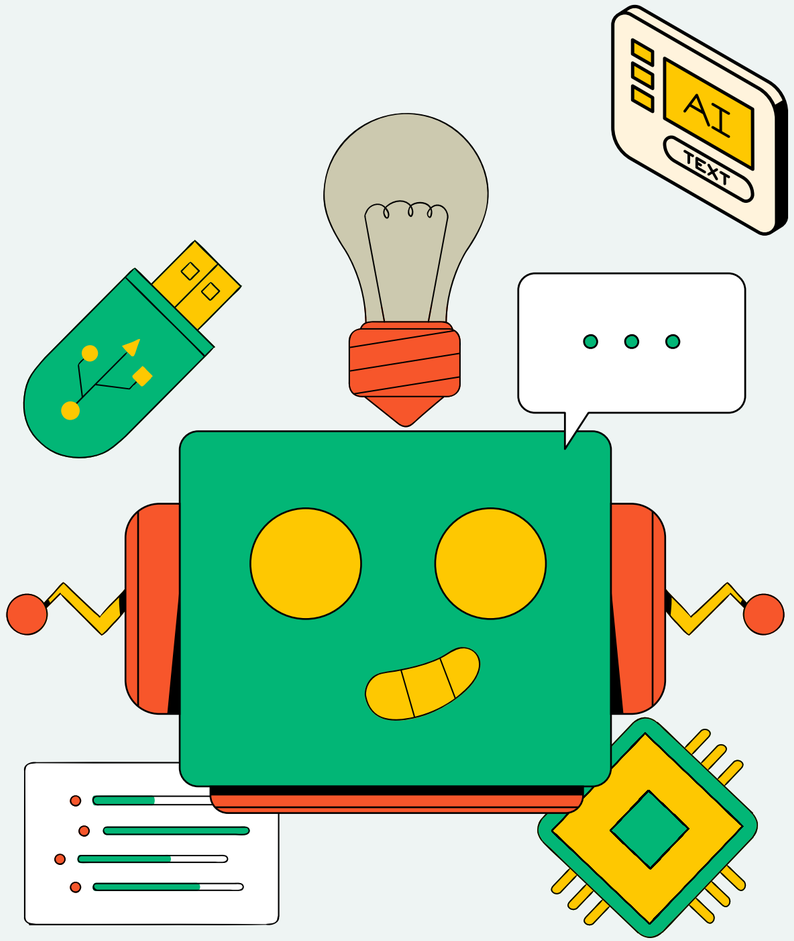
WE LEARN FOR THE FUTURE

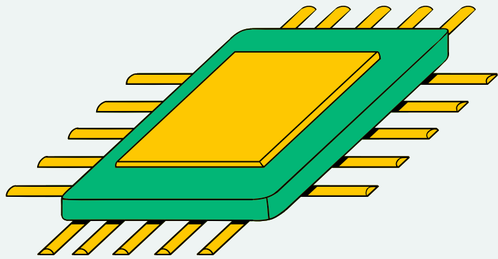
DEEPFAKE DETECTION USING DL AND ML

COMPARING EFFICIENTNET CNN VS. RANDOM FOREST CLASSIFICATION



PRESENTED BY:

JONATHAN MONROE



THE UNIVERSITY OF CHICAGO

MASTERS IN COMPUTATIONAL SOCIAL SCIENCE

HTTPS://DRIVE.GOOGLE.COM/FILE/D/1BE7AVPMGT-EBT4GRQQSNTZ4GEHV007G1/VIEW?USP=SHARING

INTRODUCTION: RESEARCH QUESTION

The Problem: Deepfakes are increasingly, and maliciously, used for misinformation and fraudulent purposes online.

Detecting Deepfake images is essential for security, media integrity, and policy formation

This leads me to ask: **How well can ML and DL models detect deepfake images? Which approach performs better?**

BACKGROUND

Deepfakes: AI-generated media that manipulates visual/audio content
-misinformation -> security threats
-ethical considerations

Literature on social impact:

- Chapagain, Devendra, Naresh Kshetri, and Bindu Aryal (2024).
- Wazid, Mohammad, Amit Kumar Mishra, Noor Mohd, and Ashok Kumar Das (2024).

Literature supporting ML as an approach:

- Vaidya, Anusha O., Monika Dangore, Vishal Kisan Borate, Nutan Raut, Yogesh Kisan Mali, and Ashvini Chaudhari (2024).
- Heidari, Arash, Nima Jafari Navimipour, Hasan Dag, and Mehmet Unal (2024).

DATASET OVERVIEW

Data Source:

Deepfake Detection Challenge (DFDC) dataset (from Kaggle)

Dataset Size and Features:

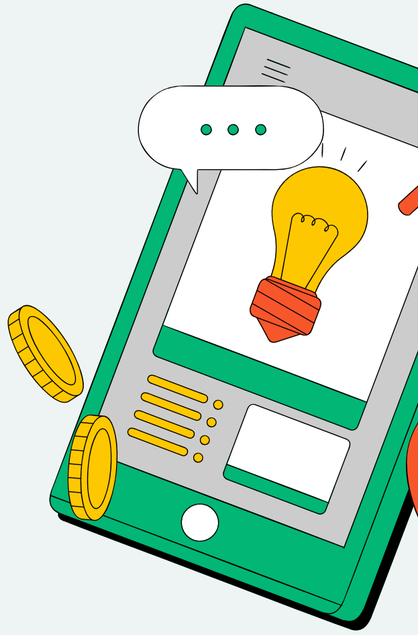
-Total Frames: 52,591

-Feature Type: Image pixel values converted into feature vectors

DATA PREPROCESSING

- Extracted frames from videos
- Merged metadata (real versus fake labels)
- Converted images into numerical feature vectors (tensor for DL; arrays for M)
- Standardized pixel values for DL model

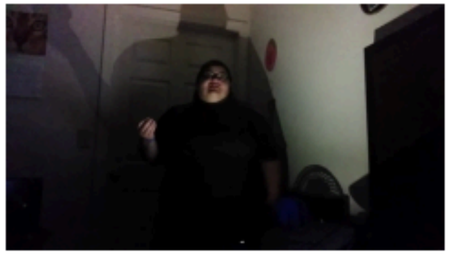
SAMPLE DATA REPRESENTATION



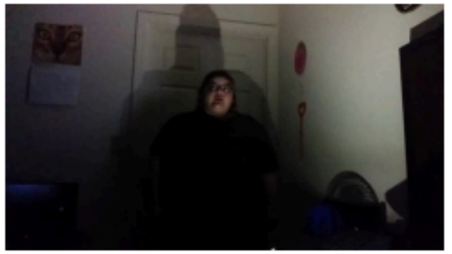
Example Frames: 2 REAL (Left) & 3 FAKE (Right)

Label: REAL

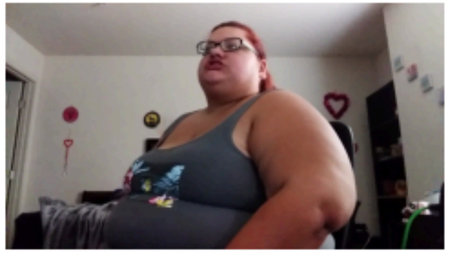
Label: REAL



Label: FAKE



Label: FAKE



Label: FAKE

CONT.

Numerical matrix representation of pixel intensity values

–Darker pixels –> lower intensity and vice versa for brighter pixels

This transformation enables ML and DL to process and learn patterns from images

Pixel Matrix Representation (REAL)

Numerical Matrix Sample (REAL)

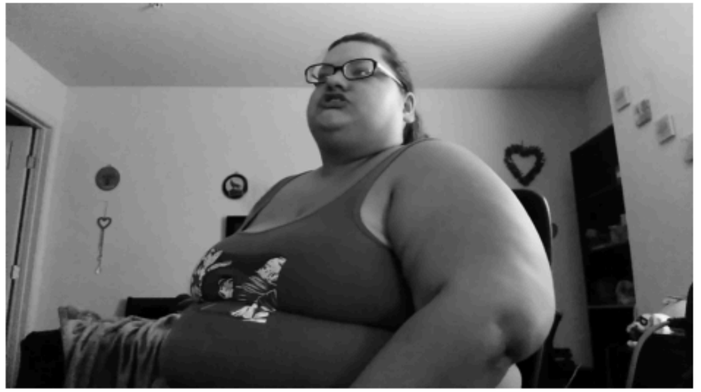
	0	1	2	3	4	5	6	7	8	9
0	206	206	206	206	206	206	206	206	207	207
1	206	206	206	206	206	206	206	206	207	207
2	206	206	206	206	206	206	206	206	207	207
3	206	206	206	206	206	206	206	206	207	207
4	206	206	206	206	206	206	206	206	207	207
5	206	206	206	206	206	206	206	206	207	207
6	207	207	207	207	207	207	207	207	208	208
7	208	208	208	208	208	208	208	208	208	208
8	207	207	207	207	207	207	207	207	207	207
9	208	208	208	208	208	208	208	208	208	208

Pixel Matrix Representation (FAKE)

Numerical Matrix Sample (FAKE)

	0	1	2	3	4	5	6	7	8	9
0	41	44	48	50	50	50	56	62	47	43
1	35	39	46	52	54	52	47	45	43	40
2	34	37	43	50	53	50	44	39	39	39
3	37	39	42	45	46	46	45	45	41	44
4	42	43	43	42	42	43	45	47	49	52
5	47	45	42	41	42	46	49	51	56	59
6	47	46	44	43	45	49	52	53	55	57
7	43	46	50	50	49	50	50	49	48	51
8	45	47	49	49	49	49	46	44	39	38
9	48	50	51	51	50	47	44	40	39	38

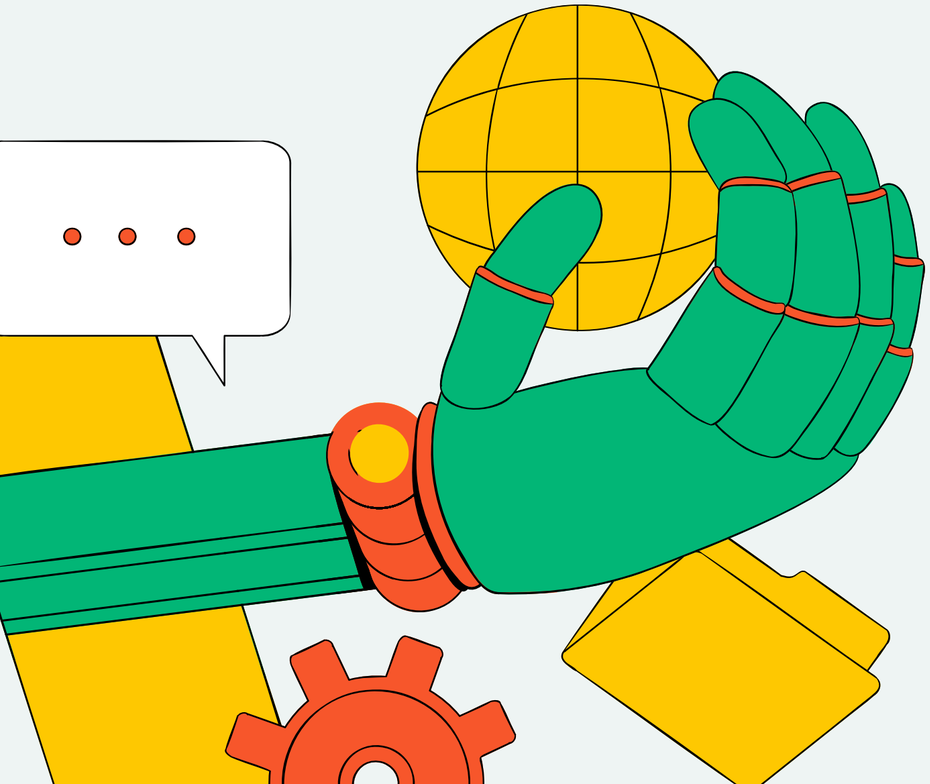
Pixel Matrix Representation (FAKE)



Numerical Matrix Sample (FAKE)

	0	1	2	3	4	5	6	7	8	9
0	129	130	132	133	134	133	131	129	129	131
1	128	129	130	131	131	130	129	128	128	130
2	128	128	128	128	127	127	127	127	126	130
3	128	128	128	126	125	126	127	129	128	130
4	130	130	129	127	126	126	128	130	130	131
5	130	130	129	127	126	126	128	130	130	131
6	128	128	127	126	126	126	127	128	129	130
7	126	126	126	125	125	125	125	126	128	130
8	125	124	124	124	125	126	127	127	127	130
9	126	126	125	125	126	126	125	125	125	127

DEEP LEARNING MODEL (EFFICIENTNET-BASED CNN)



Why EfficientNet?

–Optimized CNN architecture for image classification; optimally scales depth, width, and resolution

–faster training; reduced computational cost

Training Process:

–Input frames

–feature extraction from EfficientNet; numerical feature representations

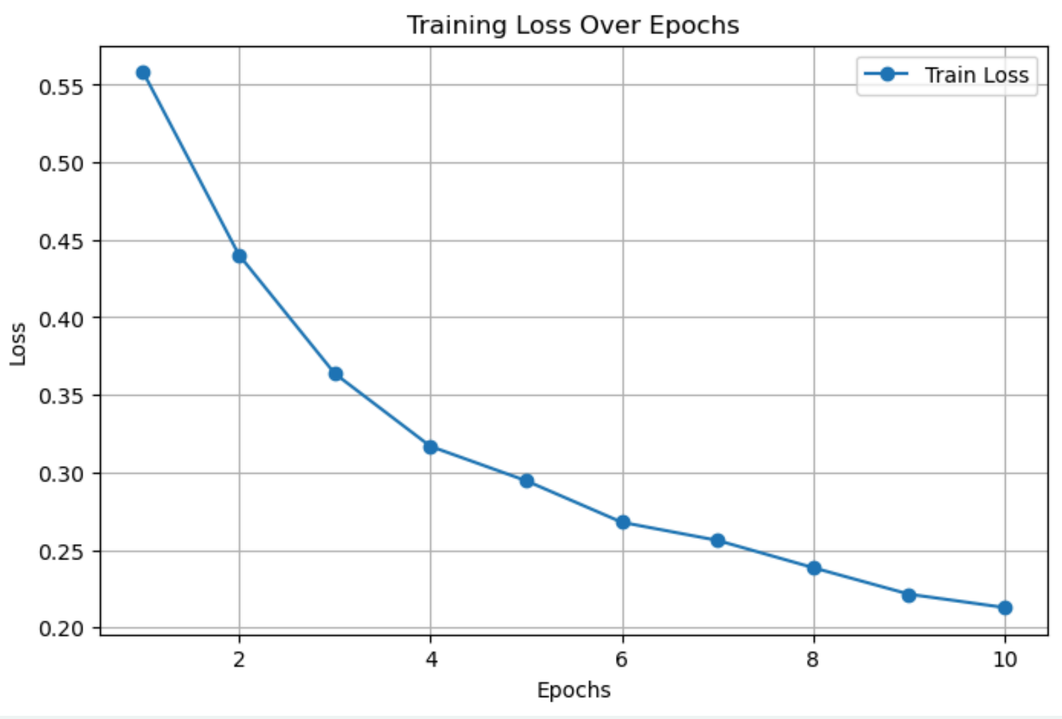
–train model on deepfake recognition

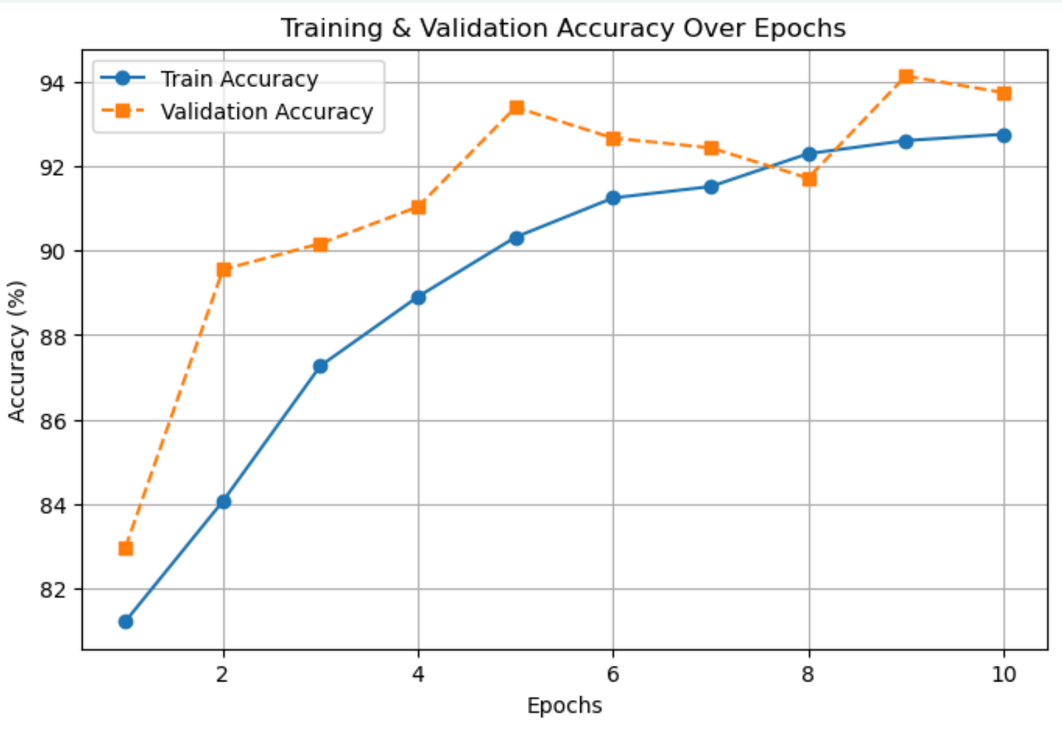
DL MODEL TRAINING EVALUATION

Loss curve –> convergence over epochs (stabilizaton); model is learning with each epoch

Training and validation accuracy over epochs increasing in unison

–Absence of major divergence between training and validation could indicate slight overfitting





DL MODEL RESULT EVALUATION

High precision –> most predicted fake frames are indeed fake

high recall –> captures most of the actual fake frames, but some are still misclassified

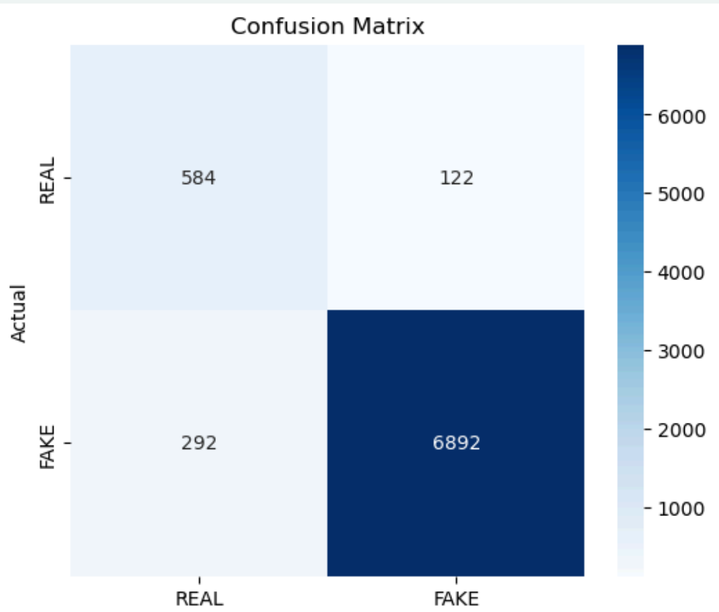
F1-score –> strong overall deepfake detection

Confusion matrix:

lots of correctly identified fake frames

Overall decent performance, some FP some FN, but most are TP and TN

```
Precision: 0.9826
Recall: 0.9594
F1-score: 0.9708

Classification Report:
              precision    recall  f1-score   support

        REAL       0.67      0.83      0.74       706
        FAKE       0.98      0.96      0.97      7184

    accuracy                           0.95      7890
   macro avg       0.82      0.89      0.85      7890
weighted avg       0.95      0.95      0.95      7890
```



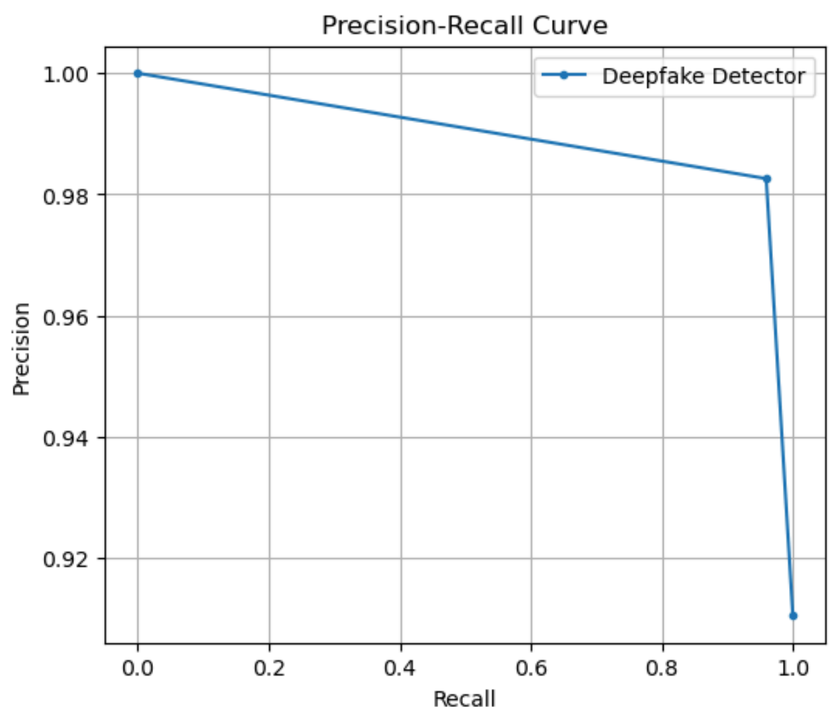
DL MODEL RESULT EVALUATION CONT.

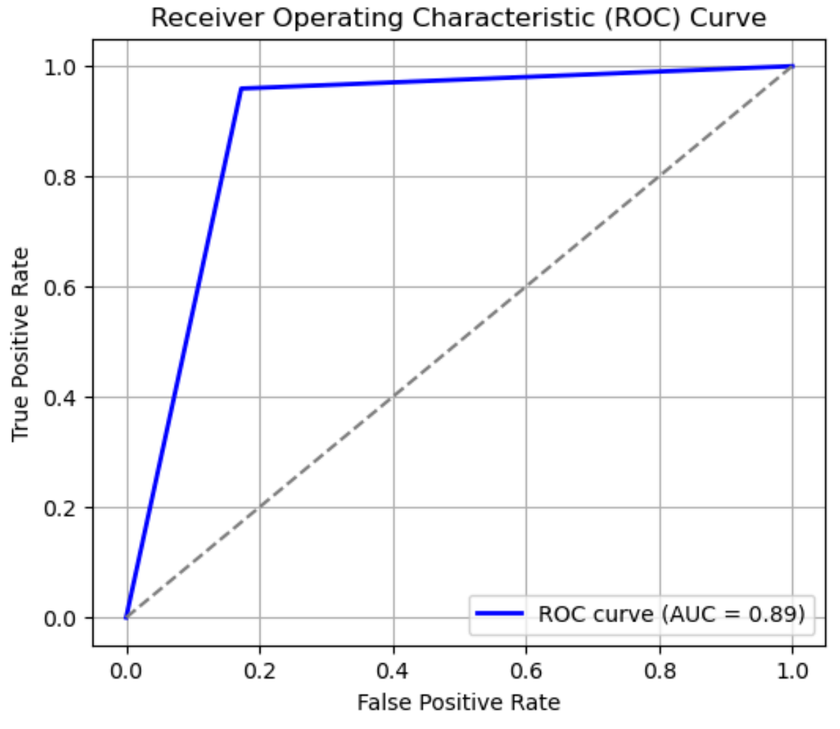
Precision-Recall Curve gradually declines -> model prioritizes detecting fake frames

-model is more recall focused, ensuring most fake frames are detected at the cost of more false positives

AUC = 0.89 -> strong model performance; effectively separating real and fake frames

-though some misclassification still present





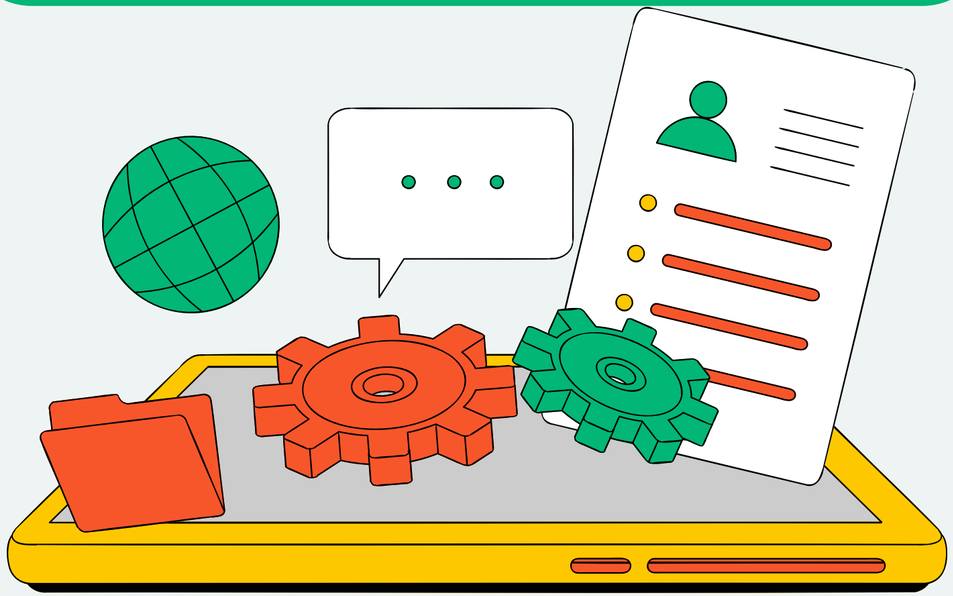
MACHINE LEARNING MODEL: RANDOM FOREST

handles high dimensional data efficiently

strong generalization with ensemble learning

robust from random feature selecton and DT averaging

EfficientNet-generated vectors (arrays)



FIRST, HYPERPARAMETER TUNING

Optimizes model performance

Controls model complexity

RandomizedSearchCV over GridSearchCV:

–faster and more computationally efficient

–Limits n_estimators (trees), max depth, and leaf nodes

```
Starting Randomized Search...
Fitting 2 folds for each of 10 candidates, totalling 20 fits
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time= 1.3min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time= 1.3min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time= 2.3min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time= 2.3min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=1, min_samples_split=5, n_estimators=200; total time= 4.6min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=1, min_samples_split=5, n_estimators=200; total time= 4.7min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time= 5.8min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time= 5.9min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time= 4.7min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time= 4.7min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=1, min_samples_split=5, n_estimators=100; total time= 6.3min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=100; total time= 6.3min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time= 4.6min
[CV] END max_depth=20, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time= 4.6min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time= 2.5min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=200; total time= 2.4min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time= 2.1min
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time= 2.0min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time= 7.6min
[CV] END max_depth=None, max_features=sqrt, min_samples_leaf=2, min_samples_split=2, n_estimators=200; total time= 7.8min
Hyperparameter tuning completed!
Best Parameters Found: {'n_estimators': 200, 'min_samples_split': 2, 'min_samples_leaf': 2, 'max_features': 'sqrt', 'max_depth': 10}
Optimized ML Model Saved.
Partial training results saved to 'random_search_results.csv'.
```

RF ML MODEL TRAINING EVALUATION

Picked other parameters besides most optimal parameter to cross reference with testing results later
-Testing uses hyperparameter results
-Decent performance across the board

Takeaways:
-Favor for fake detection at expensive of classifying real
-more complex -> more real frames at the cost of fake frames, but does not improve overall accuracy
-Bias toward deepfake detection

```
Training Model 1 with Params: {'n_estimators': 50, 'max_depth': 5, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 'sqrt'}

Training Model 2 with Params: {'n_estimators': 100, 'max_depth': 10, 'min_samples_split': 5, 'min_samples_leaf': 2, 'max_features': 'sqrt'}

Training Model 3 with Params: {'n_estimators': 200, 'max_depth': None, 'min_samples_split': 2, 'min_samples_leaf': 2, 'max_features': 'sqrt'}
```

	Model	Accuracy	Precision_Real	Recall_Real	F1_Real	Precision_Fake	Recall_Fake	F1_Fake
0	Config 1	0.917089	1.000000	0.070234	0.131250	0.916567	1.000000	0.956467
1	Config 2	0.923800	0.837209	0.180602	0.297111	0.925498	0.996562	0.959716
2	Config 3	0.924247	0.660714	0.309365	0.421412	0.935730	0.984447	0.959470

RF ML MODEL TESTING EVALUATION

Extreme bias for detecting fake frames

-near perfect detection of of fake

-significant misclassification of real

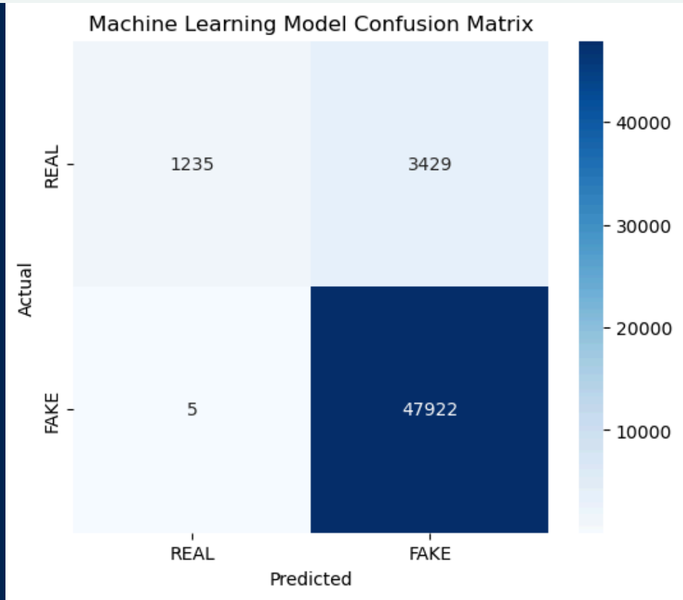
high accuracy score -> could be misleading due to bias, as recall for real frames is low

Confusion matrix:

lots of misclassified real frames

very small number of misclassified fake frames (further illustrating bias)

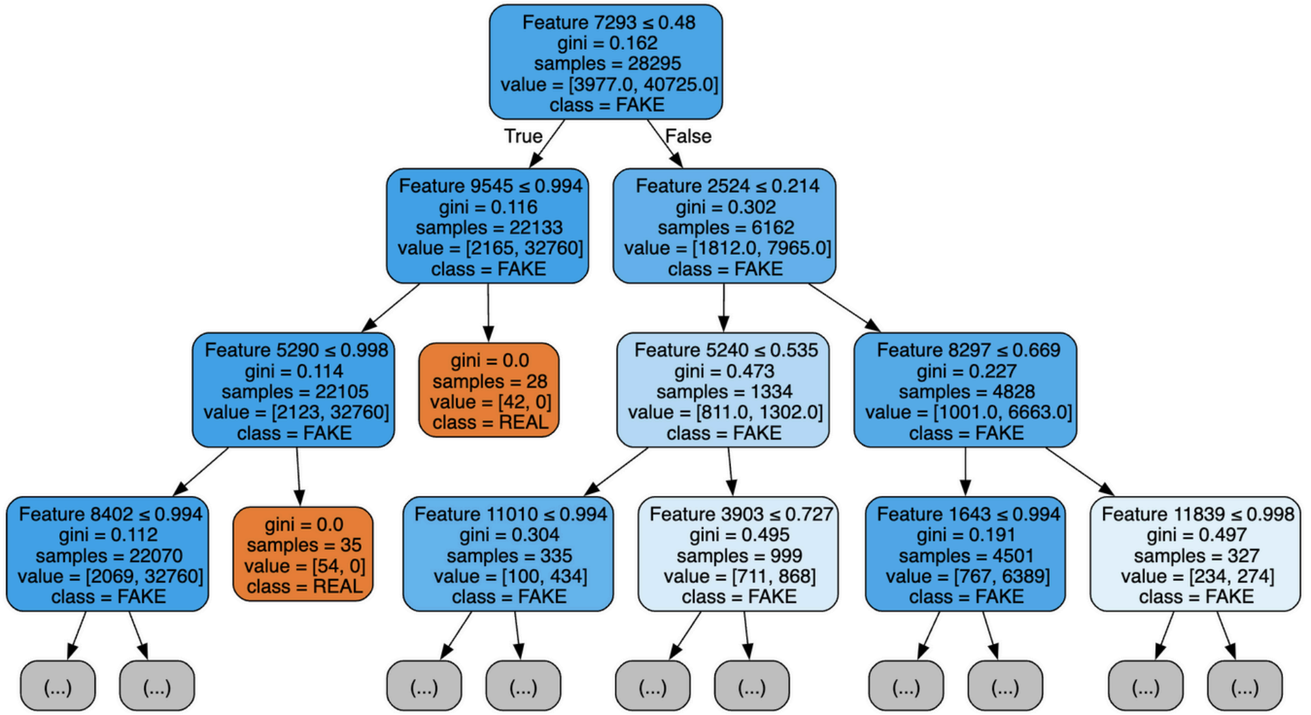
```
Optimized ML Model Performance
Precision: 0.9332, Recall: 0.9999, F1-score: 0.9654

Optimized ML Classification Report
              precision    recall  f1-score   support

        REAL       1.00      0.26      0.42      4664
        FAKE       0.93      1.00      0.97     47927

    accuracy                           0.93     52591
   macro avg       0.96      0.63      0.69     52591
weighted avg       0.94      0.93      0.92     52591
```

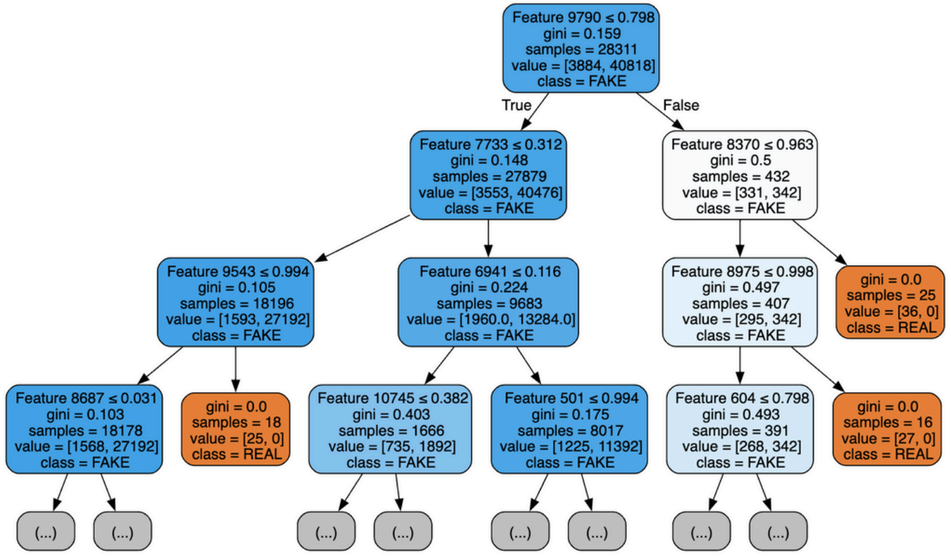


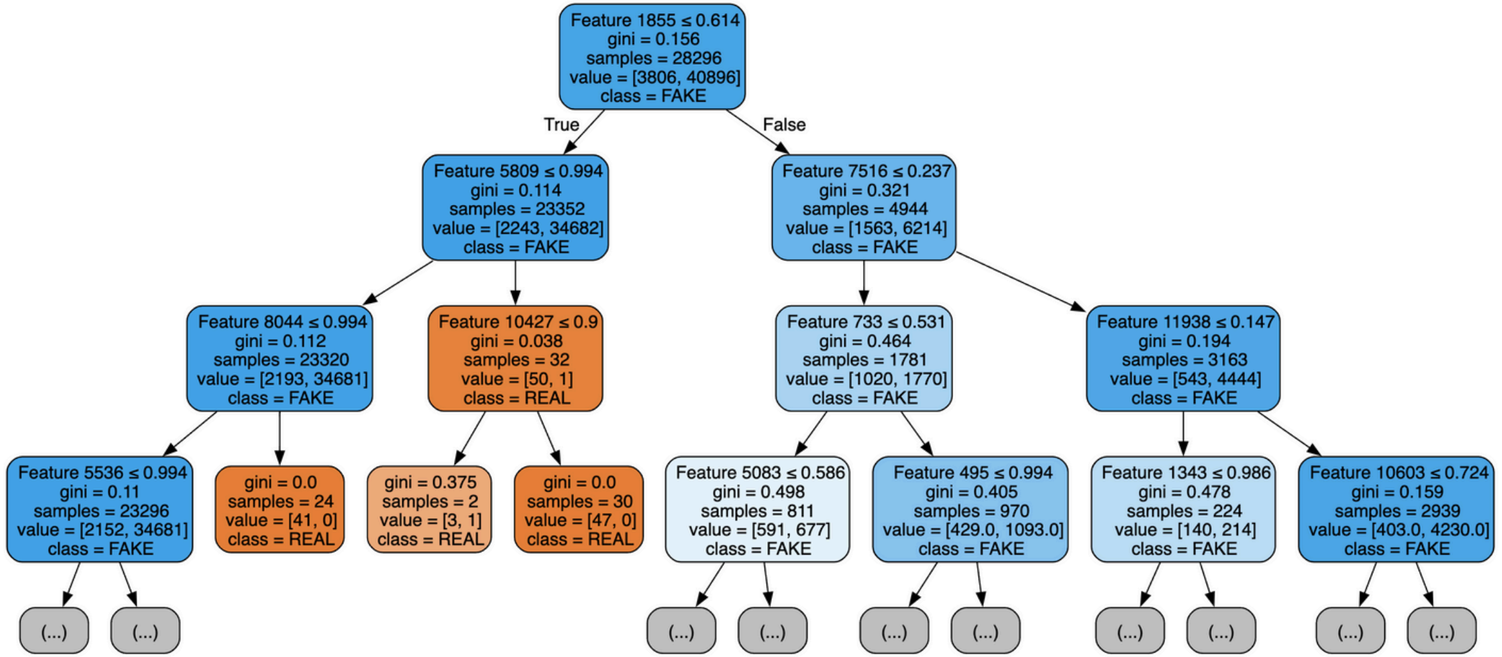
SAMPLED TREE EXAMPLES (FROM FOREST)

Specific threshold decision making

Splits start from the root node (most important, highest information gain feature)





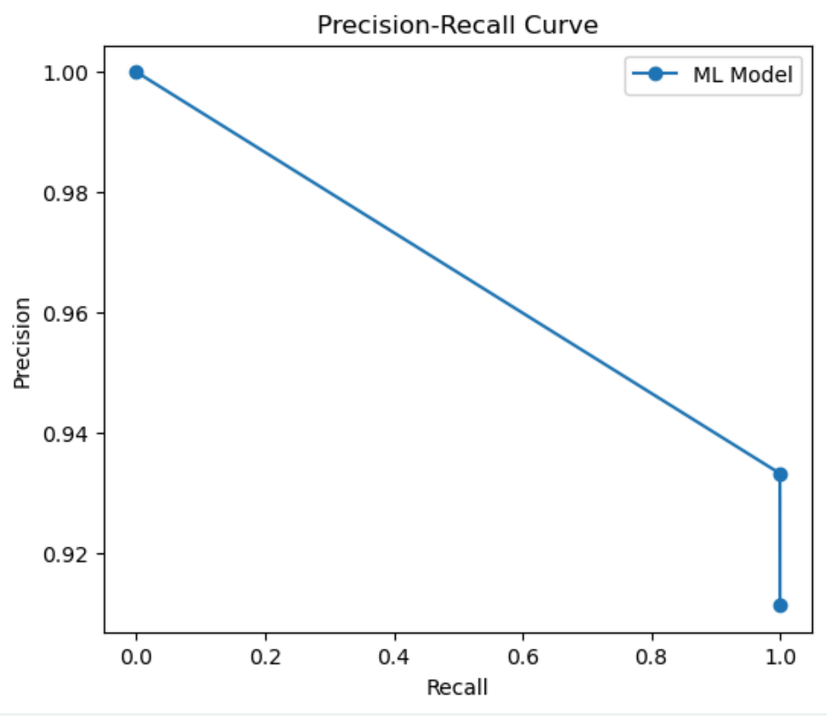


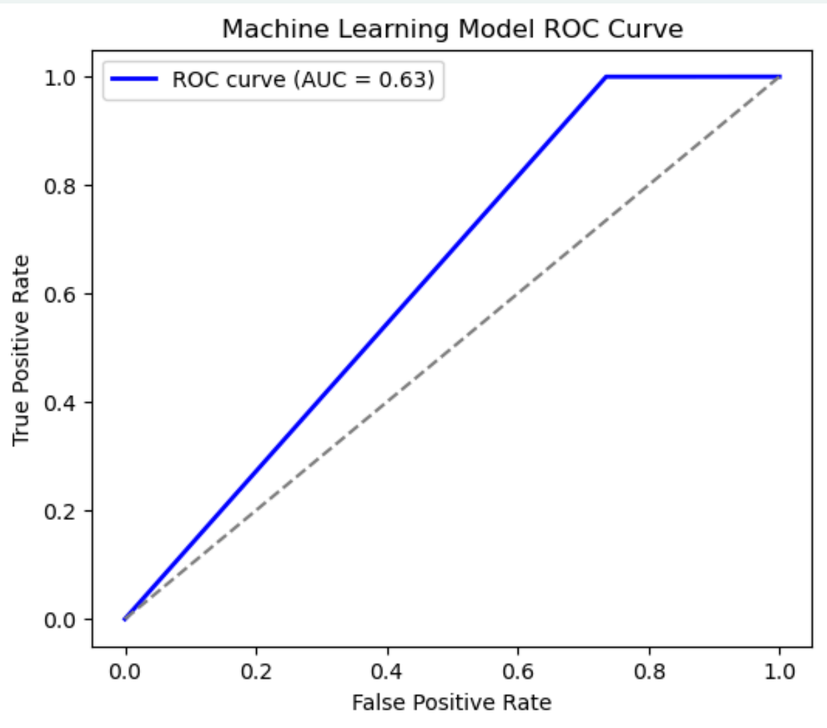
ML MODEL RESULT EVALUATION

Precision-Recall Curve shows steep decline in precision as recall increases -> misclassifying many real fames while capturing most fake frames

AUC = 0.63 -> only slightly better than random guessing (AUC - 0.5)

-model struggles with distinguishing real frames from fake frames





ML VS DL FOR DEEPFAKE CLASSIFICATION

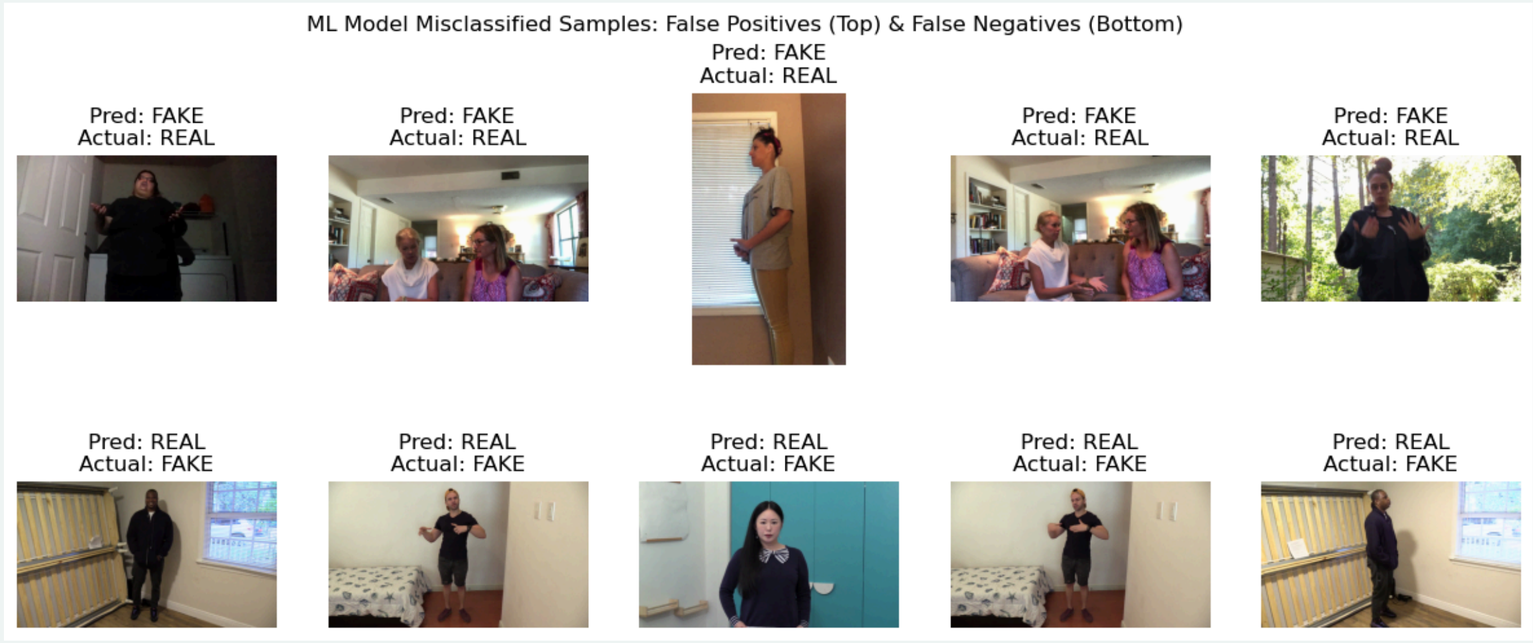
RANDOM FOREST (ML) VS. CONVOLUTIONAL NEURAL NETWORK (DL)

- accuracy: 93% vs 95%
- Precision (Fake): 93% vs 98%
- Recall (Fake): 100% vs 96%
- Precision (Real): 67% vs 67%
- Recall (Real): 26% vs 83%

KEY TAKEAWAY: DL MODEL IS BETTER

ERROR ANALYSIS: EXAMPLES

ML Model Misclassified Samples: False Positives (Top) & False Negatives (Bottom)



DL Model Misclassified Samples: False Positives (Top) & False Negatives (Bottom)

CONCLUSIONS AND FUTURE WORK

CNN WITH EFFICIENTNET

Better overall generalization

higher real frame recall

RANDOM FOREST

good at fake detection

struggles with real images

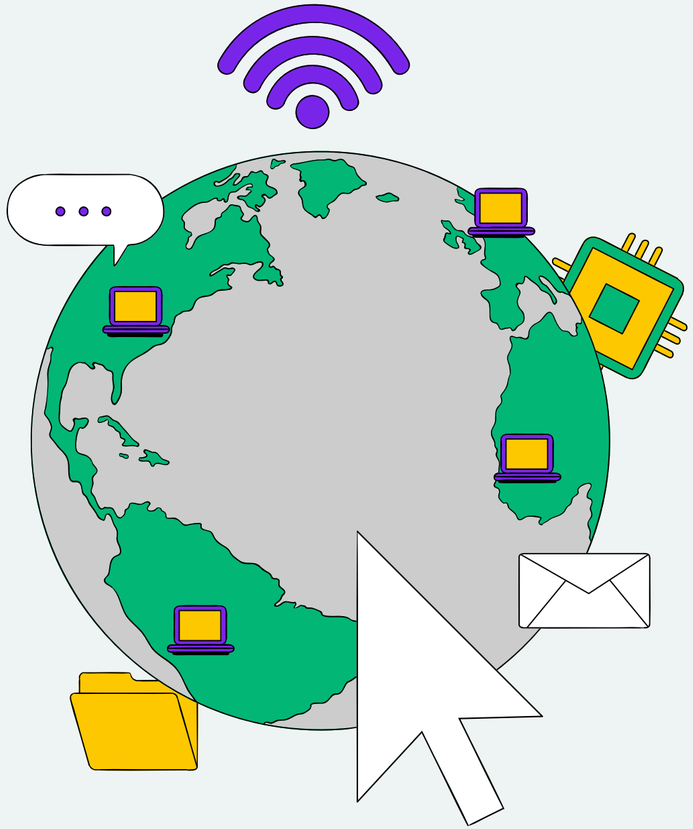
FUTURE WORK

Address false positives in RF model (fixing bias)

demographic bias detection

Analyzing other forms of deepfake material (i.e. voice)

BIBLIOGRAPHY



Chapagain, Devendra, Naresh Kshetri, and Bindu Aryal. 2024. "Deepfake Disasters: A Comprehensive Review of Technology, Ethical Concerns, Countermeasures, and Societal Implications." 2024 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC).

Heidari, Arash, Nima Jafari Navimipour, Hasan Dag, and Mehmet Unal. 2024. "Deepfake detection using deep learning methods: A systematic and comprehensive review." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 14 (2): e1520.

Vaidya, Anusha O, Monika Dangore, Vishal Kisan Borate, Nutan Raut, Yogesh Kisan Mali, and Ashvini Chaudhari. 2024. "Deep Fake Detection for Preventing Audio and Video Frauds Using Advanced Deep Learning Techniques." 2024 IEEE Recent Advances in Intelligent Computational Systems (RAICS).

Wazid, Mohammad, Amit Kumar Mishra, Noor Mohd, and Ashok Kumar Das. 2024. "A secure deepfake mitigation framework: Architecture, issues, challenges, and societal impact." Cyber Security and Applications 2: 100040.

EfficientNet documentation:
https://arxiv.org/abs/1905.11946

Pytorch image processing documentation:
https://pytorch.org/vision/stable/transforms.html