FACEGUARD-AI

Detecting AI-generated Faces using Machine Learning and Pattern Recognition

Presenters: Vani Seth and Pari Patel
Course Name: Introduction to Machine Learning and Pattern Recognition
Course Instructor: Dr. Yunxin Zhao

Problem Statement: The Rise of AI-Generated Images

- Advances in **AI tools like GANs and diffusion models** have made it easy to generate **hyper-realistic fake images**.
- These images are often **indistinguishable from real photos**, posing major risks:
  - **Misinformation** and fake news
  - **Identity theft** and privacy violations
  - **Erosion of public trust** in visual media

Stats That Matter

- Deepfake content has surged by **900%** since 2018 *(Deeptrace, 2023)*
- Over **96%** of AI-generated deepfakes are used in **non-consensual contexts**
- **63%** of Americans worry about synthetic media spreading false information *(Pew Research Center)*

The need for accurate detection methods is therefore more urgent than ever.

Introduction

- AI models like GANs and Stable Diffusion create hyper-realistic human faces
- Real vs fake faces are now difficult to distinguish by eye
- Our goal: Build a system to detect AI-generated faces automatically
- Approach: Use machine learning and pattern recognition for detection

Our Dataset

- 140K Real and Fake Faces Dataset from Kaggle [1]
- contains 70,000 real faces from the Flickr dataset [2] collected by Nvidia and 70,000 fake faces generated by StyleGAN [3]
- Each image in the dataset is in RGB format and originally sized at 128×128 pixels.
- The dataset is pre-split into training and test sets for model development and evaluation.

Training Pipeline

- The training process is done for 4 different models:
  - Training a simple CNN model
  - Training a Deep CNN model
  - Training a model with transfer learning
    - ResNet50
    - MobileNetV2
- All models were trained using the Adam optimizer and binary cross-entropy loss function.
- We used a batch size of 32 and trained for 10 epochs.
- To prevent overfitting, dropout layers are included.

Simple CNN Model

- The Simple CNN model consists of three convolutional layers, each followed by a max-pooling operation.
- After flattening the features, we used a fully connected dense layer with 128 neurons and a dropout layer to prevent overfitting.
- The final output layer uses a sigmoid activation function to perform binary classification.
- This architecture contains approximately 11.17 million trainable parameters.

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 222, 222, 32)	896
max_pooling2d_3 (MaxPooling2D)	(None, 111, 111, 32)	0
conv2d_4 (Conv2D)	(None, 109, 109, 64)	18,496
max_pooling2d_4 (MaxPooling2D)	(None, 54, 54, 64)	0
conv2d_5 (Conv2D)	(None, 52, 52, 128)	73,856
max_pooling2d_5 (MaxPooling2D)	(None, 26, 26, 128)	0
flatten_1 (Flatten)	(None, 86528)	0
dense_2 (Dense)	(None, 128)	11,075,712
dropout_1 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 1)	129

Total params: 11,169,089 (42.61 MB)
Trainable params: 11,169,089 (42.61 MB)
Non-trainable params: 0 (0.00 B)

Deep CNN Model

- The Deep CNN model extends the Simple CNN by incorporating additional convolutional layers
- It uses a total of six convolutional layers with increasing filter sizes and two fully connected layers
- It includes a dense layer with 256 neurons
- This results in a significantly higher capacity model with around 19.16 million trainable parameters

===== Deep CNN Architecture =====

Model: "sequential_9"

Layer (type)	Output Shape	Param #
conv2d_33 (Conv2D)	(None, 222, 222, 32)	896
batch_normalization_12 (BatchNormalization)	(None, 222, 222, 32)	128
max_pooling2d_21 (MaxPooling2D)	(None, 111, 111, 32)	0
dropout_21 (Dropout)	(None, 111, 111, 32)	0
conv2d_34 (Conv2D)	(None, 109, 109, 64)	18,496
batch_normalization_13 (BatchNormalization)	(None, 109, 109, 64)	256
max_pooling2d_22 (MaxPooling2D)	(None, 54, 54, 64)	0
dropout_22 (Dropout)	(None, 54, 54, 64)	0
flatten_7 (Flatten)	(None, 186624)	0
dense_18 (Dense)	(None, 128)	23,888,000
dropout_23 (Dropout)	(None, 128)	0
dense_19 (Dense)	(None, 1)	129

Total params: 23,907,905 (91.20 MB)
Trainable params: 23,907,713 (91.20 MB)
Non-trainable params: 192 (768.00 B)

MobileNetV2 Model

- MobileNetV2 is a lightweight architecture optimized for mobile and embedded devices.
- Like ResNet50, it was used with pretrained weights from ImageNet, and its layers were frozen.
- We used a custom head for classification.
- With only 2.42 million total parameters and 0.16 million trainable ones, this model is also a good fit for training images.

```
===== MobileNetV2 Architecture =====

Model: "sequential_4"
```

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functional)	(None, 7, 7, 1280)	2,257,984
global_average_pooling2d_1 (GlobalAveragePooling2D)	(None, 1280)	0
dense_8 (Dense)	(None, 128)	163,968
dropout_4 (Dropout)	(None, 128)	0
dense_9 (Dense)	(None, 1)	129

```
Total params: 2,422,081 (9.24 MB)
Trainable params: 164,097 (641.00 KB)
Non-trainable params: 2,257,984 (8.61 MB)
```

ResNet50 Model

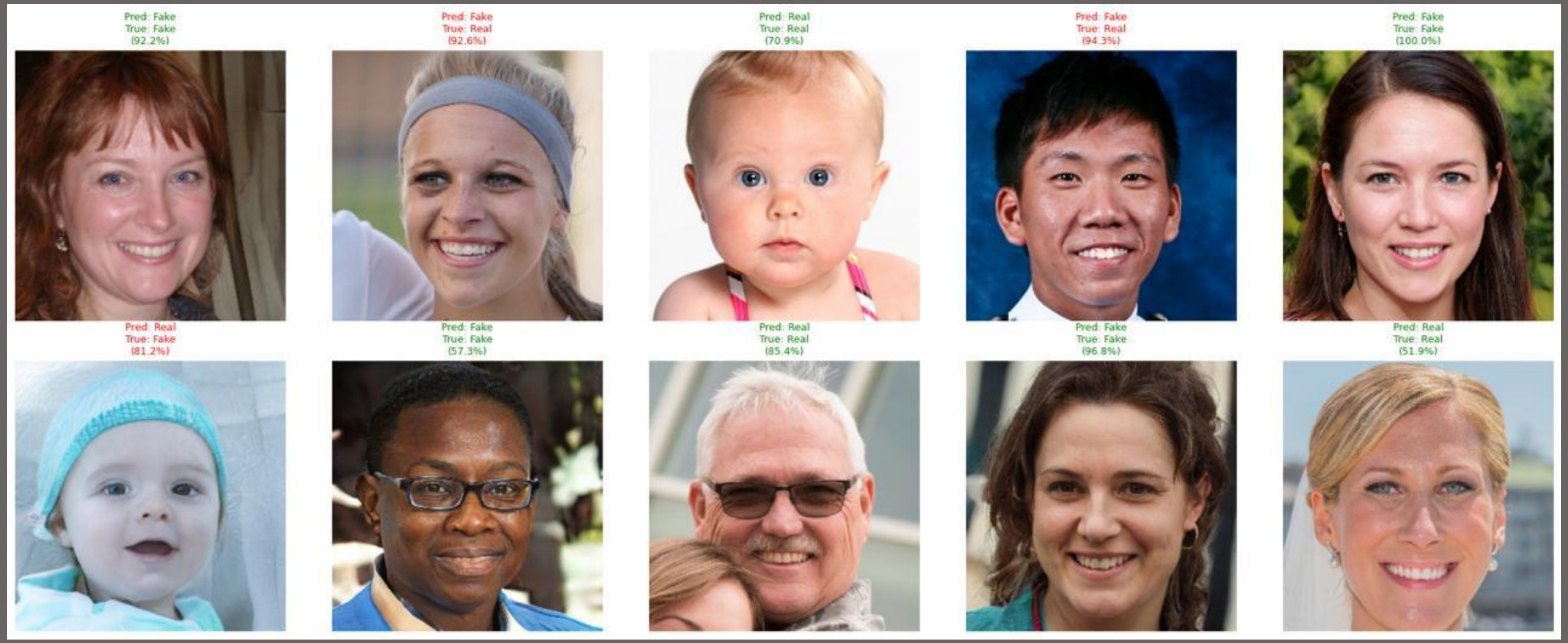
- For the ResNet50 model, we used a pretrained network trained on ImageNet as a fixed feature extractor.
- All the layers in ResNet50 were frozen, and we added a custom classification head consisting of a global average pooling layer, a dense layer with 128 units, a dropout layer, and a final sigmoid output layer.
- Although the total parameter count is approximately 23.8 million, only 0.26 million parameters are trainable, making it efficient to train

```
===== ResNet50 Architecture =====

Model: "sequential_3"
```

Layer (type)	Output Shape	Param #
resnet50 (Functional)	(None, 7, 7, 2048)	23,587,712
global_average_pooling2d (GlobalAveragePooling2D)	(None, 2048)	0
dense_6 (Dense)	(None, 128)	262,272
dropout_3 (Dropout)	(None, 128)	0
dense_7 (Dense)	(None, 1)	129

```
Total params: 23,850,113 (90.98 MB)
Trainable params: 262,401 (1.00 MB)
Non-trainable params: 23,587,712 (89.98 MB)
```

Model Predictions



Performance Evaluation

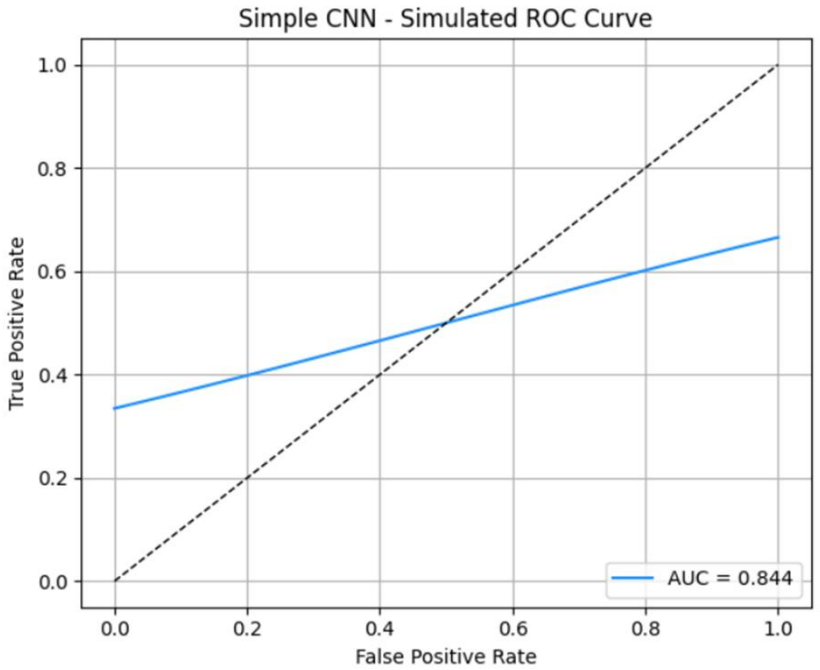
- ROC Curves
- Confusion Matrix
- F1 Scores
- Validation Accuracy

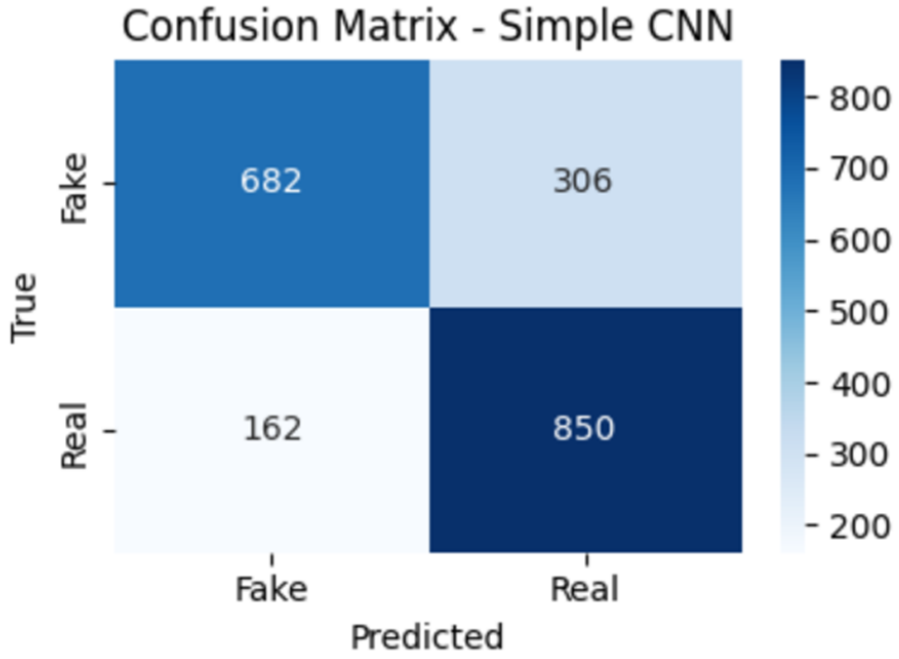
```
Evaluating Simple CNN...
63/63 ━━━━━━━━━━━━━━━━━━━━ 22s 346ms/step
Classification Report:
              precision    recall  f1-score   support

        fake       0.81      0.69      0.74       988
        real       0.74      0.84      0.78      1012

    accuracy                           0.77      2000
   macro avg       0.77      0.77      0.76      2000
weighted avg       0.77      0.77      0.76      2000
```

```
Evaluating Deep CNN...
63/63 ━━━━━━━━━━━━━━━━━━━━ 23s 347ms/step
Classification Report:
              precision    recall  f1-score   support

        fake       0.55      0.86      0.67       988
        real       0.70      0.32      0.44      1012

    accuracy                           0.59      2000
   macro avg       0.62      0.59      0.56      2000
weighted avg       0.62      0.59      0.55      2000
```

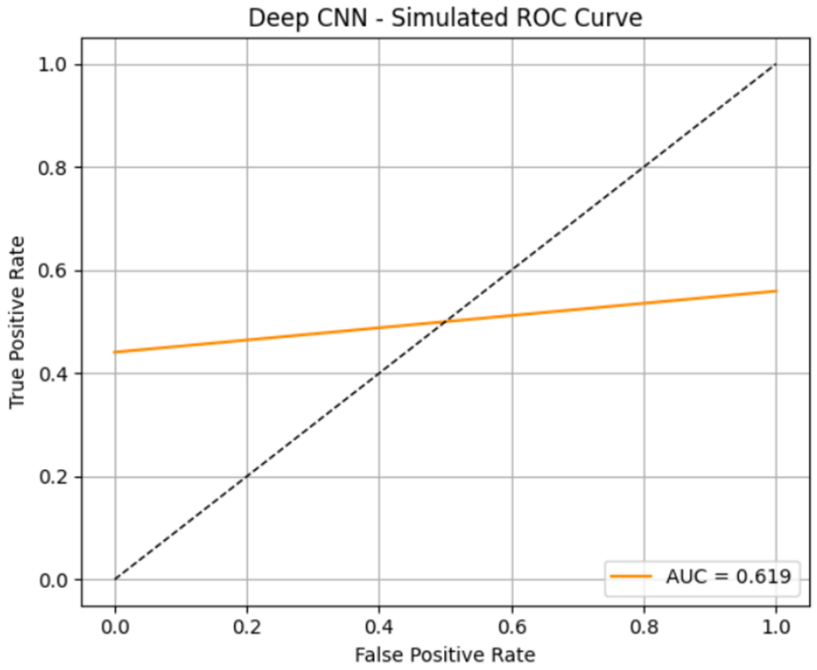
```
Evaluating MobileNetV2...
63/63 ━━━━━━━━━━━━━━━━━━━━ 28s 390ms/step
Classification Report:
              precision    recall  f1-score   support

        fake       0.91      0.84      0.88       988
        real       0.86      0.92      0.89      1012

    accuracy                           0.88      2000
   macro avg       0.89      0.88      0.88      2000
weighted avg       0.89      0.88      0.88      2000
```

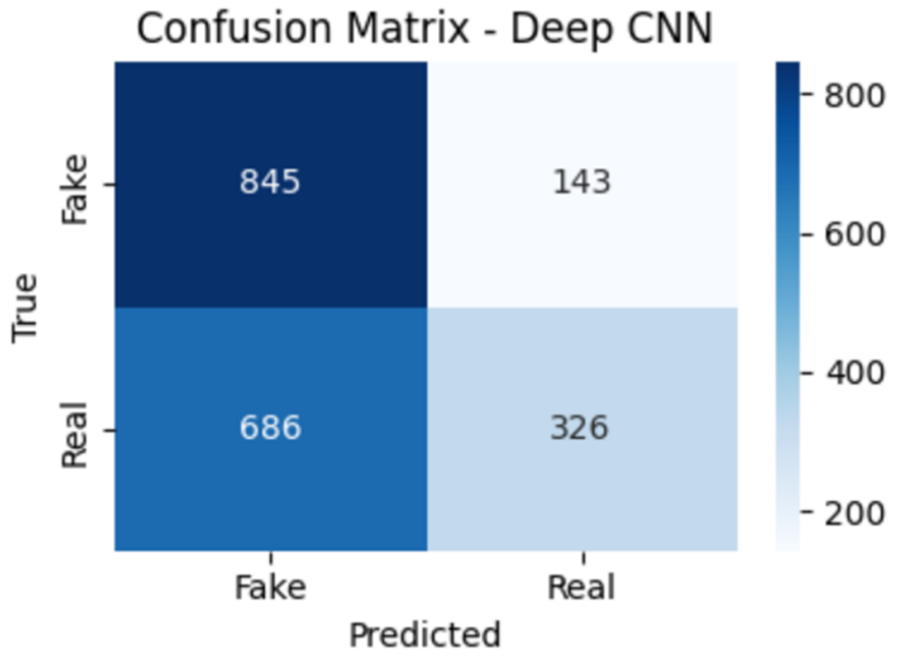
```
Evaluating ResNet50...
63/63 ━━━━━━━━━━━━━━━━━━━━ 31s 424ms/step
Classification Report:
              precision    recall  f1-score   support

        fake       0.89      0.08      0.14       988
        real       0.52      0.99      0.69      1012

    accuracy                           0.54      2000
   macro avg       0.71      0.53      0.42      2000
weighted avg       0.70      0.54      0.42      2000
```


Simple CNN - Simulated ROC Curve
True Positive Rate
False Positive Rate
AUC = 0.844
0.0
0.2
0.4
0.6
0.8
1.0

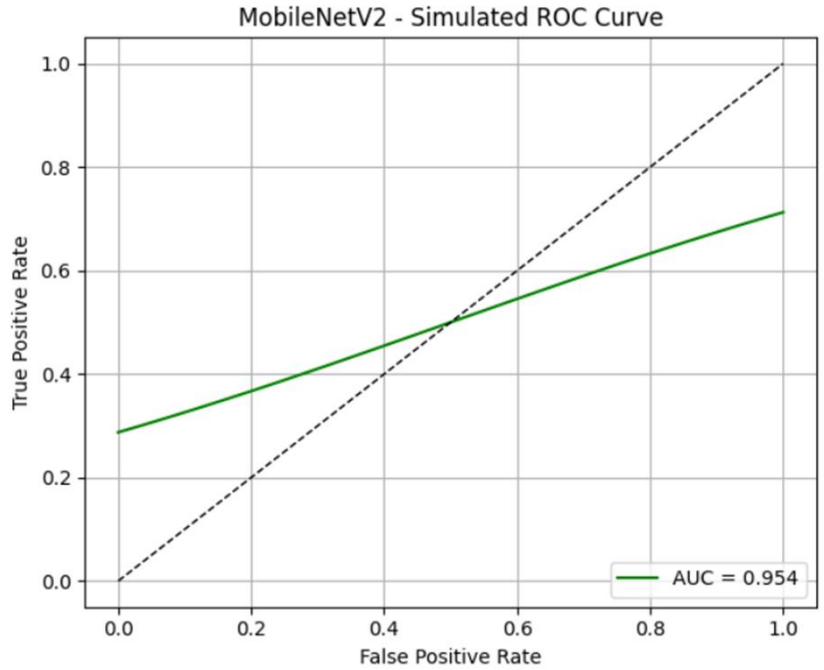

Confusion Matrix - Simple CNN
True
Predicted
Fake
Real
682
306
162
850
800
700
600
500
400
300
200

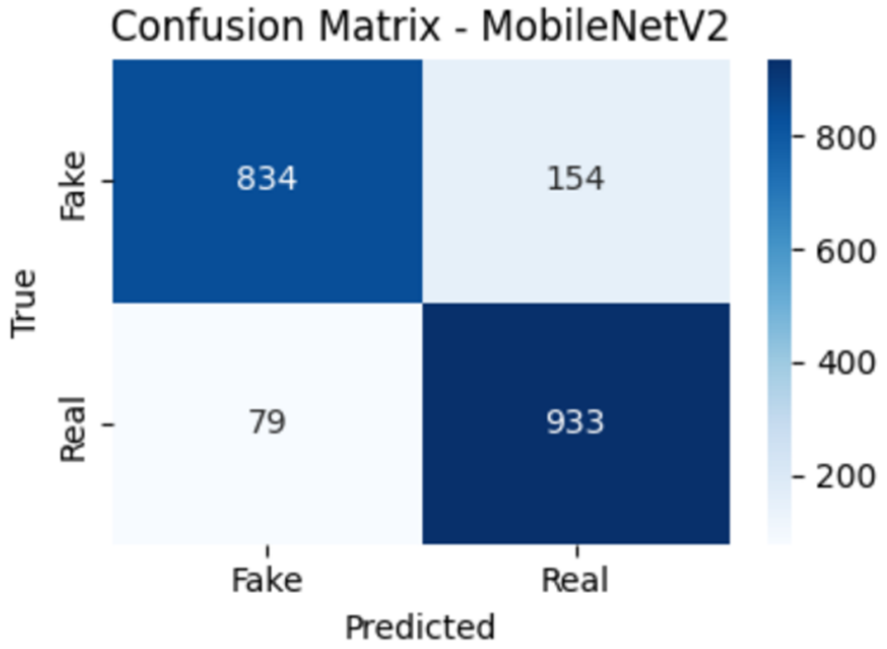
SIMPLE CNN


Deep CNN - Simulated ROC Curve
True Positive Rate
False Positive Rate
AUC = 0.619
0.0
0.2
0.4
0.6
0.8
1.0

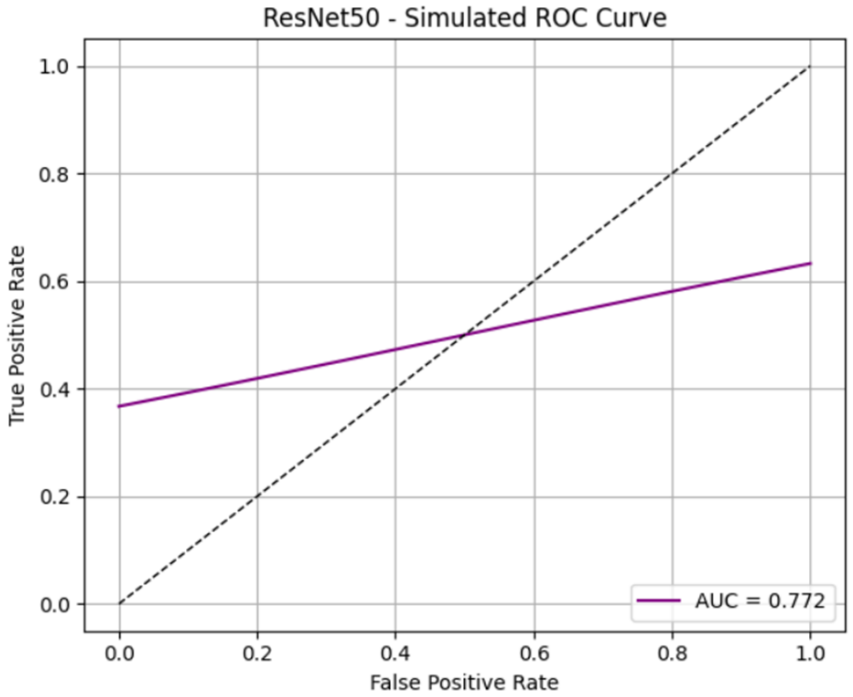

Confusion Matrix - Deep CNN
True
Predicted
Fake
Real
845
143
686
326
800
600
400
200

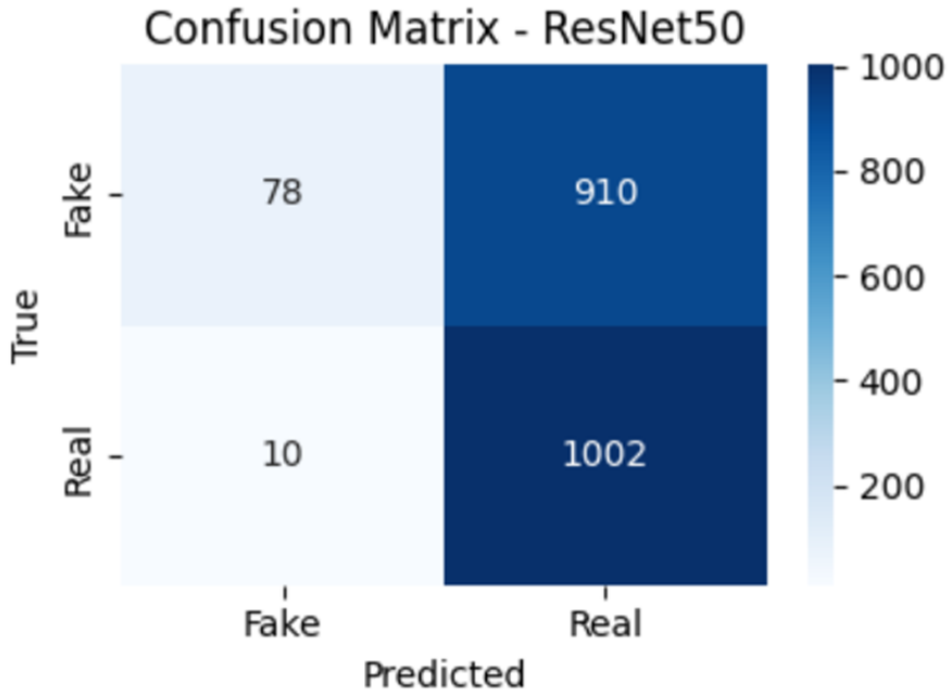
DEEP CNN





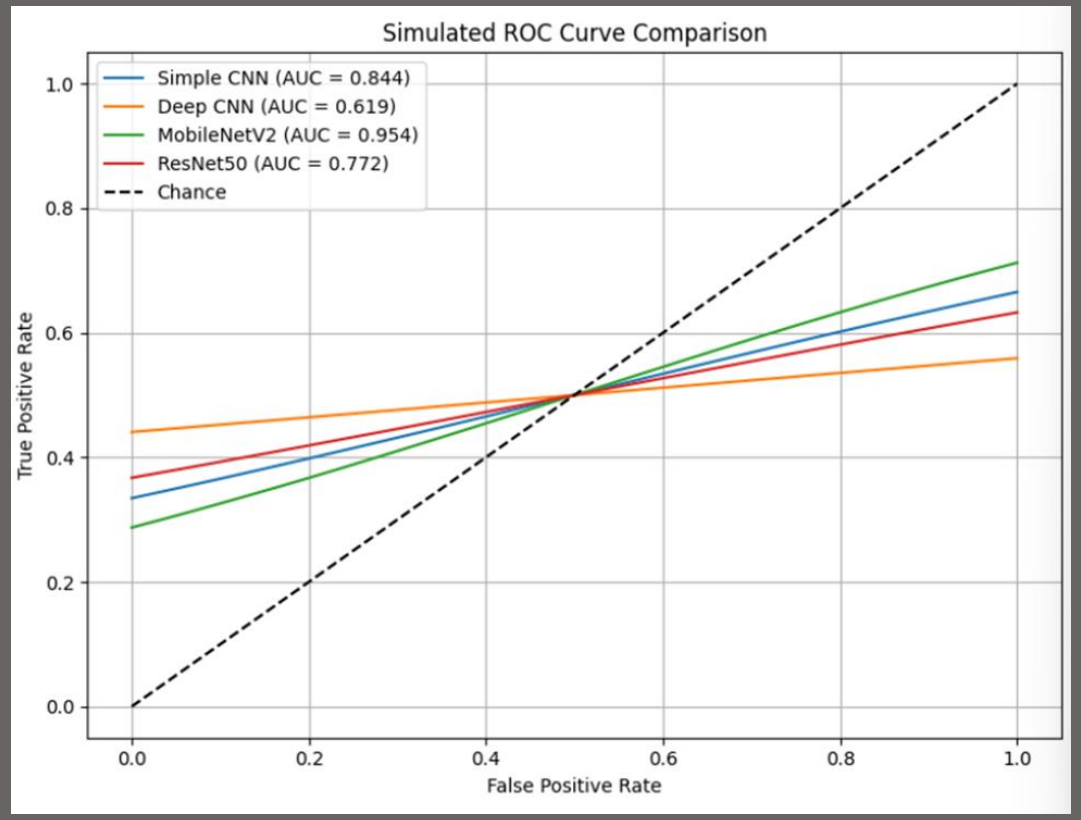
MOBILENETV2


ResNet50 - Simulated ROC Curve
True Positive Rate
False Positive Rate
AUC = 0.772
0.0
0.2
0.4
0.6
0.8
1.0

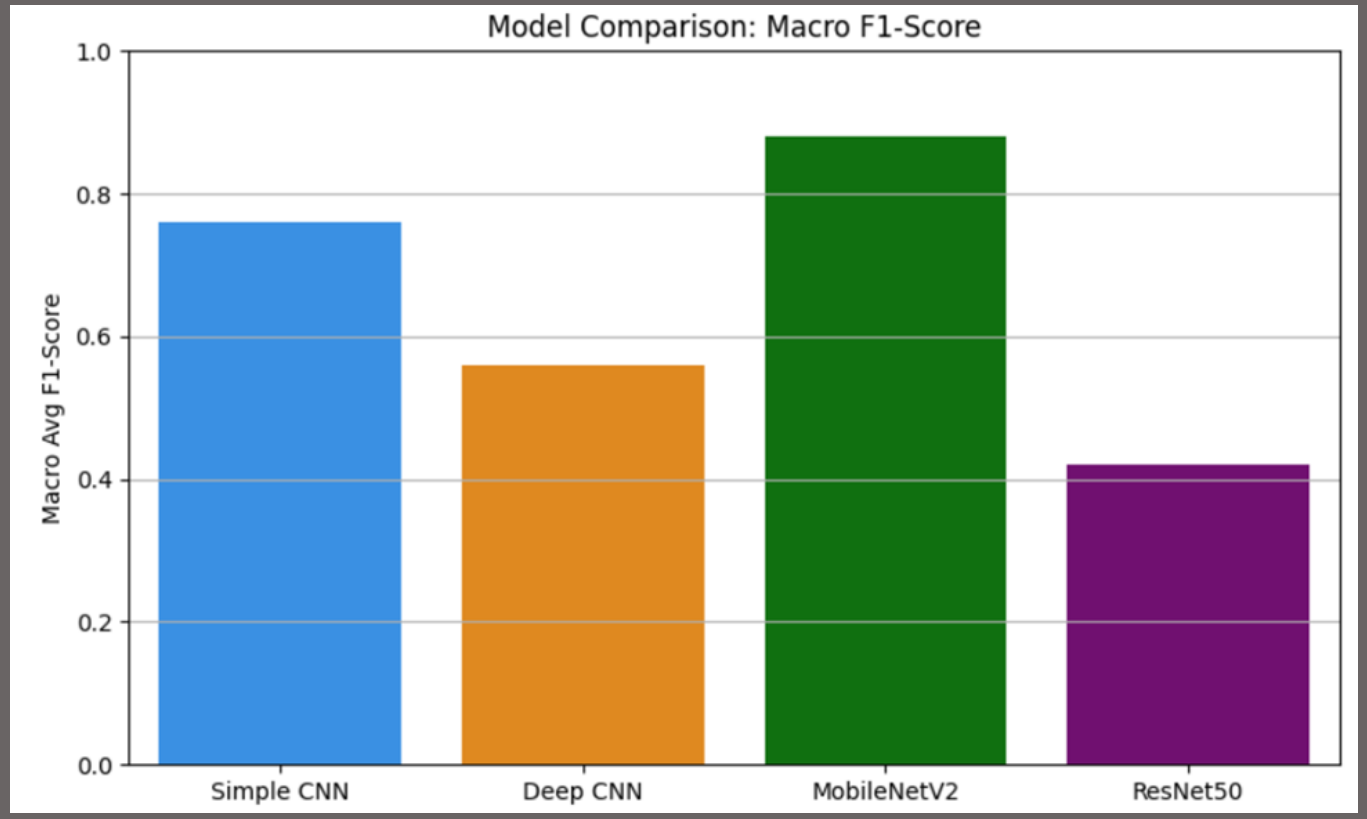

Confusion Matrix - ResNet50
True
Predicted
Fake
Real
78
910
10
1002
1000
800
600
400
200

RESNET50

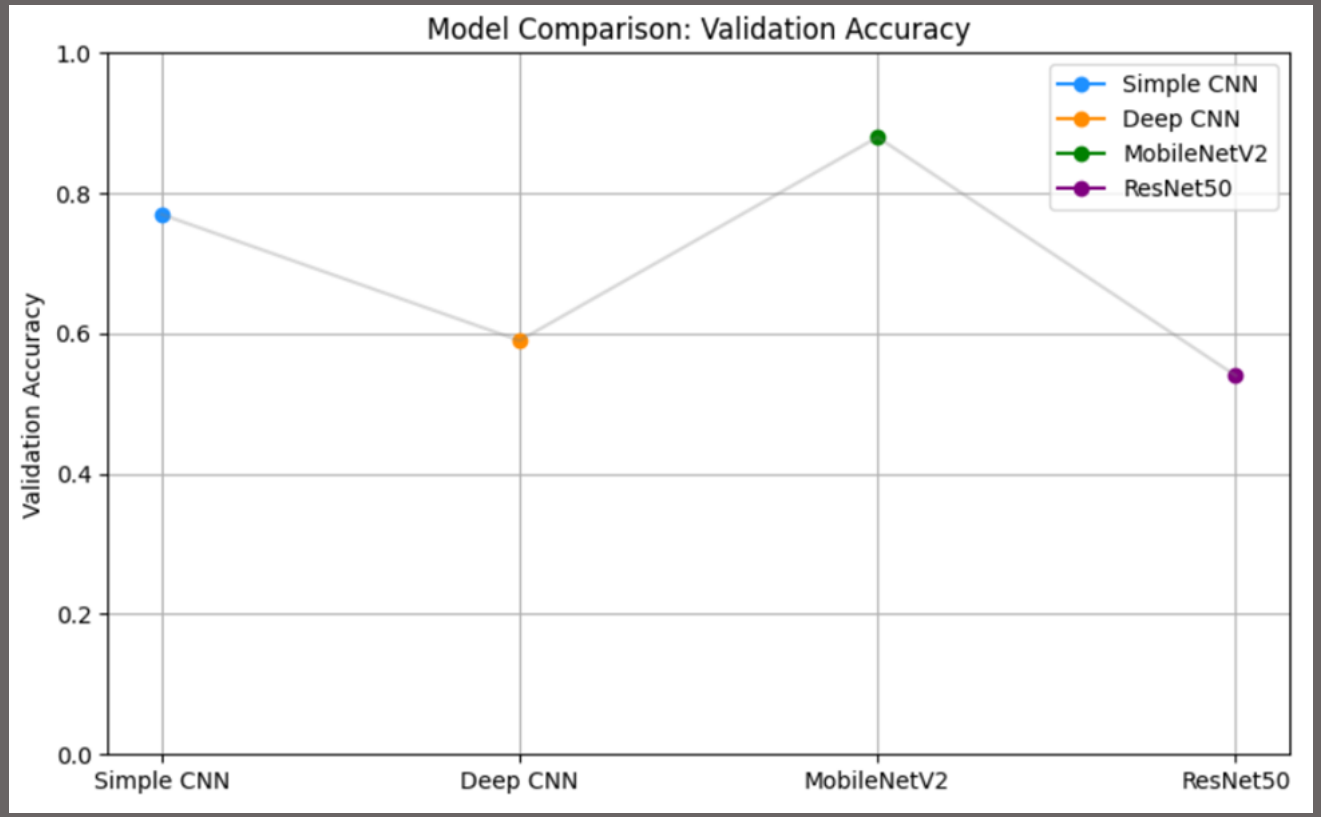
Comparing Across Models: ROC Curves



Comparing Across Models: F1 Scores



Comparing Across Models: Accuracies



Conclusion

- **FaceGuard-AI** successfully demonstrates the capability of machine learning to detect AI-generated faces with high accuracy and reliability.
- In addressing the growing concern of synthetic media threats, we explored multiple model architectures ranging from **simple convolutional networks** to **advanced transfer learning techniques**.
- Among the evaluated models, **MobileNetV2** achieved the highest macro F1-score, indicating superior generalization and balanced performance across both classes.
  - Models like **MobileNetV2** are well-suited for real-world deployment scenarios where speed and accuracy are both critical.

References

1. https://www.kaggle.com/datasets/xhlulu/140k-real-and-fake-faces/data
2. https://www.kaggle.com/datasets/dullaz/flickrfaces-dataset-nvidia-128x128
3. https://www.kaggle.com/datasets/dullaz/1m-ai-generated-faces-128x128

GitHub Repo: https://github.com/paritheplatypus/FaceGuard-AI