Machine Learning for Deepfake Detection

Shannon K. Gallagher, PhD
skgallagher@sei.cmu.edu
Copyright 2022 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.


DM22-0747
We learned why we need detectors, but why do we need machine learning?

Potential Dangers:

- >700k hours of video uploaded to YouTube daily
- Deepfake apps can be run with push of a button
- Deepfakes are generated with ML, logical then to think that we can detect them with ML
- Castle defense

Image from DW.com. The righthand side image is an example of a deepfake used to impersonate the mayor of Kyiv. Brackets are ours.

We need scalable detectors!
First, a crash course in modeling
Problem set-up: Is this a deepfake?

Input

Model/Alg./Blackbox

Output

Yes (96%)
Problem set-up: Now with some math

Model/Alg./Blackbox

Input

X is an image

Output

f(X)

Yes (96%)

Y is between 0 and 1 (0 = real, 1 = fake)
We need to first specify a *model* for \( f(X) \)

Example: Logistic Regression

\[
f(X) = \logit^{-1}(X\beta) = \frac{1}{1 + e^{-X\beta}}
\]

\( X = \) vector of features
\( \beta = \) vector of parameters to learn

This Photo by Unknown Author is licensed under CC BY
Often \( f(X) \) is a neural net we need to learn

\[
f: \mathcal{X} \rightarrow [0,1] \quad X \mapsto f(X; \theta)
\]

\( X \) = image

\( \theta \) = parameters we need to learn
Statistical and ML models let us estimate $\theta$ from data

$\text{Data} = \{(X_i, Y_i)_{i=1\ldots n}\}$
We learn by minimizing a loss function

\[ \hat{Y}_i = f(X_i, \theta) \]

\[ L(\hat{Y}_i, Y_i) = \text{distance between predicted and actual value} \]

\[ \hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{n} L(\hat{Y}_i, Y_i) \]
Our favorite loss is binary cross entropy

$$L(y_i, \hat{y}_i) = -[y_i \cdot \log(\hat{y}_i) + (1 - y_i) \cdot \log(1 - \hat{y}_i)]$$

If $y_i = 1$
- if $\hat{y}_i$ is close to 1 then term is small
- if $\hat{y}_i$ is close to 0 then term is large

If $y_i = 0$
- if $\hat{y}_i$ is close to 1 then term is large
- if $\hat{y}_i$ is close to 0 then term is small

Goal: We want loss to be small
In summary, we need only three things

1. We have a set of labeled data
2. We specify a function class that has parameters we need to learn
3. Using data, we minimize a loss function to estimate parameters in neural net

The devil is in the details
These two are hard problems, but we have lots of help

2. We specify a function class that has parameters we need to learn (e.g. neural net)
3. Using data, we minimize a loss function to estimate parameters in neural net
So the devil is in the data

Problem 1: Overfitting

Solution: Train and Test Data
Train and Test data

*Idea:* Don’t ‘train’ your model on all the data. Leave some for testing.
Problem 2. Affine transformations

**Idea:** a deepfake is still a deepfake if the face is big/large, rotated, upside down, off-center, etc.

**Solution:** Augmented data

- Rotation
- Mirroring
- Scaling
- Translation
- Shearing
- Non-uniform Scaling
Problem 3: Spurious features

Idea: Data are biased. Sometimes the machine finds coincidental features, not real ones.

Solution: standardization and masking
Standardization

- Align and center faces
- Subtract the ‘average’

Takeaway: we want real differences to stand out

AVG-REAL - AVG-FAKE
Masking

Takeaway: we want real differences to stand out
There’s a very clear pattern in deepfake detection

1. Gather labeled data
2. Transform data to emphasize useful features and mitigate biases
3. Train a model on some data
4. Test the model on separate data

We call this process the deepfake detection pipeline
Gather labeled data
Gathering deepfake data is harder than it may seem

- Ethical issues
- Proprietary issues
- Accessibility issues

**Consequence:** There are about a dozen datasets the public effectively uses for deepfake detection
## Popular datasets for deepfake detection

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Format</th>
<th>Labels</th>
<th>Size (GB)</th>
<th>Size (#)</th>
<th>Resolution</th>
<th>GAN?</th>
<th>Gen.</th>
<th>Faces?</th>
<th>Year</th>
<th>Access</th>
<th>Ex.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flick-Faces-HQ</td>
<td>Images</td>
<td>.png and .json</td>
<td>Real</td>
<td>2 TB total, 90 GB for a condensed set</td>
<td>210k files and 76k in condensed yet</td>
<td>Variable</td>
<td>No</td>
<td>Yes</td>
<td>2020</td>
<td>Google Drive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deepfake Detection Challenge (DFDC)</td>
<td>Video</td>
<td>.mp4</td>
<td>Real/Fake</td>
<td>470GB compressed</td>
<td>1080p</td>
<td>yes</td>
<td>1-3</td>
<td>yes</td>
<td>2020</td>
<td>Register on Kaggle</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MetaFaces</td>
<td>Images</td>
<td>.png and .json</td>
<td>Real (paintings of faces)</td>
<td>150GB</td>
<td>2021 files</td>
<td>1024x1024</td>
<td>No</td>
<td>Yes (but paintings)</td>
<td>2020</td>
<td>See here</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeeperForensics</td>
<td>Video</td>
<td>.mp4</td>
<td>Real/Manipulated</td>
<td>300GB</td>
<td>60k videos, 100 individuals</td>
<td>Yes</td>
<td>Yes</td>
<td>2020</td>
<td>Google form/license</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DeepFake Detection Dataset (DFD)</td>
<td>Video</td>
<td>.mp4</td>
<td>Real/Manipulated</td>
<td>~50GB compressed</td>
<td>Variable</td>
<td>yes</td>
<td>2019</td>
<td>Google form</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Flickr-Face HQ (Real portrait photos)
StyleGAN2 (Synthetic individuals, portrait style)
Deepfake Detection Challenge (Real)
DeepFake Detection Challenge (Fake)
Celeb DF v2

Real

Deepfake
Abstracting a video/image into computer representation

- **Inputs** of dimension (W, H, C, F)
  - W = Pixel width
  - H = Pixel height
  - C = Channel (RGB)
  - F = Frame #
Data Transformations
How do we extract useful features?

Thought: Only small sections of images/videos are ‘deepfaked’
Problem: Extract the ‘area’ where we think deepfake will take place

For us this usually translates to extracting faces
Haar cascades

From Ngo et al. (2009)
Edge Detectors

From this canny edge detection article
Entirely separate Neural Nets

Facial boundary detection from MTCNN
Then we can augment the data

From Zeno, Kalinovskiy, and Matveev (2021)
Modeling
Training Models – largely pre-trained Neural Nets

AlexNet  RegNet  ResNet  ResNeXt  ShuffleNet V2  SqueezeNet  SwinTransformer  VGG  VisionTransformer  Wide ResNet

ConvNeXt
DenseNet
EfficientNet
EfficientNetV2
GoogLeNet
Inception V3
MNASNet
MobileNet V2
MobileNet V3
Testing/Evaluation
Prototype results: data bias makes generalization hard

### Accuracy (%) of fine-tuned ResNet

<table>
<thead>
<tr>
<th>Trained on</th>
<th>Data Set</th>
<th>Celeb DF v1</th>
<th>Stylegan2</th>
<th>Stylegan3-t</th>
<th>Stylegan3-r</th>
<th>DFDC Pt. 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Celeb DF v1</td>
<td>99.1</td>
<td>44.2</td>
<td>44.2</td>
<td>44.0</td>
<td>51.2</td>
<td></td>
</tr>
<tr>
<td>Stylegan2</td>
<td>24.1</td>
<td>98.7</td>
<td>52.9</td>
<td>48.4</td>
<td>57.4</td>
<td></td>
</tr>
<tr>
<td>Stylegan3-t</td>
<td>16.7</td>
<td>69.7</td>
<td>96.7</td>
<td>84.0</td>
<td>7.0</td>
<td></td>
</tr>
<tr>
<td>Stylegan3-r</td>
<td>16.9</td>
<td>68.0</td>
<td>89.0</td>
<td>97.2</td>
<td>7.0</td>
<td></td>
</tr>
<tr>
<td>DFDC Pt. 0</td>
<td>68.1</td>
<td>57.4</td>
<td>57.5</td>
<td>57.5</td>
<td>88.7</td>
<td></td>
</tr>
</tbody>
</table>
Testing: How do the best models do??

Great**

**In controlled scenarios
In closing

• Deepfake detection can be completely adapted to the ML modeling framework
• In theory, deepfake detection is a simple four step process
  • Data collection
  • Data transformation
  • Modeling
  • Evaluation
• But the devil is in the details

And Dr. Bernaciak will show you exactly how!
The GAN problem

Red makes a generator to create deepfakes
Blue makes a detector
Red uses results from blue’s detector to make generator better
Blue uses new red images to improve detector

...
GAN set-up

\[ X' = G(Z) = \text{generator fake image} \]
\[ X = \text{real image} \]
\[ D(X) = \text{discriminator in } [0,1] \]
\[ Y = 0, Y' = 1 \text{ (0 is real, 1 is fake)} \]

Data = \{(X_i, 0)_{i=1}^{m}, (X'_j, 1)_{j=1}^{n}\}
\[ L(x, y) = \text{loss function} \]

Fig. 18.1.1 Generative Adversarial Networks

Fig. from d2l.ai
GAN Set-up

Round 0: Generator introduces fakes

Round 1:

Discriminator turn: Use generated data to get best discriminator

$$\hat{D}^1|\hat{G}^0 = \arg\min_D \sum_{i=1}^{m} L(D(X_i), 0) + \sum_{j=1}^{n} L(D(X'_j), 1)$$

Generator turn: Directly try to deceive discriminator

$$\hat{G}^1|\hat{D}^1 = \arg\min_G \sum_{j=1}^{n} L(\hat{D}^1(X'_j), 0)$$

$$= \arg\min_G \sum_{j=1}^{n} L(\hat{D}^1(G(Z_j)), 0)$$

Repeat