Machine Learning for Deepfake Detection

Shannon K. Gallagher, PhD

skgallagher@sei.cmu.edu

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

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.

DM22-0747

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

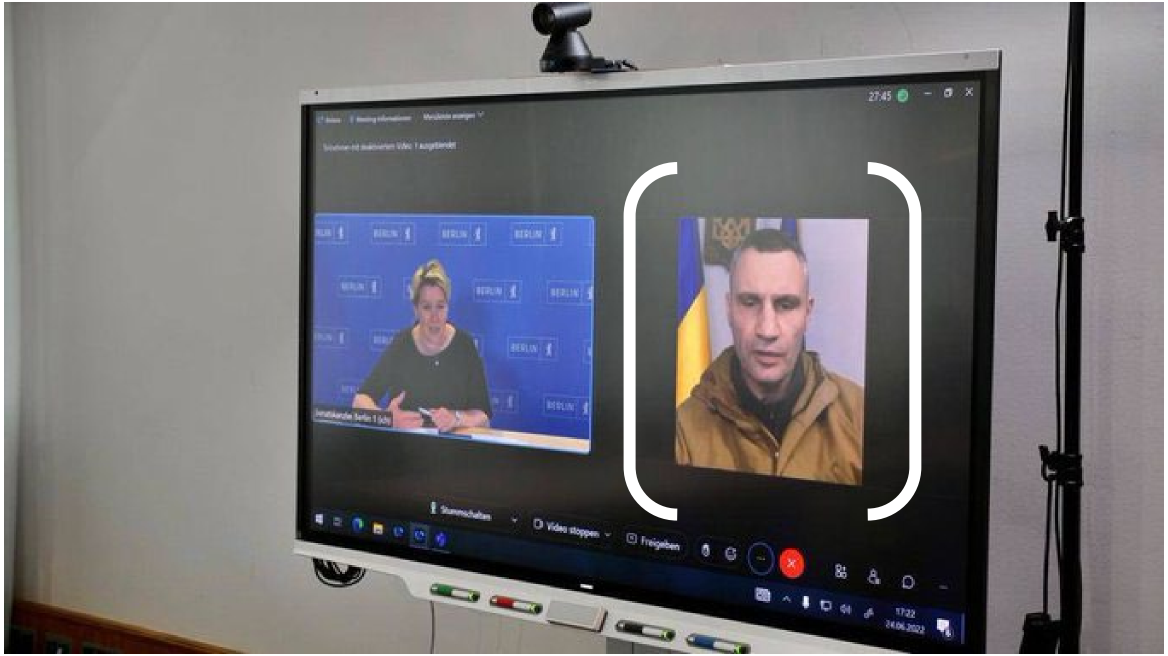


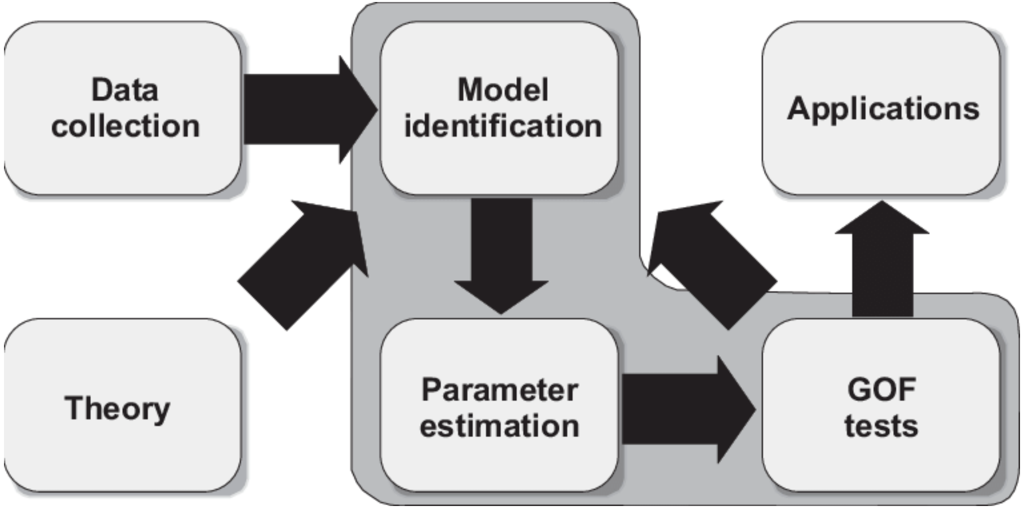
Image from DW.com. The righthand side image is an example of a deepfake used to impersonate the mayor of Kyiv. Brackets are ours.

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

We need *scalable* detectors!

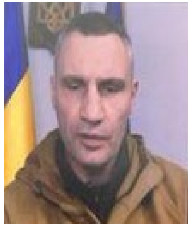
First, a crash course in modeling



Problem set-up: Is this a deepfake?

Model/Alg./Blackbox

Input



????

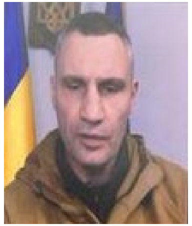
Output

Yes
(96%)

Problem set-up: Now with some math

Model/Alg./Blackbox

Input



X is an image

f(X)

Output

Yes (96%)

Y is between 0 and 1
(0 = real, 1 = fake)

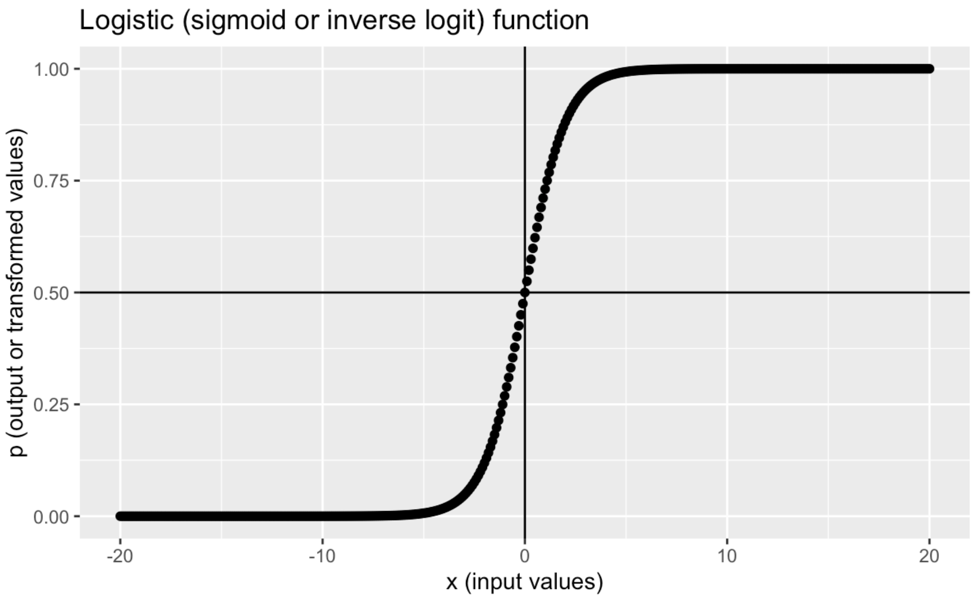
We need to first specify a *model* for f(X)

Example: Logistic Regression

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

X = vector of features

$\boldsymbol{\beta}$ = vector of parameters to learn

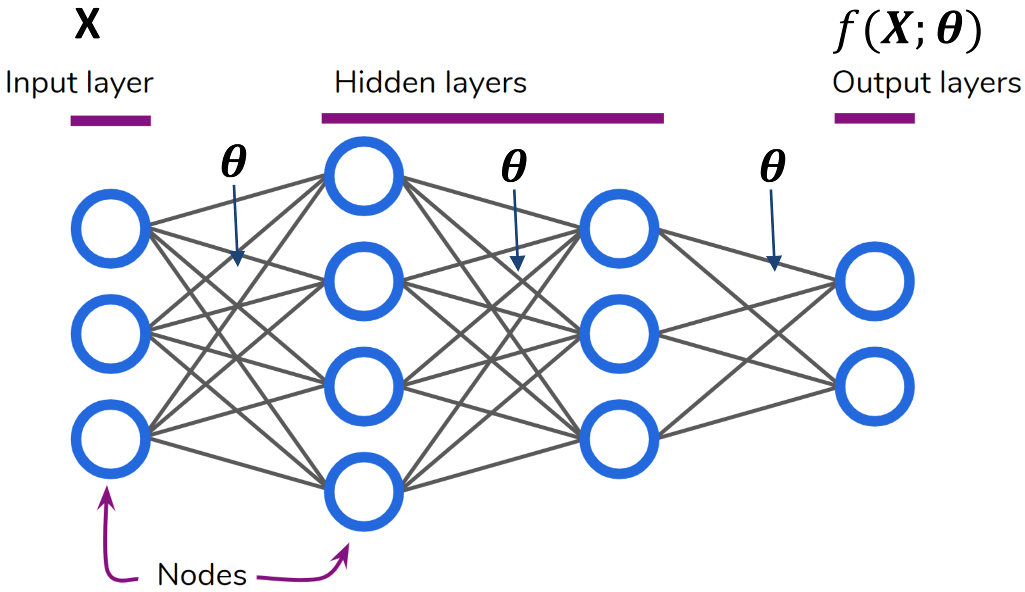


Often f(X) is a neural net we need to learn

$$f: \mathcal{X} \to [0,1]$$
$$\boldsymbol{X} \mapsto f(\boldsymbol{X}; \boldsymbol{\theta})$$

X = image

$\boldsymbol{\theta}$ = parameters we need to *learn*



Statistical and ML models let us estimate $\boldsymbol{\theta}$ from **data**

Data $=\{(X_i, Y_i)_{i=1\ldots n}\}$

Data =

, FAKE

, REAL

, REAL

, FAKE

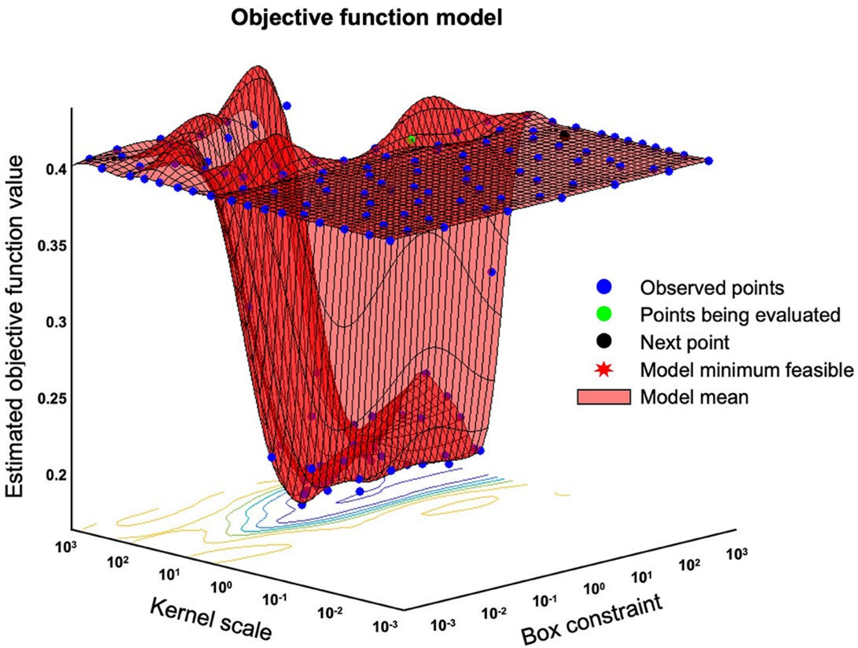
Contact: skgallagher@sei.cmu.edu

We learn by minimizing a loss function

$\hat{Y}_i = f(\boldsymbol{X_i}, \boldsymbol{\theta})$

$L(\hat{Y}_i, Y_i)$ = distance between *predicted* and actual value

$$\hat{\boldsymbol{\theta}} = argmin_{\boldsymbol{\theta}} \sum_{i=1\ldots n} L\left(\hat{Y}_i, Y_i\right)$$



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

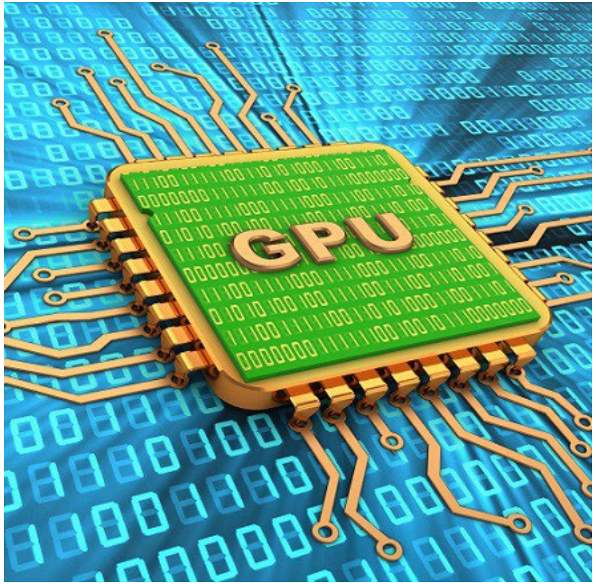
The devil is in the details

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

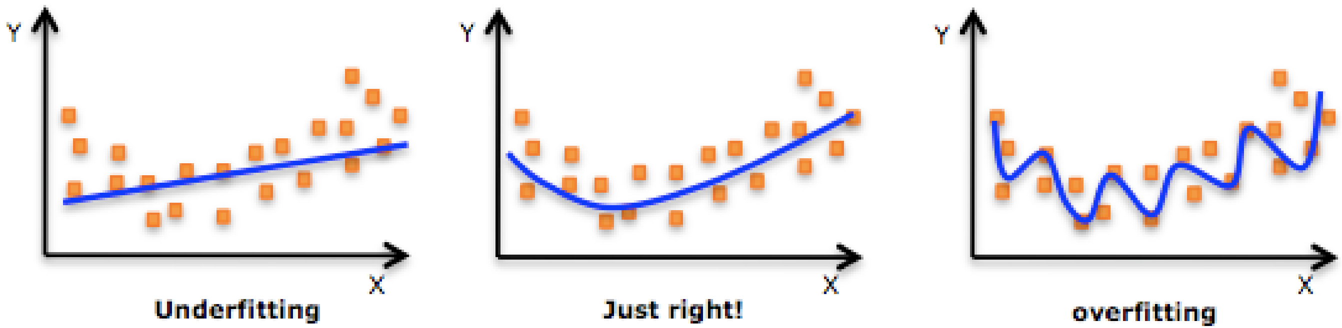
PyTorch



Contact: skgallagher@sei.cmu.edu

So the devil is in the data

Problem 1: Overfitting



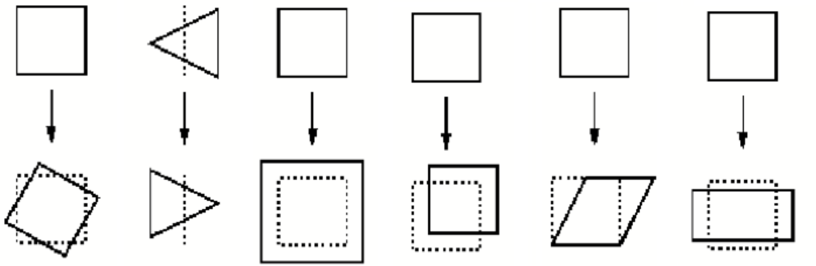
Solution: Train and Test Data

Train and Test data

Idea: Don't 'train' your model on all the data. Leave some for testing.

Problem 2. Affine transformations

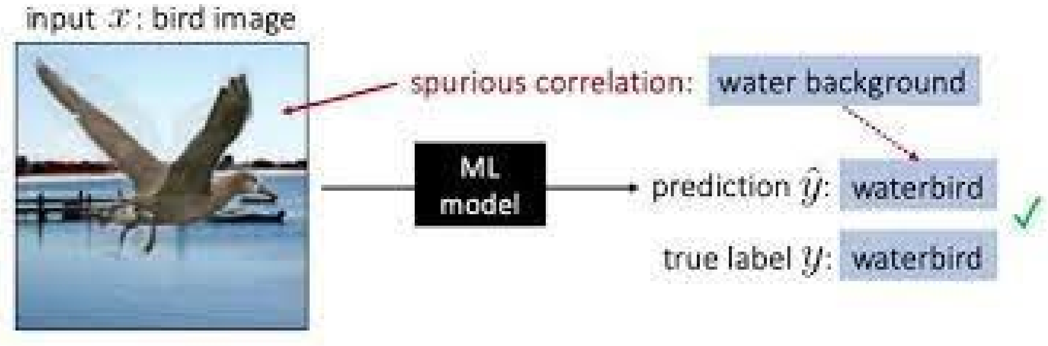
Idea: a deepfake is still a deepfake if the face is big/large, rotated, upside down, off-center, etc.



Solution: Augmented data

Problem 3: Spurious features

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

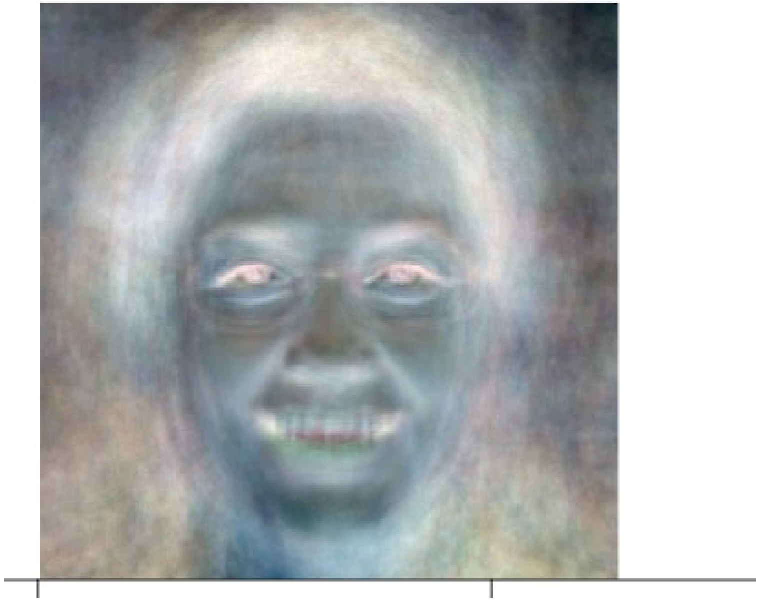


Solution: standardization and masking

Standardization

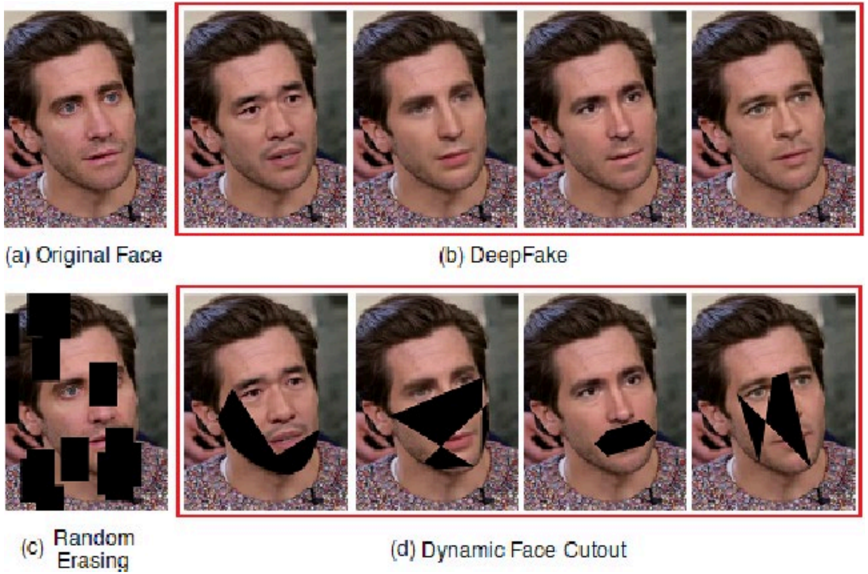
- Align and center faces
- Subtract the 'average'

AVG(REAL) - AVG(FAKE)



Takeaway: we want *real* differences to stand out

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

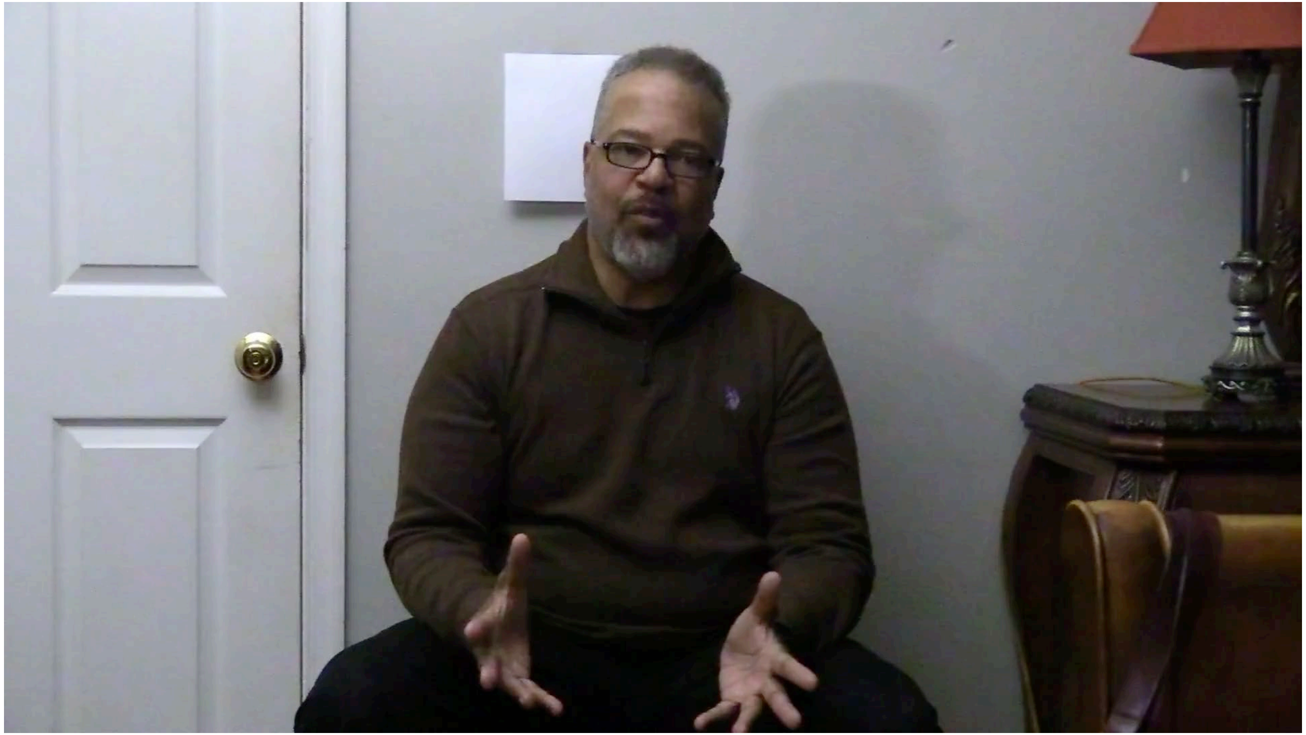
Popular datasets for deepfake detection

Name	Type	Format	Labels	Size (GB)	Size (#)	Resolution	GAN?	Gen.	Faces?	Year	Access	Ex.
have better provenance)												
Flickr-Faces-HQ	Images	.png and .json	Real	2 TB total, 90 GB for a condensed set	210k files and 70k in condensed yet	Variable	No		Yes	2020	Google Drive	
Deepfake Detection Challenge (DFDC)	Video	.mp4	Real/Fake	470GB compressed	100k+ 10s clips 3426 unique actors	1080p (mostly)	yes	1-3	yes	2020	Register on Kaggle	ahfazfbntc.mp4
MetFaces	Images	.png and .json	Real (paintings of faces)	15GB	2621 files	1024x1024	No		Yes (but paintings)	2020	See here	
DeeperForensics	Video	.mp4	Real/Manipulated	300GB	60k videos, 100 individuals		Yes		Yes	2020	Google form/license	
DeepFake Detection Dataset (DFD) **Note: The data is Face Forensics++**	Video	.mp4	Real/Manipulated	~50GB compressed ~2 TB raw	363 original videos and 3068 manipulated	Variable	yes		yes	2019	Google form	

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 #

Contact: skgallagher@sei.cmu.edu

Data Transformations

Contact: skgallagher@sei.cmu.edu

How do we extract useful features?

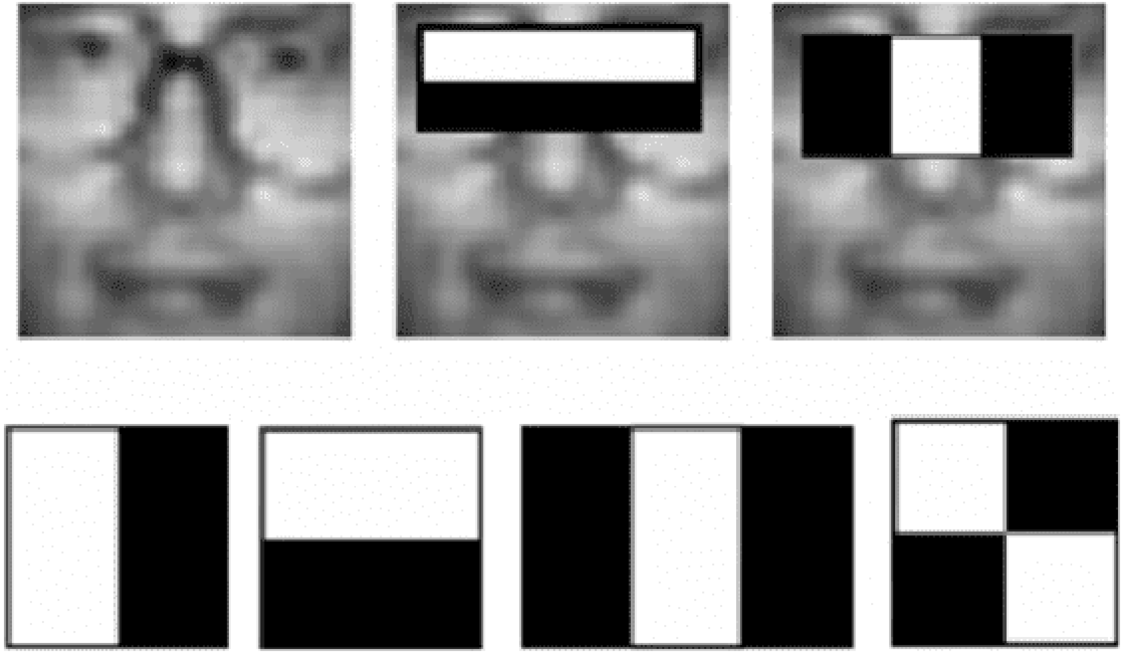
Thought: Only small sections of images/videos are ‘deepfaked’

Problem: Extract the ‘area’ where we think deepfake will take place

For us this usually translates to extracting faces

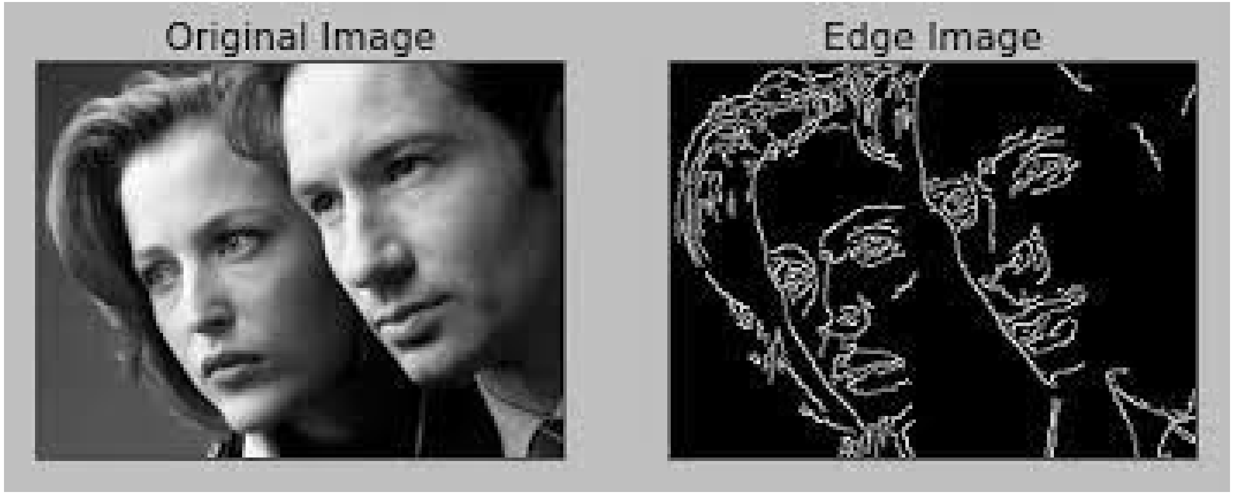
Contact: skgallagher@sei.cmu.edu

Haar cascades



From Ngo et al. (2009)

Edge Detectors



From this [canny edge detection article]

Entirely separate Neural Nets

Facial boundary detection from MTCNN

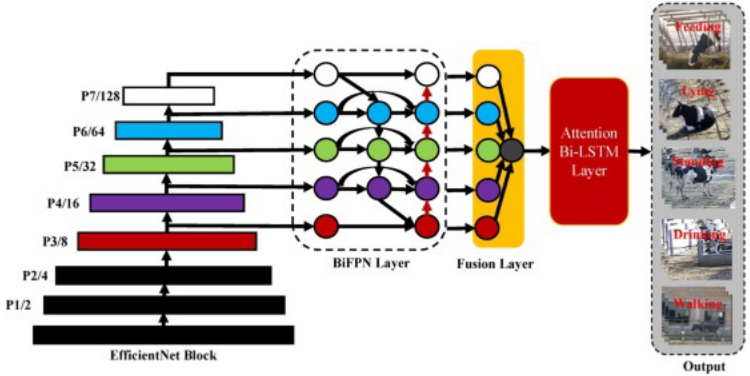
Then we can augment the data

From Zeno, Kalinovskiy, and Matveev (2021)

Modeling

Training Models – largely pre-trained Neural Nets

AlexNet
ConvNeXt
DenseNet
EfficientNet
EfficientNetV2
GoogLeNet
Inception V3
MNASNet
MobileNet V2
MobileNet V3
RegNet
ResNet
ResNeXt
ShuffleNet V2
SqueezeNet
SwinTransformer
VGG
VisionTransformer
Wide ResNet



Testing/Evaluation

Prototype results: data bias makes generalization hard

Accuracy (%) of fine-tuned ResNet

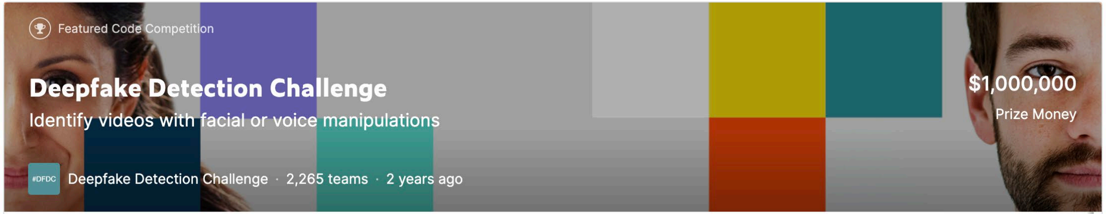
Tested on

Trained on

Data Set	Celeb DF v1	Stylegan2	Stylegan3-t	Stylegan3-r	DFDC Pt. 0
Celeb DF v1	99.1	44.2	44.2	44.0	51.2
Stylegan2	24.1	98.7	52.9	48.4	57.4
Stylegan3-t	16.7	69.7	96.7	84.0	7.0
Stylegan3-r	16.9	68.0	89.0	97.2	7.0
DFDC Pt. 0	68.1	57.4	57.5	57.5	88.7

Testing: How do the best models do??

Great**



Overview Data Code Discussion Leaderboard Rules

Leaderboard

Raw Data Refresh

Search leaderboard

This competition is closed for submissions. The Private Leaderboard was based on a re-run of participants' code by the host on a privately-held test set.
This competition has completed. This leaderboard reflects the final standings.

Prize Winners

#	Team	Members	Score	Entries	Last	Code
1	Selim Seferbekov		0.42798	2	2Y	
2	\WM/		0.42842	2	2Y	
3	NtechLab		0.43452	2	2Y	

**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

…

Who wins?

GAN set-up

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

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

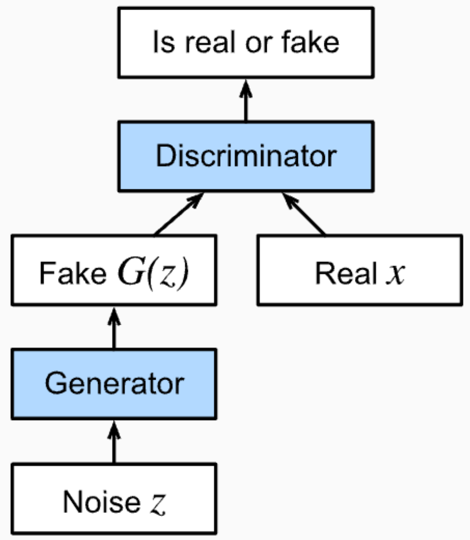


Fig. 18.1.1 Generative Adversarial Networks

Fig. from d2l.ai

GAN Set-up

Round 0: Generator introduces fakes

Round 1:

Discriminator turn: Use generated data to get best discriminator

$$\widehat{D}^1 | \widehat{G^0} = argmin_D \sum_{\{i=1\}}^{m} L(D(X_i), 0) + \sum_{\{j=1\}}^{n} L(D(X_j'), 1)$$

Generator turn: Directly try to deceive discriminator

$$\hat{G}^1 | \widehat{D^1} = argmin_G \sum_{\{j=1\}}^{n} L(\widehat{D}^1(X_j'), 0)$$

$$= argmin_G \sum_{\{j=1\}}^{n} L(\widehat{D}^1(G(Z_j)), 0)$$

Repeat