EasyDeep: An IoT Friendly Robust Detection Method for GAN Generated Deepfake Images in Social Media

Presenter: Alakananda Mitra

A. Mitra[1], S. P. Mohanty[2], P. Corcoran[3], and E. Kougianos[4]

University of North Texas, Denton, TX , USA.[1,2,4] and

National University of Ireland, Galway, Ireland[3].

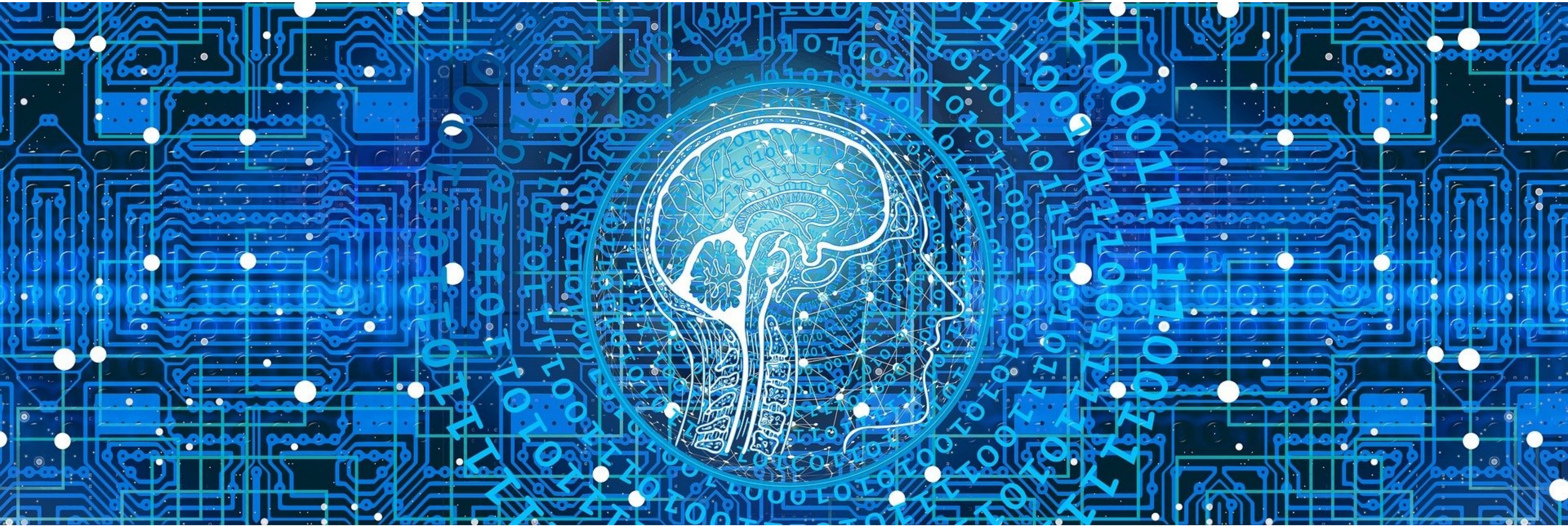
Email: alakanandamitra@my.unt.edu[1], saraju.mohanty@unt.edu[2], peter.corcoran@nuigalway.ie[3], and elias.kougianos@unt.edu[4]

Outline


image: Freepik.com

- Deepfake Image Detection in IoT
- Deepfake Image Creation
- Analysis of GAN Generated Deepfake Images
- EasyDeep Implementation at Edge Computing Platform
- Conclusions & Future Work

Deepfake Image



- **Deepfake Image Detection & IoT**
- **How is Deepfake Image created?**

Deepfake Image Detection in IoT

- **Deepfake = Deep Learning + Fake**

- Sophisticated Images
  - **Generative Adversarial Networks (GANs)**

- Threat to individual's identity, reputation & national security.

- Smart Phones & Handheld Devices
  - Checking Social Media Anywhere & Anytime
  - Spread of Misinformation

Solution

Check Authenticity of Image/Video Anywhere & Anytime

Deepfake Detection System at Edge

How Is Deepfake Image Created?

- Generative Models in Supervised Way.
- Made with CNN.
  - Convolutional Layer
  - Pooling Layer
  - FC Layer
- GANs Used
  - CycleGAN
  - StarGAN

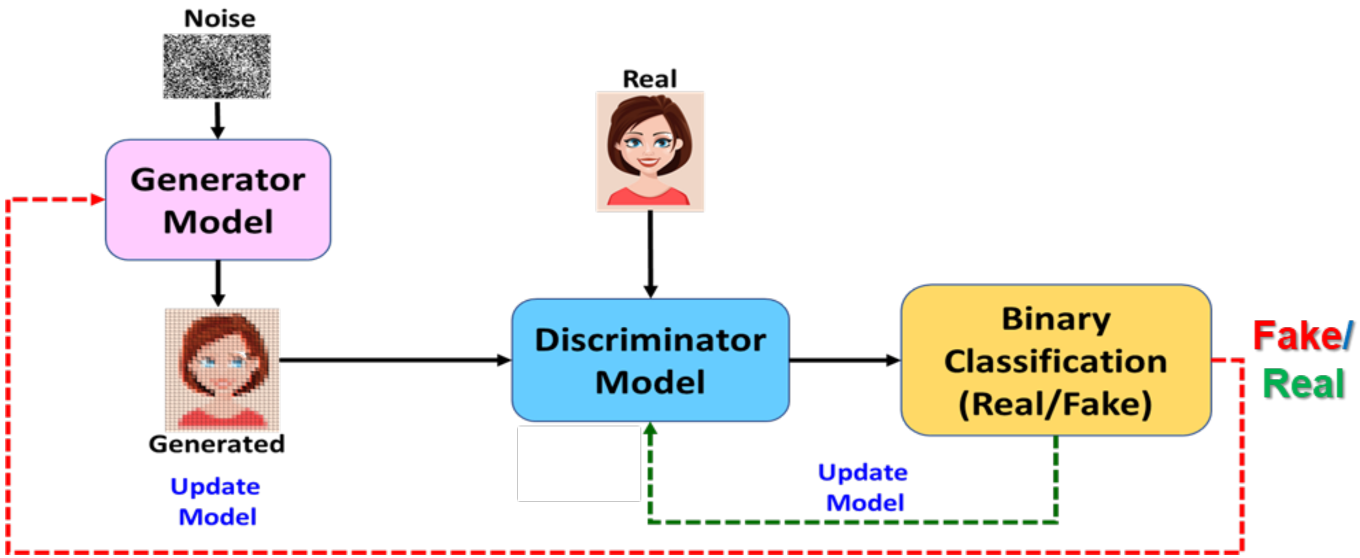


Image-To-Image Translation

Image of Domain 1 → Image of Domain 2

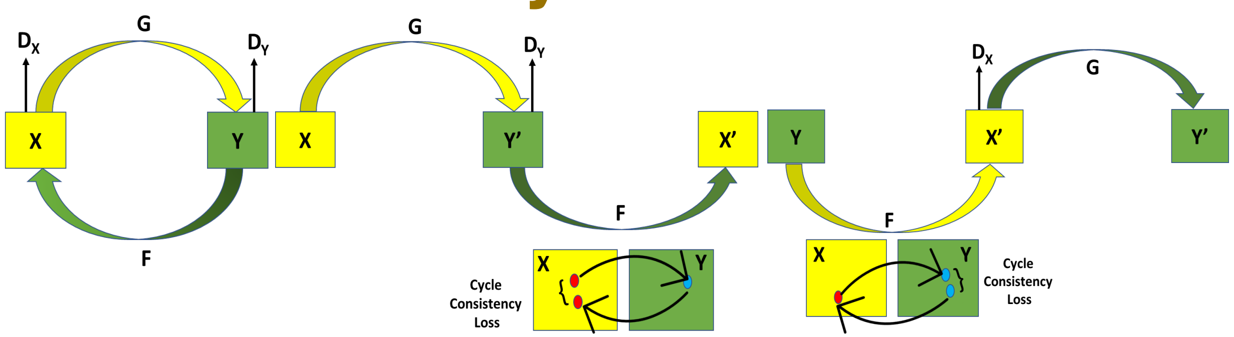
Analysis of GAN Generated Deepfake Images

- Image-To-Image Translation

GANs For Image-To-Image Translation

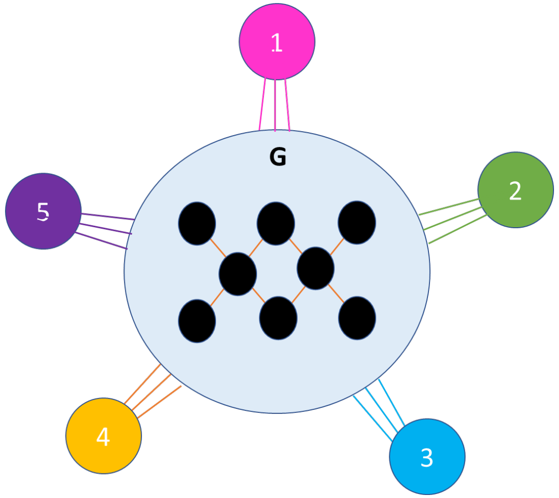
- Generator for translation from X to Y and Generator for reconstructing X given Y.
- Two Discriminator models
- No paired image data.

Cycle GAN



- Unified GAN
- Multi Domain Image to Image Transfer (Photo → Gogh, Monet, Ukiyoe)
- Simultaneous Training of Different Datasets of Different Domains.

Star GAN

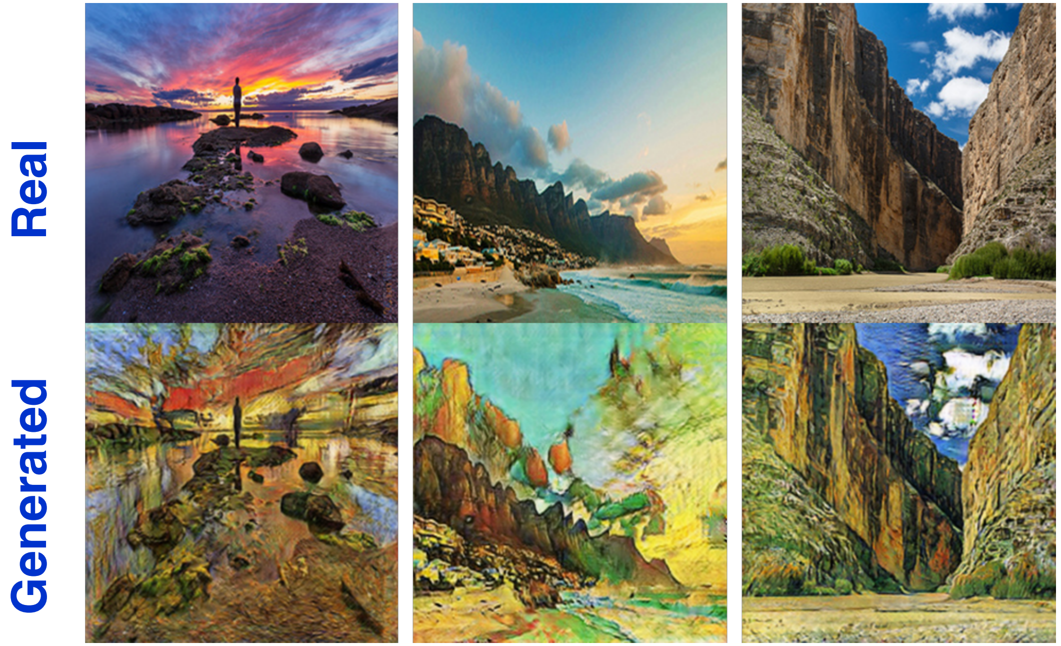


Implementation Details

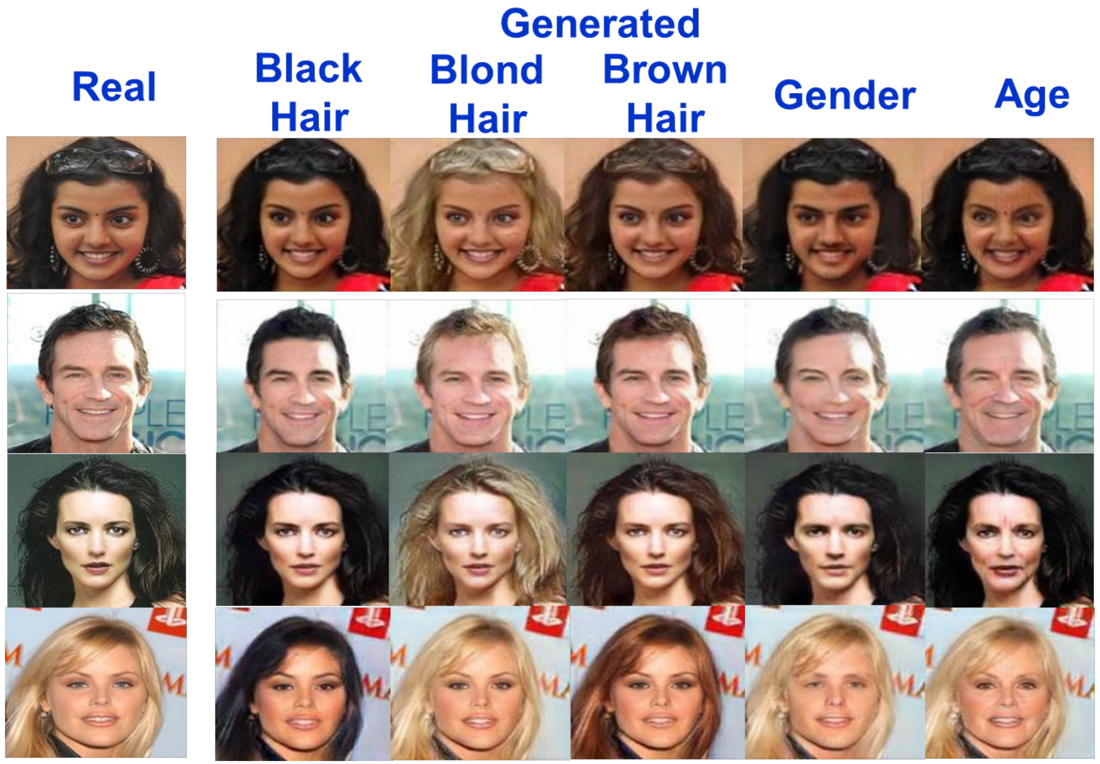
Packages	Version
Python	3.7.7
Torch	1.7.1+cu101
Torchvision	0.8.2+cu101
TensorFlow-gpu	1.13.1
Operating System	Linux

Hardware	Details/ Version
NVIDIA GPU Processor	GP100
GPU Processor Architecture	Pascal
GPU Generation	Tesla
Bus Type	PCIe
cuda	11.2
CPU	2-coreXeon 2.2GHz
Memory	13G
Disc Space	34G

GANs Generated Images



CycleGAN



StarGAN

GANs Generated Data

CycleGAN

	apple2 orange	horse2 zebra	monet2 photo	vangogh2 photo	winter2 summer	ukiyoe	cezanne
Real	2237	2349	671	1738	194	295	343
Generated	2239	2348	672	1736	193	294	342

StarGAN

Class	Number of Images	Source
Real	6,000	CelebA
Generated	30,000	StarGAN

Total

Type of GAN	# of Real Images	# of Generated Images
StarGAN	6000	30000
CycleGAN	9812	9809

Analysis of GAN Generated Images

- Texture Analysis
- Shannon's Entropy
- Haralick's Texture Features from GLCM
  - Contrast
  - Homogeneity
  - Dissimilarity
  - Correlation

$$E = -\sum_{i=0}^{n-1} p_i \log_b p_i,$$

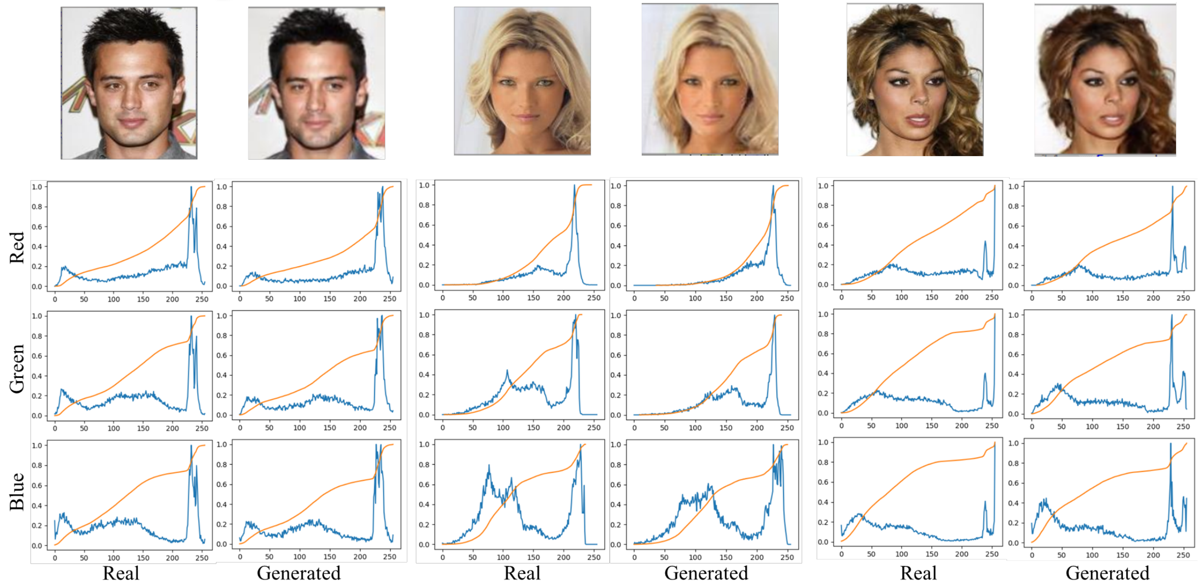
$$CON = \sum_{i,j=0}^{n-1} p(i,j)(i-j)^2$$

$$HOM = \sum_{i,j=0}^{n-1} \frac{p(i,j)}{(1+(i-j)^2)}$$

$$DIS = \sum_{i,j=0}^{n-1} p(i,j)|i-j|$$

$$COR = \sum_{i,j=0}^{n-1} p(i,j)\left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}}\right]$$

Results - Analysis of GAN Generated Images



GAN	Data	ΔE_1
Cycle GAN	apple2orange	1.8177
	horse2zebra	0.7999
	monet	0.5779
	vangogh	0.5779
	ukiyoe	0.4335
	facades	1.448
	cezanne	0.6253
StarGAN		0.4579

GAN	Data	Contrast		Dissimilarity		Homogeneity		Correlation	
		Δ_{min}	Δ_{max}	Δ_{min}	Δ_{max}	Δ_{min}	Δ_{max}	Δ_{min}	Δ_{max}
Cycle GAN	apple2orange	0.1759	4067.51	0.0025	19.2035	0.6417	4.0771	0.6303	3.3104
	horse2zebra	0.0089	16649.09	0.0002	65.8922	0.00001	0.3731	0.8593	6.9169
	monet	0.9867	2559.67	0.0026	18.8634	0.00001	0.4919	0.0004	0.3023
	vangogh	0.6193	2532.19	0.0025	27.3638	0.0002	0.6615	0.0004	0.6378
	ukiyoe	1.2918	2343.93	0.03824	23.4140	0.00005	0.4478	0.0003	0.4959
	facades	1.2550	1982.35	0.0061	21.2044	0.0014	0.6379	0.00005	0.3006
	cezanne	1.0449	1964.87	0.0071	19.0935	0.00003	0.4021	0.0016	0.4070
	Average	**0.7689**	**4585.66**	**0.0085**	**27.8621**	**0.0919**	**1.0131**	**0.2132**	**1.7673**
StarGAN		0.0003	3175.85	0.0026	24.2543	0.0001	0.4029	0.0001	0.3212

$\Delta \rightarrow$(Difference of a texture property)

Observations From Analysis

- StarGAN generates more robust (less varied entropy than real images) fake images than CycleGAN.
- The average difference of contrast, dissimilarity, correlation, and homogeneity are much larger in CycleGAN than StarGAN.
- StarGAN generated images have varied texture features than real images too.
- GAN generated images vary in colors from the real images.

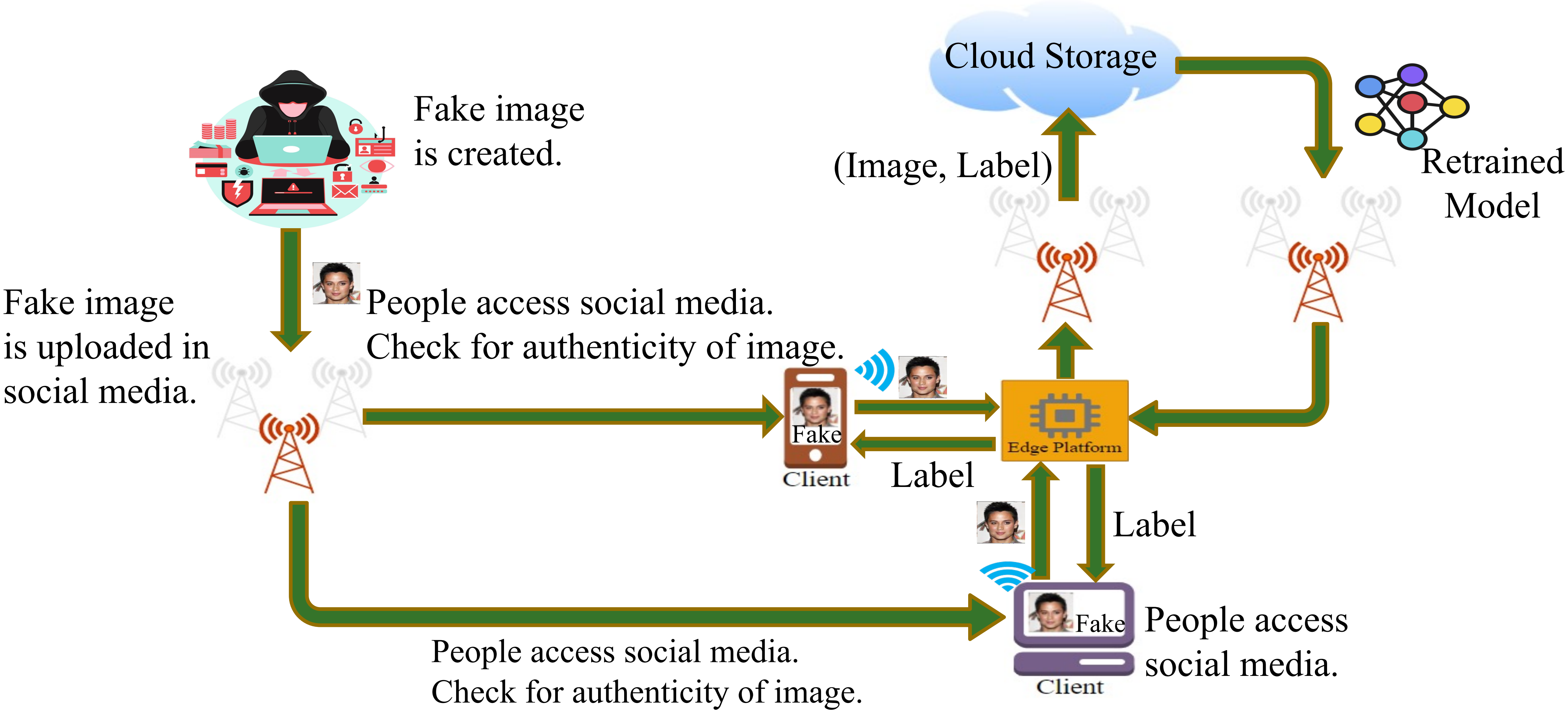
EasyDeep

- GAN Generated Deepfake Image Detection at IoT Platform

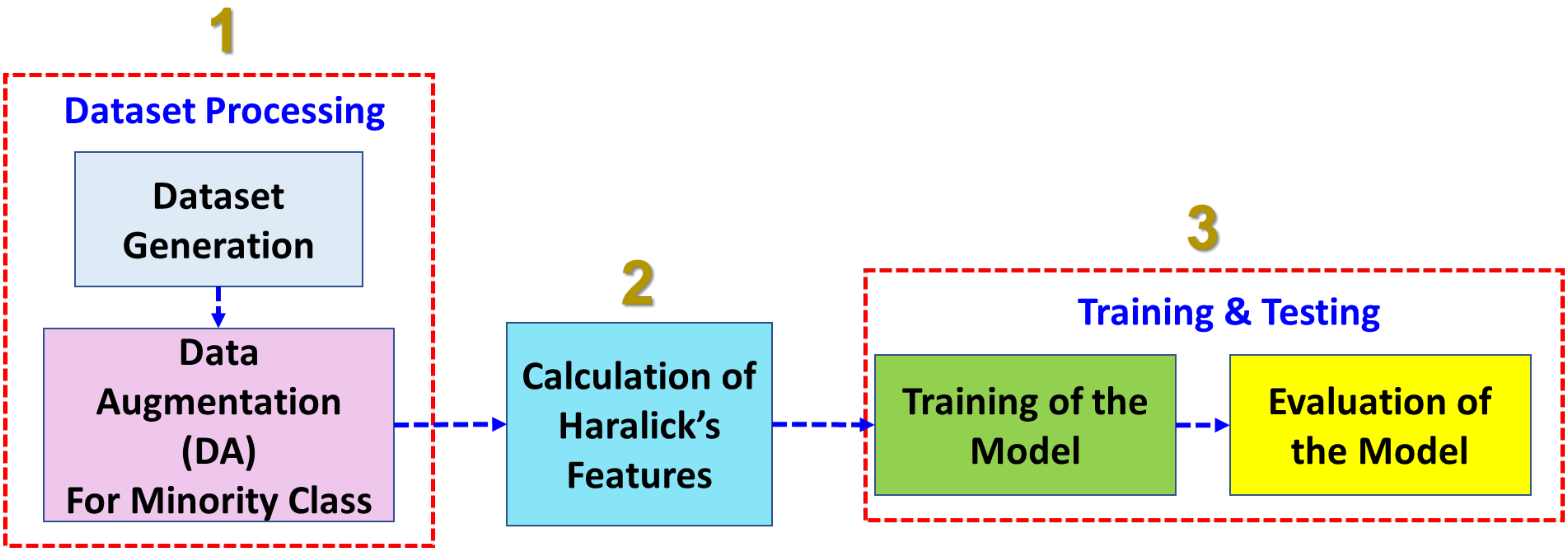
Related Works

Papers	Source of Deepfake Images/Videos	Remarks	IoT Implementation
Nataraj et al. [2019]	GAN	GLCM on RGB Channels + DNN	No
Liu et al. [2020]	GAN	Gram Block + ResNet	No
Wang et al. [2020]	GAN	FakeSpotter: Monitoring neuron behavior	No
He et al. [2019]	GAN	Ensemble deep learning technique via a Random Forest classifier + Computing Intensive.	No
Mitra et.al. [2020, 2021]	Auto-encoder	For compressed social media videos. Less Computation and high accuracy.	No
EasyDeep [2021]	**GAN**	**Textural Analysis + LightGBM Classifier**	**Yes**

EasyDeep Overview

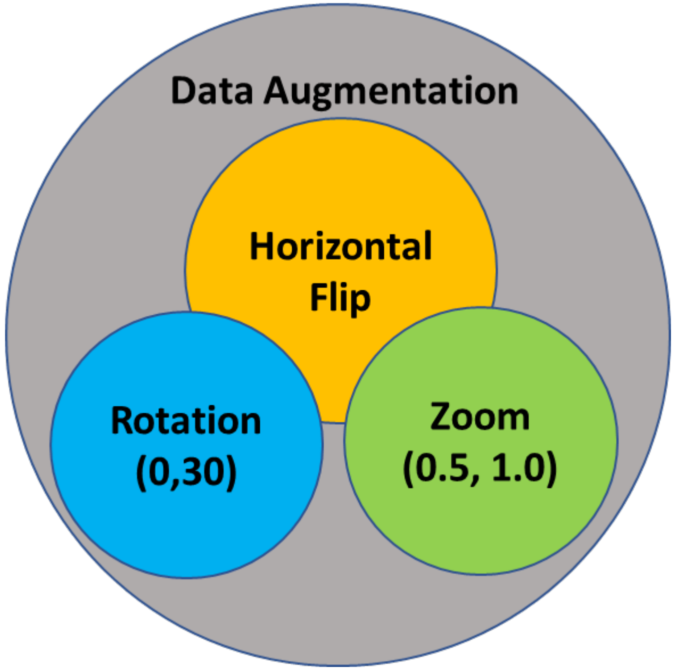


EasyDeep Implementation Workflow



EasyDeep: Data Generation & Augmentation

- Features Used for Fake Image Generation by StarGAN
- Hair color (black, blond, brown), Gender, and Age



Class	Number of Images	Source
Real	6,000	CelebA
Generated	30,000	StarGAN

Class	# of Images (Before DA)	DA	# of Images (After DA)	Source
Real	6,000	Yes	30,000	CelebA
Generated	30,000	No	30,000	StarGAN

EasyDeep: Attributes for StarGAN Images

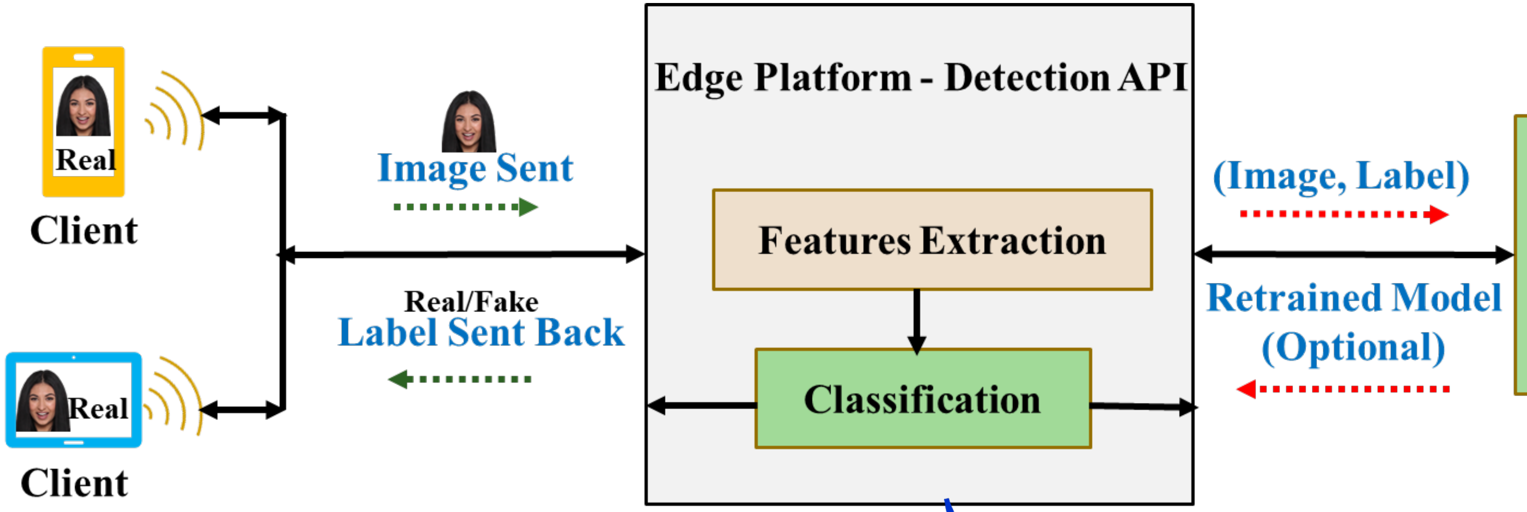
Sample CelebA Images

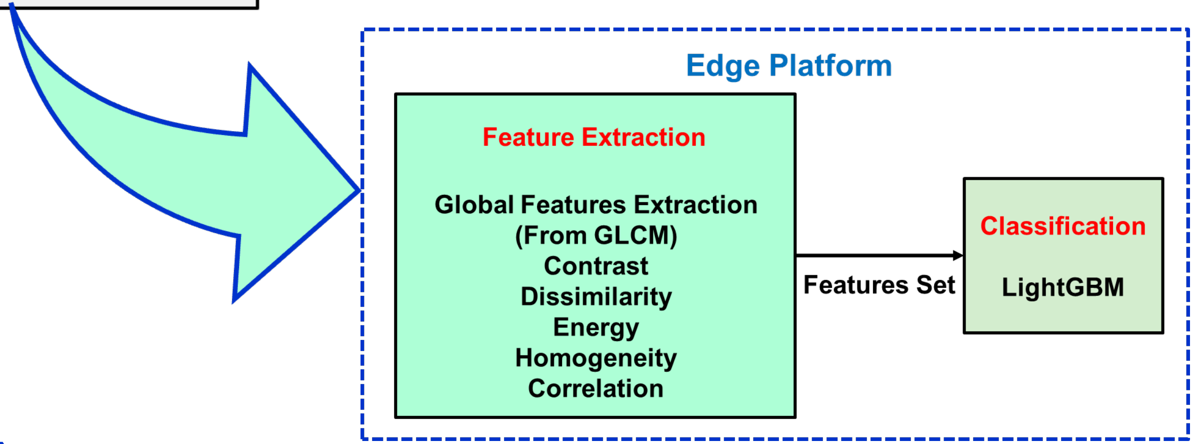
- Source CelebA Dataset

Attributes Used to Generate Images : 5

Original | Black Hair | Blond Hair | Brown Hair | Gender | Age

EasyDeep: Features Extraction & Classification



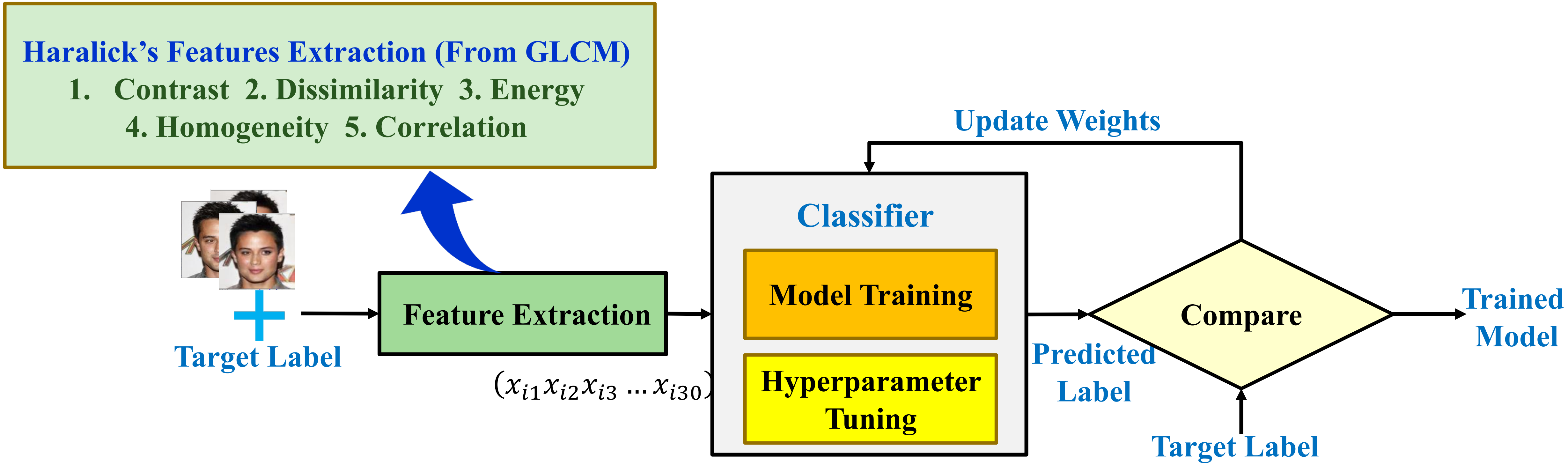


- GLCM for
  - four distances d=1, 2, 3, 5,
  - three angles θ=0, π/4, π/ 2
- Gradient Boosting Decision Tree (gbdt)

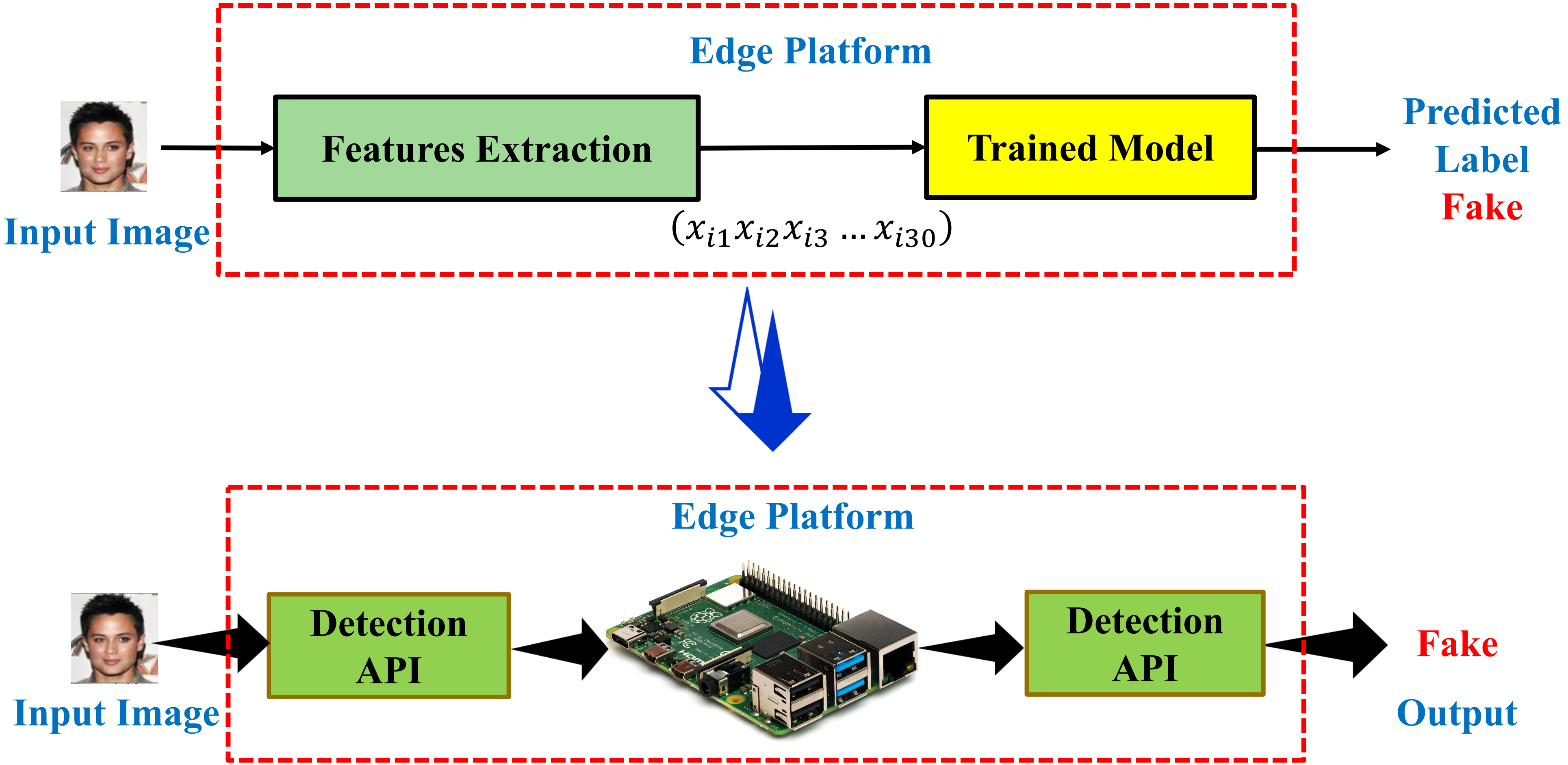
EasyDeep: Why LightGBM?

- Uses histograms to learn.
- Cost effective as time complexity α number of bins once histograms are made.
- – Use of discrete bins reduces the memory usage which is a limiting factor at an edge device.
- – Training is very fast as it is distributed.

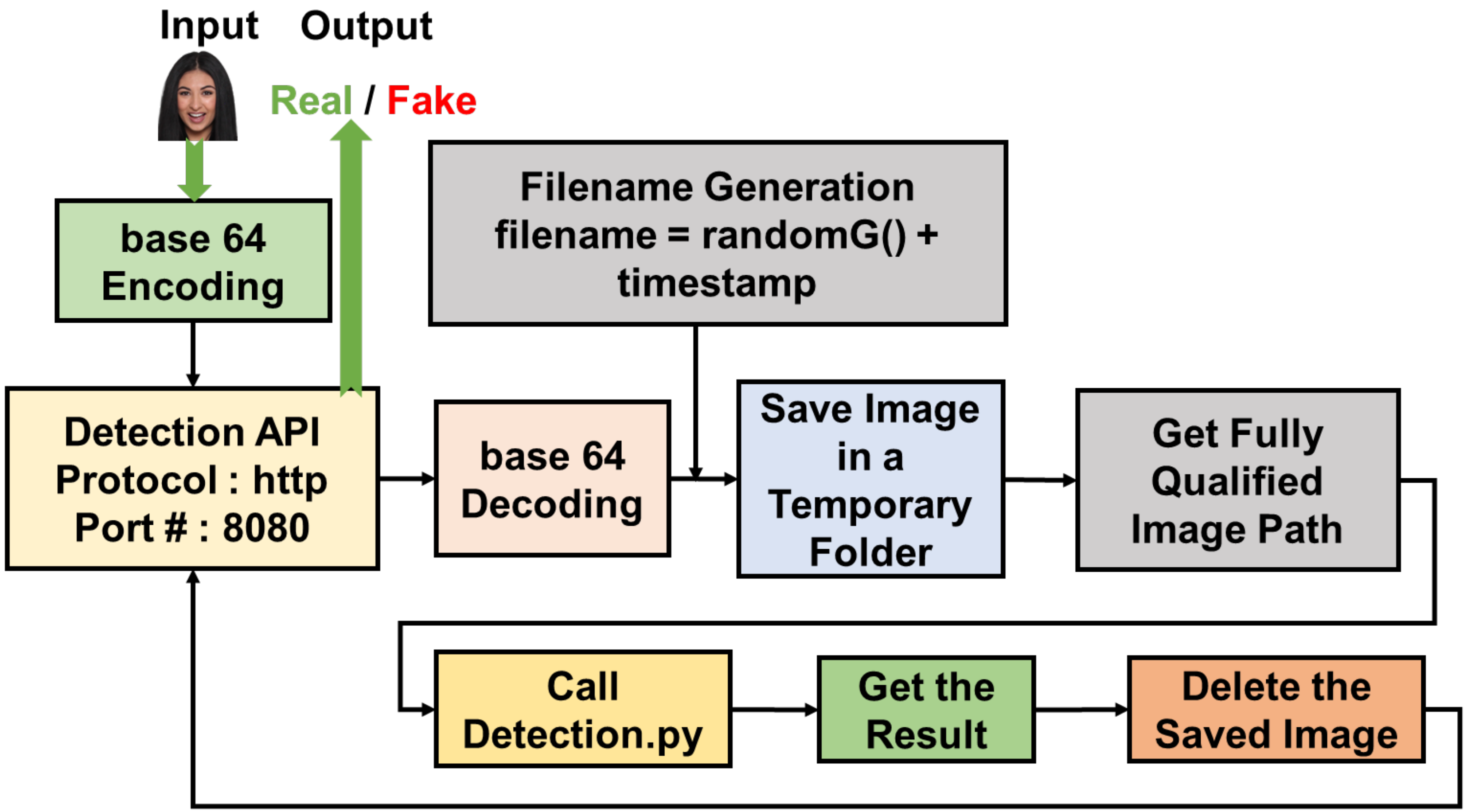
EasyDeep Training Workflow



EasyDeep Testing Workflow



EasyDeep: Detection API



EasyDeep: Implementation

- Implemented on a 4GB Raspberry pi 4.
- Input image provided through the Detection API.
- Detection result has been given back through the API.
- Detection API in Java.
- RGB Image → Gray Level Image
- Resized to 256 x 256.
- ImageDataGenerator() of Keras API used for data augmentation for minority class.
- The features set is constructed from Haralick's texture features.
- The feature set is of size 48,000 x 30 for training data.

EasyDeep: Implementation (Contd..)

- Initial training on a PC (16GB memory & Intel Core i7-9750 processor).
- No GPU used.
- 48,000 images for training + 10,000 images for validation + 2000 images for testing.
- Total time for training and validation of the model was 27 minutes.
- Learning Rate of the classifier = 0.05
- # of Trees = 600 ; Maximum Depth = 13 ; Number of Leaves = 8,500.
- Boosting algorithm = 'Gradient Boosting Decision Tree'.
- Detection method in Python.

EasyDeep: Detection Metrices

	Predicated Label	
True Label	True Positive (**TP**): Reality : **Fake** Model predicted : **Fake**	False Negative (**FN**): Reality : **Fake** Model predicted : **Real**
	False Positive (**FP**): Reality : **Real** Model predicted : **Fake**	True Negative (**TN**): Reality : **Real** Model predicted : **Real**

$$Accuracy = \left(\frac{TP+TN}{TP+TN+FP+FN}\right)\times 100\%$$

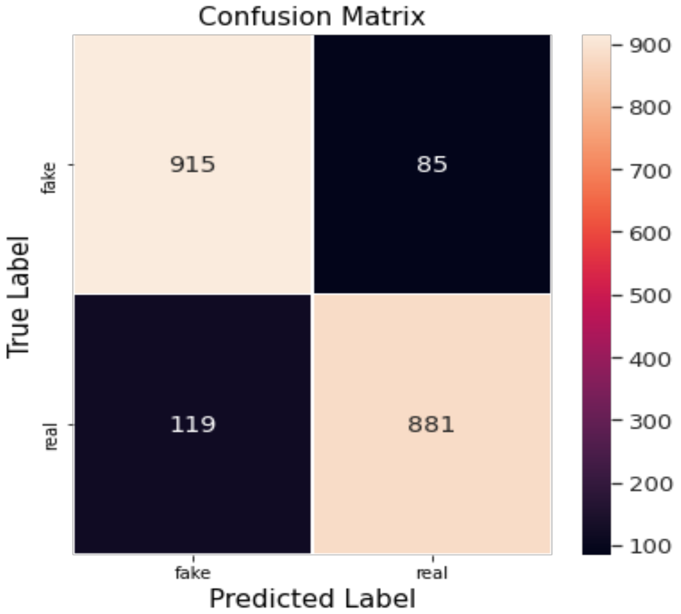
$$Precision = \left(\frac{TP}{TP+FP}\right)\times 100\%$$

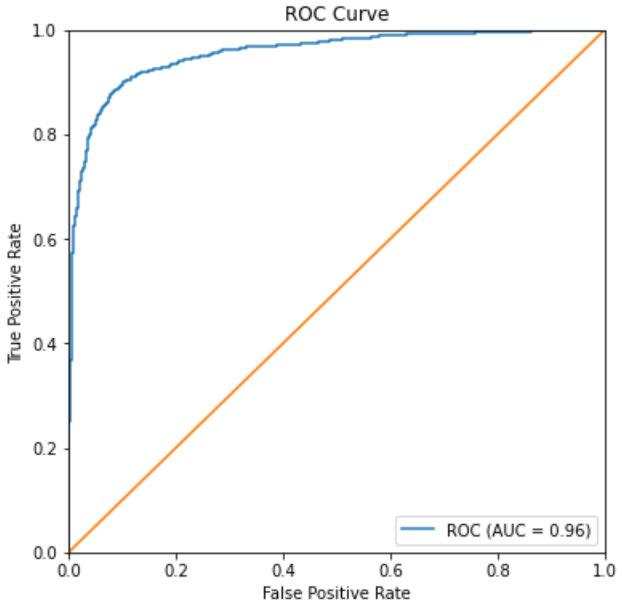
$$Recall = \left(\frac{TP}{TP+FN}\right)\times 100\%$$

$$F1-score = \left(\frac{2}{\frac{1}{Precision}+\frac{1}{Recall}}\right)\times 100\%$$

EasyDeep: Classification Report On Test

Test Images	Precision%	Recall%	F1-score%
1000 Fake	88.0	92.0	90.0
1000 Real	91.0	88.0	90.0
Macro Average	90.0	90.0	90.0
Weighted Average	90.0	90.0	90.0
Total 2000	Accuracy %	90.0	
Total 2000	AUC Score %	96.0	





EasyDeep: Accuracy Variation With Tree

Number of Trees	Max Tree Depth	Number of Leaves	Algorithm Boosting	Accuracy %	Model Size (MB)
100	8	255	dart*	79.4	3.2
100	10	1000	dart	80.4	6.7
100	11	2500	dart	81.8	12.4
100	12	4200	dart	82.1	15.8
100	13	8500	dart	82.9	19.0
100	14	17000	dart	82.7	22.2
100	13	8500	gbdt*	85.5	14.3
100	14	17000	gbdt	85.9	16.4
200	13	8500	gbdt	87.4	21.5
300	13	8500	gbdt	88.2	27.3
400	13	8500	gbdt	89.0	32.8
600	13	8500	gbdt	90.0	43.7

dart* (Dropouts meet Multiple Additive Regression Trees)
gbdt* (Gradient Boosting Decision Tree)

EasyDeep & Other Works

Papers	IoT Implementation	Remarks	Accuracy / AUC
Nataraj et al. [2019]	No	Heavy Computation than Ours	93.42 % 99.49 %
Liu et al. [2020]	No	Heavy Computation than Ours	87.52% - 95.51%
Wang et al. [2020]	No	Heavy Computation than Ours	90.6 % 93.1 %
He et al. [2019]	No	Heavy Computation Not suitable for IoT	99.35%
EasyDeep [2021]	**Yes**	**Less Computation Training Time < 30 minutes**	**96%**

Conclusions & Future Work

- Light Weight and Less Computing Model.
- IoT Friendly.
- High AUC.
- Training Time is Very Low.
- Accuracy will Improve by Increasing # of Trees & # of Features.
- Inference Time can be improved by Sending Images in Binary Format.
- With More Number of Features Generalizability can be Obtained.
- Mobile Apps will be made in future.

Thank You !!