Neural Style Transfer using Generative Adversarial Network on Mobile Device

Real Face to Anime Drawing Style

Nguyen Trung Hieu

July 7, 2021

Introduction

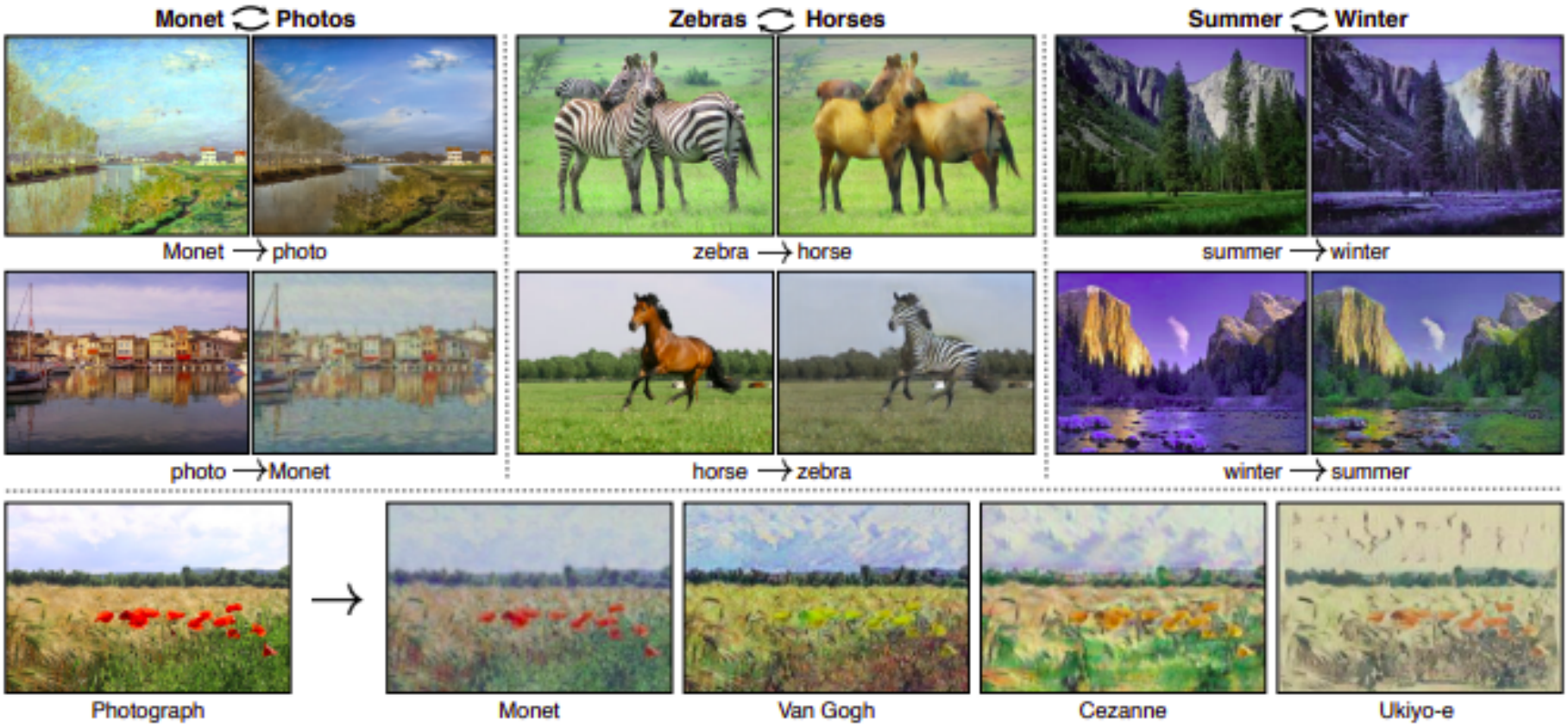


Figure: Unpaired Image to Image Translation[1]

[1] https://arxiv.org/pdf/1703.10593.pdf

Convert face to anime style

Face2Anime

- Input: Human face from real life
- Output: Anime - drawing style of that face image
- Why: Entertaining

Unpaired image-to-image translation

- Mapping from one image domain to another (e.g. zebras-horses, apple-orange, summer-winter)
- Unpaired Data: 2 piles of data, each from a domain.
- The network should learn the common features (content) and unique features (style) of each domain to perform mapping

Convert face to anime style

Data: Unpaired image-to-image translation

- Real-face dataset: CelebFaces Attributes (CelebA) Dataset[2]
- Drawing-anime dataset: Anime Face Dataset[3]

[2] https://www.kaggle.com/jessicali9530/celeba-dataset

[3] https://www.kaggle.com/splcher/animefacedataset

Generative Adversarial Network (GAN)

Generative Adversarial Network (GAN)

- Given that we want a mapping from $X \rightarrow Y$.
- A GAN is constructed from a Generator $G_Y(X)$ and a Discriminator D_y
- A generator $G_Y(X)$ maps X to Y
- Discriminator is a classifier, output whether G(X) is real or fake

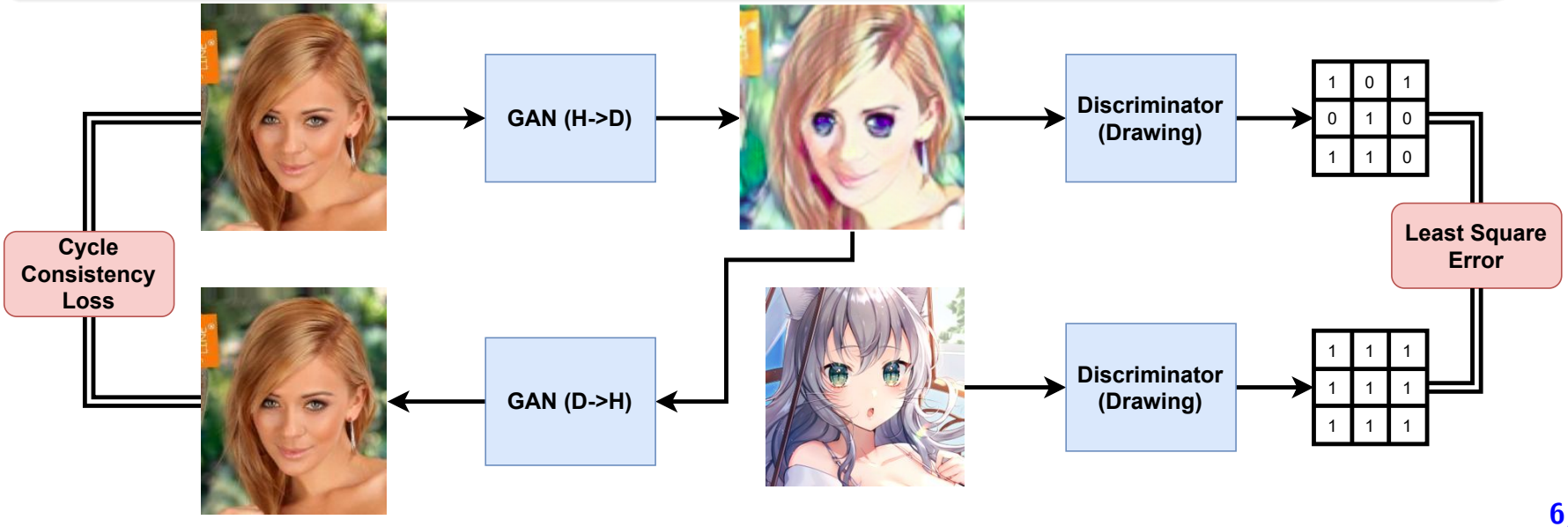
Training

- $\mathcal{L} = E_Y(log(D(y))) + E_X(log(1 - D(G_Y(X))))$
- Intuition: $G_Y(X)$ tries to fool D_Y, D_Y tries to classify G(X) as fake

Approach: CycleGan[4]

Idea

- H: Human face, D: Drawing Face
- Train two GANs at the same time: $GAN_{H \to D} + GAN_{D \to H}$
- Cycle consistency loss + Least Square Adversarial Loss + Identity Loss
- Generator Loss $= \mathcal{L}_{cycle} + \mathcal{L}_{Adversarial} + \mathcal{L}_{Identity}$
- Discriminator loss $= \mathcal{L}_{Adversarial}$



Approach: CycleGan

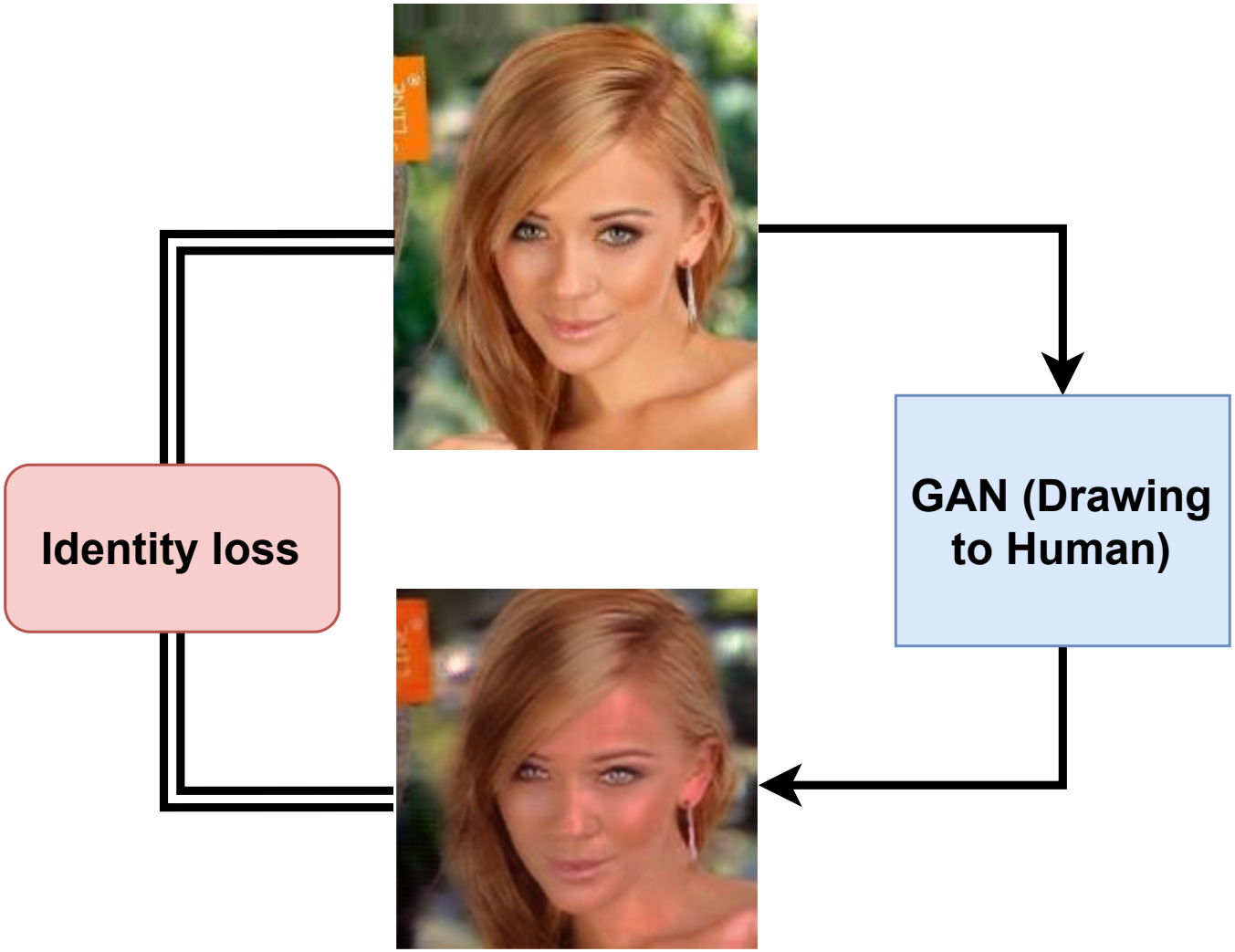


Figure: Identity Loss

Approach: CycleGan

Network Architecture

- Generator: U-NET
- Discriminator: PatchGAN[5]

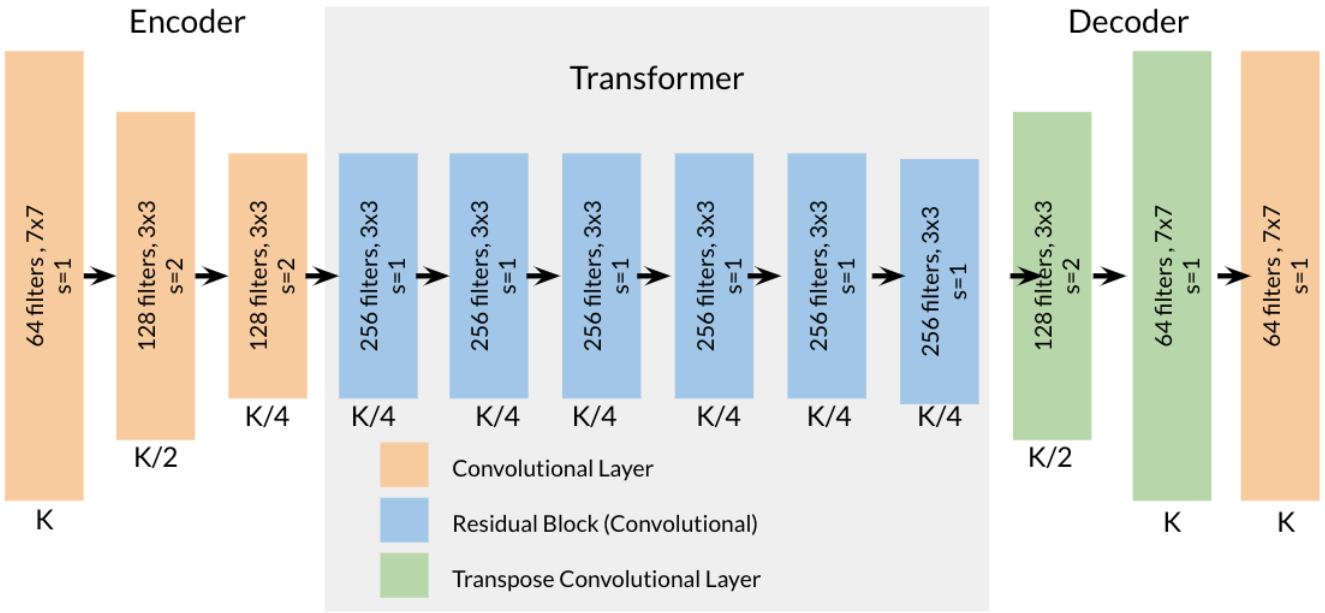


Figure: Generator (U-NET)[6]

[5] https://arxiv.org/pdf/1803.07422.pdf
[6] coursera.org/learn/apply-generative-adversarial-networks-gans

Approach: CycleGan

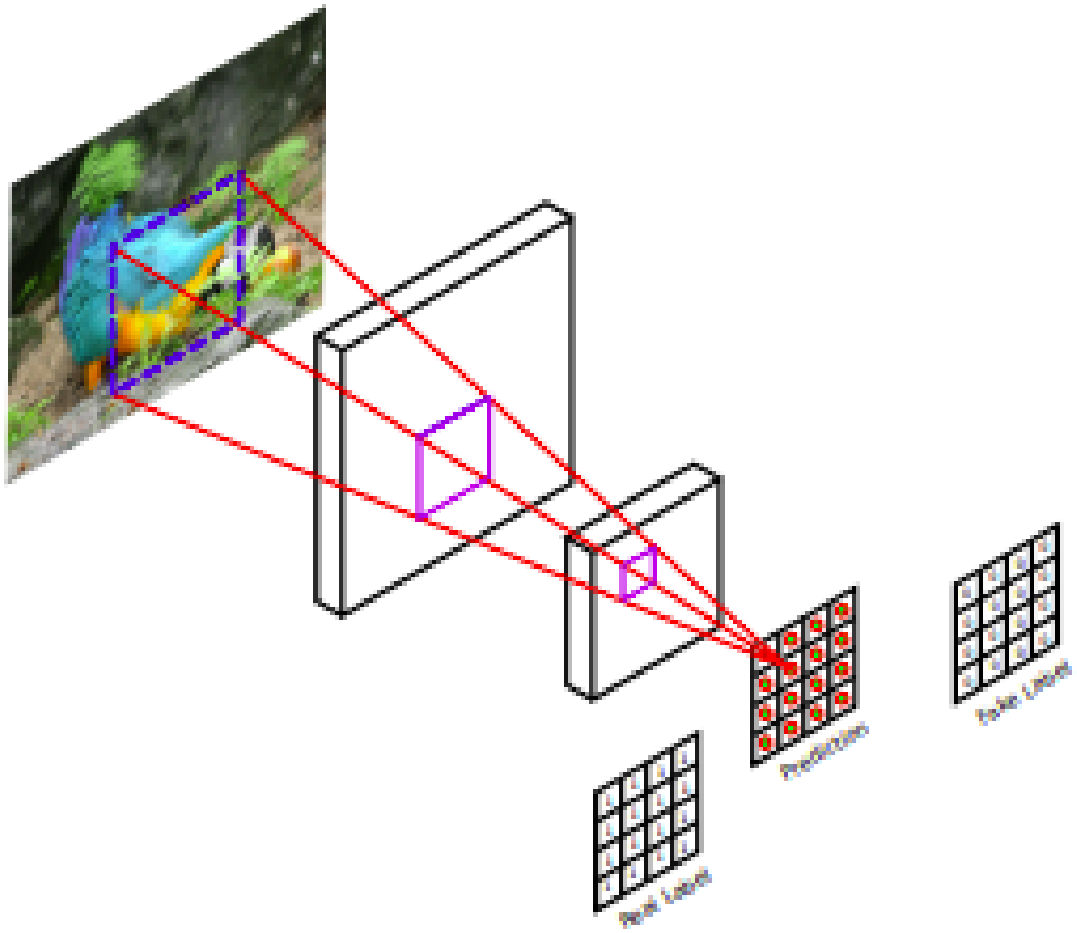


Figure: PatchGan[7]

[7] https://arxiv.org/pdf/1803.07422.pdf

Our Workflow

Figure: Our WorkFlow[8]

Resources

- GPU: Tesla P100-PCIE Google Colab
- Data: 10,000 face images in total
  - 5000 Anime Images (around 75% Female)[9]
  - 5000 CelebA images (around 61% Female)[10]
- Image resize to 256x256 resolution
- Train for 100 epochs (nearly 1.5 day)

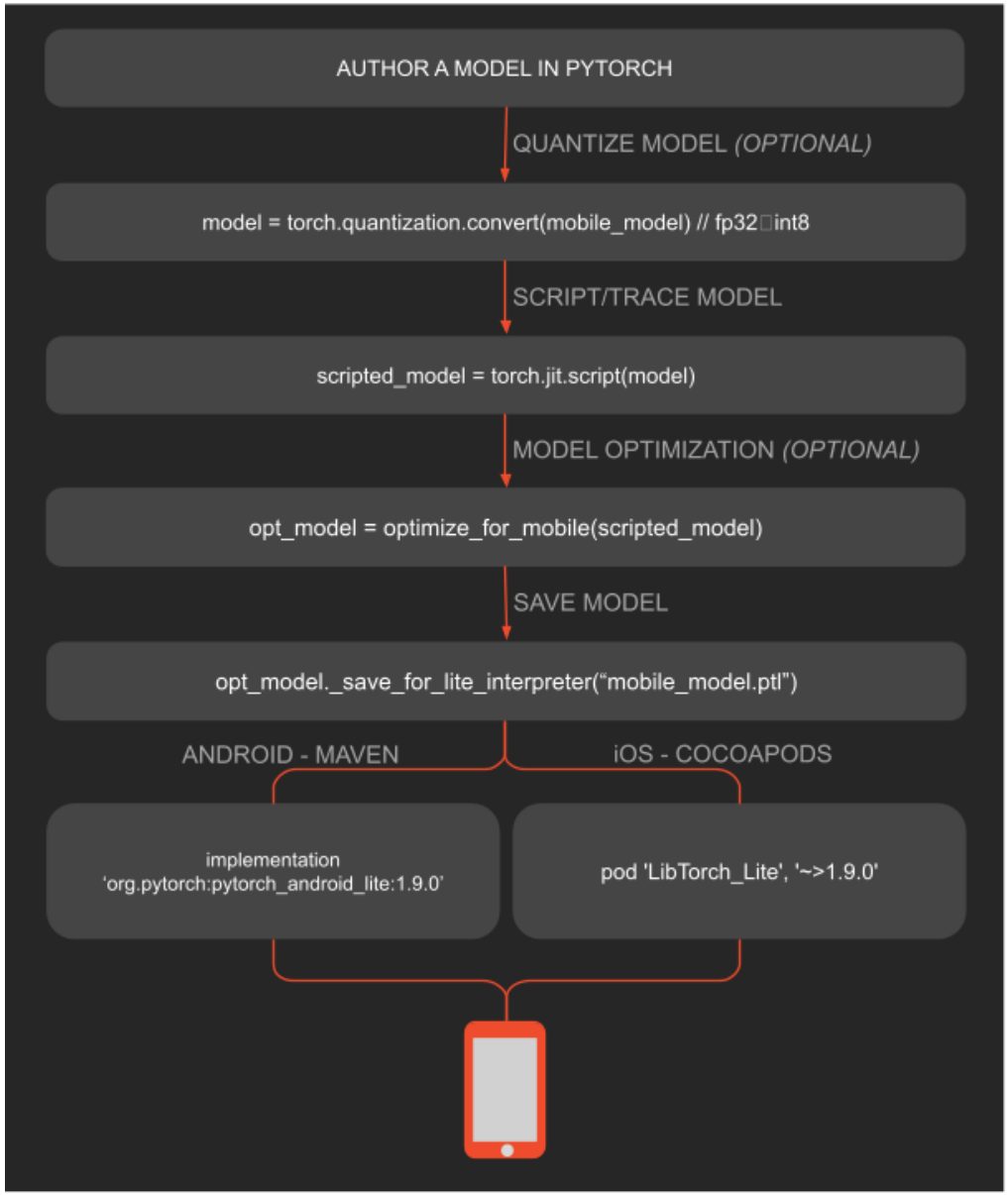
[8] approximate by random sampling

Some Results

- Eyes changing, Hair Color

- little change result (mode collapse)

Optimize Model for Mobile Device



App Specification

Competitor Name	Inference Time (ms)	RAM	Model size
Emulator	15,979.25	2048MB	44,500 Kb
Real Android	3,500	2048MB	44,500 Kb
Hieu's Laptop	2,530.71	8Gb Ram	300 MB

Table: Mobile model comparison

Success and Failure

- We successfully optimize trained cycleGAN model to deploy on our mobile devices
- We have trained from scratch a CycleGan model (nearly 30 Millions parameters)
- We failed to address some mode collapse cases

The End.

Thank you for your attention!