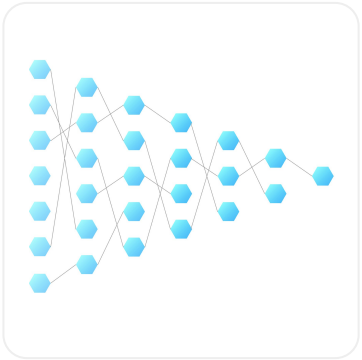
NEUROMATION Synthesis.ai

Synthetic Data in Deep Learning

Sergey Nikolenko

October 4, 2019

Building AI Computer Vision Applications

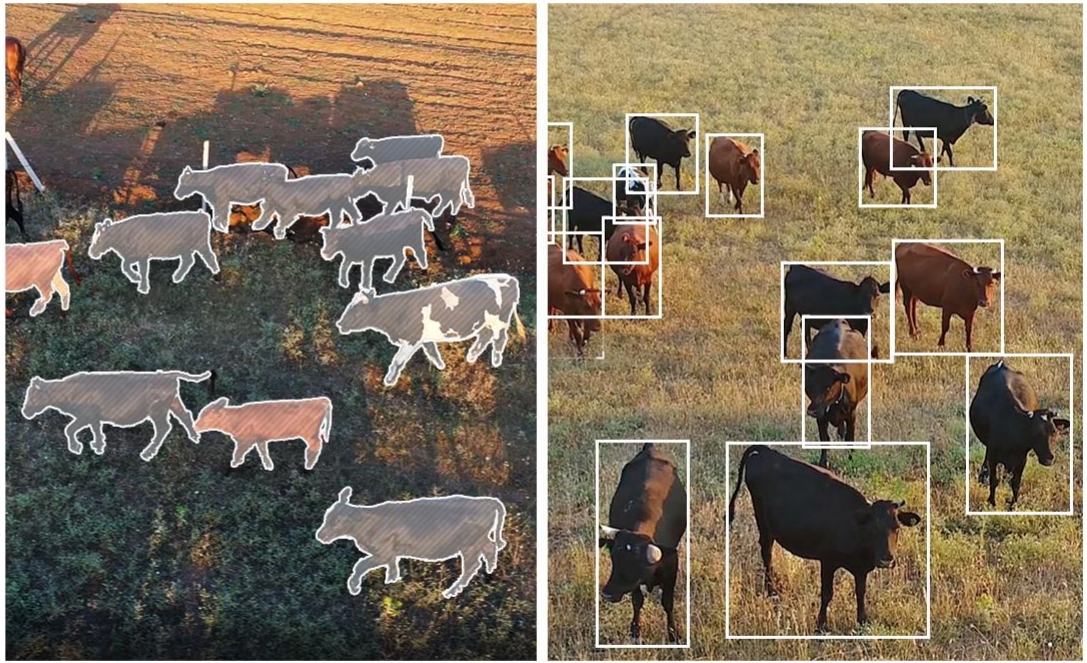


DATA → ANNOTATION → TRAINING → DEPLOYMENT

NEUROMATION

Manual Data Labeling is Labor Intensive

- The deep learning revolution was in a large part due to new large datasets available for training
- To create datasets for supervised learning, we need to *label* the training samples
- Labeling is a very laborious manual process, especially for some problems such as, e.g., segmentation



Major Limiting Factors for AI Adoption

1

Availability
of large data sets

2

Accuracy
and availability of labels

3

Non-representative data
(lack of diversity)

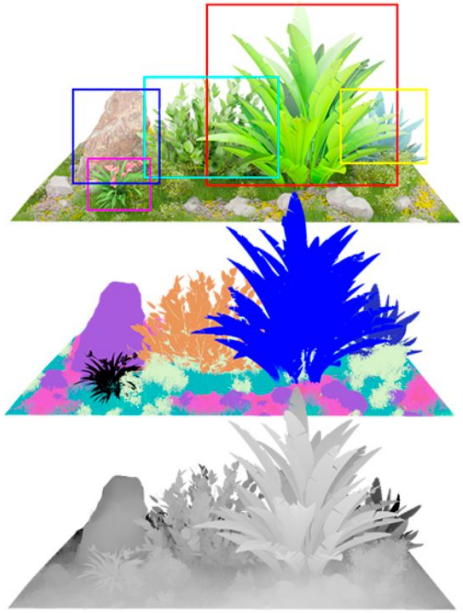
Data is the limiting factor

Here at Neuromation, we are addressing these issues with...

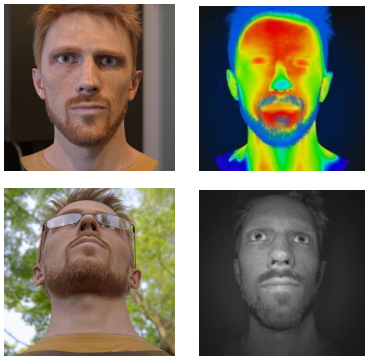
Synthetic Data

DATA
Vast Scale & Diversity

LABELS
Pixel-Perfect



MODELS
Unbiased and Robust

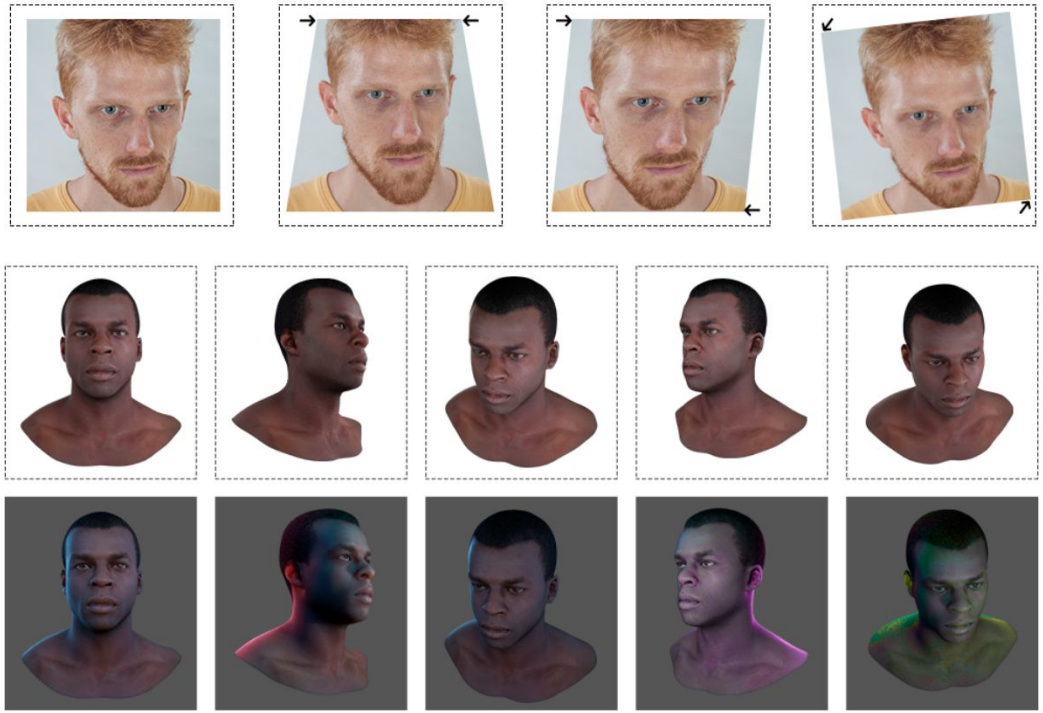


NEUROMATION

Scale: Easy to Add New Objects

Diversity: Variation Leads to Robustness

- Deep learning models for computer vision always use *data augmentation*
- But data augmentation methods are very restricted: we have to be able to transfer the *labels* (correct answers) automatically
- Synthetic data can support a *much* more varied set of transformations...



NEUROMATION

Combinatorial Power of Synthetic Data

Camera attributes

- Imaging modality (optical, IR, multispectral, etc)
- Resolution
- Field of view
- Capture angle
- Distance
- ISO / white balance / exposure / hue & saturation

Environment

- Lighting type (outdoor, time of day, indoor, LED, etc)
- Light source/position & intensity
- Background
- Surface properties

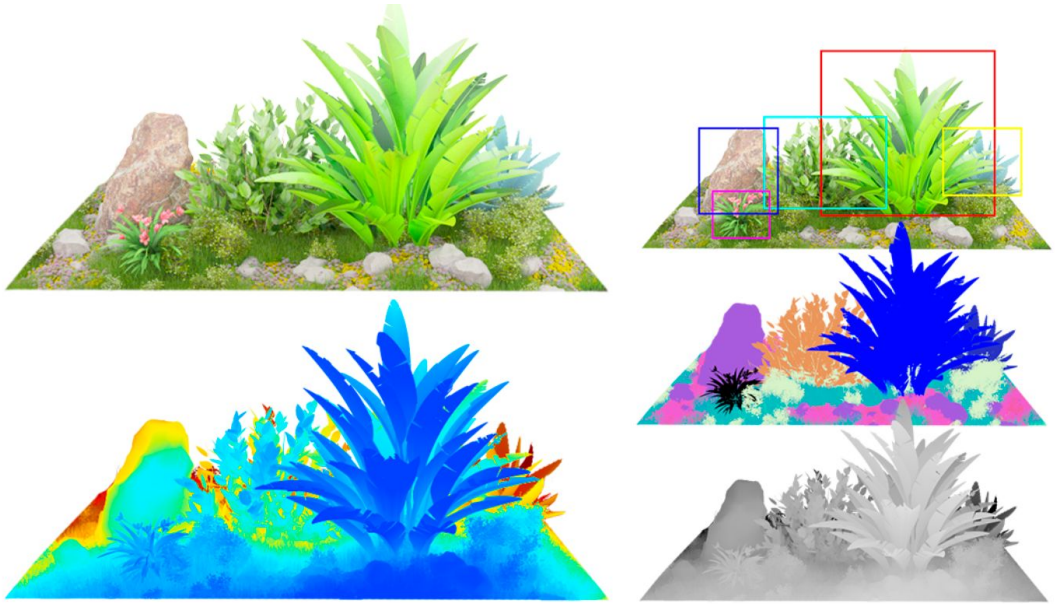
Scene complexity

- # and type of objects
- Object position with respect to other objects
- Confounds & view obstruction
- Dynamic vs static

Limitless number of images per object

Labeling Accuracy: Pixel-Perfect Labels

- Synthetic data comes with free pixel-perfect labeling for:
  - object detection
  - segmentation
  - key points and specific features
- Moreover, it can have labeling impossible for humans:
  - depth map
  - camera viewpoint
  - fully labeled 3D scene
  - objects hidden from view



NEUROMATION

Key Ideas in Synthetic Data Development

- **Domain randomization**: let's try vary the synthetic distribution so much as to cover the real data
- **Domain adaptation**: let's try to make synthetic data more realistic and/or useful (later)
- **Generation feedback loop**: let's try to learn how to generate synthetic data (later yet)
- **Synthetic data for privacy**: can we generate a synthetic dataset that has similar distributions but won't provide information about real data? (we won't go there today)
- And many more…
- Now for the obligatory shameless plug:

Synthetic Data for Deep Learning

Sergey I. Nikolenko[1,2]

[1]Synthesis.ai, San Francisco, CA
[2]Steklov Institute of Mathematics at St. Petersburg, Russia

snikolenko@synthesis.ai

Broad Set of Current Applications

Face Recognition/
Verification

Autonomous
Vehicles

Satellite/
Drone Imagery

Gesture
Recognition

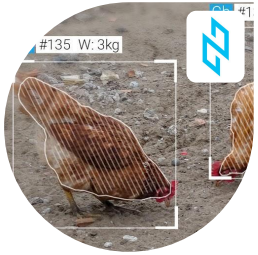
Interior Mapping/
Navigation

Medical
Imaging

Retail
Applications

Gaze
Estimation

Robotics



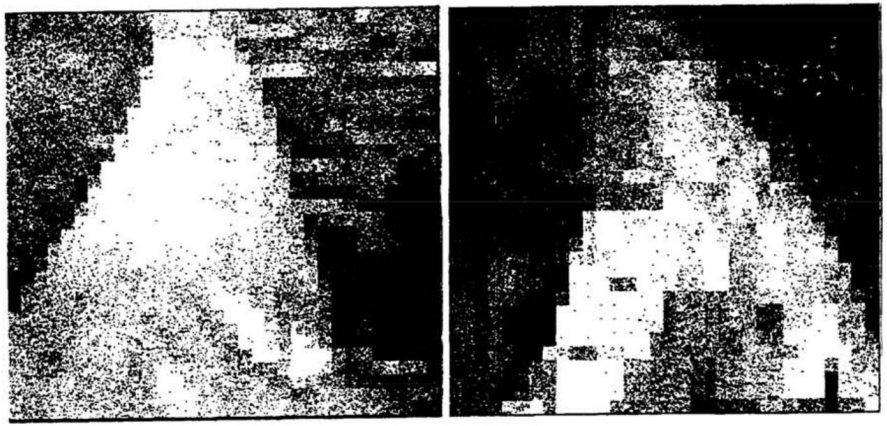
Animal
Applications

Security

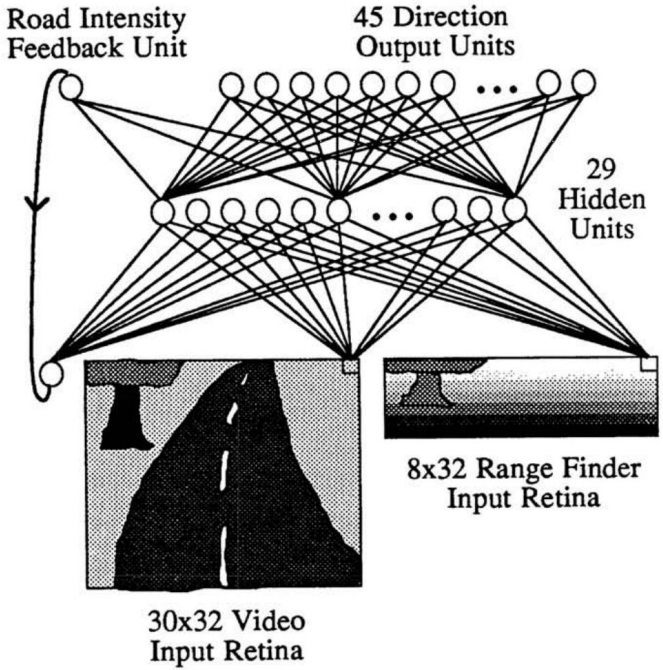
Mobile Recognition
(Object, Landmark)

Case Study: Outdoor Scenes

- Specific example: learning to drive, autonomous vehicles, and so on
- Computer vision problems: object detection and segmentation for outdoor (urban) scenes
- Interesting history -- ALVINN: An Autonomous Land Vehicle in a Neural Network (Pomerleau, 1989)
  - 30x32 videos used as input, and already synthetic!
  - "...training on actual road images is logistically difficult because... the network must be presented with a large number of training exemplars... under a wide variety of conditions..."



Real Road Image Simulated Road Image



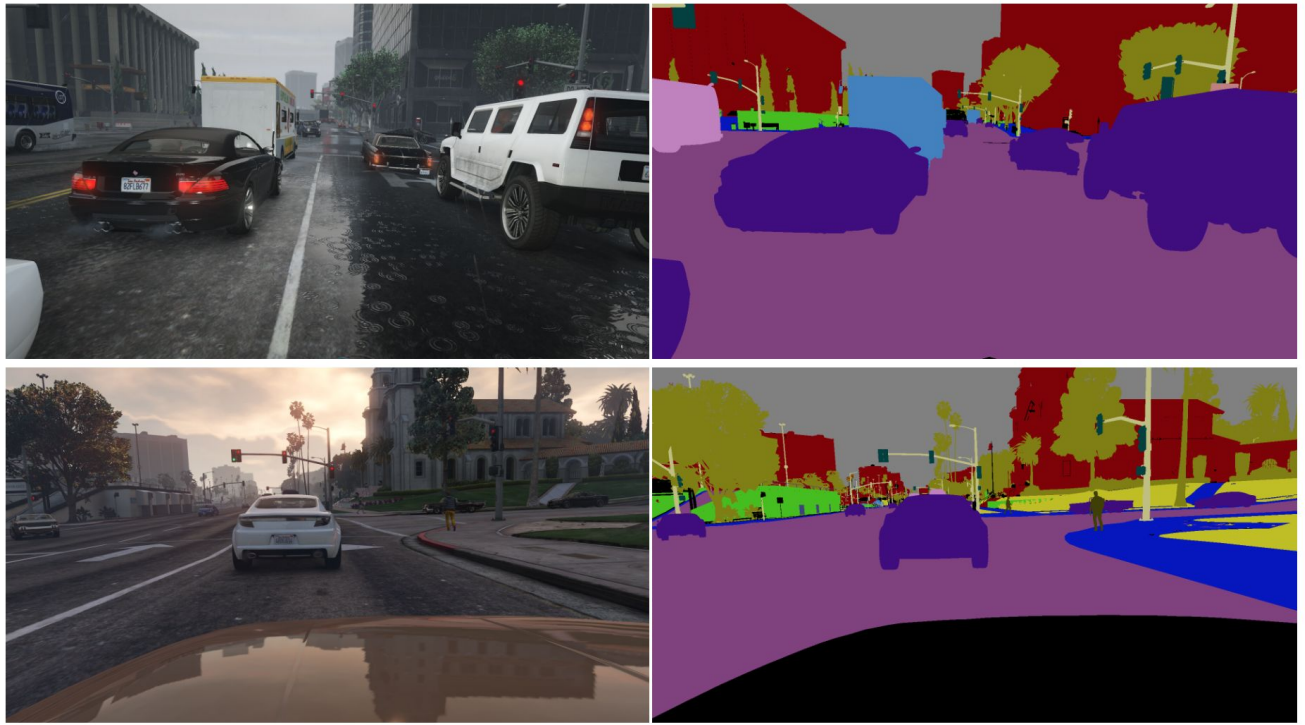
Case Study: Outdoor Scenes

- Virtual KITTI (Gaidon et al., 2016), modeled after the KITTI benchmark suite:
  - Unity game engine, 5 virtual outdoor environments, 50 photorealistic synthetic videos
  - way too small to actually train models on; used for benchmarking
  - e.g., Gaidon et al. tested multi-object trackers and found relatively small real-to-virtual gap

NEUROMATION

Case Study: Outdoor Scenes

- GTAVision (Johnson-Robertson et al., 2016): “Playing for Benchmarks” and “Playing for Data”
  - GTA V game engine
  - large enough datasets to train on
  - 250K 1920x1080 images annotated with optical flow, segmentation, 3D scene layout etc.
  - positive results: Faster RCNN could improve performance on real validation sets (KITTI) by training on synthetic data
- Similar: SYNTHIA and SYNTHIA-SF



Case Study: Outdoor Scenes

- VEIS (Saleh et al., 2018): Virtual Environment for Instance Segmentation
  - interesting distinction: not all classes are equally suited for synthetic data
  - things (people, cars, bikes) work well with object detectors but segmentation suffers since the textures are hard
  - stuff (grass, road surface, sky) has a high degree of texture realism
  - therefore, they use detection-based segmentation (Mask R-CNN) for the foreground and DeepLab for background

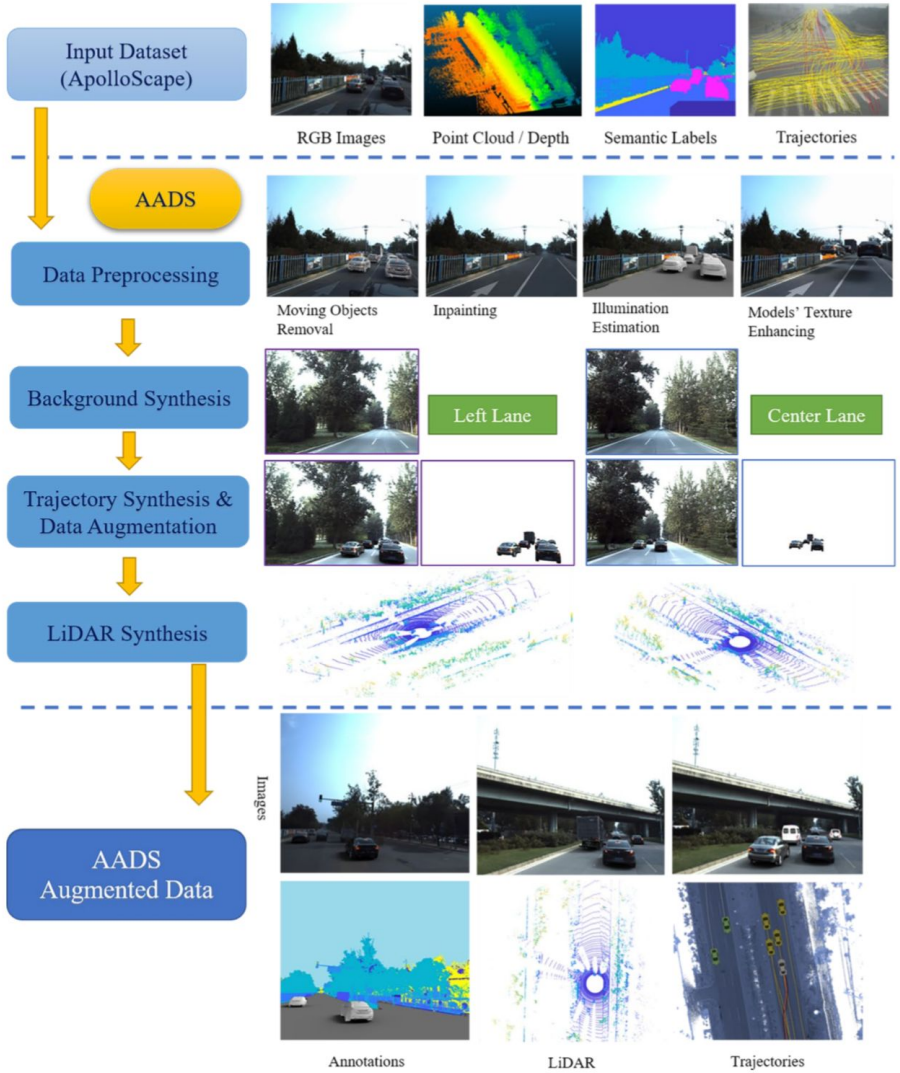
Case Study: Outdoor Scenes

- AADS (Li et al., 2019): Augmented Autonomous Driving Simulation
  - inserting synthetic traffic on real RGB images
  - starting from a real dataset, apply a lot of models to remove objects, inpaint backgrounds, estimate where and how to place new synthetic traffic etc.
- Similar to (Abu Alhaija et al., 2018):

Real scene (KITTI)

Synthetic scene (Virtual KITTI)

Real scene augmented with synthetic cars (Ours)



Case Study: Outdoor Scenes

- Synscapes (Wrenninge, Unger, 2019): does realism help?
  - photorealistic renderings of urban scenes with unbiased path tracing for rendering, special models for light scattering effects in camera optics, motion blur, and more
  - conclude that additional effort for photorealism does indeed improve object detection over GTA-based datasets

NEUROMATION

Case Study: Outdoor Scenes

- Apart from datasets, we also need simulations; they are, of course, always synthetic
- One of the first still relevant examples is The Open Racing Car Simulator (TORCS):
  - started as a game for Linux
  - has involved physics simulation, including mechanical details, friction profiles for tyres, a realistic aerodynamic model etc.
  - currently one of the default classical simulators for learning to drive

Case Study: Outdoor Scenes

- CARLA (Dosovitsky et al., 2017): CAR Learning to Act
  - open source layer over Unreal Engine 4
  - provides RGB cameras (with customizable positions), depth maps, semantic segmentation maps with 12 classes (road, lane marking, traffic sign, sidewalk etc.), bounding boxes for dynamic objects, and measurements of the agent (vehicle location and orientation).

Case Study: Outdoor Scenes

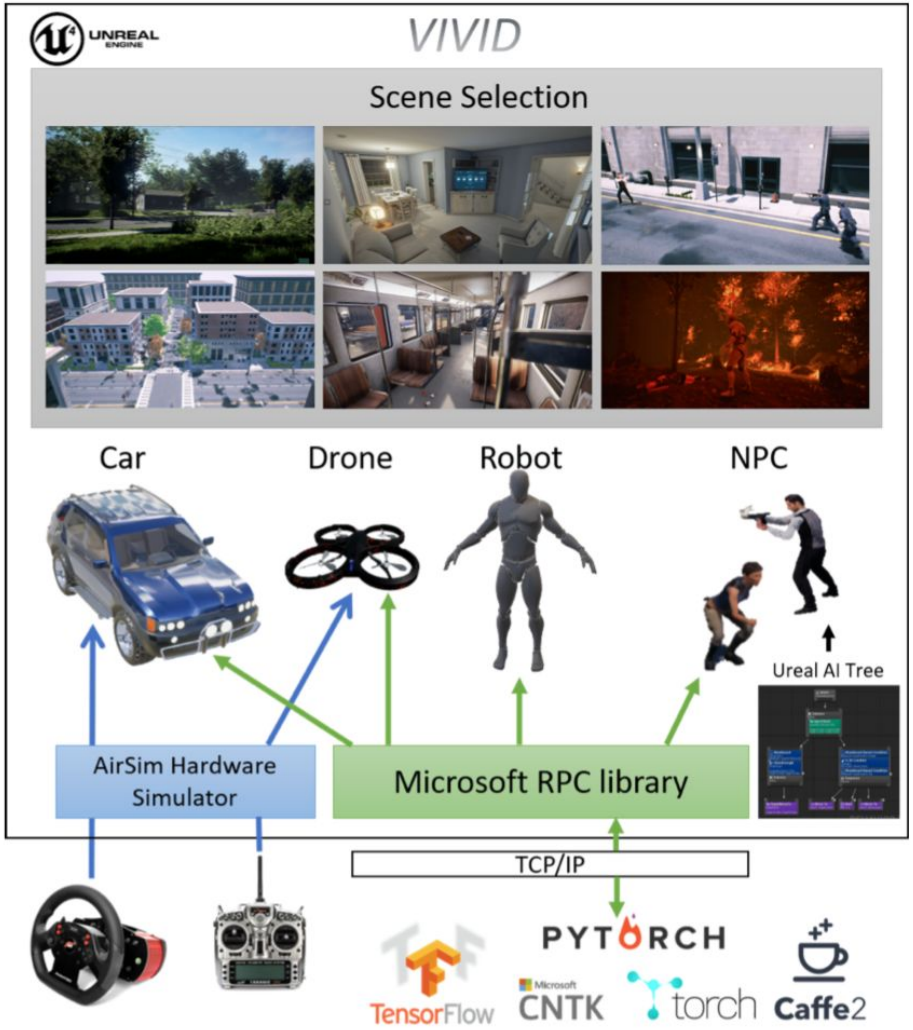
- VIVID (Lai et al., 2018): VIrtual environment for VIsual Deep learning
  - also based on the Unreal Engine but adds people interacting in various ways

(a) Gun shooting detection

(b) Searching in warehouse

(c) Forest fire rescue

(d) Pedestrian detection



NEUROMATION

Domain Transfer

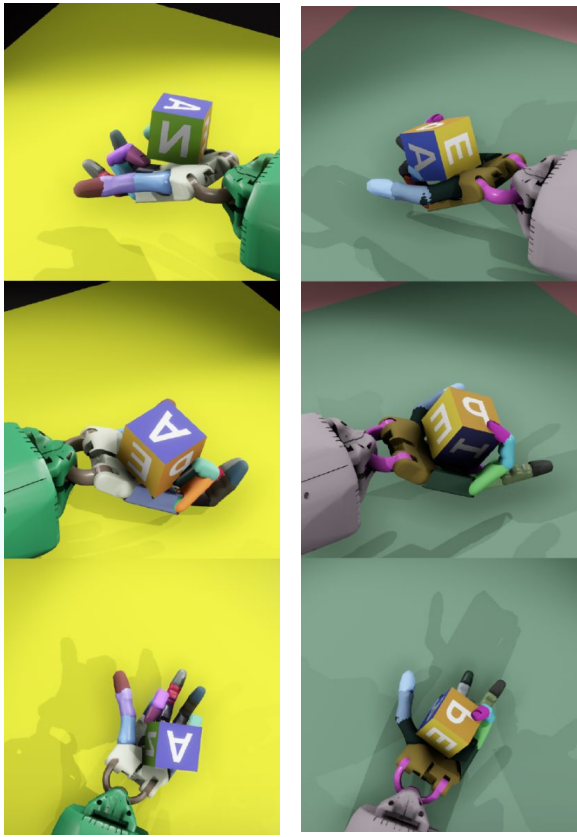
- Still, there are problems
- Synthetic data is not perfectly realistic, and it is extremely hard (and expensive) to make it realistic at the rendering stage
- Sometimes sufficient **domain randomization** is enough (Dactyl)
- But often we face the problem of **synthetic-to-real domain transfer**
- Two main ways to address it:

Refinement

Make synthetic data more realistic with some automated procedure

Domain adaptation

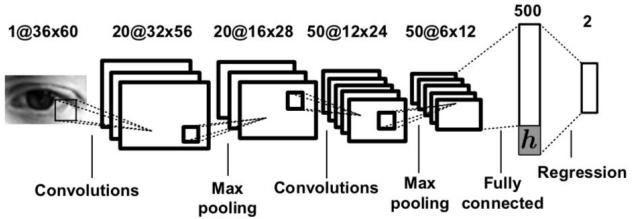
Change the model or its training process to address the transfer

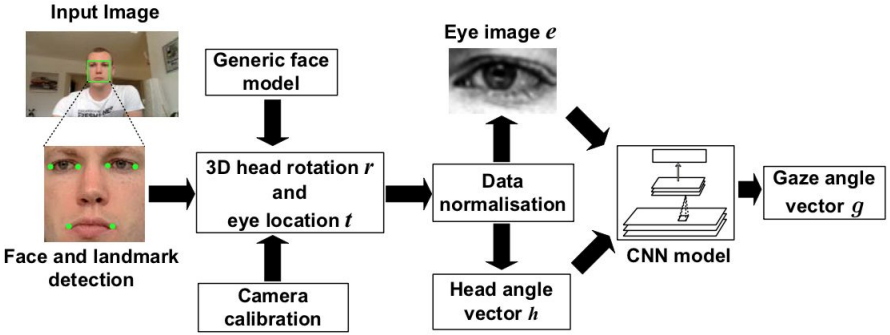


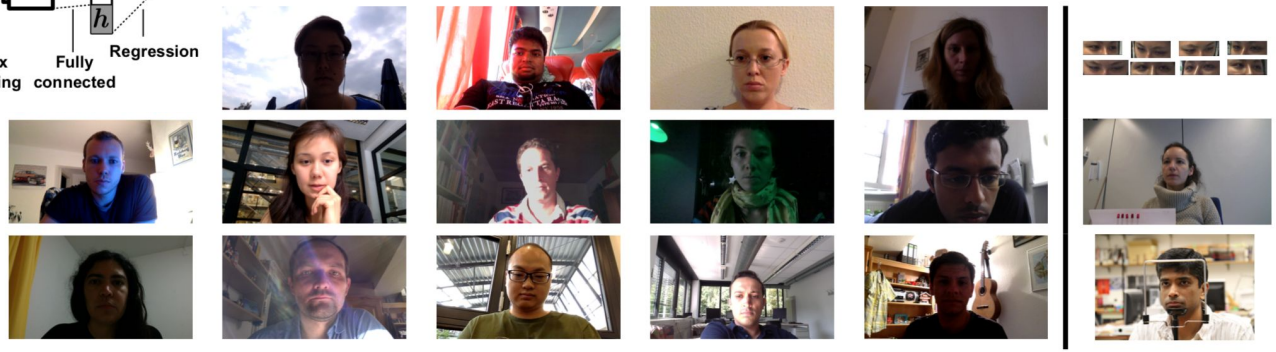
NEUROMATION

Refinement for Gaze Estimation

- Sample problem: **gaze estimation**
- (Zhang et al., 2015): gaze estimation in the wild; convolutional architecture

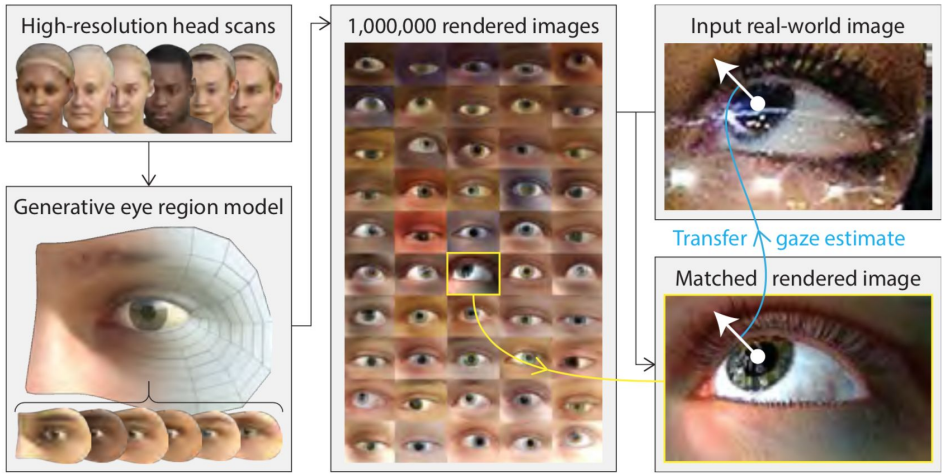


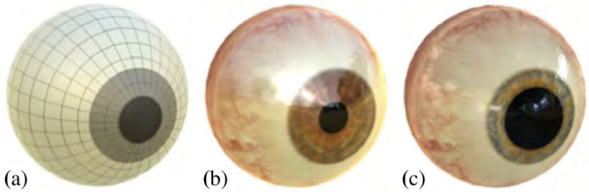




Refinement for Gaze Estimation

— (Wood et al., 2016): learning gaze estimation from synthetic images generated by a special 3D modeling system **UnityEyes**





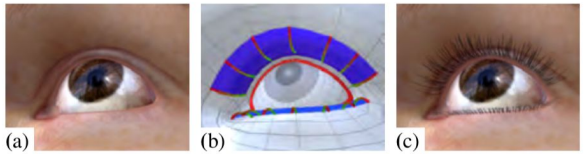


Figure 8: *We include eyelashes and eye wetness for realism. (a) shows a render without these, (b) shows eye wetness (red) and eyelash geometry (blue), (c) shows the final render.*

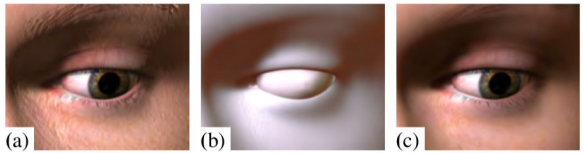
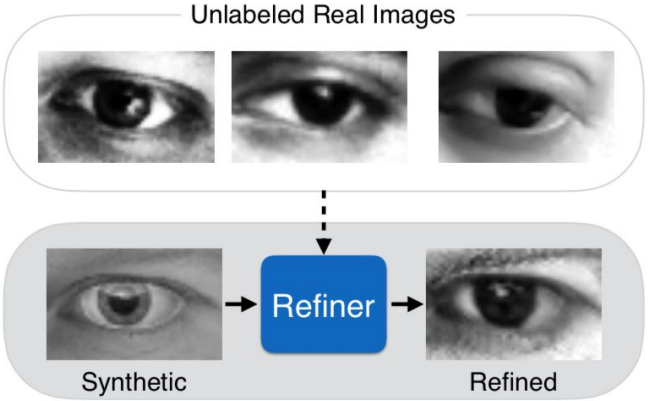


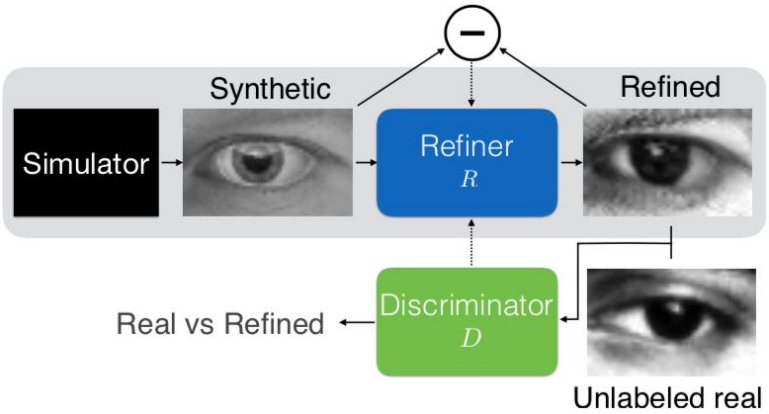
Figure 9: *We use pre-integrated skin shading for realism (c). Without it, skin appears too hard (a). (b) shows the scattered light through skin – this causes the skin to appear soft.*

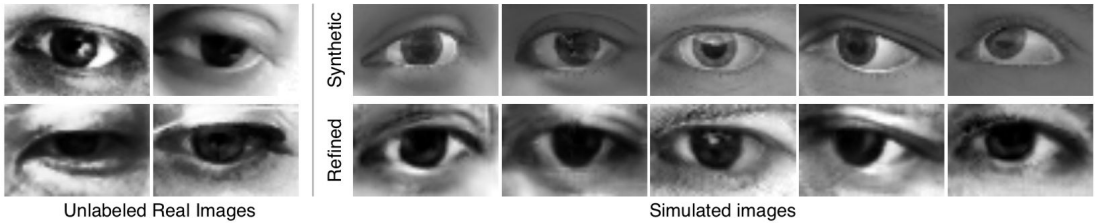
NEUROMATION

Refinement for Gaze Estimation

- (Shrivastava et al., 2017): Apple learns a gaze estimation model on **synthetic images refined by SimGAN** that trains to fool D while self-regularizing to keep refined similar to original synthetic images

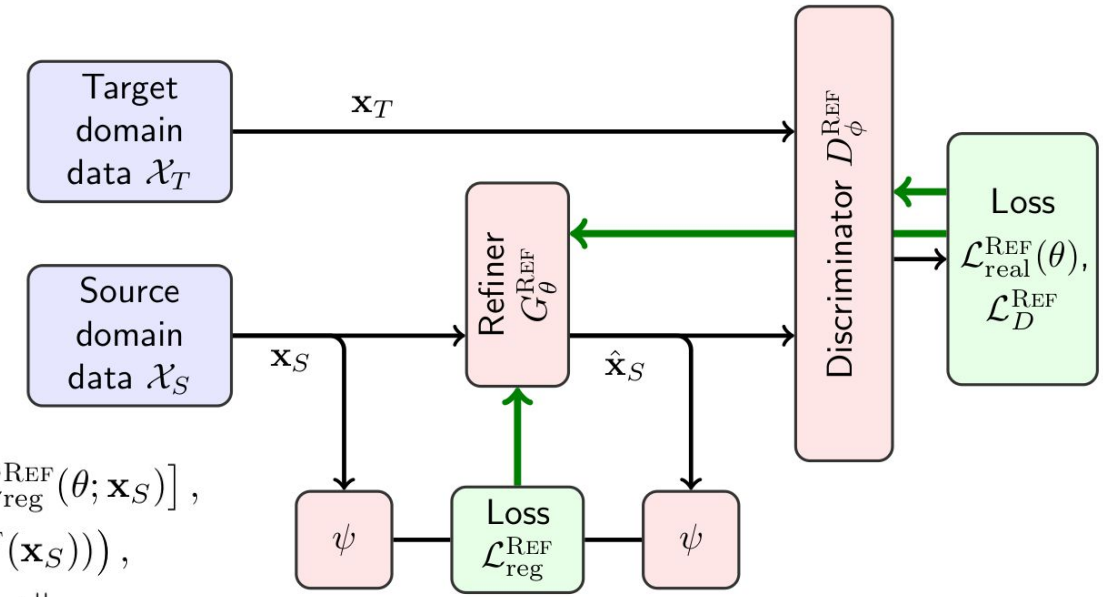






Refinement for Gaze Estimation

- (Shrivastava et al., 2017): here is the SimGAN architecture
- Very straightforward GAN
- The only interesting novelty here are the ψ functions that compute the gaze and the corresponding loss function to preserve gaze direction



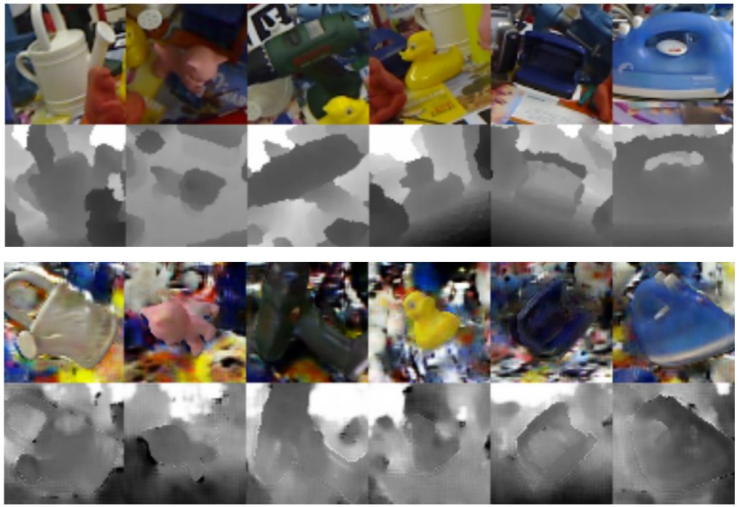
- Loss functions:

$$\mathcal{L}_G^{\text{REF}}(\theta) = \mathbb{E}_S\left[\mathcal{L}_{\text{real}}^{\text{REF}}(\theta; \mathbf{x}_S) + \lambda \mathcal{L}_{\text{reg}}^{\text{REF}}(\theta; \mathbf{x}_S)\right],$$
$$\mathcal{L}_{\text{real}}^{\text{REF}}(\theta; \mathbf{x}_S) = -\log\left(1 - D_\phi^{\text{REF}}(G_\theta^{\text{REF}}(\mathbf{x}_S))\right),$$
$$\mathcal{L}_{\text{reg}}^{\text{REF}}(\theta; \mathbf{x}_S) = \left\|\psi(G_\theta^{\text{REF}}(\mathbf{x}_S)) - \psi(\mathbf{x}_S)\right\|_1,$$

General Synthetic Data Refinement

- (Bousmalis et al., 2016): PixelDA, early work on refinement
- Pixel-level domain adaptation via style transfer
- Alternating optimization

$$\min_{\theta_G,\theta_T} \max_{\phi} \lambda_1 \mathcal{L}_{\text{dom}}^{\text{PIX}}(D^{\text{PIX}}, G^{\text{PIX}}) + \lambda_2 \mathcal{L}_{\text{task}}^{\text{PIX}}(G^{\text{PIX}}, T^{\text{PIX}}) + \lambda_3 \mathcal{L}_{\text{cont}}^{\text{PIX}}(G^{\text{PIX}})$$



- Loss functions:
  - domain loss

$$\mathcal{L}_{\text{dom}}^{\text{PIX}}(D^{\text{PIX}}, G^{\text{PIX}}) = \mathbb{E}_{\mathbf{x}_S \sim p_{\text{syn}}} \left[\log\left(1 - D^{\text{PIX}}(G^{\text{PIX}}(\mathbf{x}_S; \theta_G); \phi)\right)\right] + \mathbb{E}_{\mathbf{x}_T \sim p_{\text{real}}} \left[\log D^{\text{PIX}}(\mathbf{x}_T; \phi)\right];$$

  - task-specific loss

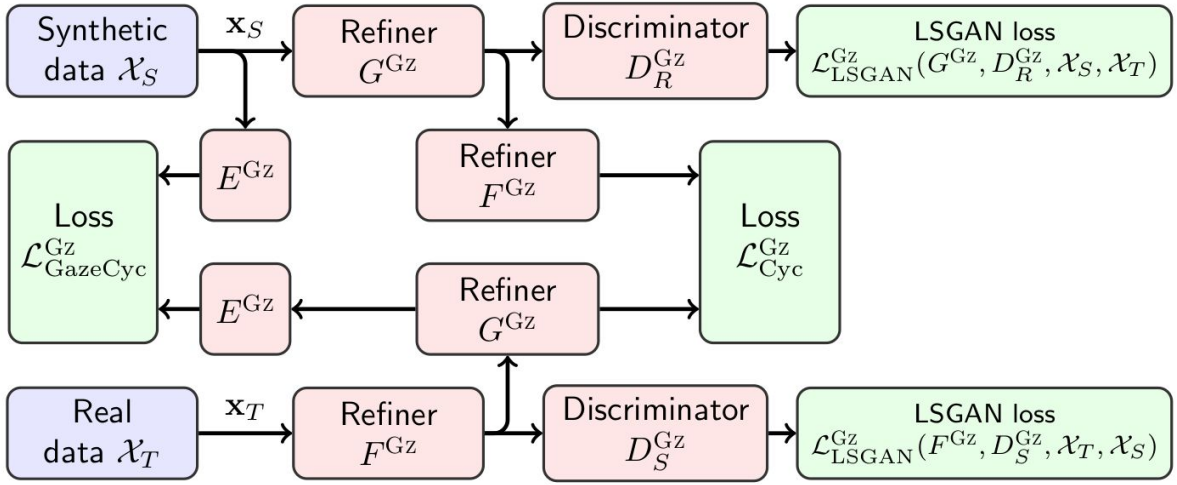
$$\mathcal{L}_{\text{task}}^{\text{PIX}}(G^{\text{PIX}}, T^{\text{PIX}}) = \mathbb{E}_{\mathbf{x}_S, \mathbf{y}_S \sim p_{\text{syn}}} \left[-\mathbf{y}_S^\top \log T^{\text{PIX}}(G^{\text{PIX}}(\mathbf{x}_S; \theta_G); \theta_T) - \mathbf{y}_S^\top \log T^{\text{PIX}}(\mathbf{x}_S; \theta_T)\right]$$

  - content similarity loss (right)

$$\mathcal{L}_{\text{cont}}^{\text{PIX}}(G^{\text{PIX}}) = \mathbb{E}_{\mathbf{x}_S \sim p_{\text{syn}}} \left[\frac{1}{k} \left\|(\mathbf{x}_S - G^{\text{PIX}}(\mathbf{x}_S; \theta_G)) \odot \mathbf{m}(\mathbf{x})\right\|_2^2 - \frac{1}{k^2} \left((\mathbf{x}_S - G^{\text{PIX}}(\mathbf{x}_S; \theta_G))^\top \mathbf{m}(\mathbf{x})\right)^2\right]$$

Refinement for Gaze Estimation

- (Sela et al., 2017): GazeGAN added CycleGAN ideas to SimGAN
- Now we have two refiners and a cycle that should be idempotent
- LSGAN loss for the discriminators, cycle consistency and gaze consistency
- Loss functions:



$$\mathcal{L}_{\mathrm{Cyc}}^{\mathrm{Gz}}(G^{\mathrm{Gz}}, F^{\mathrm{Gz}}) = \mathbb{E}_{\mathbf{x}_S \sim p_{\mathrm{syn}}} \left[\| F^{\mathrm{Gz}}(G^{\mathrm{Gz}}(\mathbf{x}_S)) - \mathbf{x}_S \|_1 \right] + \mathbb{E}_{\mathbf{x}_T \sim p_{\mathrm{real}}} \left[\| G^{\mathrm{Gz}}(F^{\mathrm{Gz}}(\mathbf{x}_T)) - \mathbf{x}_T \|_1 \right];$$

$$\mathcal{L}_{\mathrm{GazeCyc}}^{\mathrm{Gz}}(G^{\mathrm{Gz}}, F^{\mathrm{Gz}}) = \mathbb{E}_{\mathbf{x}_S \sim p_{\mathrm{syn}}} \left[\| E^{\mathrm{Gz}}(F^{\mathrm{Gz}}(G^{\mathrm{Gz}}(\mathbf{x}_S))) - E^{\mathrm{Gz}}(\mathbf{x}_S) \|_2^2 \right]$$

$$\mathcal{L}_{\mathrm{LSGAN}}^{\mathrm{Gz}}(G, D, S, R) = \mathbb{E}_{\mathbf{x}_S \sim p_{\mathrm{syn}}} \left[\left(D(G(\mathbf{x}_S)) - 0.9 \right)^2 \right] + \mathbb{E}_{\mathbf{x}_T \sim p_{\mathrm{real}}} \left[D(\mathbf{x}_T)^2 \right]$$

General Synthetic Data Refinement

- (Zheng et al., 2018): T^{2}Net for syn-to-real refinement, for single-image depth estimation
- Loss functions:
  - GAN loss

$$\mathcal{L}_{\text{GAN}}^{\text{T2}}(G_S^{\text{T2}}, D_T^{\text{T2}}) = \mathbb{E}_{\mathbf{x}_S \sim p_{\text{syn}}}\left[\log(1 - D_T^{\text{T2}}(G_S^{\text{T2}}(\mathbf{x}_S)))\right] + \mathbb{E}_{\mathbf{x}_T \sim p_{\text{real}}}\left[\log D_T^{\text{T2}}(\mathbf{x}_T)\right];$$

  - feature-level GAN loss

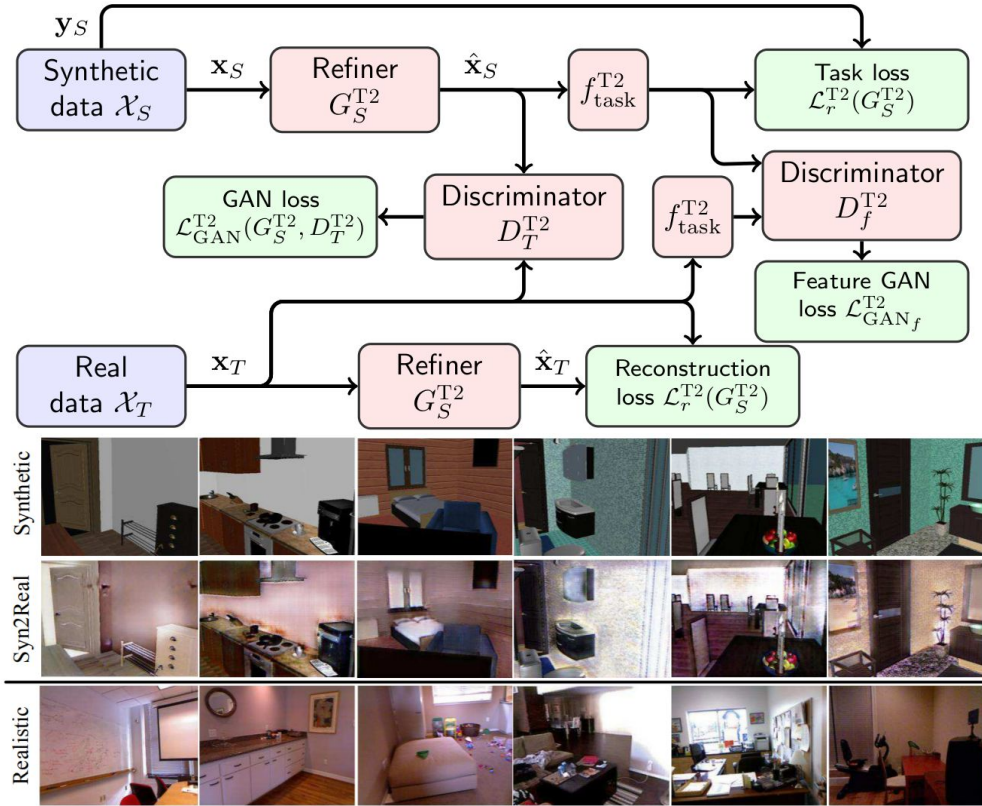
$$\mathcal{L}_{\text{GAN}_f}^{\text{T2}}(f_{\text{task}}^{\text{T2}}, D_T^{\text{T2}}) = \mathbb{E}_{\mathbf{x}_S \sim p_{\text{syn}}}\left[\log D_f^{\text{T2}}(f_{\text{task}}^{\text{T2}}(G_S^{\text{T2}}(\mathbf{x}_S)))\right] + \mathbb{E}_{\mathbf{x}_T \sim p_{\text{real}}}\left[\log(1 - D_f^{\text{T2}}(f_{\text{task}}^{\text{T2}}(\mathbf{x}_T)))\right];$$

  - reconstruction loss $\mathcal{L}_r^{\text{T2}}(G_S^{\text{T2}}) = \left\| G_S^{\text{T2}}(\mathbf{x}_T) - \mathbf{x}_T \right\|_1$
  - task loss for depth estimation on synthetic images

$$\mathcal{L}_r^{\text{T2}}(f_{\text{task}}^{\text{T2}}) = \left\| f_{\text{task}}^{\text{T2}}(\hat{\mathbf{x}}_S) - \mathbf{y}_S \right\|_1$$

  - task loss for depth estimation on real images

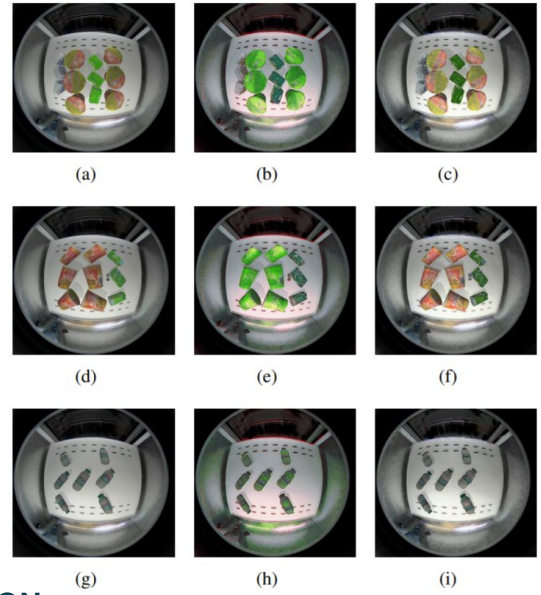
$$\mathcal{L}_s^{\text{T2}}(f_{\text{task}}^{\text{T2}}) = \left|\partial_x f_{\text{task}}^{\text{T2}}(\mathbf{x}_T)\right|^{-|\partial_x \mathbf{x}_T|} + \left|\partial_y f_{\text{task}}^{\text{T2}}(\mathbf{x}_T)\right|^{-|\partial_y \mathbf{x}_T|}$$

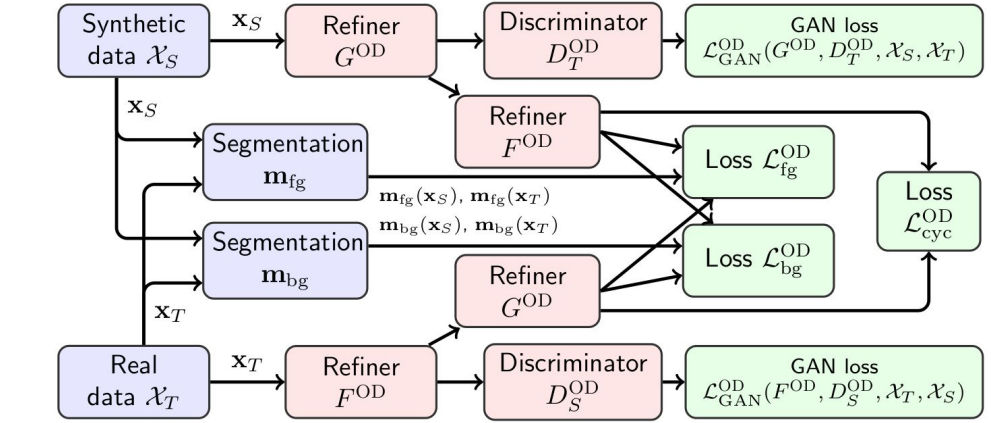


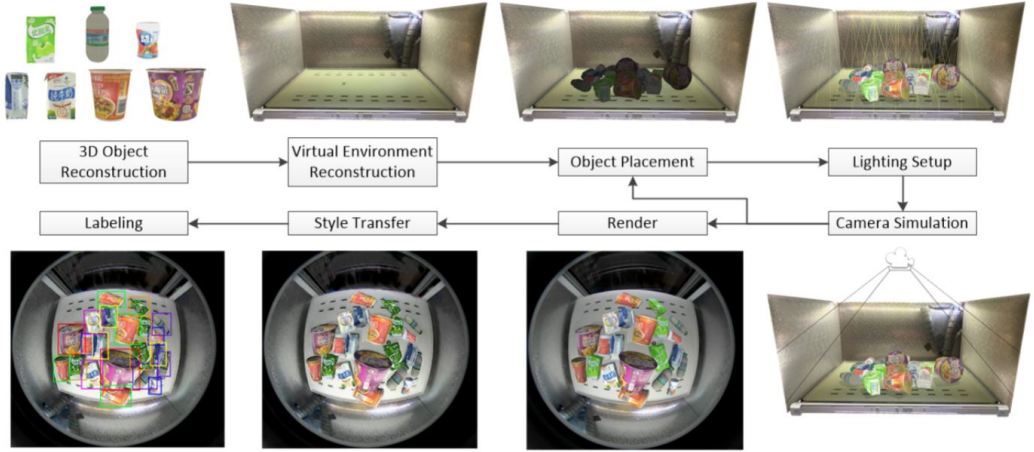
NEUROMATION

General Synthetic Data Refinement

- (Wang et al., 2019): syn-to-real refinement for automatic vending machines
- Create randomized layouts for virtual fisheye cameras with deformed objects etc.
- Example (left to right: render, classical style transfer, Wang et al.)







NEUROMATION

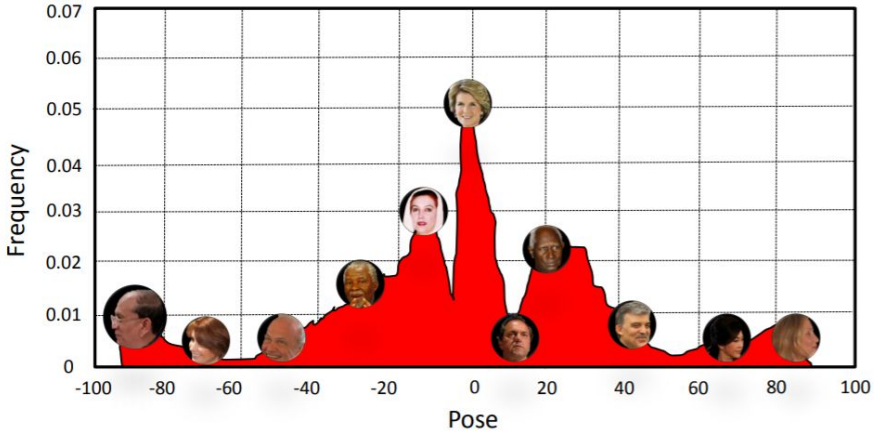
Refinement in the Other Direction

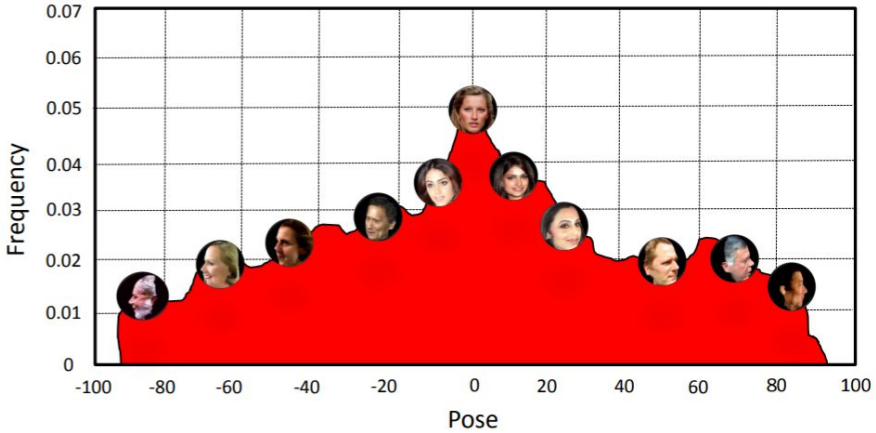
- We can also try to do refinement in the opposite direction: real-to-syn
- Sure we can, but why would we want to do anything like that?

Refinement in the Other Direction

- We can also try to do refinement in the opposite direction: real-to-syn
- Sure we can, but why would we want to do anything like that?

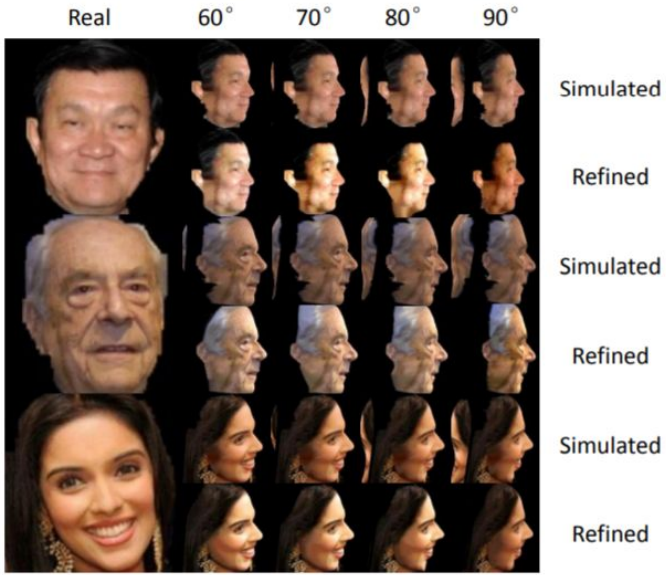
- One answer is to generate realistic data that can fill in the holes in the data distribution
- (Zhao et al., 2018): face recognition in the wild; classical problem, large datasets…
- But they are highly imbalanced; e.g., with respect to pose (angle); can we move from left to right?

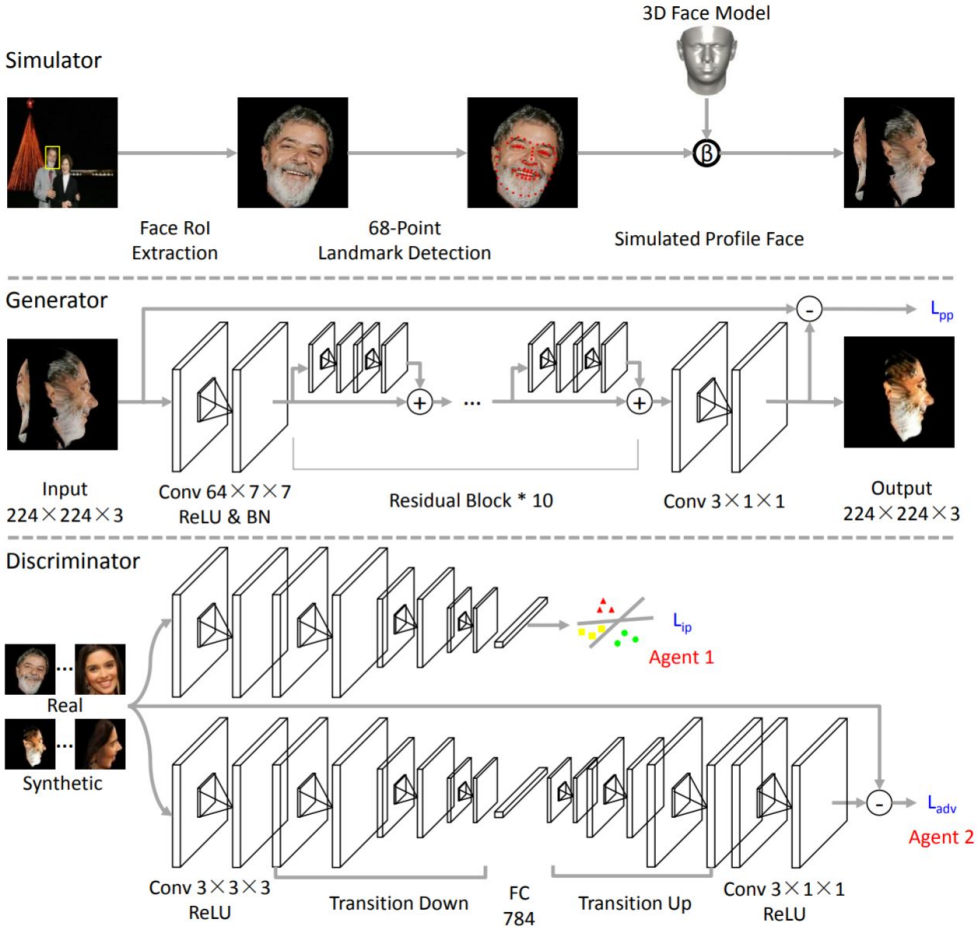




Refinement in the Other Direction

- (Zhao et al., 2018): Dual-Agent GAN
- Let's make new synthetic images by "rotating" real faces with a GAN

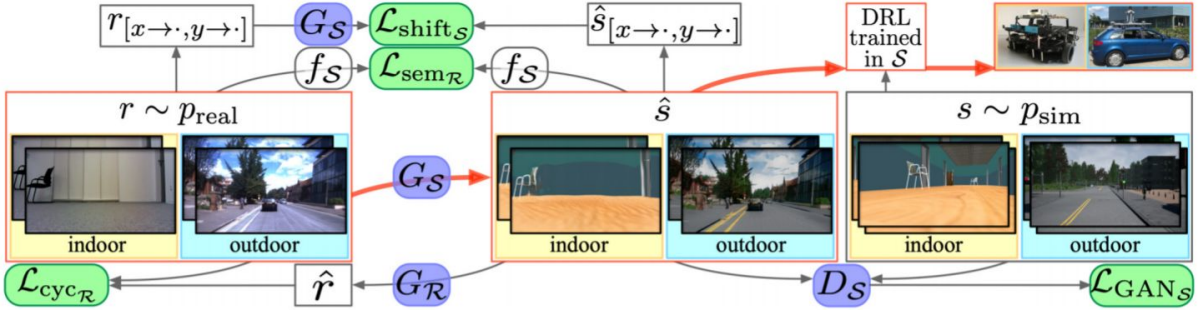




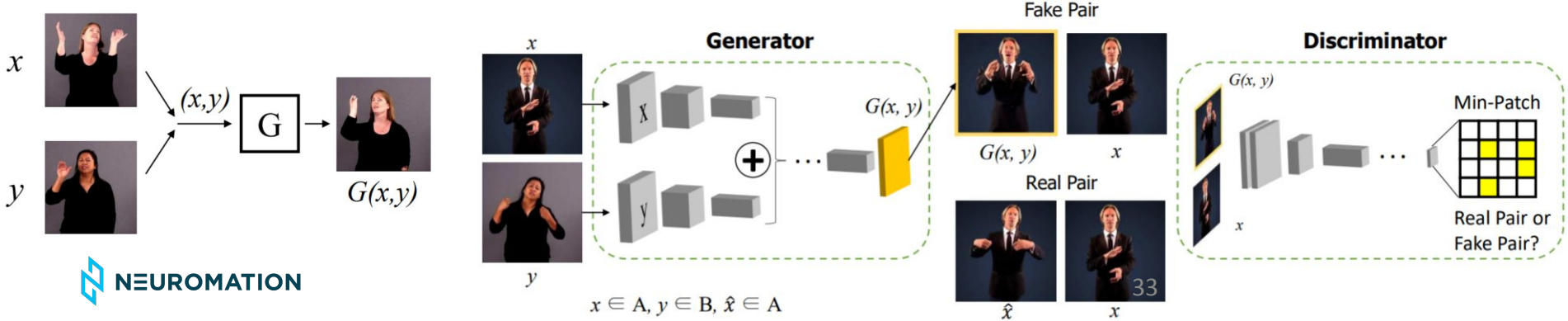
NEUROMATION

Refinement in the Other Direction

- (Zhang et al., 2019): VR Goggles for Robots, again a CycleGAN-based architecture



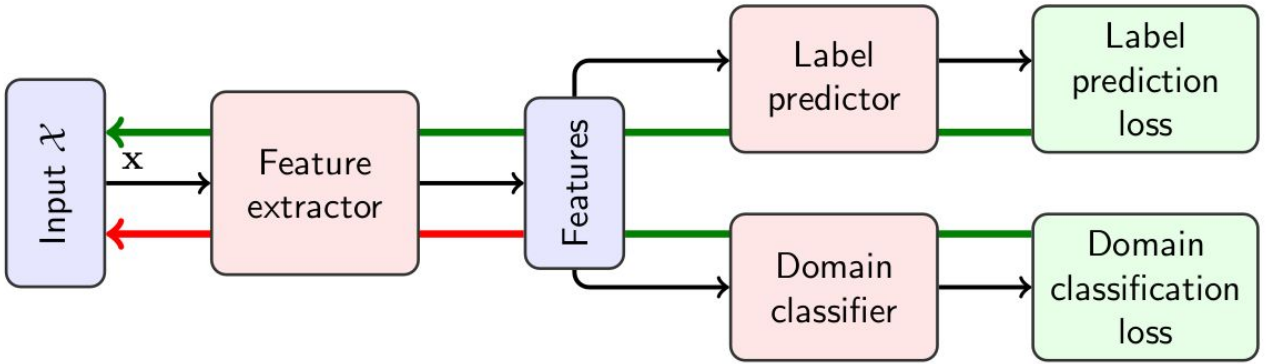
- (Joo et al., 2018): FusionGAN for generating synthetic data from real



NEUROMATION

Model-Level Domain Adaptation

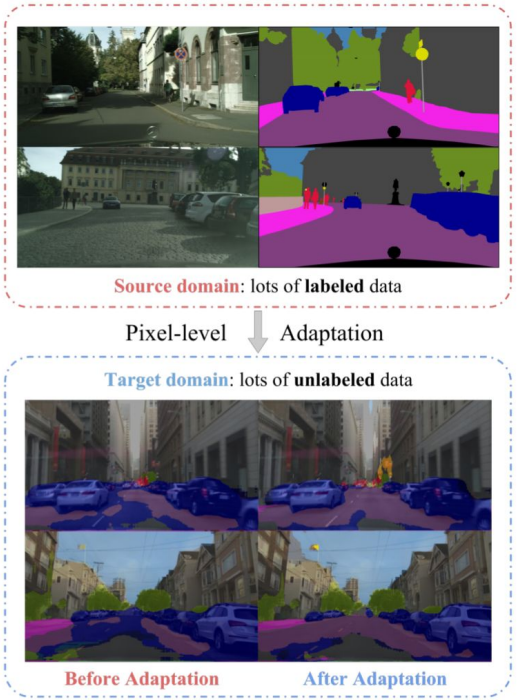
- Another direction is to do domain adaptation in the feature space or during training
- Simplest idea — share weights in feature extractors or learn explicit mappings between features
- Alternative — gradient reversal layers for domain classifiers (Ganin & Lempitsky, 2014):

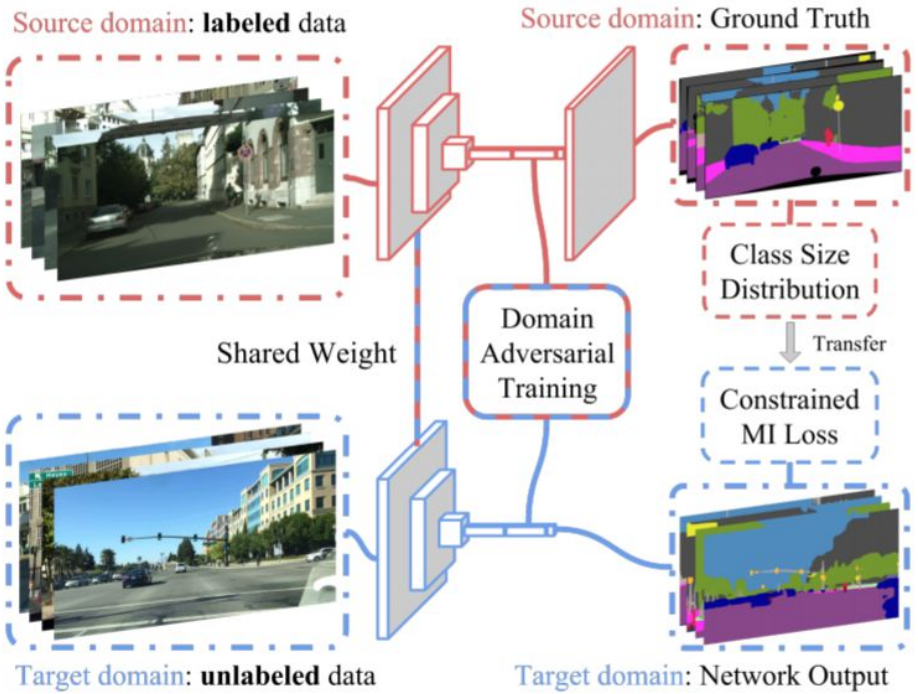


- Such approaches have been used many times for synthetic-to-real adaptation

Model-Level Domain Adaptation

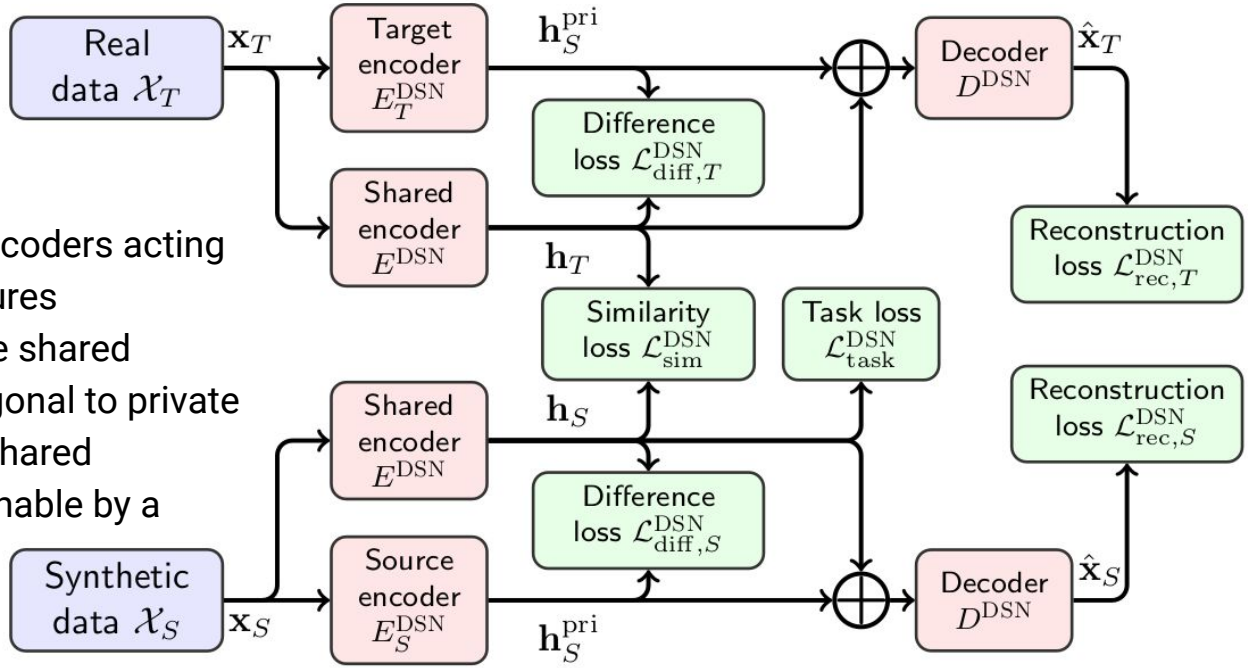
- (Hoffman et al., 2016): FCNs in the Wild
- Shared weights + domain classifier + multiple instance loss for the class size distribution





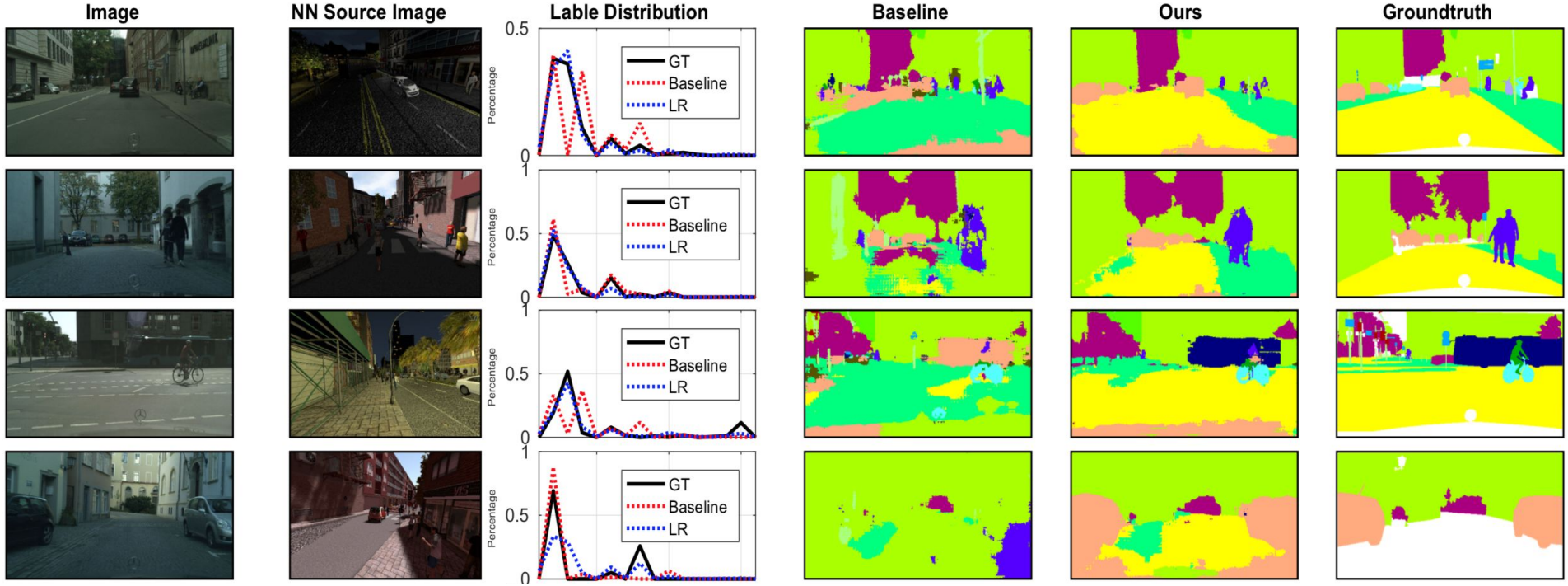
Model-Level Domain Adaptation

- (Bousmalis et al., 2016): **Domain Separation Networks**
- Explicit disentanglement via three encoders, one shared and two private
- Supervised task loss in synthetic data
- Reconstruction loss after decoders acting on the shared + private features
- Difference loss to encourage shared representations to be orthogonal to private
- Similarity loss to make the shared representations indistinguishable by a domain classifier



Model-Level Domain Adaptation

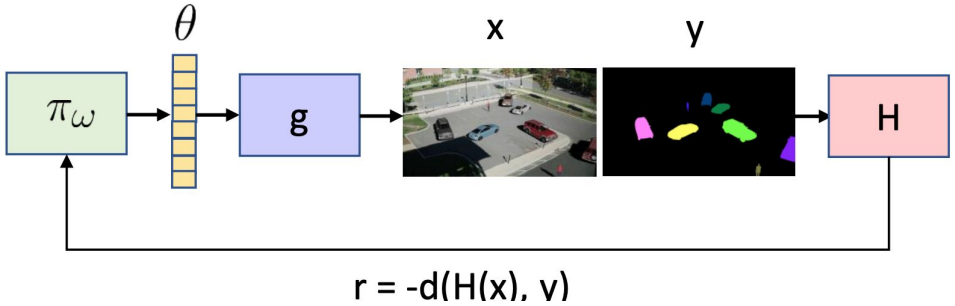
— (Zhang et al., 2019): **Curriculum Domain Adaptation**; let's make the label distribution match



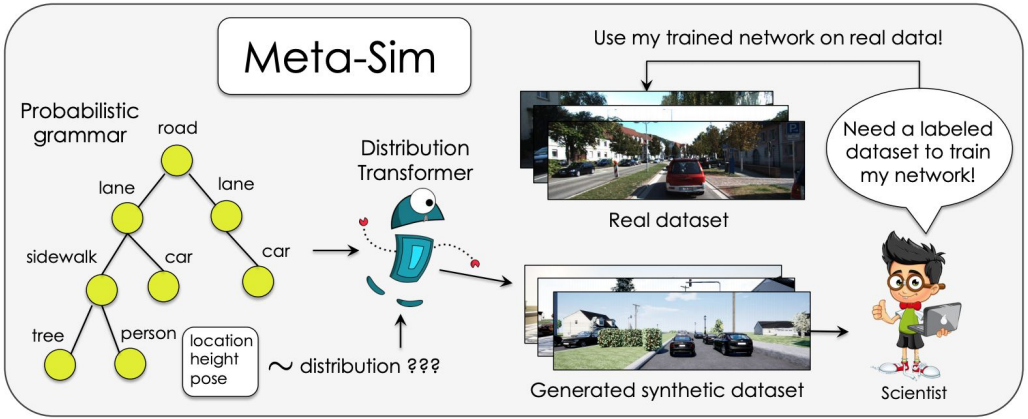
NEUROMATION

Closing the Generation Feedback Loop

- Recent interesting idea: what if we don't go for realism, but go for the goal itself?
- Let's refine synthetic data based on the actual performance of models trained on it!
- (Khirodkar et al., 2018): **VADRA** (Visual Adversarial Domain Randomization and Augmentation)
- **Domain randomization** is the idea of making the synthetic data distribution as wide as possible to (hopefully) cover real data
- VADRA updates synthetic data parameters based on the current policy learned on current synthetic data distribution



- (Kar et al., 2019) **Meta-Sim**: Learning to Generate Synthetic Datasets
- Closing the **content gap** by generating scenes similar to real data via learning probabilistic scene grammars



Conclusions

- Synthetic data is great
- But it incurs the domain transfer problem: how do we train on synthetic and apply on real?
- A big part of the answer is **domain adaptation**:
  - **refinement** that changes the **data**, bringing the real and synthetic distributions closer
    - GAN-based refiner architectures: SimGAN, Cycle-GAN based, etc.
    - real-to-synthetic adaptation
  - **domain adaptation** that changes the **models** in feature space or during training
    - shared weights, gradient reversal layers
    - disentanglement, regularizing with general statistics etc.
- What's next? One interesting direction is to close the **generation feedback loop**
- More? Questions?

Where androids dream of electric sheep

THANK YOU!

neuromation.io