Project name: Learning from Simulated and Unsupervised Images through Adversarial Training

Mohan Krishna Sunkara

UIN: 01206224

Introduction:

Large labeled training datasets are becoming increasingly important with the recent rise in high capacity deep neural networks. However, labeling such large datasets is expensive and time-consuming. Thus the idea of training on synthetic instead of real images has become appealing because the annotations are automatically available. Human pose estimation with Kinect and, more recently, a plethora of other tasks have been tackled using synthetic data. However, learning from synthetic images can be problematic due to a gap between synthetic and real image distributions – synthetic data is often not realistic enough, leading the network to learn details only present in synthetic images and fail to generalize well on real images. One solution to closing this gap is to improve the simulator. However, increasing the realism is often computationally expensive, the renderer design takes a lot of hard work, and even top renderers may still fail to model all the characteristics of real images. This lack of realism may cause models to overfit to 'unrealistic' details in the synthetic images.

Simulated+Unsupervised (S+U) learning is used, where the goal is to improve the realism of synthetic images from a simulator using unlabeled real data. The improved realism enables the training of better machine learning models on large datasets without any data collection or human annotation effort. In addition to adding realism, S+U learning should preserve annotation information for training of machine learning models – e.g. the gaze direction should be preserved. Moreover, since machine learning models can be sensitive to artifacts in the synthetic data, S+U learning should generate images without artifacts.

BACKGROUND:

Adversarial Network:

Generative modeling is the use of artificial intelligence (AI), statistics and probability in applications to produce a representation or abstraction of observed phenomena or target variables that can be calculated from observations. Generative modeling is used in unsupervised machine learning as a means to describe phenomena in data, enabling computers to understand the real world. This AI understanding can be used to predict all manner of probabilities on a subject from modeled data. In unsupervised machine learning, generative modeling algorithms process volumes of training data and make reductions about the data into its digital essence. These models generally are run on neural networks and can come to naturally recognize the natural distinctive features of the data. The neural networks take these reduced fundamental understandings of real world data and then use them to model data that is similar or indistinguishable from real world data

Classification is also traditionally referred to as discriminative modeling. This is because a model must discriminate examples of input variables across classes; it must choose or make a decision as to what class a given example belongs. Alternately, unsupervised models that summarize the distribution of input variables may be able to be used to create or generate new examples in the input distribution. As such, these types of models are referred to as generative models.

GAN

The main focus for GAN (Generative Adversarial Networks) is to generate data from scratch, mostly images but other domains including music have been done. But the scope of application is far bigger than this. Just like the example below, it generates a zebra from a horse. In reinforcement learning, it helps a robot to learn much faster. Generative adversarial networks are based on a game theoretic scenario in which the generator network must compete against an adversary or opponent. The generator network directly produces samples. Its adversary or opponent, the discriminator network, attempts to distinguish between samples drawn from the training data and samples drawn from the generator. The GAN model architecture involves two sub-models: a generator model for generating new examples and a discriminator model for classifying whether generated examples are real, from the domain, or fake, generated by the generator model. Generator. Model that is used to generate new plausible examples from the problem domain. Discriminator. Model that is used to classify examples as real (from the domain) or fake (generated)

Architecture:

SIM GAN Overview:

The output of the simulator is refined with a refiner neural network, R, that minimizes the combination of a local adversarial loss and a 'self-regularization' term. The adversarial loss 'fools' a discriminator network, D, that classifies an image as real or refined. The self-regularization term minimizes the image difference between the synthetic and the refined images. The refiner network and the discriminator network are updated alternately. Simulated+Unsupervised (S+U) learning, not only adding realism to the synthetic images, but also preserving the annotation.

Discriminator D

We got a set of unlabeled real images yi∈Y to learn a refiner Rθ (x) that refines a synthetic image x, and ~x is the refined image.

The discriminator network, D-, is trained to classify the images as real vs refined. Thus, for discriminator,

$$\mathcal{L}_D(\phi) = -\sum_i \log(D_{\boldsymbol{\phi}}(\tilde{\mathbf{x}}_i)) - \sum_j \log(1 - D_{\boldsymbol{\phi}}(\mathbf{y}_j))$$

it is the cross-entropy error for a two class classification problem: where DΦ-(.) is the probability of the input being a synthetic image, 1 - DΦ-(.) that of a real one.

Refiner R (Generator)

A refiner Rθ(x) that refines a synthetic image x, where θ is the parameters of refiner network R.

$$\tilde{\mathbf{x}} := R_{\boldsymbol{\theta}}(\mathbf{x})$$

where ~x is the refined image. The refined image ~x should look like a real image in appearance while preserving the annotation information from the simulator. The θ, the parameters of refiner network R, is learnt minimizing a combination of two losses:

$$\mathcal{L}_R(\boldsymbol{\theta}) = \sum_i \ell_{\text{real}}(\boldsymbol{\theta}; \mathbf{x}_i, \mathcal{Y}) + \lambda \ell_{\text{reg}}(\boldsymbol{\theta}; \mathbf{x}_i),$$

The first part of the cost, lreal, adds realism to the synthetic images, while the second part, lreg, preserves the annotation information. To add realism to the synthetic image, we need to bridge the gap between the distributions of synthetic and real images. To do this, the realism loss function lreal is:

$$\ell_{\text{real}}(\boldsymbol{\theta}; \mathbf{x}_i, \mathcal{Y}) = -\log(1 - D_{\boldsymbol{\phi}}(R_{\boldsymbol{\theta}}(\mathbf{x}_i))).$$

By minimizing this loss function, the refiner forces the discriminator to fail classifying the refined images as synthetic. In addition to generating realistic images, the refiner network should preserve the annotation information of the simulator. For example, for gaze estimation the learned transformation should not change the gaze direction, and for hand pose estimation the location of the joints should not change. A self-regularization loss, that minimizes per-pixel difference between a feature transform of the synthetic and refined images, is used:

$$\ell_{\text{reg}} = \|\psi(\tilde{\mathbf{x}}) - \mathbf{x}\|_1,$$

where Ψ is the mapping from image space to a feature space, and l1 norm is used. The feature transform Ψ can be identity mapping, image derivatives, mean of color channels, or a learned transformation. Thus, the overall refiner loss function becomes:

$$\begin{aligned}\mathcal{L}_R(\boldsymbol{\theta}) = &-\sum_i \log(1 - D_{\boldsymbol{\phi}}(R_{\boldsymbol{\theta}}(\mathbf{x}_i))) \\ &+\lambda \|\psi(R_{\boldsymbol{\theta}}(\mathbf{x}_i)) - \psi(\mathbf{x}_i)\|_1.\end{aligned}$$

Rθ is as a fully convolutional neural net without striding or pooling, modifying the synthetic image on a pixel level, rather than holistically modifying the image content as in e.g. a fully connected encoder network, thus preserving the global structure and annotations.

Local Adversarial Loss

When training a single strong discriminator network, the refiner network tends to over-emphasize certain image features to fool the current discriminator network, leading to drifting and producing artifacts. Yet, any local patch sampled from the refined image should have similar statistics to a real image patch. Therefore, rather than defining a global discriminator network, a discriminator network should be defined that classifies all local image patches separately. The discriminator D is designed to be a fully convolutional network that outputs w×h dimensional probability map belonging to the fake class, where w×h are the number of local patches in the image. The adversarial loss function is the sum of the cross-entropy losses over the local patches.

Updating Discriminator using a History of Refined Images

Another problem of adversarial training is that the discriminator network only focuses on the latest refined images. This lack of memory may cause (i) divergence of the adversarial training, and (ii) the refiner network re-introducing the artifacts that the discriminator has forgotten about. A method to improve the stability of adversarial training is to update the discriminator using a history of refined images. Let B be the size of the buffer and b be the mini-batch size. The discriminator loss function is computed by sampling b/2 images from the current refiner network, and sampling an additional b/2 images from the buffer to update parameters Φ. After each training iteration, b/2 samples in the buffer is randomly replaced with the newly generated refined images.

Dataset: Dataset used in this paper was unity Eyes to generate ~1.2 million synthetic images. The dataset of real image's used is the MPIIGaze Dataset. They use the normalized images provided in this dataset which are stored in matlab files.

The synthetic images were generated with the windows version of UnityEyes
http://www.cl.cam.ac.uk/research/rainbow/projects/unityeyes/tutorial.html

The real images were taken from

https://www.mpi-inf.mpg.de/departments/computer-vision-and-machine-learning/research/gaze-based-human-computer-interaction/appearance-based-gaze-estimation-in-the-wild

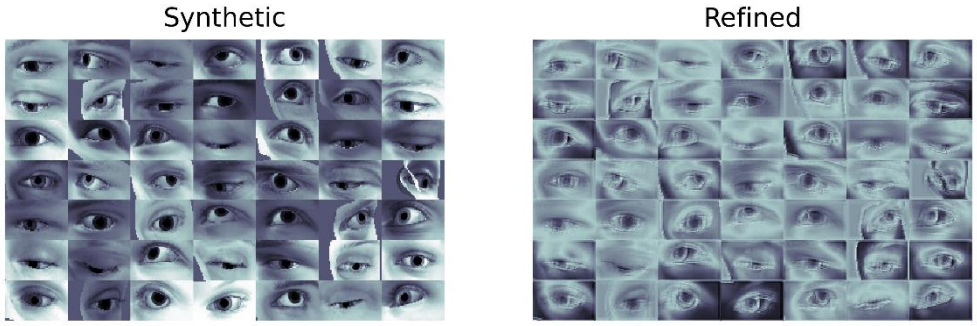
Language Used: Python

Packages Used: Tensorflow, Keras and Numpy and from keras we used Applications, Layers, Models, Optimizers, Image

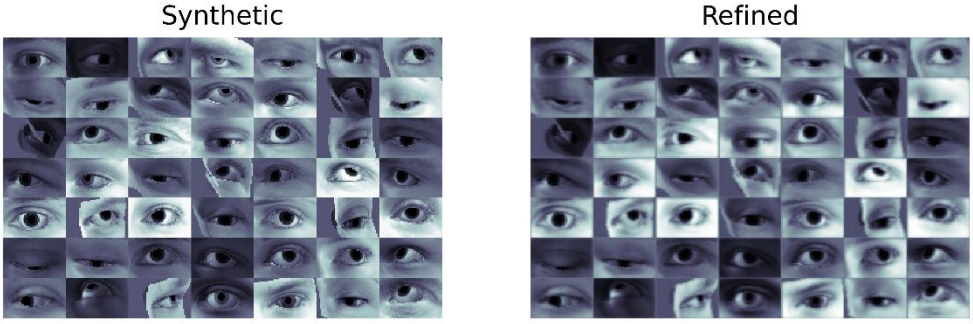
Results:

Visual Turing Test: To quantitatively evaluate the visual quality of the refined images, we designed a simple user study where subjects were asked to classify images as real or refined synthetic. Each subject was shown a random selection of few real images and few refined images in a random order and was asked to label the images as either real or refined.

Synthetic images and refined images before start of training:



Synthetic images and refined images during training:



Conclusion:

We have proposed Simulated+Unsupervised learning to add realism to the simulator while preserving the annotations of the synthetic images. We described SimGAN, our method for S+U learning, that uses an adversarial network and demonstrated state-of-the-art results without any labeled real data. In future, we intend to explore modeling the noise distribution to generate more than one refined image for each synthetic image, and investigate refining videos rather than single images.