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

Student name: Mohan Krishna Sunkara

Student UIN: 01206224

1. An introduction about the background knowledge and why this project is important.

Answer: 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.

2. What is the project to be conducted? What are the data to be used? What are the goals?

Answer: With recent progress in graphics, it has become more tractable to train models on synthetic images, potentially avoiding the need for expensive annotations. However, learning from synthetic images may not achieve the desired performance due to a gap between synthetic and real image distributions. To reduce this gap, Simulated+Unsupervised (S+U) learning is proposed, where the task is to learn a model to improve the realism of a simulator's output using unlabeled real data, while preserving the annotation information from the simulator. A method called S+U learning that uses an adversarial network similar to Generative Adversarial Networks (GANs), but with synthetic images as inputs instead of random vectors. Several key modifications was made to the standard GAN algorithm to preserve annotations, avoid artifacts and stabilize training:

(i) a 'self-regularization' term, (ii) a local adversarial loss, and (iii) updating the discriminator using a history of refined images. It is showed that this enables generation of highly realistic images, which we demonstrate both qualitatively and with a user study. It is showed that a significant improvement over using synthetic images, and achieve state-of-the-art results on the MPIIGaze dataset without any labeled real data.

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.

3. What methods are your going to use for the project? What are the main steps to be worked out?

Answer: We use Simulated+Unsupervised (S+U) learning. A method called S+U learning that uses an adversarial network similar to Generative Adversarial Networks (GANs), but with synthetic images as inputs instead of random vectors. Below are the main steps to be worked out.

Loading Data: First we create synthetic data and collect real eye images and then load them into the program.

Network Architectures: Then we define refiner and discriminator network. They work against each other to constantly improve the results of the other one.

Combining Models: Adversarial training of refiner network $R\theta$ and discriminator network $D\phi$ and combining them into single loss-functions and models that can be trained.

Compile Models: Here we combine the models and compile them with the loss functions defined above binding them into single loss-functions and models that can be trained.

Data Generators: Here we setup the pipeline to feed new images to both models.

Pretraining: Here we pretraining the models before we start the full-blown training procedure.

Full Training: Here we run the full training of the model (normally for thousands of iterations, but here just a few). The images are saved in the output directory for examining the training.

4. What are potential results, in what format?

Answer: We evaluate our method for appearance-based gaze estimation in the wild on the MPIIGaze dataset. We use fully convolutional refiner network with ResNet blocks for all our experiments. Output of the model was refined images.

‘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. The subjects were constantly shown few examples of real and refined images while performing the task. The subjects found it very hard to tell the difference between the real images and the refined images. In contrast, when testing on original synthetic images vs real images, we showed 10 real and 10 synthetic images per subject, and the subjects chose correctly and was easily able to tell the difference between them.

5. What kind of the packages are needed to be used in your project development?

Answer: We used

1. Tensorflow,
2. Keras
3. Numpy

And from keras we used

1. Applications
2. Layers
3. Models
4. Optimizers
5. Image