Sim-to-Real

David Oniani
www.github.com/oniani/sim-to-real

Abstract

Can one transfer knowledge between simulation data and real data in order to improve the classification ability of said real world data? With the growing field of sim-to-real research, scientists argue that it is not only possible, but useful too. We use dataset which is comprised of simulation (e.g., video games, 3D virtual worlds) and real-world images. There are approximately 8941 images in total. Each of these two classes have 6 subclasses: Living Room, Bathroom, Staircase, Forest, Field, Computer Lab.

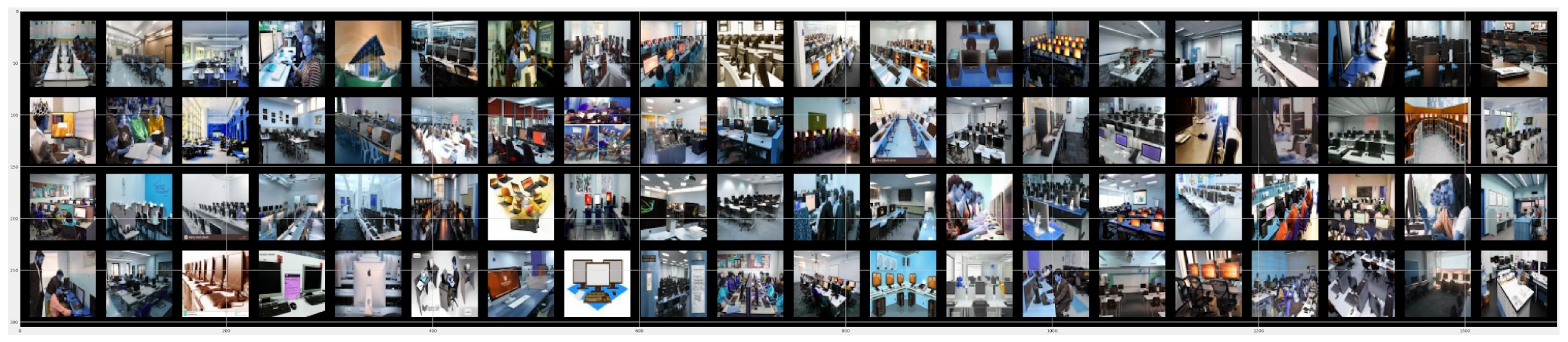
We attempt to answer the following questions:
- Can we classify images based on the given 6 classes?
- Is it possible to differentiate real and simulation images?
- Does the combination of real and simulation images perform better than using any one of them alone?
- Do DL models outperform traditional ML techniques?

Dataset

Notice that classes are unbalanced. In order to balance out both classes, we randomly select 369 images from from both categories across all classes. Therefore, we end up with 369 x 6 x 2 = 4428 images in total. However, it is generally suggested that every class has at least 1000 images. Indeed, performing the training without data augmentation has given us worse results which are not included in this poster.

Class	Real	Virtual
Living Room	369	1065
Bathroom	597	1034
Staircase	491	708
Forest	583	1007
Field	377	1085
Computer Lab	596	1029

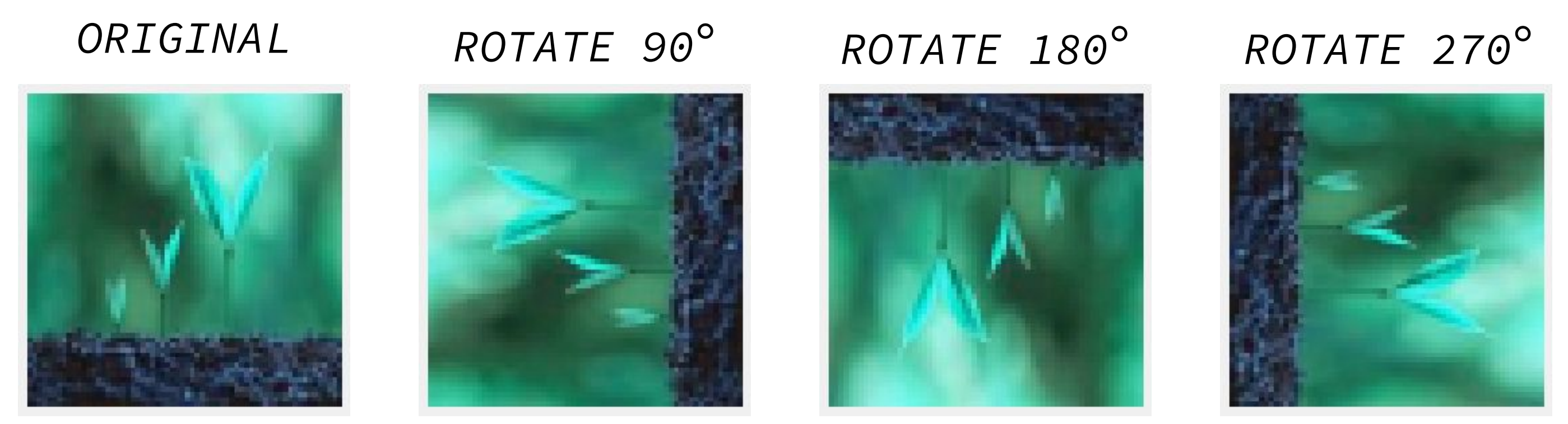
Below find some of the images from the Computer Lab class. They were randomly selected from the 64x64 real images.



Data Augmentation

Data augmentation is a technique to increase the amount of data by adding slightly modified copies of already existing data or newly created synthetic data from existing data. In this case, we will consider three different image rotations. Namely, 90°, 180°, and 270° rotations.

Class	Real	Virtual
Living Room	1476	1476
Bathroom	1476	1476
Staircase	1476	1476
Forest	1476	1476
Field	1476	1476
Computer Lab	1476	1476



Traditional Machine Learning

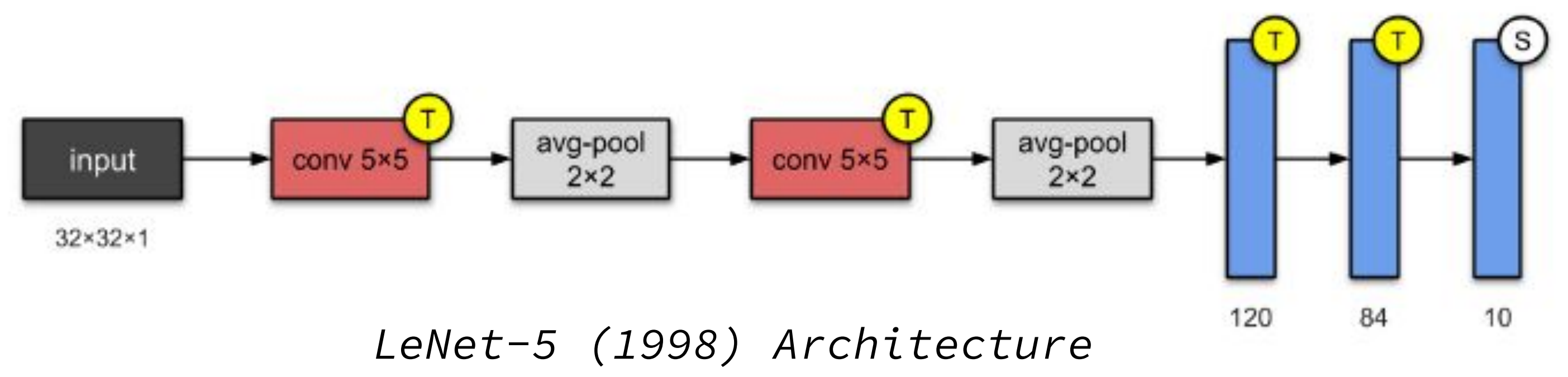
From the traditional approaches, we have applied SVM classifier. We have used `SGDClassifier` of `scikit-learn` that implements a plain stochastic gradient descent learning routine. It was trained with the hinge loss, equivalent to a linear SVM. We have run the parallelized training for 10000 iterations on the training dataset and present the metrics obtained on the test dataset below (80/20 split).

Variation	Accuracy	Precision	Recall	F1 Score
Real	0.35	0.34	0.34	0.33
Virtual	0.36	0.46	0.35	0.25
Real + Virtual	0.28	0.38	0.29	0.24

Given results show that all three models performed poorly. We will proceed by applying Convolutional Neural Networks (CNN).

Convolutional Neural Networks

Convolutional Neural Network (CNN) is a deep learning architecture that has been proved to work well on image data. It is often composed of multiple layers that include weights, biases, activation functions, and the loss function. An example of a CNN is shown below.



LeNet-5 (1998) Architecture
Y. LeCun, et al.

Convolutional Neural Networks allow for automatically learning a large number of filters in parallel specific to a training dataset under the constraints of a specific predictive modeling problem, such as image classification. The result is highly specific features that can be detected anywhere on input images.

Custom Model Architecture

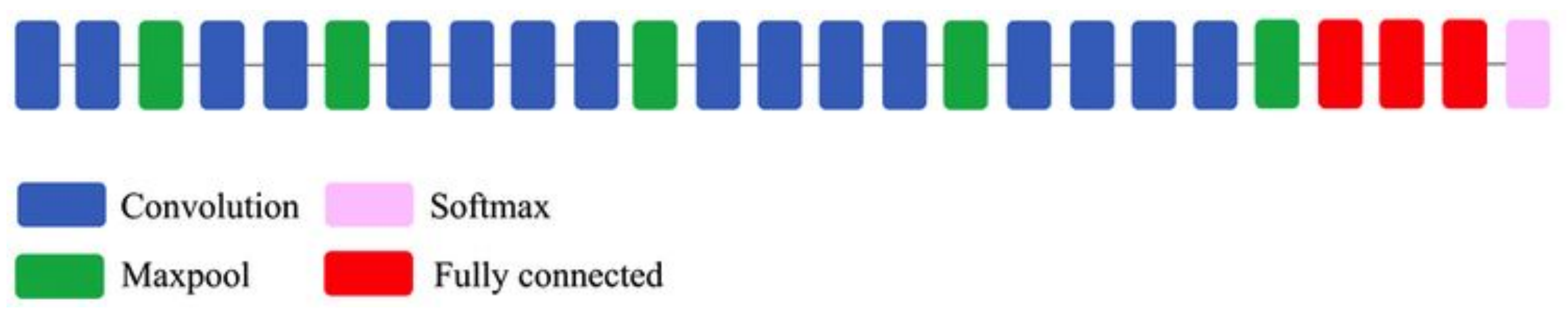
We started with a simple architecture comprised of three convolutional, three batch normalization, two linear, one ReLU, and one log softmax layer. We trained the CNN with the following parameters:
- Epochs: 25
- Optimizer: Adam
- Learning Rate: 0.0001
- Loss Function: Cross Entropy Loss

The results of the training are shown below:

Variation	Accuracy	Precision	Recall	F1 Score
RealNet	0.57	0.58	0.57	0.57
VirtNet	0.73	0.73	0.73	0.73
AllNet	0.63	0.64	0.63	0.63

VGG19 Architecture

VGG19 is a variant of VGG model which consists of 19 layers:
- 16 convolution layers
- 3 fully connected layers
- 5 MaxPool layers
- 1 SoftMax layer



Training such on the CPU takes a lot of time. Therefore, it is generally recommended to delegate the work onto the GPU. NVIDIA P100-PCIE was used as the GPU.

VGG19 Training Results

The following parameters were used for the training:
- Epochs: 25 (50 for `AllNet`)
- Optimizer: Adam
- Learning Rate: 0.0003
- Loss Function: Cross Entropy Loss

Variation	Accuracy	Precision	Recall	F1 Score
VGG19 RealNet	0.79	0.80	0.79	0.79
VGG19 VirtNet	0.96	0.96	0.96	0.96
VGG19 AllNet	0.85	0.85	0.85	0.85

As shown in the table above, `VGG19` has shown significant improvements over linear SVM as well as the custom CNN. Values for all metrics have nearly doubled as compared with the `SGDClassifier`. `VGG19` reported, on average, 30% better statistics than the custom CNN. The loss in `VGG19 RealNet` and `VGG19 VirtNet` converged at 25 epochs while it took approximately 50 epochs for `VGG19 AllNet` loss to converge.

Outcomes

We have found several important outcomes:
- Traditional machine learning approaches underperform significantly in comparison with deep learning models
- Custom-made CNN performed a lot better than the traditional ML approach
- A tried-and-true VGG19 outperformed custom-made model
- We have classified 6 classes of images with high accuracy within each category (real/virtual)
- It appears that combining images gives even higher accuracy meaning that it might be helpful to combine real and virtual images

References

- Scene Classification: Simulation to Reality
- From Simulation to Reality: CNN Transfer Learning for Scene Classification
- Very Deep Convolutional Networks for Large-Scale Image Recognition