21 January 2018

Object Recognition and Computer Vision
Training CNNs using synthetic images of people

Marc Etheve
Master 2 MVA
ENS Paris-Saclay
marc.etheve@ens-paris-saclay.fr

Jean-Baptiste Remy
Master 2 MVA
ENS Paris-Saclay
jean-baptiste.remy@ens-paris-saclay.fr

1. Introduction

Neural Networks, as good as they may perform, require a huge amount of training data. In case of supervised learning, which accounts for the majority of the tasks, these training data have to be labeled and therefore are more expensive. To tackle this issue, the article *Learning from Synthetic Humans* [1] proposes to train a model on computer generated images of humans and then to test it on real data.
We propose to explore two possibilities to enhance the performance of the model on real images of humans, namely increasing the resemblance between source and target data and learning features which are pertinent on both the distributions. In the following, we first modify the synthetic images in order to make them more realistic. This is achieved by aligning the brightness of the human on background, creating occlusions and perform data augmentation (Marc Etheve). Secondly, we use a domain adversarial version of the stacked hourglass model to perform 2D pose estimation, i.e. pixel level identification of the joints (Jean-Baptiste Remy).

2. Enhancing and preprocessing images

In order to improve the generalization to real images of a model trained on synthetic ones, the easiest way is to increase the resemblance between synthetic and real images.

2.1. Rationalizing the brightness

The SURREAL dataset [1] is composed of MoCap human images pasted on random backgrounds. The lighting of the background is random, as well as the coefficients in the Spherical Harmonics model [2] used to lighten the human body.
As a consequence, a proper way of rationalizing the lighting between the human and the background would be to fit a Spherical Harmonics model on the background, estimate the coefficients and apply them to the body. This procedure being quite difficult to implement, we propose here only to center the human's brightness (Equation 2) on the background's average (Equation 1), N_A denoting the number of pixels in set A and $\mathcal{P}$ the set of pixels belonging to i (see an example in Figure 6.2).

$$B_A = \frac{1}{3N_A} \sum_{p=1}^{N_A} (R_p + G_p + B_p) \tag{1}$$

$$RGB_i \leftarrow RGB_i - B_{hum} + B_{bg} \; \forall \; i \in \mathcal{P}_{hum} \tag{2}$$

2.2. Create random occlusions

A second feature to add to the SURREAL dataset is occlusions. The most realistic way of creating random occlusions would be to randomly pick segmented objects and paste them on SURREAL images, with appropriate brightness.
Again, this strategy requires a lot of engineering, hence an easier procedure has been preferred. First, we define a rectangle area using the segmentation matrix to occlude the human, in a random proportions and at a random location. Then, we accordingly define available areas (Figure 5) in which we can randomly pick a corresponding area without overlapping with the human. Finally, we paste the selected area on top of the human to create an occlusion (Figure 6).
This procedure allow us not to use external data, can be performed with modularity (probability of occlusion, proportions' range of the occlusions, spectific location of the occlusions) and at low cost.

2.3. Preprocessing images to feed the Neural Network

Finally, we create both uniformity and randomness in the framing of the pictures to feed the Neural Network. This is done by resizing them using bilinear interpolation, centering the image on the human by cropping and randomly rotating the image around the human's barycenter.

3. Deep Neural Network Architecture

Two dimensional pose estimation is the task of finding the articulations (joints) of a human on a frame. The model we use here performs pixel level classification for every joint.

3.1. Stacked Hourglass

To perform this task we use the stacked hourglass architecture as described in *Stacked Hourglass Networks for Human Pose Estimation* [3]. This network is composed of a succession of hourglasses. Each hourglass performs several rounds of max pooling to downgrade the resolution of the features down to a minimal resolution of 4×4, then upscales the resolution the same number of times to output features of the original resolution. Throughout the hourglass, the network branches off at every max pooling layer to keep the spatial information, the unpooled features being then added with the upscaled ones. Each hourglass outputs probability heat maps of presence for each joint in the image.

3.2. Domain Adversarial Learning

To train the network to adapt to real images of humans we use domain adversarial training. Let's consider the classification task $\phi(x_i) = \hat{y}_i$, with L possible labels. Two domains are identified, S being the labeld source domain used to learn the predictors, and T being the unlabeld target domain on which we the prediction may be performed. We can formulate this as

$$S = \{(x_i, y_i)\}_{i=1..n} \sim (D_S)^n \quad T = \{x_i\}_{i=n+1..N} \sim (D_T^X)^{n'} \tag{3}$$

where D_A stands for the sample's distribution over the domain A, and D_A^X the marginal distribution of X over domain A. $N = n + n'$ is the total number of samples, and we want to minimize the classification risk :

$$R_{D_T}(\phi) = \mathbb{P}_{D_T}\Big(\phi(x) \neq y\Big). \tag{4}$$

To measure the divergence between S and T, one can use the $\mathcal{H}$-divergence[1]. If $\mathcal{H}$ is a class of classifiers, and if $Y \in \{0, 1\}$:

$$d_{\mathcal{H}}(D_S^X, D_T^X) = 2 \sup_{\phi \in \mathcal{H}} \Big|\mathbb{P}_{D_S^X}[\phi(x) = 1] - \mathbb{P}_{D_T^X}[\phi(x) = 1]\Big| \tag{5}$$

We define $\mathcal{H}\Delta\mathcal{H} = \{g / g(x) = 1 - \mathbb{1}_{\phi(x)=\phi'(x)}$ with $\phi, \phi' \in \mathcal{H}\}$. Then we can bound the risk of a classifier on the target distribution :

$$R_{D_T}(\phi) \leq R_{D_S}(\phi) + \frac{1}{2}\hat{d}_{\mathcal{H}\Delta\mathcal{H}}(D_S^X, D_T^X) + 4\sqrt{\frac{4d\log(2m) + \log(\frac{2}{\delta})}{m}} + \beta \tag{6}$$

Where m is the size of both unlabeled sets U_S (here $\emptyset$) and U_T, $1 - \delta$ is the probability to draw from U_S over U_T, and $\beta = \min_{\phi' \in \mathcal{H}} R_{D_T}(\phi') + R_{D_S}(\phi')$.
This results tells us that, in order to minimize the risk on the target domain, we must achieve a low risk on both sets at the same time. It also tells us that for a class of finite Vapnik-Chervonenkis dimension we can find a classifier by minimizing a trade-off between the source risk and the empirical $\mathcal{H}\Delta\mathcal{H}$-divergence.

In practice (see [5]), the output of the network is fed into a fully connected layer to classify between the source and the target set, and a gradient reversal layer inverses the gradient of the loss of this classification before it is propagated to the rest of the network. In that way the model learns domain independent features, which do not allow to discriminate the source from the target.

3.3. Implementation

As we wanted to fully understand how the network is built and to properly control the adding of the domain adversarial module, we chose to implement the full network from scratch using Tensorflow[2] (see Figure 7). The input of the network is classically preprocessed to perform data augmentation (rotation and cropping) then resized to fit the input dimension of 256×256. The labels take the form of a one hot vector telling on which pixel is the joint. The training loss used is the cross entropy classification loss, fed into a RMSprop optimizer with decaying learning rate. At first, we used a constant learning rate, which led the network to predict every joints at (0,0) during the training process. This issue has been fixed using a decaying learning rate. To compute the training and prediction of the algorithm we used an Azure virtual machine with an Nvidia Tesla K80 GPU.

4. Results

The experiment we ran aimed at comparing the results between : (i) a classical Stacked Hourglass network trained on raw images from the SURREAL dataset, (ii) a classical Stacked Hourglass trained on enhanced images, and (iii) a Stacked Hourglass with Domain Adversarial Learning trained on raw images.
We did not have time to train very deep networks, so we

[1] Ben-David and al., 2006, 2010; Kifer and al. 2004

[2] Complete code can be found on GitHub (github.com/JbRemy/Leaning_on_synthetic_humans)

Figure 1. Raw training data

Figure 2. Preprocessed training data

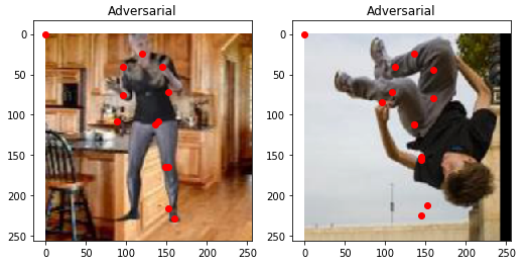


Figure 3. Raw training data with Domain Adversarial Learning

trained single-stack networks on a small subset of 10000 (randomly extracted) and just a few epochs. Their performances are then quite low, but we are more interested here in the comparison between the models.

4.1. Images

First, it is always preferable to compare the results of the networks on synthetic data, to ensure they effectively learnt. Then we can observe the results on a real image instance. Figures 1, 2 and 3 shows that the three models perform very well on synthetic data, but quite poorly on real images. Let's notice that the domain adversarial version seems to be the worst on synthetic images, but the best on real images.

4.2. Metrics

Two metrics have been used to compare the models. First, the Root Mean Squared Error (RMSE) of the joints position, but it appeared to not really fit to pose estimation. Indeed, when a joint is badly predicted, it does not mean much if it is close or far from the real one. We then chose to add a better fitting metric, namely the Percent of Detected Joints (PDJ), where a joint is detected when the distance between the predicted and true joint is lower than a proportion of the torso width. Here the PDJ is calculated on the elbows. We show these metrics in Table 1.

Model	PDJ .05	PDJ .2	PDJ .4	RMSE
Classic	.005	.05	.2	52.69
Enhanced Data	.005	.06	.2	45.01
Adversarial	**.005**	**.07**	**.36**	**43.36**

Table 1. Evaluated metrics on test data

5. Conclusion

Even though the results must be taken into account with caution due to the low training time the network were given, our analysis indicates that a good way to enhance the performances of a network trained on synthetic data is to use domain adversarial learning. In the article *Learning from Synthetic Humans* [1], a model trained on a mix of synthetic and real images is also proposed. This model could surely have better performances if it were trained with a domain adversarial module. The advantage is that it is easy to implement on top of an already existing model.

6. Appendix

6.1. References

[1] Varol, G., Romero, J., Martin, X., Mahmood, N., Black, M., Laptev, I., & Schmid, C. (2017). Learning from synthetic humans. arXiv preprint arXiv:1701.01370.

[2] Green, R. (2003, March). Spherical harmonic lighting: The gritty details. In Archives of the Game Developers Conference (Vol. 56, p. 4).

[3] Newell, A., Yang, K., & Deng, J. (2016, October). Stacked hourglass networks for human pose estimation. In European Conference on Computer Vision (pp. 483-499). Springer International Publishing.

[4] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., ... & Lempitsky, V. (2016). Domain-adversarial training of neural networks. Journal of Machine Learning Research, 17(59), 1-35.

[5] Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., & Vaughan, J. W. (2010). A theory of learning from different domains. Machine learning, 79(1), 151-175.

6.2. Figures

(a) Non-centered image

(b) Centered image

Figure 4. Centering human brightness

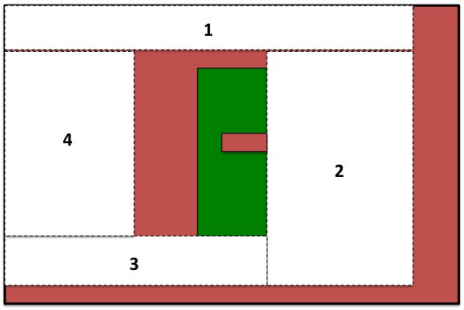
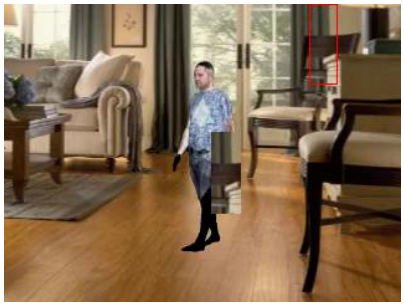


Figure 5. Definition of available areas according to the occlusion area



(a) Create random occlusions (a)

(b) Create random occlusions (b)

Figure 6. Create random occlusions using the background

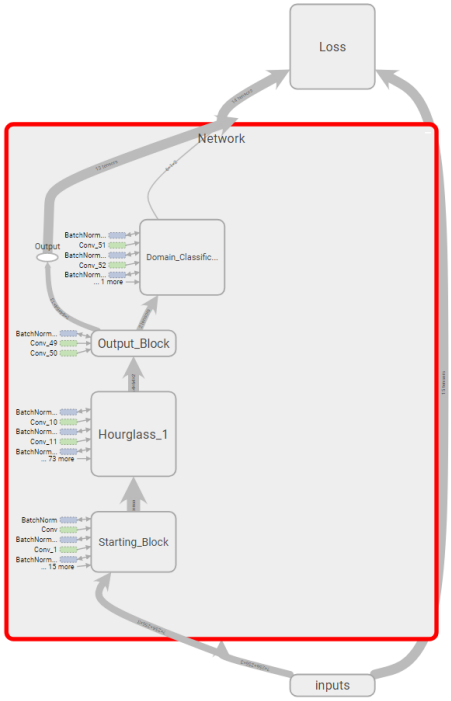


Figure 7. Domain adversarial stacked hourglass