The 33rd Annual Conference of the Japanese Society for Artificial Intelligence, 2019

Likelihood distribution of Pedestrian Trajectories rendered by Variational Autoencoder

Yasunori Yokojima[*1] Tatsuhide Sakai[*2]

[*1] Siemens K.K. [*2] Great Wall Motor

We studied applicability of Variational Autoencoder (VAE) to capture stochastic nature of pedestrian moves in a public space without explicit labels. Movies for training the network were recorded in a public pedestrian street and an exhibition booth. These movies were converted to grayscale images representing observed pedestrian locations and occupied areas. VAE was trained on 90% of data and rest of data was kept for validation. The validation result showed satisfactory reconstruction performance of pedestrian distributions in video frames. We propose a novel method to render our expectation of finding a pedestrian in a crowd as 2-D images by utilizing the trained network. Images rendered by this method correspond to subjective images usually only captured in our mind.

1. Introduction

Building reliable traffic system requires a realistic description of pedestrian behaviors. For example, one of challenges in advanced safety system developments is to find rare but important corner cases, which are likely to be traced back to a subtle but non-negligible pedestrian behavior.

Pedestrian behaviors in a public space are explained by logical relationships and stochastic factors. A logical relationship of pedestrian trajectories is studied for years and various types of models are proposed to analyze universal traffic phenomena. Those models include Social force model, Optimal velocity model, Car-following model, and Cellular Automaton (CA) based model.

The stochastic and statistic nature of pedestrian behaviors is likely to be influenced by individual experiences, subjective recognitions of regional trends, or expectations in mind. For example, there can be a trend to yield right side in a public space in a certain region when two pedestrians are passing each other. Although such stochasticity can be taken into account by assuming a probability distribution and parameters, is it possible to learn the probability distribution directly from pedestrian data?

In the meantime, we occasionally recognize certain patterns while we stare at pedestrians in a public crowd. This is a subjective process of building expectations to pedestrian moves. These expectations give intuitive motive for a pedestrian to decide behaviors in a crowd. To take this into account in a pedestrian modeling, is it possible to mimic this subjective process by unsupervised learning on pedestrian data?

For these backgrounds, our goal is twofold: firstly study applicability of unsupervised machine learning to capture stochastic aspect of pedestrian behaviors, and secondly render learned content as a representation of our expectations to find an individual in a crowd. In the following sections, we propose a method to learn pedestrian distribution in a public space using VAE [Kingma 2014].

Contact: Yasunori Yokojima, Siemens K.K., Address, Phone number, yasunori.yokojima@siemens.com

2. Learning pedestrian trajectories

2.1 Data Preprocessing

Our focus is to train a network on pedestrian locations and occupied areas. For this purpose, training data is prepared in the following process.

Pedestrian movies are recorded at 30 FPS by a single eye camera mounted at a fixed location with a certain elevation angle to a floor of pedestrian traffic. Movies are recorded at two different locations by this setup: movies recorded at a public pedestrian street have approximately 90 second length (Street data, hereafter). Movies recorded at an exhibition booth have approximately 30 minutes length (Booth data, hereafter).

For each frame in these movies, we applied YOLO v3 [Redmon 2015] to detect pedestrians and selected middle points of lower bases of detected bounding boxes as approximated pedestrian positions. Pedestrian positions are transformed to positions in a 2-D rectangular area while keeping the aspect ratio same as the original area. The resulted view is corresponding to a perspective perpendicular to the floor. Based on these pedestrian positions we generate grayscale images with white circles represent areas occupied by pedestrians on a black rectangular background. These images are resized to 10% in size of the original images by using INTER_AREA algorithm in OpenCV. Pixels in an image take value from 0 to 255. These images are fed as inputs to train the VAE network (Figure 3).

2.2 VAE network

VAE network structure is shown in Figure 1. We assume the latent distribution to follow a Gaussian distribution [Doersch 2016]. A loss function is defined as the following form:

$$L(x) = -D_{KL}\big(q_\phi(z|x)\|p(z)\big) + E_{q_\phi(z|x)}[\log p_\theta(x|z)] \quad (1)$$

, where first term represents a KL divergence regularization and second term represent a reconstruction error. As an explicit form of the reconstruction term, a mean-squared error function is chosen:

$$\mathrm{MSE} = \frac{1}{n}\sum(x - x')^2 \tag{2}$$

, where x is input data and x' is reconstructed data, and the summation runs over all n data points in x.

Once the network is trained properly, the reconstruction cost quantifies a distance between training data and an input fed to the trained network. When the input is not alike the training dataset, the reconstruction cost gives a high value. Using this reconstruction cost, we define a likelihood of x as follows:

$$f(x) = 1 - \left[\frac{\mathrm{MSE} - \min(\mathrm{MSE})}{\max(\mathrm{MSE}) - \min(\mathrm{MSE})}\right] \tag{3}$$

, where $f(x)$ takes values from zero to one and higher value means a higher likelihood for a given x.

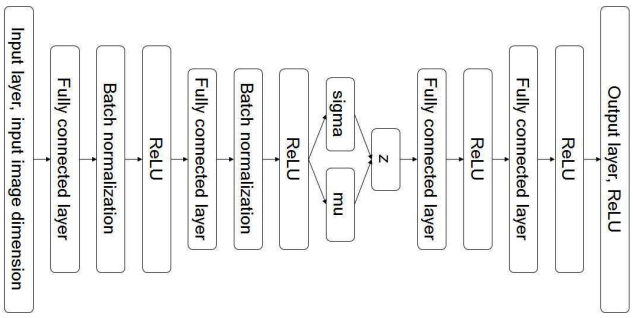

Figure 1 – **VAE network overview**: Input and Output layers have a same dimension as an input image. Two intermediate dimensions and latent dimension are 1000, 300 and 100, respectively. The rectified linear unit (ReLU) is used as the output activation.

2.3 Training results

VAE was successfully trained on Street data and Booth data (Figure 2) and demonstrated satisfactory reconstruction performance (Figure 3). However, a rate of training loss vs validation loss shows different trends for Street data and Booth data: the rate is about 20% for Street data, while it is about 5% for Booth data. This will be discussed in Section 3.1.

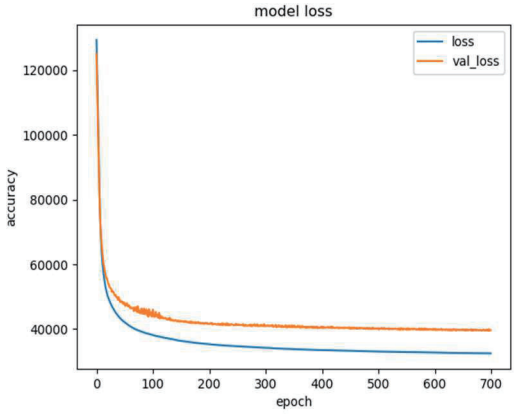

Figure 2 – **Example of loss change during a training**: Training loss (Orange) and validation loss (Blue) change during training on Street dataset for 700 epochs.

The trained VAE network recovers an original input from an incomplete input, in which some circles are removed from the original input as shown in Figure 3. In Figure 3, the reconstruction seems to be performing well considering 20% of gap between training and validation error for Street data. This can be interpreted as that the training on image pixel values is suitable for the current network and we are complementing the reconstructed image in our mind.

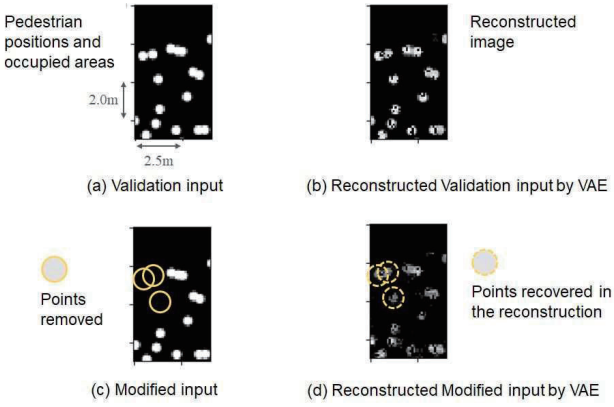

Figure 3 – **Street Data Reconstruction by the trained network**: a) Original input not exposed during the training, b) Reconstructed image by VAE, c) Modified input by removing three points from the original input, and d) Reconstruction of the modified input by VAE

3. Analysis

3.1 Generalization capabilities

The difference in the rate of training loss vs validation loss represents a generalization capability of the trained network. First, Booth data have longer duration than Street data and a longer training data have an advantage for generalization. Second, pedestrians in Street data are constantly and randomly moving in the view. In contrast, people in Booth data sometimes stop and see at specific locations in the view. Thus, the Booth data have advantages over Street data in regard to generalization and this resulted in the smaller difference in the training and validation loss.

Although the network is not over-fitted (Figure 2), the cost function after training is highly sensitive to similarity of an input data to the training dataset. If a training input is slightly shifted before reconstruction by the trained network, the network tends to recover the original input by canceling out the amount of shift. In other words, the VAE network learned the pedestrian distribution specific to the training dataset and it tries to interpret any input by learned patterns.

This generalization capability will be improved by preparing larger dataset and data augmentation.

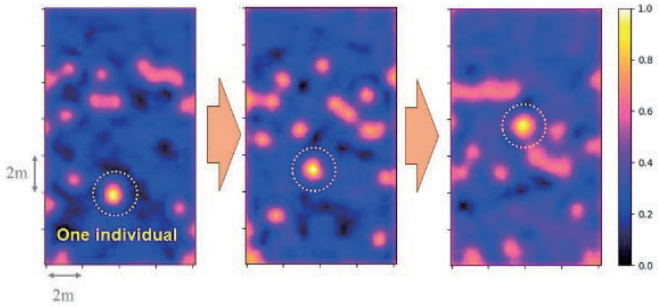

Figure 4 – **Likelihood distribution over 2-D area**: images are ordered in time sequence from left to right. Color shows likelihood of finding an individual in this area.

3.2 Likelihood distribution on expected trajectories

When an input x is fed to the trained VAE and reconstructed, likelihood of x can be estimated by $f(x)$. To estimate expectation to find an individual within a crowd, first one specific circle in a frame is selected and removed to create a base input x_0. Then, we put a new circle on x_0 to generate a new test input x_1. By evaluating $f(x_1)$, one can estimate a likelihood of x_1, which is a quantified expectation to find the removed individual at the position where the new circle is placed.

By repeating this process while moving a new circle over the area of x_0, one can estimate a likelihood distribution over 2-D area. Images generated via this procedure are presented in Figure 4. These images can be seen as our expectation for a pedestrian location when other pedestrians are distributed as x_0

By processing multiple frames in sequence, one can estimate an expected trajectory of an individual based on observations (Figure 4).

In the process described above, only one circle is removed and the likelihood distribution is estimated. The same process is applicable when removing a cluster of circles to generate x_0 and putting a new area, which is not necessary to be a circle, to create x_1. In this case, the resulted 2-D likelihood distribution means our expectation to find a cluster of people around the area instead of an individual. This can be regarded as our 'soft-focused' expectation over a crowd.

4. Conclusion

In the present study, we discussed unsupervised training of VAE on pedestrian traffic data. Since our focus was training the network only by pedestrian positions and occupied areas in a public space, movies for training are preprocessed to generate input images representing only these features. As a result, the training was made computationally less intensive and VAE network was successfully trained with relatively small amount of data.

The trained VAE captures features of a pedestrian distribution in a frame and it is utilized to render our expectation to the trajectory of an individual in a crowd. Images rendered by this method correspond to subjective images usually only captured in our mind.

Our goal is to capture stochastic aspect of pedestrian behaviors in a crowd by machine learning. In the present study, outcomes of our first attempts to embody this concept are reported. Further analysis is required to achieve an appropriate level of a generalization capability. In addition, the current method is based only on an instantaneous pedestrian distribution, and does not take into account temporal correlations. Deep neural network based models are applied to predict pedestrian trajectories, and prediction performance of various architecture is compared in a recent study [Becker 2018]. These are areas to extend the present study.

References

[Yokojima 2018] Yasunori Yokojima, and Toshihiko Nakazawa, Applying deep learning to test design process in hydraulic systems design 深層学習を応用した油圧システム設計におけるテスト設計プロセス. Journal of the Japan Fluid Power System Society **49**, 71-74, 2018

[Sakai 2019] Sakai and Yokojima, Development of Deep Learning technology for a pedestrian trajectory study without requiring explicit labeling, Annual Congress(Spring), JSAE, 2019.

[Kingma 2014] Diederik P. Kingma and Max Welling, Auto-Encoding Variational Bayes, arXiv:1312.6114v10, 2014.

[Doersch 2016] Carl Doersch, Tutorial on Variational Autoencoders, arXiv:1606.05908v2, 2016.

[Mochihashi 2017] Daichi Mochihashi, "Variational Bayesian methods for Natural Language Processing", ATR SLC. 2005-6-21, http://chasen.org/~daiti-m/paper/vb-nlp-tutorial.pdf, accessed 2019-01-24.

[kenmatsu4 2017] kenmatsu4, "Variational Autoencoder 徹底説", Qiita, 2017-8-1, https://qiita.com/kenmatsu4/items/b029d697e9995d93aa24, accessed 2019-01-24.

[Nitta 2016] Kazuki Nitta, "Variational Autoencoder", www.SlideShare.net, 2016-11-11, https://www.slideshare.net/KazukiNitta/variational-autoencoder-68705109, accessed 2019-01-24.

[Tatsuno 2016] Sho Tatsuno, "猫でもわかる Variational Autoencoder", www.SlideShare.net, 2016-7-29, https://www.slideshare.net/ssusere55c63/variational-autoencoder-64515581, accessed 2019-01-24.

[Nozawa 2016] Kento NOZAWA, "Variational Auto Encoder", Red Green Black and White, 2016-12-1, https://nzw0301.github.io/notes/vae.pdf, accessed 2019-01-24.

[fchollet 2018] fchollet, "Deep Learning for humans", GitHub, 2018-11-1, https://github.com/keras-team/keras, accessed 2019-01-24.

[taehoonlee 2018] taehoonlee, "variational_autoencoder.py", GitHub, 2018-11-1, https://github.com/keras-team/keras/blob/master/examples/variational_autoencoder.py, accessed 2019-01-24.

[Chollet 2019] Francois Chollet, "Building Autoencoders in Keras", The Keras Blog, 2016-5-14, https://blog.keras.io/building-autoencoders-in-keras.html, accessed 2019-01-24.

[qqwweee 2018] qqwweee, “A Keras implementation of YOLOv3 (Tensorflow backend)”, GitHub, 2018-11-1, https://github.com/qqwweee/keras-yolo3, accessed 2019-01-24.

[iss-f 2018] iss-f, “VAE を keras で実装”, Qiita, 2018-2-12, https://qiita.com/iss-f/items/c43b23815fc6d15ae22e, accessed 2019-01-24.

[Redmon 2015] Joseph Chet Redmon, “YOLO: Real-Time Object Detection”, Darknet Neural Network framework, 2015, https://pjreddie.com/darknet/yolo/, accessed 2019-01-24.

[Redmon 2013] Joseph Redmon, “Convolutional Neural Networks http://pjreddie.com/darknet/”, GitHub, 2013-11-3, https://github.com/pjreddie/darknet/, accessed 2019-01-24.

[Kathuria 2018] Ayoosh Kathuria, “A PyTorch implementation of the YOLO v3 object detection algorithm”, GitHub, 2018-2-25, https://github.com/ayooshkathuria/pytorch-yolo-v3/, accessed 2019-01-24.

[Bergstra 2012] James Bergstra: Random Search for Hyper-Parameter Optimization, Journal of Machine Learning Research 13, 2012.

[Patterson 2017] Josh Patterson: Deep Learning A Practitioner’s Approach, O’Reilly Media, p.1-403, 2017.

[Sugomori 2017] Yusuke Sugomori: Deep Learning: Practical Neural Networks with Java, Packt Publishing, 2017.

[Sugomori 2016] Yusuke Sugomori: Deep Learning Java programming, Packt publishing, 2016.

[Henrik 2017] Henrik B.: Machine Learning, Impress corporation, 2017.

[Fujita 2016] K. Fujita, A.Takahara: 実装ディープラーニング, Ohmsha, 2016.

[Saito 2016] Y. Saito: Deep Learning–Python, O’Reilly Japan, 2016.

[Becker 2018] Stefan Becker, Ronny Hug, Wolfgang H¨ubner and Michael Arens, Notes on the TrajNet Benchmark, arXiv:1805.07663v6, 2018.

[Newell 2002] G.F.Newell, A simplified car-following theory: a lower order model, Transportation Research Part B: Methodological, 36, 2002.

[Sakai 2018] Tatsuhide Sakai: An Automatic Search to EV design variables using Reinforcement Learning, EVS31 Kobe Japan, 2018.