


Robotics project II
Semester project

Generation of multimodal distribution using cVAE and Normalizing Flows

Artur **Jesslen**

Supervisor:
Kothari *Parth*

Professor:
Pr. Alahi *Alexandre*

Code available at:
`https://github.com/Arturjssln/trajnetplusplus`

Academic year 2020-2021
Autumn semester - 10 ECTS credits

March 27, 2021

Contents

1 Introduction

The *theory of mind*[1] is a strong social-cognitive skill that involves the capacity to reason about other people's actions in terms of their mental states. It implies that generally, pedestrians do not follow linear trajectories. To avoid collisions, they need to accommodate the avoidance of obstacles and the presence and the state of mind of fellow pedestrians. They may take different paths given the same scene, as illustrated in Figure 1. Accounting for this multi-modal nature of trajectory is a prerequisite to safe and socially-aware robotic navigation.

Since autonomous systems do not have this ability, imbuing autonomous systems with this capability is challenging because social interactions are complex and often subtle but would allow making proactive decisions. Predicting someone's possible trajectories can be considered as a sequence generation task, in which, based on his past positions and other pedestrians' past positions, we predict future trajectories by taking into account social interactions.

In recent research in computer vision, Kitani et al.[2] have demonstrated that having semantic information about the scenes (i.e. presence of crosswalk, the position of sidewalks, etc) helps predict the trajectory of pedestrians in future instants more accurately than a model that ignores the scene information. Some works[3,4] proposed models of *human-human* interactions to increase robustness and accuracy in multi-target tracking problems. However, these works were limited by the following assumptions. (1) They use hand-crafted functions to model "interactions" for specific settings rather than inferring them in a data-driven fashion. (2) They focus on modeling interactions among people in close proximity to each other (to avoid immediate collisions). With the recent success of Long Short-Term Memory networks (LSTM) for sequence prediction, Alahi et al. used them to overcome these limitations.[5] To capture dependencies between multiple correlated sequences, they introduced a "social" pooling layer that learns typical interactions that take place among trajectories that coincide in time.

In this work, we will go further on the work from Alahi et al. and we will make use of conditional Variational Autoencoders (cVAE) to see if we can improve the performance of the prediction by using cVAEs.

This work is using the interaction-centric benchmark framework TrajNet++.[6] We will test the performance using synthetic data (generated by the TrajNet++ framework) as well as real-world data.

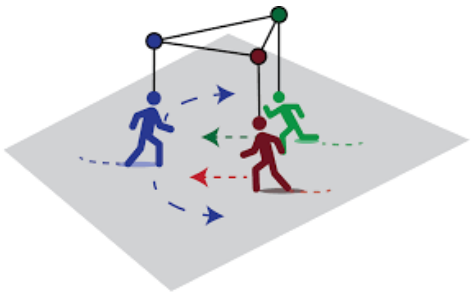

Figure 1: Illustration of multimodal behavior where the blue pedestrian, who must make a decision about which direction he will take to avoid the red-green pedestrian group.

2 Related work

Human-human interaction: As evoked before, some works[3,7] have been focusing on simple social forces to predict how humans are interacting with each other and thus, allowing to take this behavior into account for robots to plan a path in a crowded environment. Some later work[8] supposes that the moving pedestrians detected in the environment follow typical motion patterns that are represented by Gaussian processes which have been learned by an off-board platform before navigation and to be known by the robot that is trying to minimize the probability of collision using its knowledge. Most of these models provide hand-crafted energy potentials based on relative distances and rules. Using some more data-driven model could lead to better results and capture elements that we could not think of or see.

Koppula and Saxena[4] showed that using anticipatory temporal conditional random field (ATCRF) that models the rich spatial-temporal relations through object affordances. This last work was concentrating on objects (e.g. the butter will probably end up in the fridge if it is currently on a table) and is therefore not applicable for pedestrians.

RNN for sequence prediction: Deep Neural Networks (DNN) are powerful models that have achieved excellent performance on difficult learning tasks. However, when speaking about mapping sequences to sequences, Recurrent Neural Networks (RNN) such as LSTMs outperform DNNs as shown by Sutskever et al.[9] They also achieved remarkable results in speech recognition[10] or translation.[11]

Alahi et al.[5] demonstrate the power of RNNs for sequence-to-sequence modeling by learning to generate sequential future prediction outputs. The approach in this work is similar to the approaches from Cho et al.[12] or Kothari et al.,[6] using an encoder-decoder structure to embed a hidden representation for encoding and decoding variable-length inputs and outputs.

Generative Adversarial Networks (GAN): GANs have been introduced by Goodfellow et al.[13] in 2014. They introduce a model composed of two different models trained simultaneously: a generative model G capturing the data distribution and a discriminative model D that learns to determine whether a sample is from the model distribution or the data distribution. GANs implicitly encode the multimodality by encouraging the generative network to spread its distribution and cover the space of possible paths while being consistent (verified by the discriminator) with the observed inputs.[14,15]

Variational Autoencoders (VAE): Conditional VAE[16] consists of a deep recurrent backbone architecture with a latent variable model, but unlike GANs, they explicitly encode multimodality. Some previous works are based on this architecture.[17–22] In section 3, we will develop in further detail the implementation of *DESIRE*[17] and *Trajectron++*[22] frameworks.

Improving VAEs: Using conditional VAEs are very promising. The main weakness of this architecture, and generally of all DNNs is that they are working like a black-box that is not interpretable. In the following paragraph, we will introduce, similarly to what Dai and Wipf did,[23] some works that allow better interpretability of the latent variables.

Attribute based regularization: A promising way to structure the latent space is

attribute based regularization[24] that consists of encoding different continuous-valued attributes explicitly. As this is explicit, we need to encode some features (e.g the thickness of a digit) manually. In the case of scenes with pedestrians, it is difficult to find meaningful features. We could use the encode the distance with a neighbor to be taken into account in the navigation, the time for which an agent will respond to the presence of other agents, etc.

β-VAEs: In 2016, Higgins et al. introduced a novel framework for automated discovery of interpretable factorized latent representations working in a completely unsupervised manner, the β-VAE.[25] A framework that outperforms other state-of-the-art frameworks to disentangled factor learning on a variety of datasets. A big flaw of this new framework is that the great performance in disentangling factors implies a trade-off in reconstruction accuracy. Two years later, Burgess et al.[26] procured intuitions of the emergence of disentangled representation as well as some modification to the training regime of β-VAE that progressively increases the information capacity of the latent code and thus, remove the previous trade-of in reconstruction accuracy.

Correlation: By decomposing the evidence lower bound (ELBO), Chen et al.[27] show that a term measuring the total correlation between latent variables exists. Knowing that, they introduced the β-TCVAE (Total Correlation VAE) algorithm requiring no hyper-parameters. They also show some evidence that independence between latent variables can be strongly related to disentanglement.

Factorizing: Another approach of having disentangled latent dimensions is factorization.[28,29] This approach allows overcoming one of the main drawback of β-VAE evoked previously, the trade-off between disentanglement and data-likelihood (reconstruction quality).

SOM-VAE (Markov Process): A way to address the problem of predicting high-dimensional time series has been introduced in 2019, the SOM-VAE.[30] It uses a self-organizing map (SOM) and a Markov transition model is learned to predict the next discrete representation given the current one. The organization in the SOM allows having a disentangled representation of the data. However, as this method is only using the last representation, to predict the next, it is not using all the information it has from the previous states and is thus, not as efficient as some other methods.

Conditional flow VAEs (CF-VAE): Normalizing flows are a powerful class of density estimation methods with exact inference.[31] They can be used in parallel with VAEs. Indeed, Non-Linear Squared Flows have been introduced by Ziegler et al.,[32] and are used in a new framework *Conditional Flow Variational Autoencoders (CF-VAE)*[33] that allows modeling complex multi-modal conditional distributions over sequences. We will focus more on this implementation in the following section.

ODE^2VAE: Yildiz et al. introduce an Ordinary Differential Equation Variational Auto-Encoder[34] with a latent second-order ODE model for high-dimensional sequential data. The main benefit from this method and the following (GPPVAE) are that contrary to all the presented that operate in discrete-time, they operate in continuous-time and is thus closer to most of the real-world. They model the latent dynamic ODE state decomposed as position and momentum.

Gaussian process prior VAEs (GPPVAE): The GPPVAE[35] aims to combine the power of VAEs with the ability to model correlations afforded by Gaussian Process (GP)

priors. Indeed, it solves the limitations of the VAE model that latent representations of samples are independent and identically distributed (iid) which, for example, is not the case in autonomous driving because high dimensional images are correlated in time (i.e. two images that were taken closer in time should have more similar latent representations than images taken further apart). However, in this work, this problem is solved using another approach.

DESIRE and Trajectron++ approaches

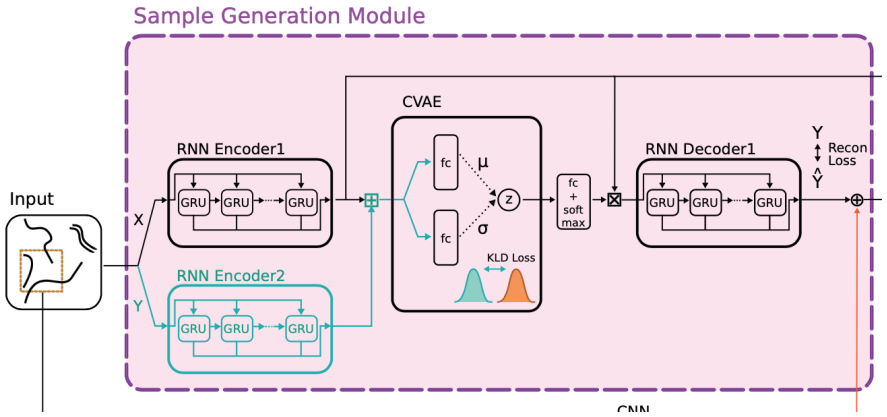

Figure 2: Architecture of the DESIRE framework implementation.

As can be seen in Figure 2, the DESIRE framework implements a simple cVAE. First, the input trajectories (observations or groundtruth) are encoded through RNN Encoder(s), then the cVAE encoder outputs a latent distribution (approximated as Standardized Normal distribution using Kullback–Leibler divergence loss). Then the decoders are reconstructing the predicted trajectories that are compared with the groundtruth using reconstruction loss.

The second framework, Trajectron++, uses a similar architecture but the main difference lies in the fact that the latent distribution is not considered as Standardized Normal distribution, but just as a Gaussian distribution in the latter. Another characteristic of the latter implementation is that the encoding $\mathcal{H}_x$ (the output given by the LSTM encoding the observations) and the latent sample z (sample of the distribution given by the latent space) are concatenated together as a way to incorporate noise in the model to produce a single node representation vector.

Both approaches are a basis for the actual framework implementation that will be presented in the next sections.

3 Implementation

3.1 Contextual information

For prediction tasks, it is necessary to take into account sources of contextual information (e.g. past trajectories, trajectories of interactive agents, ...). The current data that are being used are composed of the x-y positions of all primary and neighbour pedestrians (as represented in Fig. 3). In the case where the data contains heterogeneous sources (e.g. environmental map), it is easy to include them using specific networks to extract fixed length vectors from each source that can easily be concatenated in a single vector $X = \{X_a, X_b, ...\}$. In this project, the x-y positions of all primary and neighbour pedestrians are processed through an LSTM network, interaction between each other are computed using *Social pooling*.[6] In this report, past trajectories will be denoted as $X = \{X_1, X_2, ..., X_n\}$ where n denotes the number of pedestrians (n=1 corresponds to the primary pedestrian). The multiple agents' future trajectories are denoted as $Y = \{Y_1, Y_2, ..., Y_n\}$. The past trajectory of an agent i is defined as $X_i = \{x_{i,0}, x_{i,1}, ..., x_{i,t}\}$ and the future trajectory as follow $Y_i = \{y_{i,t+1}, y_{i,t+2}, ..., y_{i,t+T}\}$. Here, each element of a trajectory (e.g. $x_{i,t}$) is a vector in $\mathbb{R}^2$ representing the coordinates of agent i at time t.

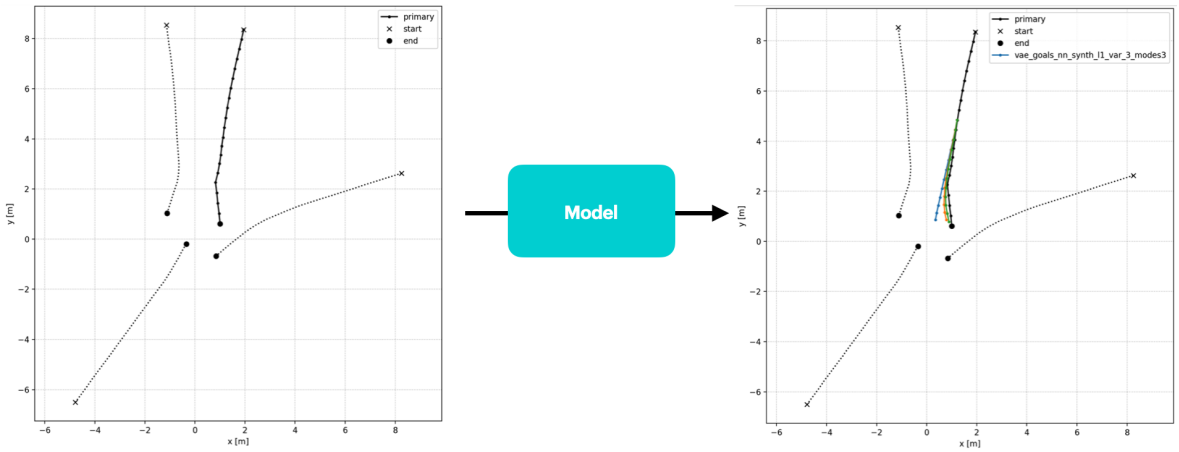

Figure 3: Left figure represents the groundtruth input data composed of X-Y positions of all primary and neighbour pedestrians (during training/evaluation only the first N points are given, for synthetic data: $N = 9$). **Right figure** shows output data that consists in the predicted X-Y positions of the primary pedestrian.

3.2 Generating trajectories with cVAE

Trajectory prediction, as said before, are multimodal and can be inherently ambiguous and has multiple plausible scenarios given the same past situation (as illustrated in Fig.1 previously). Thus, learning a deterministic function f that directly maps X to Y is not possible. It would mean that the potential prediction are under-represented, would represent the average trajectory of all possible trajectories or that it is overfitting the training data. In order to tackle this challenge, instead of predicting trajectories

Y, we will predict a probability distribution $p(\mathrm{Y}_i|\mathrm{X}_i)$ of Y_i conditioned on the input X_i by introducing a stochastic latent variable z_i. The prior of the latent variables z_i is modulated by the input X_i, however, this can be relaxed to make the latent variables statistically independent of input variables, i.e., $P_\nu(z_i|\mathrm{X}_i) = P_\nu(z_i)$.[36] This latent variable follow a distribution $q_\phi(z_i|\mathrm{X}_i, \mathrm{Y}_i)$ that is learned during the training such that it gives higher probability to z_i that is likely to produce a reconstruction $\hat{\mathrm{Y}}_i$ close to actual prediction given the full context X_i and Y_i. In parallel, the distribution $q'_\theta(z_i|\mathrm{X}_i)$ is also learned, it represents the latent distribution conditioned only on the input X_i . ϕ, θ, ψ denote deep neural network weights that parameterize their respective distributions. On one hand the DESIRE[17] architecture that assumes that the distribution $q'_\theta(z_i|\mathrm{X}_i) \sim \mathcal{N}(0, I)$, while on the other hand the Trajectron++[22] architecture assumes that the distribution $q'_\theta(z_i|\mathrm{X}_i) \sim \mathcal{N}(\mu, \Sigma)$. In the current implementation, we don't make such strong assumptions, instead, we are using normalizing flows (explained in more details in the following section 3.3).

At test time z_i is sampled randomly from the learned prior distribution and decoded through the decoder network to produce a prediction of the trajectory $\hat{\mathrm{Y}}_i$. This enables probabilistic inference which serves to handle multi-modalities in the prediction space.

3.3 Normalizing flow

As explained in Huang et al.,[37] research on constructing NFs focuses on finding ways to parametrize flows which meet the following requirements while being maximally flexible in terms of the transformations which they can represent. For efficient training, the following operations must be tractable and cheap:

1. Sampling $x \sim p_X(x)$
2. Computing $y = f(x)$
3. Computing the gradient of the log-likelihood of $y = f(x); x \sim p_X(x)$ under both $p_Y(y)$ and $p_{target}(y)$
4. Computing the gradient of the log-determinant of the Jacobian of f

An extension of the normalizing flow is the Non-Linear Squared (NLSq) Flow.[32] They propose to replace the affine scalar transformation with an invertible NLSq transformation with five pseudo parameters (a, b, c, d, g) instead of two for the affine transformation (i.e. $f(\epsilon) = a + b\epsilon$):

$$f(\epsilon) = z = a + b\epsilon + \frac{c}{1 + (d\epsilon + g)^2} \tag{1}$$

Under conditions on the scale parameter c the function can be guaranteed to be invertible, and the analytical inverse is the solution to a cubic equation (see Ziegler et al.[32] for more precision).

Algorithm 1 Non Linear Normalizing Flow

```
procedure NormalizingFlow(x,z)
    (zL, zR) ← split_coupling(z)
    coefs ← f(x, zR)
    ϵ, logdet ← flow_encoding(zL, coefs)
    zL , logdet ← flow_decoding(ϵ, coefs)
    z ← split_coupling((zL, zR), reverse=True)
```

3.4 Training phase

Firstly, the past and futures trajectories of an agent i, X_i and Y_i respectively, are encoded through two different LSTM encoders (LSTM encoder 1 and 2 respectively in Figure 4). The resulting encoding, $\mathcal{H}_x$ and $\mathcal{H}_y$ resp. are concatenated and passed through the cVAE Encoder. Two side-by-side fully-connected (fc) layers are followed to produce both the mean μ_{z_i} and the standard deviation σ_{z_i} over z_i. The distribution over z_i is modeled as Gaussian distribution (i.e. $z_i \sim q_\phi(z|\mathrm{X}_i, \mathrm{Y}_i) = \mathcal{N}(\mu_{z_i}, \sigma_{z_i})$). The encoding $\mathcal{H}_x$ is concatenated with half of the sample z_i (i.e. $z_i \sim q_\phi(z|\mathrm{X}_i, \mathrm{Y}_i)$), these values are passed through an fc layer that outputs the coefficients of the normalizing flows (i.e. a, b, c, d, g as shown in eq. 1 for each layer). The normalizing flow is learning the following distribution of $z \sim q'_\theta(z|\mathrm{X}_i)$. The output will finally be passed through the cVAE decoder to yield multimodal trajectories that are part of the distribution $p_\psi\left(\mathrm{Y}_i|\mathrm{X}_i, z\right)$.
There are two loss terms used during the training of the model:

- Reconstruction loss: $\ell_{recon} = \frac{1}{K}\sum_k \left\|\mathrm{Y}_i - \hat{\mathrm{Y}}_i^{(k)}\right\|$. This loss measures how far the generated samples are from the actual ground truth.
- $\mathcal{KLD}$ Loss: $\ell_{KLD} = D_{\mathcal{KL}}\left(q_\phi(z_i|\mathrm{X}_i, \mathrm{Y}_i)||q'_\theta(z_i|\mathrm{X}_i)\right)$. This regularization loss measures how close the sampling distribution at test time is to the distribution of latent variable that we learn during training.

The last loss, as developed by Bhattacharyya et al.,[33] can be decomposed as shown in equation 2.

$$-D_{\mathcal{KL}}\left(q_\phi(z_i|\mathrm{X}_i, \mathrm{Y}_i)||q'_\theta(z_i|\mathrm{X}_i)\right) = \mathcal{H}(q_\phi) + \mathbb{E}_{q_\phi(z_i|\mathrm{X}_i,\mathrm{Y}_i)} \log(p(\epsilon|\mathrm{X}_i) + \sum_{i=1}^{n} \log(|\det J_i|) \quad (2)$$

where, $\mathcal{H}(q_\phi)$ is the entropy of the variational distribution, the Jacobian J_i of each layer i of the transformation f_i from the the normalizing flow and n is the number of transformation.

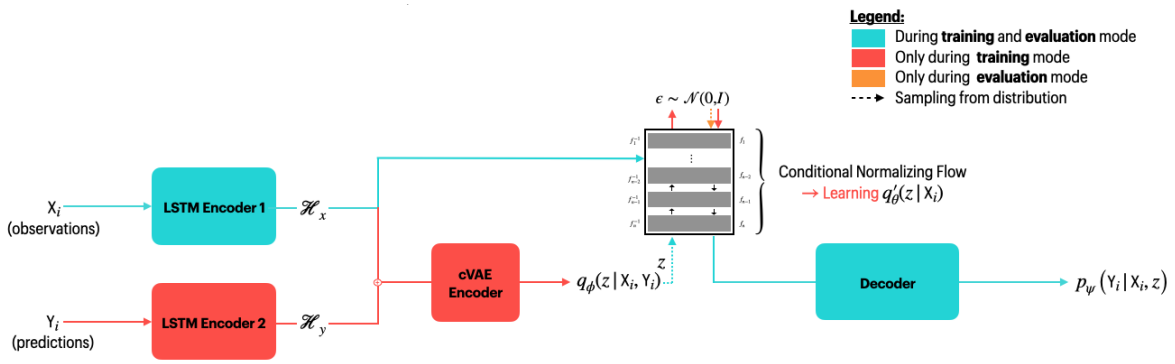

Figure 4: Architecture used in the final implementation (CF-VAE). The major difference with the baseline model shown in Figure 5 lies in the fact that the normalizing flow allows to better depict the data distribution while the baseline consider the data distribution as a gaussian distribution.

The final model has been trained with 54513 different scenes, with 20 epochs, a step size of 6, a batch size of 8 on a CPU.

3.5 Testing phase

During the testing phase, the groundtruths of the trajectories are not available. Thus, the only information available as input, are the actual observations, X_i. Consequently, from the encoding $\mathcal{H}_x$, and the right part of the sample z_i (i.e. sampled from the distribution $z_i \sim q_\phi(z|X_i, Y_i) = \mathcal{N}(\mu_{z_i}, \sigma_{z_i})$), we compute the coefficients of the different layers of the NF. Then, we sample $\epsilon \sim \mathcal{N}(0, I)$ and decode this sample using the forward path of the normalizing flow. Finally, following the same process as in the training part, we will decode the output by passing it through the cVAE decoder to yield multimodal trajectories that are part of the distribution $p_\psi(Y_i|X_i, z)$.

4 Experiments

In this section, we will do some experiments proving that the implemented baseline is actually working and some experiments that justify the architecture choices that have been done. The code for these experiments can be found on Github[1]. All experiments have been done with both synthetic[2] and real-world data. From the moment a parameter is considered as better than the others, it becomes the new base parameter and replaces the old base parameter. The base architecture consists of using:

- cVAE (without normalizing flows)
- Trajectron approach
- Nearest Neighbour (NN) pooling
- $k = 3$ (i.e. number of predicted trajectories)
- L2 Loss for per trajectory loss (Unimodal metric)
- Reconstruction loss for multimodal loss (Multimodal metric)

[1]Code available on : `https://github.com/Arturjssln/trajnetplusplus`
[2]Synthetic data are generated using TrajNet++ dataset[6]

4.1 Evaluation metrics

To evaluate the the performance of a model, we need good evaluation metrics[3]. The ones that are used in this work are describe in the two following sub-sections.

4.1.1 Unimodal metrics (single prediction)

Average Displacement Error (ADE): Average L2 distance between the ground truth and prediction of the primary pedestrian over all predicted time steps. Lower is better.
Final Displacement Error (FDE): The L2 distance between the final ground truth coordinates and the final prediction coordinates of the primary pedestrian. Lower is better
Prediction Collision (Col-I): Calculates the percentage of collisions of primary pedestrian with neighbouring pedestrians in the scene. The model prediction of neighbouring pedestrians is used to check the occurrence of collisions. Lower is better.
Ground Truth Collision (Col-II): Calculates the percentage of collisions of primary pedestrian with neighbouring pedestrians in the scene. The ground truth of neighbouring pedestrians is used to check the occurrence of collisions. Lower is better.

4.1.2 Multimodal metrics (multiple prediction)

Topk Average Displacement Error (Topk ADE): Given k output predictions for an observed scene, the metric calculates the ADE of the prediction which is closest to the groundtruth trajectory in terms of ADE. Lower is better.
Topk Final Displacement Error (Topk FDE): Given k output predictions for an observed scene, the metric calculate the FDE of the prediction which is closest to the groundtruth trajectory in terms of ADE. Lower is better.

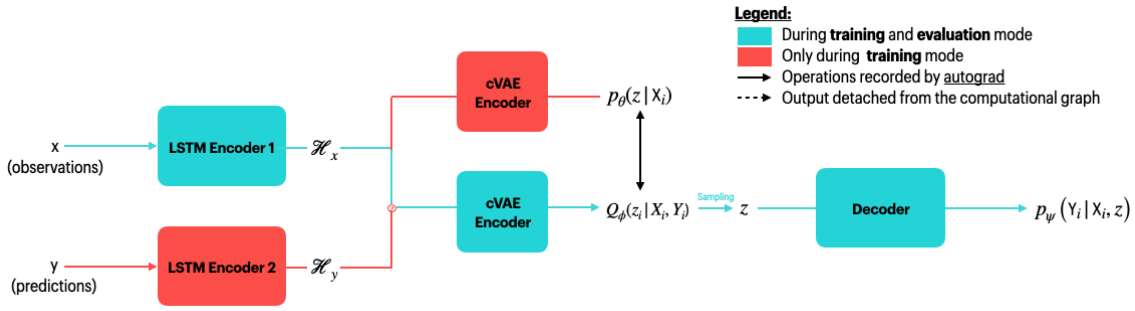

Figure 5: Architecture used as a baseline (cVAE). The architecture represented here is very similar to the Trajectron++[22] implementation.

[3]Metrics and their description are taken from the Trajnet++ Challenge that is available using the following link: `https://www.aicrowd.com/challenges/trajnet-a-trajectory-forecasting-challenge` (Last accessed: 31/12/20)

4.2 Social interaction

From this section (sec. 4.2) to section 4.5, the architecture of the model is the one represented in Figure 5. For the final section 4.6, the model represented is the final model, which includes Conditional Normalizing flows and shown in Figure 4. As a first step, it is substantial to assess if the baseline is actually performing better using social interaction rather than predicting the trajectory without considering social interaction (e.g. mostly predicting straight trajectories). As can be seen in table 1, the baseline is performing better using *social pooling*[6] than when using vanilla pooling (no interaction between pedestrians). On average, the displacement errors (i.e. ADE, FDE, Top3 ADE and Top3 FDE) are reduced by 0.72 while the collision rates (Col-I and Col-II) are 3.4 times smaller.

	ADE	FDE	Col-I	Col-II	Top3 ADE	Top3 FDE
Vanilla pooling	0.32	0.61	18.40	6.55	0.28	0.57
Social pooling	0.27	0.52	3.31	5.41	0.23	0.47

Table 1: Performance of the model using vanilla pooling (i.e. no interaction between pedestrians) and social pooling (i.e. considering other pedestrians). For all metrics, lower is better. Best results are highlighted in green.

4.3 Noise incorporation

To induce multimodality and to be robust to noise there are different strategies possible. As in the trajectron approach, the encoding $\mathcal{H}_x$ and the sample z are concatenated together as a way to incorporate noise in the model. There are different ways to mix information carried by these two vectors:

1. Hadamard product : $\mathcal{H}_x \odot z$
2. Concatenation of $[\mathcal{H}_x, z]$ and passing though a fc layer
3. Sum : $\mathcal{H}_x + z$

As can be seen in Table 2 for synthetic data, the Hadamard product performs better or as good as others in most of the metrics we are using. Results for real data show the same behavior and are shown in Appendix B, Table 7. Thus, we will use this noise incorporation method from now on.

	ADE	FDE	Col-I	Col-II	Top3 ADE	Top3 FDE
Hadamard product	0.26	0.51	1.47	5.64	0.23	0.46
Concat. and fc layer	0.26	0.52	1.95	6.02	0.23	0.47
Sum	0.26	0.52	1.72	5.56	0.23	0.47

Table 2: Performance of the model with synthetic data using Hadamard product, Concatenation followed by fully connected (fc) layer and summation as a way to incorporate noise in the model. For all metrics, lower is better. Best results are highlighted in green.

4.4 Combination of per trajectory loss with multi-modal loss

In our architecture, we have different losses. The first loss concerns the loss per trajectory while the second loss aims at improving the multimodal predictions. The candidate for the different losses are summed up in Table 3 (complementary results can be found in Appendix B, Table 8). As it can be seen, the results for synthetic data and real data are different (opposed to the experiment in section 4.3 where results are consistent for both datasets). When for synthetic data, it is very obvious that the reconstruction loss gives the best results for the multimodal loss, we cannot conclude as for the per trajectory loss since no loss that is better for all metrics. Considering the real data, the Variety loss is the best multimodal loss and L1 loss can be considered as the best per trajectory loss.

Synthetic data		**Real data**	
Per trajectory loss	**Multimodal loss**	**Per trajectory loss**	**Multimodal loss**
L1	Reconstruction	L1	Reconstruction
L2	Variety	L2	Variety
Gaussian		Gaussian	

Table 3: Performance of the model using Hadamard product, Concatenation followed by fully connected (fc) layer and summation as a way to incorporate noise in the model. Best results are highlighted in green.

4.5 Disentanglement

The main adavantage of the VAE compared to GANs is that VAE allows to have an explicit latent space vector. An interseting field in VAEs is trying to understand, what does this latent space represents exactly? To try and answer this question, we implemented the β-VAE. The results of this model are presented in Table 4

β	ADE	FDE	Col-I	Col-II	Top3 ADE	Top3 FDE
0.1	0.26	0.51	2.10	5.43	0.23	0.46
1.0	0.26	0.52	2.25	5.56	0.23	0.46
10.0	0.26	0.52	2.28	5.41	0.23	0.47

Table 4: Performance of the model using β-VAE. For all metrics, lower is better. Best results are highlighted in green.

The results show that the β-VAE doesn't induce any notable change compared to the initial model. Moreover, even if the parameters are more disentangled (this has not been calculated, we should use metrics that estimate the disentanglement of the parameters to illustrate our words), it would be very hard to interpret them. Indeed, when using other types of data such as handwriting, the thickness or the size of the writing could be interpretable parameters. In our case, it is much harder to find that kind of parameters.

4.6 CF-VAE

Until now, the distribution of the samples in the latent space were considered as Gaussian distribution. However, the data distribution might be more complex than this. Thus, as explained before (see Section 3.3), Normalizing Flow has been introduced in the model to deal with this issue.

Model	ADE	FDE	Col-I	Col-II	Top3 ADE	Top3 FDE
cVAE	0.26	0.52	2.25	5.56	0.23	0.46
CF-VAE	0.26	0.52	1.39	5.31	0.23	0.47

Table 5: Performance of the model using the best cVAE model found in section 4.4 and CF-VAE with synthetic data. For all metrics, lower is better. Best results are highlighted in green. Worst results are highlighted in red.

In the light of the results shown in Table 5, for the cVAE without normalizing flow, we observe very similar results for all metrics compared to the one with NF. However, for the collision-I metric where we observe a decrease of 40% of its collision rate (i.e. cVAE: 2.25%, CF-VAE: 1.39%) as well as a slight decrease in the collision-II metric (i.e. cVAE: 5.56%, CF-VAE: 5.31%). Using the synthetic dataset, there are no trajectories that are colliding, thus a model capturing the essence of trajectories prediction would predict no colliding trajectories at all. It suggests that CF-VAEs are capable of predicting trajectories that are colliding 40% less than the precedent model, and it would mean that having a more complex latent distribution closer to the actual data distribution leads to better results, and in particular, regarding the collisions. Considering the real-world data (results can be found in Appendix B, Table 9), we first observe stable (top3-)ADE and (top3-)FDE metrics and a decrease in the collision rate. Although the gap is not as important as for the synthetic data, we still observe the same behavior on the collision rate.

5 Conclusion

To sum up all the experiments, Table 6 shows the results of the different models we have been using throughout all experiments. As we could expect, the vanilla-VAE model (i.e. not taking interaction into account) is performing the worst compared to all other models. cVAE and β-VAE (with $\beta = 10$) have very similar performances for all metrics. It shows that taking interaction into account increase the capacity of the model to predict different trajectories. This supports what would one expect. On average, the displacement errors are reduced by 0.72 while the collision rates (Col-I and Col-II) are 3.4 times smaller.

Finally, the final model, the CF-VAE, has similar performances on almost every metric compared to the original cVAE. However, it distinguishes itself in the predicted collision metric (i.e. Col-I) meaning that the output trajectories are less susceptible to have a collision. This suggests that the better capacity of CF-VAE to model complex distribution compared to cVAE allows having a better knowledge of the essence of the trajectories that is that they should not collide.

Model	ADE	FDE	Col-I	Col-II	Top3 ADE	Top3 FDE
Vanilla-VAE	0.32	0.61	18.40	6.55	0.28	0.57
cVAE	0.26	0.52	2.25	5.56	0.23	0.46
β-VAE	0.26	0.52	2.28	5.41	0.23	0.47
CF-VAE	0.26	0.52	1.39	5.31	0.23	0.47

Table 6: Performance of the model using Vanilla pooling (not taking into consideration other pedestrians, normal cVAE with noise incorporation technic and losses found in sections 4.3 and 4.4), β-VAE (with $\beta = 10$) and CF-VAE. For all metrics, lower is better. Best results are highlighted in green. Worst results are highlighted in red.

Further work

Although the disentanglement analysis has not been pushed very far, it could be interesting to go further by finding some interpretable parameters. Indeed, parameters such as the distance between pedestrians before interaction, or the time horizon before detecting obstacles/other pedestrians could be such parameters.
No hyperparameters optimization has been computed in this project. It could be even more interesting to play with the different hyperparameters, in particular the ones that are controlling the loss function, and see how the results could be improved.
In the current work, all metrics are numerical. Using more visual analysis to have a better idea of what would need to be improved, and thus, could give a better idea of what needs to be corrected (i.e. the distribution of the prediction as an example).

References

[1] H. Gweon and R. Saxe. Developmental Cognitive Neuroscience of Theory of Mind. In *Neural Circuit Development and Function in the Brain*, pages 367–377. Academic Press, Cambridge, MA, USA, Jan 2013.

[2] Kris M. Kitani, Brian D. Ziebart, James Andrew Bagnell, and Martial Hebert. Activity Forecasting. In *Computer Vision – ECCV 2012*, pages 201–214. Springer, Berlin, Germany, Oct 2012.

[3] Dirk Helbing and Péter Molnár. Social force model for pedestrian dynamics. *Phys. Rev. E*, 51(5):4282–4286, May 1995.

[4] Hema S. Koppula and Ashutosh Saxena. Anticipating Human Activities Using Object Affordances for Reactive Robotic Response. *IEEE Trans. Pattern Anal. Mach. Intell.*, 38(1):14–29, May 2015.

[5] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971, 2016.

[6] Parth Kothari, Sven Kreiss, and Alexandre Alahi. Human trajectory forecasting in crowds: A deep learning perspective. 2020.

[7] Stefano Pellegrini, Andreas Ess, and Luc Van Gool. *Improving Data Association by Joint Modeling of Pedestrian Trajectories and Groupings.* Sep 2010.

[8] Jorge Rios-Martinez, Anne Spalanzani, and Christian Laugier. Understanding human interaction for probabilistic autonomous navigation using Risk-RRT approach. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems.*

[9] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to Sequence Learning with Neural Networks. *arXiv*, Sep 2014.

[10] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech Recognition with Deep Recurrent Neural Networks. *arXiv*, Mar 2013.

[11] Xing Huang, Huobin Tan, Guangyan Lin, and Yongfen Tian. A LSTM-based bidirectional translation model for optimizing rare words and terminologies. *IEEE*, pages 26–28, 2020.

[12] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv*, Jun 2014.

[13] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv*, Jun 2014.

[14] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. *arXiv*, Mar 2018.

[15] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints, 2019. [Online; accessed 30. Dec. 2020].

[16] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv*, Dec 2013.

[17] Namhoon Lee, Wongun Choi, Paul Vernaza, Christopher B. Choy, Philip H. S. Torr, and Manmohan Chandraker. DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[18] Boris Ivanovic, Edward Schmerling, Karen Leung, and Marco Pavone. Generative Modeling of Multimodal Multi-Human Behavior. *arXiv*, Mar 2018.

[19] Boris Ivanovic and Marco Pavone. The Trajectron: Probabilistic Multi-Agent Trajectory Modeling With Dynamic Spatiotemporal Graphs. *arXiv*, Oct 2018.

[20] Nachiket Deo and Mohan M. Trivedi. Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs. *arXiv*, May 2018.

[21] Nicholas Rhinehart, Rowan McAllister, Kris Kitani, and Sergey Levine. PRECOG: PREdiction Conditioned On Goals in Visual Multi-Agent Settings. *arXiv*, May 2019.

[22] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data. *arXiv*, Jan 2020.

[23] Bin Dai and David Wipf. Diagnosing and Enhancing VAE Models. *arXiv*, Mar 2019.

[24] Ashis Pati and Alexander Lerch. Attribute-based Regularization of Latent Spaces for Variational Auto-Encoders. *arXiv*, Apr 2020.

[25] beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework, Nov 2016. [Online; accessed 29. Dec. 2020].

[26] Christopher P. Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in β-VAE. *arXiv*, Apr 2018.

[27] Ricky T. Q. Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating Sources of Disentanglement in Variational Autoencoders. *arXiv*, Feb 2018.

[28] Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational Inference of Disentangled Latent Concepts from Unlabeled Observations. *arXiv*, Nov 2017.

[29] Hyunjik Kim and Andriy Mnih. Disentangling by Factorising. *arXiv*, Feb 2018.

[30] Vincent Fortuin, Matthias Hüser, Francesco Locatello, Heiko Strathmann, and Gunnar Rätsch. SOM-VAE: Interpretable Discrete Representation Learning on Time Series. *arXiv*, Jun 2018.

[31] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation. *arXiv*, Oct 2014.

[32] Zachary M. Ziegler and Alexander M. Rush. Latent Normalizing Flows for Discrete Sequences. *arXiv*, Jan 2019.

[33] Apratim Bhattacharyya, Michael Hanselmann, Mario Fritz, Bernt Schiele, and Christoph-Nikolas Straehle. Conditional Flow Variational Autoencoders for Structured Sequence Prediction. *arXiv*, Aug 2019.

[34] Cagatay Yildiz, Markus Heinonen, and Harri Lahdesmaki. ODE2VAE: Deep generative second order ODEs with Bayesian neural networks. *Advances in Neural Information Processing Systems*, 32:13412–13421, 2019.

[35] Francesco Paolo Casale, Adrian Dalca, Luca Saglietti, Jennifer Listgarten, and Nicolo Fusi. Gaussian Process Prior Variational Autoencoders. *Advances in Neural Information Processing Systems*, 31:10369–10380, 2018.

[36] Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. Semi-Supervised Learning with Deep Generative Models. *arXiv*, Jun 2014.

[37] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural Autoregressive Flows. *arXiv*, Apr 2018.

Appendices

A Abbreviations

VAE: Variational Autoencoder
ATCRF: Anticipatory Temporal Conditional Randomfield
cVAE: conditional Variational Autoencoder
CFVAE: Conditional Flow Variational Autoencoder
DNN: Deep Neural Network
ELBO: Evidence Lower Bound
fc: Fully connected
GAN: Generative Adversarial Networks
GP: Gaussian Process
GPPVAE: Gaussian Process Prior Variational Autoencoder
iid: Independent and identically distributed random variables
LSTM: Long Short-Term Memory
NLSq: Non Linear Squared
NN: Nearest Neighbour
RNN: Recurrent Neural Network
SOM: Self Organizing Map

B Complementary Tables

	ADE	FDE	Col-I	Col-II	Top3 ADE	Top3 FDE
Hadamard product	0.69	1.40	6.46	11.95	0.63	1.29
Concat. and fc layer	0.73	1.48	6.84	11.41	0.65	1.34
Sum	0.70	1.41	6.84	12.05	0.63	1.29

Table 7: Performance of the model with synthetic data using Hadamard product, Concatenation followed by fully connected (fc) layer and summation as a way to incorporate noise in the model. For all metrics, lower is better. Best results are highlighted in green.

Per trajectory loss	Multimodal loss	ADE	FDE	Col-I	Col-II	Top3 ADE	Top3 FDE
Gaussian	Recon.	0.26	0.51	3.11	5.36	0.22	0.45
	Variety.	0.41	0.83	2.17	6.88	0.24	0.49
L1	Recon.	0.26	0.51	3.46	5.31	0.23	0.45
	Variety.	0.31	0.62	4.45	5.59	0.23	0.45
L2	Recon.	0.26	0.52	1.79	5.79	0.23	0.46
	Variety	0.30	0.60	2.53	5.84	0.23	0.47

(a) Performance of the model on synthetic data using L1, L2 and Gaussian losses as per trajectory losses as well as Reconstruction and Variety losses as multimodal losses. For all metrics, lower is better. Best results are highlighted in green.

Per trajectory loss	Multimodal loss	ADE	FDE	Col-I	Col-II	Top3 ADE	Top3 FDE
Gaussian	Recon.	0.64	1.32	6.81	11.95	0.59	1.23
	Variety.	0.61	1.25	8.02	11.57	0.54	1.11
L1	Recon.	0.62	1.27	7.96	11.67	0.56	1.17
	Variety.	0.58	1.18	10.06	11.32	0.50	1.03
L2	Recon.	0.69	1.41	7.54	11.98	0.64	1.32
	Variety	0.69	1.39	7.89	12.21	0.59	1.21

(b) Performance of the model on real data using L1, L2 and Gaussian losses as per trajectory losses as well as Reconstruction and Variety losses as multimodal losses. For all metrics, lower is better. Best results are highlighted in green. Worst Col-I result is highlighted in red, it shows that L1-Variety combination performs the best for all metrics except Col-I where it performs the worst.

Table 8: Performance of the model with different combination of losses on both synthetic and real data.

Model	ADE	FDE	Col-I	Col-II	Top3 ADE	Top3 FDE
cVAE	0.58	1.18	10.06	11.32	0.50	1.03
CF-VAE	0.59	1.19	9.58	11.06	0.49	1.01

Table 9: Performance of the model using the best cVAE model found in section 4.4 and CF-VAE with real data. For all metrics, lower is better. Best results are highlighted in green.