Mixup in Agent Forecasting

David Yin Dingqi Zhang James Ma Connie Huang

[1] ***Abstract*—Agent forecasting is the task to predict future trajectories of agents based on their past counterparts. As deep neural networks have been widely applied in computer vision and natural language processing, they have also become one of the most reliable modeling techniques in agents forecasting. In this paper, motivated by the algebraic structure of trajectory data, we apply a data augmentation technique called mixup to the domain of agent predictions. In essence, in mixup, the convex combinations of training data are fed into the model instead of the raw ones. We validate the effectiveness of mixup both theoretically and experimentally. First, we show how mixup affects the inductive bias from the perspective of empirical risk minimization and why mixup is suitable for agent forecasting. Then, we conduct a series of ablation studies on the state-of-the-art model of agent forecasting Trajectron++ and demonstrate how mixup helps improve its performance and agrees with the theoretical justifications.**

***Index Terms*—agent forecasting, mixup, Trajectron++**

I. Introduction

Forecasting the future motions of agents is critical to the success of developing intelligent system that is able to interact with the real-world environment. One of the most prominent examples in the 21st century is autonomous driving: the advent of self-driving cars demands scientists to develop a universally applicable model to assist human beings control vehicles and navigate through traffic. However, building such powerful models is challenging. Traditional methods developed based on the Newton's laws can only perform well in very limited domains because they can hardly incorporate the multi-modal and dynamic behaviors of agents.

Fortunately, in the past ten years, we have witnessed an exploding development of deep neural networks. Benefited from that, researchers have achieved significant progress agent forecasting using a variety of models. Nevertheless, most of solutions solely look into the direction of model design and disregard the natural structure of trajectory data, which is one of the most important features that differentiates agent forecasting from classic domains of deep learning for decades.

For that reason, in this work, we are interested in improving the performance of the state-of-art model Trajectron++ from the perspective of data augmentation [1]. Our key contributions are two parts. First, we provide a mathematical understanding of how mixup technique regularizes the training objective and why the linear interpolation is consistent with the data themselves. Secondly, we present experimental results on two publicly available ETH [2] and UCY [3] datasets to show the advantage of using mixup [4] .

[1]The code could be founded on https://github.com/davidyinyida0609/CS-182-Final-Project

II. Related Work

A. Non-deep-learning Methods

Human trajectory prediction and planning haa received increasing attention from several communities such as self-driving vehicles and service robots. Some of the early works of human trajectory forecasting include the Social Force Model [5], in which the author proposed several force concepts as a measure for the internal motivations of the individuals to perform certain actions. In [6], a dynamic potential field was proposed to integrate global navigation with moving obstacles such as other people. These earlier works rely mostly on hand-crafted energy potential to model human motion.

B. Deep Learning Approaches

Deep learning has also been utilized to forecast human trajectories. Some choose to formulate this problem as a deterministic time-series regression problem and then solve it using Gaussian Process Regression (GPR) [7]–[9], inverse reinforcement learning (IRL) [10], and recurrent neural networks (RNNs) [11]–[14]. However, human behavior is rarely deterministic or unimodal. Therefore, more recently generative approaches have become the state-of-the-art in this field, due to their ability to generate a distribution of potential future trajectories, instead of one single future trajectory. Most of these methods use a recurrent neural network architecture with a latent variable model, such as a conditional variational auto-encoder (CVAE) [1], [15]–[19], or a generative adversarial network (GAN) [20]–[24] to encode multi-modality. There also exist works utilizing attention mechanisms. GraphAttention [25] introduced a graph-based attention model to learn the dependency of the agents in multi-agent sports games. AgentFormer [26] proposed a novel agent-aware attention mechanism that can attend to features of any agent at any previous timestep. Different from all previous sequential models, Y-net utilizes U-net architecture to encode the heatmap of trajectory data and scenes maps and reconstructs the heatmap of future trajectories [27], [28].

C. Data Augmentation

There are a variety of data augmentation techniques for sequential data in deep learning. We can categorize them into three groups. The first one employs traditional algorithms, including slicing, jittering, and scaling [29]. Approaches in this group only use the features in original data and apply predetermined transformation to them. However, the regularized effect of these methods is very limited because the simple permutation or rearrangement of old samples doesn't produce many diversified data. The second one is based on a generative

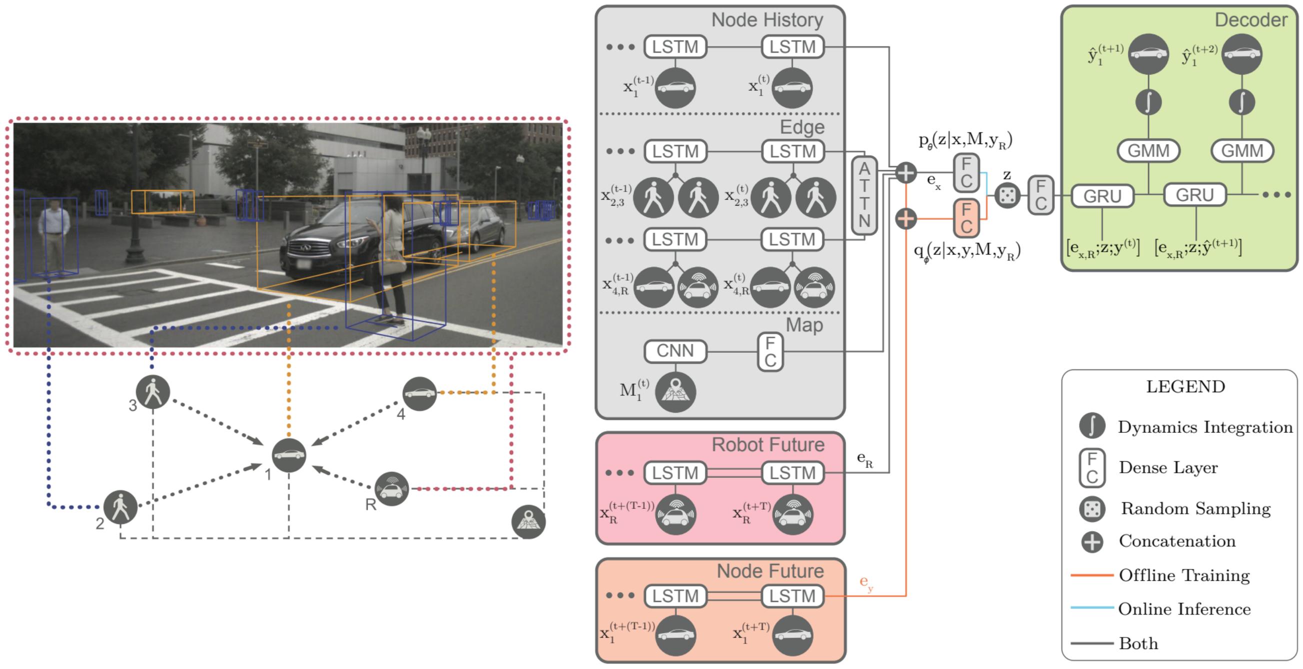

Fig. 1. An illustration of Trajectron++ model architecture and how it utilizes the graph information. The figure itself comes from Trajectron++ paper [1].

model, such as GAN and CVAE, that is trained on raw data and obtains a latent feature space to generate new data [29]. Despite its ability to create new data points, it's not efficient because the time to train such a data generator sometimes is even longer than to train a downstream model. The last one is mixup. It applies a convex combination to a pair of features and their respective targets so that the resulting mixed data point contains the characteristics from both samples. At the first glance, it has no much difference compared with traditional algorithms in that mixup solely utilizes the features of raw data. However, we will demonstrate its advantage in the problem of agent forecasting.

III. Approach

A. Problem Formation

We denote the time-varying number $\mathbf{N}(t)$ of agents at time t in the map as $\{\mathbf{A}_i\}_{i=1,\cdots,\mathbf{N}(t)}$, where t belongs to the all possible timesteps $\mathbf{T}$. For each agent $\mathbf{A}_i$, we denote its two-dimensional position at time t as $\mathbf{s}_i^t \in \mathbb{R}^2$. Naturally, for all agents, we can define their joint past dynamics, including positions, velocities, and accelerations, from time $t-H$ to t as $\mathbf{X}_{1:\mathbf{N}(t)}^t = \{(\mathbf{s}_i^t, \dot{\mathbf{s}}_i^t, \ddot{\mathbf{s}}_i^t)\}_{i=1,\cdots,\mathbf{N}(t)}^{t-H:t} \in \mathbb{R}^{(H+1)\times N(t)\times 6}$ and the future trajectories from time $t+1$ to $t+F$ as $\mathbf{Y}_{1:\mathbf{N}(t)}^t = \mathbf{s}_{1:N(t)}^{t+1:t+F} \in \mathbb{R}^{F\times\mathbf{N}(t)\times 2}$. Then, our objective is to model the joint distribution $\mathbf{p}(\mathbf{Y}_{1:\mathbf{N}(t)}^t|\mathbf{X}_{1:\mathbf{N}(t)}^t)$ for any $t \in \mathbf{T}$.

B. Trajectron++ Adoption [1]

Due to the property of Markov Decision Process, it is assumed the future trajectories for all agents are conditional independent to one another given the past joint dynamics in Trajectron++. Thus, we can just train a model to fit a collection of distributions $\forall t \in \mathbf{T}, \{\mathbf{p}(\mathbf{Y}_i^t|\mathbf{X}_{1:\mathbf{N}(t)}^t)\}_{i=1,\cdots,N(t)}$. In addition, in order to focus more on those interactive agents, Trajectron++ researchers reformulate the joint past dynamics in terms of each agent $\mathbf{A}_i$. Specifically, we create a series of partial-view graphs $\mathbf{G}_i^t = (\mathbf{V}_i^t, \mathbf{E}_{j,i}^t)$, where j is the index of the neighbor agent $\mathbf{A}_j$ in the neighborhood of the main agent $\mathbf{A}_i$. We define the neighborhood of the main agent $\mathbf{A}_i$ at time t as $\mathcal{N}(\mathbf{A}_i)_t = \{\mathbf{A}_j, j \in \{1,\cdots,\mathbf{N}(t)\} | \, ||\mathbf{s}_i^t - \mathbf{s}_j^t||_2 \le \epsilon\}$ that is the neighbor agent $\mathbf{A}_j$ is present at time t if the euclidean distance from it to the main agent is smaller than a threshold value ϵ. For notation convenience, we denote the graph processing as a function $\mathbf{g}$ such that $\mathbf{g}(\mathbf{X}_{1:\mathbf{N}(t)}^t, \mathbf{A}_i) = \mathbf{G}_i^t$. Overall, now our data is a collection of pairs of the graph-structural past dynamics for each node and its respective future trajectory, denoted as $\mathbf{D} = \{(\mathbf{G}_i^t, \mathbf{Y}_i^t)\}_{i=1,\cdots,\mathbf{N}(t)}^{t\in\mathbf{T}}$.[2]

Trajectron++ itself is a CVAE that trains on the preprocessed graph-structural data to model the prementioned joint distribution. The details about the model architecture can be found in Fig.1. Since the model itself is not the main focus for this paper, we abstract it as $\mathbf{f}$.

C. Mixup

Formally, we want to minimize the empirical risk of the model with a loss $\mathbf{L}$ and underlying data distribution $\mathbf{P}$:

$$f^* = \underset{f}{\operatorname{argmin}}\, \mathbb{E}_{(x,y)\sim\mathbf{P}}[\mathbf{L}(\mathbf{f}(x), y)]. \quad (1)$$

[2]The detail of how the partial-view graph is constructed can be found in these papers [1], [11], [17], [30]

As mentioned in [4], unfortunately, this expression is unsolvable because the underlying distribution $\mathbf{P}$ is unknown in most cases. What we have is the discrete distribution $\mathbf{D}$, so we could use the discrete samples $\mathbf{D}$ to approximate the underlying distribution $\mathbf{P}$:

$$\mathbf{P}^*(x,y) = \frac{1}{|D|} \sum_{(x_i,y_i)\in\mathbf{D}} \delta(x = x_i, y = y_i), \tag{2}$$

where $\delta(x = x_i, y = y_i)$ is a Dirac mass centered at (x_i, y_i). Then our objective could be written out by using the approximated distribution:

$$f^* = \underset{f}{\operatorname{argmin}} \frac{1}{|D|} \sum_{(x,y)\in\mathbf{P}^*} \mathbf{L}(\mathbf{f}(x), y). \tag{3}$$

However, the approximate distribution $\mathbf{P}^*$ might not fully represent the underlying one. For example, our data points could be noisy, resulting in inaccurate $\mathbf{P}^*$. To solve this issue, the idea of Vicinal Risk Minimization is raised [31]. Concretely, we approximate the underlying distribution with a vicinal distribution $\mathbf{P}_\nu$ based on $\mathbf{D}$.

$$\mathbf{P}_\nu(x,y) = \frac{1}{|D|} \sum_{(x_i,y_i)\in\mathbf{D}} \nu(x, y|x_i, y_i) \tag{4}$$

We can then minimize the objective based on the vicinal distribution[3]:

$$f^* = \underset{f}{\operatorname{argmin}} \frac{1}{|\mathbf{P}_\nu|} \sum_{(x,y)\in\mathbf{P}_\nu} \mathbf{L}(\mathbf{f}(x), y). \tag{5}$$

There are a variety of choices for the vicinal function ν. We could use a Gaussian kernel centered at each discrete data features with a specified variance. In this paper, we adopt the mixup technique. Concretely, we define the vicinal function to be:

$$\nu(x, y|x_i, y_i) = \frac{1}{|D|} \sum_{(x_j,y_j)\sim\mathbf{D}} \mathbb{E}_\lambda[x = x_{ij}, y = y_{ij}], \tag{6}$$

where $x_{ij} = \lambda x_i + (1-\lambda)x_j, y_{ij} = \lambda y_i + (1-\lambda)y_j$ and λ follows a certain distribution which is only defined in the range of $(0, 1)$.

The intuition for this is that the convex combination of features aligns with the same convex combination of targets under the distribution $\mathbf{P}$. By combining a pair of data points together, we could not only apply linearly property to the learned function $\mathbf{f}$ but also enable it to see a variety of scenarios, which is unlikely with raw data. For example, in computer vision, the combination of a cat image and a dog image with mixup produces a both dog-like and cat-like image, which encourages the model to learn how to differentiate dog features from cat features. However, the critical issue for mixup is that we can't assume the linear interpolation for features and targets is consistent in general. Nevertheless, in the following subsection, we show the consistency of this technique in the problem of agents forecasting.

[3]If $\mathbf{P}_\nu(x, y)$ is a continuous distribution, then the normalization factor should be the number of samples we draw instead of the cardinality of the distribution, which is infinite.

D. Consistence Justification

In this section, we justify why mixup is a consistent operation in agent forecasting. We define the mixup function as $h_\lambda(a, b) = \lambda a + (1-\lambda)b$. Suppose we have two arbitrary data points $(\mathbf{G}_i^t, \mathbf{Y}_i^t)$ and $(\mathbf{G}_j^k, \mathbf{Y}_j^k)$, where $\mathbf{G}_i^t = \mathbf{g}(\mathbf{X}_{1:\mathbf{N}(t)}^t, \mathbf{A}_i)$ and $\mathbf{G}_j^k = \mathbf{g}(\mathbf{X}_{1:\mathbf{N}(t)}^k, \mathbf{A}_j)$. We then apply mixup to interpolate two partial-view graphs. In other words, we have:

$$\begin{aligned} h_\lambda(\mathbf{G}_i^t, \mathbf{G}_j^k) &= \lambda \mathbf{g}(\mathbf{X}_{1:\mathbf{N}(t)}^t, \mathbf{A}_i) \\ &= \lambda \mathbf{g}(\mathbf{X}_{1:\mathbf{N}(t)}^t, \mathbf{A}_i) + (1-\lambda)\mathbf{g}(\mathbf{X}_{1:\mathbf{N}(t)}^k, \mathbf{A}_j) \\ &= \mathbf{g}(\lambda \mathbf{X}_{1:\mathbf{N}(t)}^t, \mathbf{A}_i) + \mathbf{g}((1-\lambda)\mathbf{X}_{1:\mathbf{N}(t)}^k, \mathbf{A}_j) \end{aligned} \tag{7}$$

Since constructing a partial-view graph only doesn't change the position for each agent and also the relative distance among them, we can move the mixup two factors inside the operation g in the last inequality in 7. Lastly, since the joint history and future trajectories for each agent are all truncated from the original full trajectories, applying mixup first on the raw trajectories and then truncate them into past and future ones is the same as truncate them into past and future trajectories and then applying mixup to each of them. As long as the mixup factor λ is the same for each interpolation, the mixup operation h is consistent.

IV. Experiments

A. Data Preparation

We train the model on two widely-used datasets: The ETH dataset [2], with subsets named ETH and HOTEL, and the UCY dataset [3], with subsets named ZARA1, ZARA2, and UNIV. Both datasets provide interactive human pedestrian navigation episodes and contain key information of the pedestrian position, velocity, and accelerations. The trajectories are sampled at 0.4 seconds intervals. In our experiments, the data is segmented into batches that consist of observed sequences of 8 time steps ($\mathbf{H} = 7$), which corresponds to 3.2 seconds, and future sequences of the next 12 time steps ($\mathbf{F} = 12$), which corresponds to 4.8 seconds. In the training time, we feed the model with batches of data. In the testing time, we truncate a series of episodes into 20-time-step interval and evaluate the model on how well they predict the future 12-time-step trajectories provided the past graph structural dynamics. The split between the training and testing data follows the convention in Trajectron++ [1].

B. Implementation Details

The model is adapted from Trajectron++ codebase. The model is trained using Intel Core I7 CPUs and NVIDIA GTX 1080 Ti/ RTX 2080 Ti GPUs for 100 epochs. All the hyper-parameters in the model architecture follow the convention in Trajectron++ for the sake of fariness. The learning rate is set to 0.001 initially and decayed exponentially every epoch with a decay rate of 0.9999. The model is trained using Adam gradient descent and gradients are clipped at 1.0.

Also, we adopt the in-training mixup technique from the original paper [4]. Its pseudocode is shown in Fig.2. Though

TABLE I

MEANADE/MEANFDE VALUES FOR TRAJECTRON++ AND TRAJECTRON++ WITH TWO DIFFERENT MIXUP IN BEST OF 20 MODE ON DIFFERENT DATASETS. BOLD INDICATES BEST.

	ZARA1	HOTEL	ETH	UNIV	ZARA2
baseline	**0.15/0.33**	**0.11/0.19**	**0.41**/0.83	0.20/0.45	**0.11/0.25**
uniform mixup	**0.15/0.33**	0.12/0.22	**0.41**/0.85	**0.19/0.42**	0.12/0.26
beta mixup	**0.15/0.33**	0.12/0.22	**0.41/0.82**	**0.19/0.42**	0.12/0.26

TABLE II

MEANADE/MEANFDE VALUES FOR TRAJECTRON++ PLUS DYNAMIC INTEGRATION AND TRAJECTRON++ PLUS DYNAMIC INTEGRATION WITH TWO DIFFERENT MIXUP IN BEST OF 20 MODE ON DIFFERENT DATASETS. BOLD INDICATES BEST.

	ZARA1	HOTEL	ETH	UNIV	ZARA2
baseline	0.17/**0.32**	**0.12/0.20**	0.44/0.89	0.22/0.43	**0.12/0.25**
uniform mixup	**0.16/0.32**	0.13/0.21	0.45/0.89	**0.21/0.42**	0.13/0.26
beta mixup	0.17/**0.32**	**0.13/0.20**	**0.43/0.85**	**0.21/0.42**	0.13/0.26

our data is in graph structure, we still can apply the convex combination to the positions of each pair of nodes and the weights of each pair of weights.

```
for (x1, y1), (x2, y2) in zip(loader1, loader2):
    lam = numpy.random.beta(alpha, alpha)
    x = Variable(lam * x1 + (1. - lam) * x2)
    y = Variable(lam * y1 + (1. - lam) * y2)
    optimizer.zero_grad()
    loss(net(x), y).backward()
    optimizer.step()
```

Fig. 2. Mixup Training Scheme [4]

C. Evaluation Metrics

a) Final Displacement Error (FDE): FDE is the l_2 distance between the predicted final position and the ground truth final position with future horizon F.

$$\text{FDE} = \sum_{i=1}^{\mathbf{N}(t)} \frac{1}{\mathbf{N}(t)} \|\hat{\mathbf{s}}_i^{t+F} - \mathbf{s}_i^{t+F}\|_2, \tag{8}$$

b) Average Displacement Error (ADE): ADE is the l_2 distance between the ground truth and predictions.

$$\text{ADE} = \sum_{i=1}^{\mathbf{N}(t)} \sum_{\tau=t+1}^{t+F} \frac{1}{\mathbf{N}(t)} \frac{1}{F} \|\hat{\mathbf{s}}_i^{\tau} - \mathbf{s}_i^{\tau}\|_2, \tag{9}$$

where $\mathbf{N}(t)$ is the total number of pedestrians at time t, F is the number of future horizon we want to predict for, $\hat{s}_i^{\tau}$ is the predicted trajectory for pedestrian i.

D. Quantitative Comparisons

We run a series of experiments to demonstrate the effectiveness of mixup. Specifically, there are two different versions in Trajectron++: one is with dynamic integration and the other is without. We compare each of them with the one with uniform mixup ($\lambda \sim \mathbf{U}(0, 1)$) and beta mixup ($\lambda \sim \beta(0.5, 0.5)$). The one without dynamic integration is shown in Table I and the one with it is shown in Table II. As we can observe, the majority of models with mixup report lower meanADE and meanFDE than the baselines.

In addition, in Fig.3, Fig.4, and Fig.5, the grey curve is the baseline, the orange curve is the beta mixup, and the red curve is the uniform mixup. As we can see, though the training loss for two mixup experiments are higher than the one for the baseline, as expected, the performances of the models with mixup on the validation data are much better than the baseline. Thus, the higher training loss and yet the lower validation ADE and FDE indicates the regularized effects of mixup technique in the problem of agents forecasting, confirming previous theoretical justification of vicinal risk minimization and the consistency of linear interpolation.

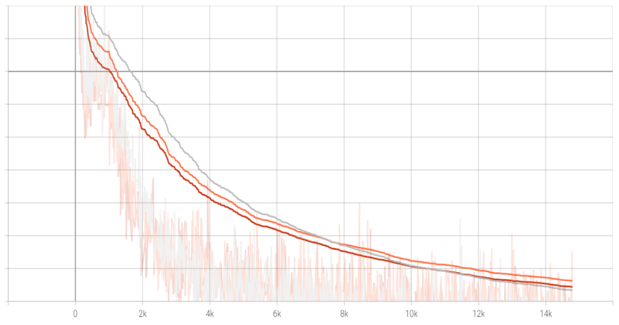

Fig. 3. Training Loss for ETH dataset

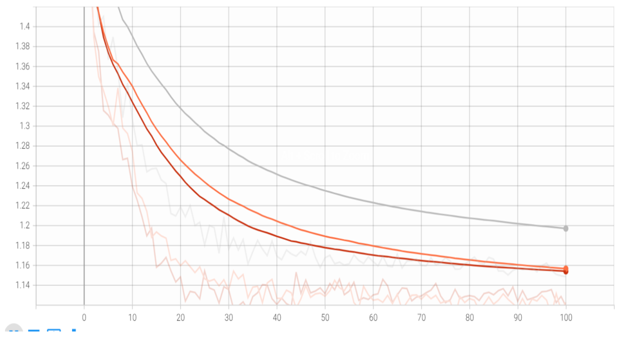

Fig. 4. Evaluation meanADE for ETH dataset

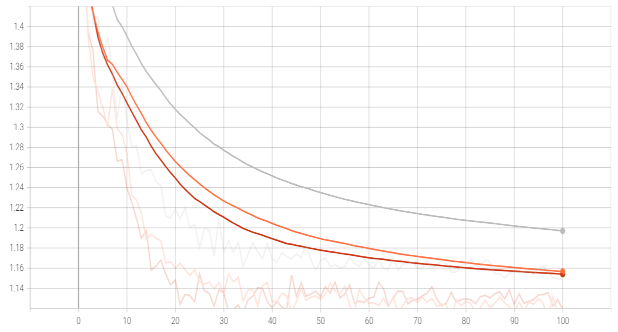

Fig. 5. Evaluation meanFDE for ETH dataset

E. Outcome Visualization

We utilize matplotlib to visualize the predictions from Trajectron++ with mixup. In Fig.6 and Fig.7, we present two sample predictions of Trajectron++ with mixup. Each green node represents an agent. Each black dotted line represents the past history of an agent; each white dotted represents the respective ground-truth future trajectory; and blue line represents a sample that the model outputs. The reason why there is a collection of samples instead of single one is because Trajectron++ equipped with a Gaussian Mixture Model component in the final layer of the decoder is not deterministic.

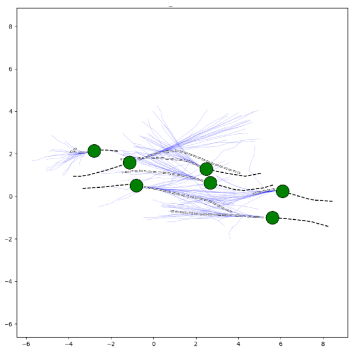

Fig. 6. Sample Prediction 1

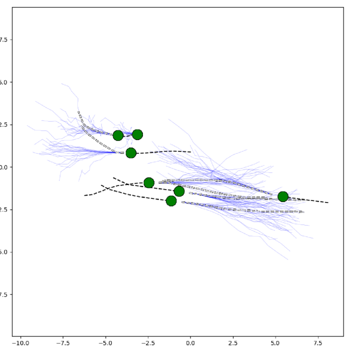

Fig. 7. Sample Prediction 2

V. Conclusion

In this paper, we present a novel approach to improve one of the state-of-the-art models for agent forecasting Trajectron++ through the perspective of data augmentation. We provide an analytic proof of the regularization of mixup technique and its approach in the domain of trajectory predictions. In addition, a series of ablation studies demonstrate how mixup technique regularizes the inductive bias of the model and achieves better generalization.

Admittedly, there is still a huge potential to further explore the mixup technique in this field. First, many generative models utilizing the graph structural data could leverage on the invariant permutation of the graph structure. Specifically, in mixup, we only mixup data points across time domain and agent domain, but the graph structural data could allow us to mixup data points across several permuted graphs. This regularized effect should be much more powerful than the counterpart with the vanilla mixup. Secondly, in this work, we treat Trajectron++ as a blackbox without touching any internal part. On one hand, it facilitates the understanding of mixup in terms of the overall objective much better. On the other hand, how mixup technique transforms the high dimensional spaces within the model itself is beyond the scope of our knowledge. Hopefully, these intricacies could be uncovered in the near future, contributing to the development of deep learning.

References

[1] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control," *CoRR*, vol. abs/2001.03093, 2020. [Online]. Available: http://arxiv.org/abs/2001.03093

[2] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *2009 IEEE 12th international conference on computer vision*. IEEE, 2009, pp. 261–268.

[3] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," in *Computer graphics forum*, vol. 26, no. 3. Wiley Online Library, 2007, pp. 655–664.

[4] H. Zhang, M. Cissé, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *CoRR*, vol. abs/1710.09412, 2017.

[5] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Physical Review E*, vol. 51, no. 5, pp. 4282–4286, may 1995.

[6] A. Treuille, S. Cooper, and Z. Popović, "Continuum crowds," *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3, pp. 1160–1168, 2006.

[7] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer school on machine learning*. Springer, 2003, pp. 63–71.

[8] J. M. Wang, D. J. Fleet, and A. Hertzmann, "Gaussian process dynamical models for human motion," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 2, pp. 283–298, 2007.

[9] K. Das and A. N. Srivastava, "Block-gp: Scalable gaussian process regression for multimodal data," in *2010 IEEE International Conference on Data Mining*. IEEE, 2010, pp. 791–796.

[10] N. Lee and K. M. Kitani, "Predicting wide receiver trajectories in american football," in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2016, pp. 1–9.

[11] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 961–971.

[12] A. Jain, A. R. Zamir, S. Savarese, and A. Saxena, "Structural-rnn: Deep learning on spatio-temporal graphs," in *Proceedings of the ieee conference on computer vision and pattern recognition*, 2016, pp. 5308–5317.

[13] J. Morton, T. A. Wheeler, and M. J. Kochenderfer, “Analysis of recurrent neural networks for probabilistic modeling of driver behavior,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1289–1298, 2016.

[14] A. Vemula, K. Muelling, and J. Oh, “Social attention: Modeling attention in human crowds,” in *2018 IEEE international Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 4601–4607.

[15] K. Sohn, H. Lee, and X. Yan, “Learning structured output representation using deep conditional generative models,” *Advances in neural information processing systems*, vol. 28, 2015.

[16] N. Deo and M. M. Trivedi, “Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms,” in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1179–1184.

[17] B. Ivanovic and M. Pavone, “The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2375–2384.

[18] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, “Desire: Distant future prediction in dynamic scenes with interacting agents,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 336–345.

[19] N. Rhinehart, R. McAllister, K. Kitani, and S. Levine, “Precog: Prediction conditioned on goals in visual multi-agent settings,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 2821–2830.

[20] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks,” *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[21] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, “Social gan: Socially acceptable trajectories with generative adversarial networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 2255–2264.

[22] V. Kosaraju, A. Sadeghian, R. Martín-Martín, I. Reid, H. Rezatofighi, and S. Savarese, “Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks,” *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[23] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, “Sophie: An attentive gan for predicting paths compliant to social and physical constraints,” in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 1349–1358.

[24] T. Zhao, Y. Xu, M. Monfort, W. Choi, C. Baker, Y. Zhao, Y. Wang, and Y. N. Wu, “Multi-agent tensor fusion for contextual trajectory prediction,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 126–12 134.

[25] D. Ding and H. H. Huang, “A graph attention based approach for trajectory prediction in multi-agent sports games,” *arXiv preprint arXiv:2012.10531*, 2020.

[26] Y. Yuan, X. Weng, Y. Ou, and K. M. Kitani, “Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 9813–9823.

[27] K. Mangalam, Y. An, H. Girase, and J. Malik, “From goals, waypoints &amp; paths to long term human trajectory forecasting,” 2020. [Online]. Available: https://arxiv.org/abs/2012.01526

[28] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” *CoRR*, vol. abs/1505.04597, 2015.

[29] E. Talavera, G. Iglesias, . Gonzlez-Prieto, A. Mozo, and S. Gmez-Canaval, “Data augmentation techniques in time series domain: A survey and taxonomy,” 2022. [Online]. Available: https://arxiv.org/abs/2206.13508

[30] B. Ivanovic, E. Schmerling, K. Leung, and M. Pavone, “Generative modeling of multimodal multi-human behavior,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3088–3095.

[31] O. Chapelle, J. Weston, L. Bottou, and V. Vapnik, “Vicinal risk minimization,” *Advances in neural information processing systems*, vol. 13, 2000.