Social-Implicit: Rethinking Trajectory Prediction Evaluation and The Effectiveness of Implicit Maximum Likelihood Estimation

Abduallah Mohamed[1], Deyao Zhu[2], and Warren Vu[1]
Mohamed Elhoseiny[2,*] Christian Claudel[1,*]

[1] The University of Texas at Austin
{abduallah.mohamed, warren.vu, christian.claudel}@utexas.edu
[2] KAUST
{deyao.zhu, mohamed.elhoseiny}@kaust.edu.sa
[3] * Equal advising

Abstract. Best-of-N (BoN) Average Displacement Error (ADE)/ Final Displacement Error (FDE) is the most used metric for evaluating trajectory prediction models. Yet, the BoN does not quantify the whole generated samples, resulting in an incomplete view of the model's prediction quality and performance. We propose a new metric, Average Mahalanobis Distance (AMD) to tackle this issue. AMD is a metric that quantifies how close the whole generated samples are to the ground truth. We also introduce the Average Maximum Eigenvalue (AMV) metric that quantifies the overall spread of the predictions. Our metrics are validated empirically by showing that the ADE/FDE is not sensitive to distribution shifts, giving a biased sense of accuracy, unlike the AMD/AMV metrics. We introduce the usage of Implicit Maximum Likelihood Estimation (IMLE) as a replacement for traditional generative models to train our model, Social-Implicit. IMLE training mechanism aligns with AMD/AMV objective of predicting trajectories that are close to the ground truth with a tight spread. Social-Implicit is a memory efficient deep model with only 5.8K parameters that runs in real time of about 580Hz and achieves competitive results.

Keywords: Motion Prediction, Motion Forecasting, Deep Graph CNNs, Evaluation, Trajectory Forecasting

1 Introduction

Trajectory prediction is an essential component for multiple applications, such as autonomous driving [5,16,4,35,28], augmented reality [18,34], and robotics [19,2]. Typically, solving this problem requires a generative model to predict the future agent's trajectories. Though there are plenty of deep models and design architectures that tackle this problem, the evaluation method used is being questioned.

Code: https://github.com/abduallahmohamed/Social-Implicit/

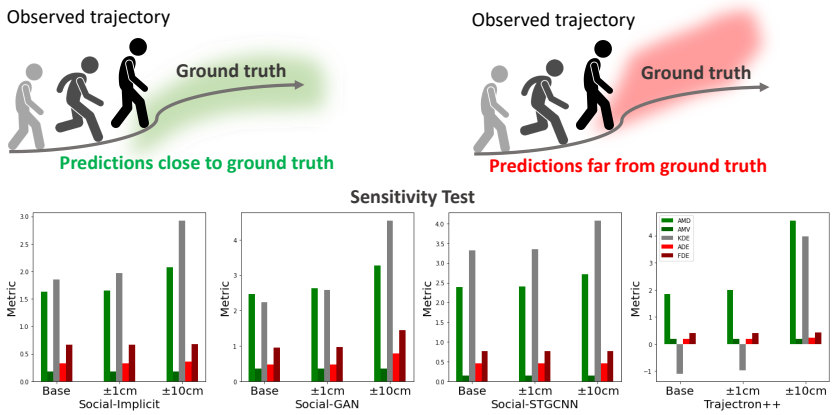

Fig. 1. The current BoN ADE/FDE metrics are not sensitive to the predicted distribution. The BoN ADE/FDE only focuses on the closest sample to the ground truth. We can see for both the green and red predictions, the BoN ADE/FDE stays the same. On the other hand, the proposed AMD/AMV metrics changes based on how close the whole predicted distribution and it is spread with respect to the ground truth. This makes the AMD/AMV a better metric to evaluate the predictions.

Typically, two metrics are used to evaluate the trajectory predictions models. The first one is Average Displacement Error (ADE) [24] which is the average L_2 distance between the predicted and ground truth trajectories. Lower ADE values means that the overall predicted trajectory is close to the ground truth. The other metric is the Final Displacement Error (FDE) [1], which is an L_2 distance between the two final predicted and ground truth locations. In other terms, it describes if the predicted agent reaches its last goal or not. Also, the lower the FDE is, the better the model in not accumulating errors during the predictions. This issue of accumulating errors, resulting in a higher FDE was noticed in prior works that used recurrent based architectures. Prior works introduced the idea of a full CNN based architecture [23] to solve this error accumulation behavior.

Yet, this ADE/FDE metric remains unsuitable for generative models. Generative models predict multiple samples of future trajectories, implicitly forming a predicted distribution. This generative behavior is suitable for the problem, as the motion of an agent or pedestrian can be a multi-modal with possible future trajectories. In order to use the ADE/FDE in generative settings, the works of [1,8] introduced the concept of Best-of-N (BoN). BoN technique chooses from N samples the closest sample to the ground truth and calculates the ADE/FDE metric on it. This has a main issue of ignoring the set of generated samples. A model might generate an outlier sample that is luckily close enough to the ground truth, while the other samples are way off from the ground truth. This approach also fails in real-life applications, as there is a lack in the assessment of the predictions. Some important components, such as motion planning and

collision avoidance, need a complete view of the predictions. Another issue we noticed that the recent models [30,37,22,20] which are state-of-the-art based on the ADE/FDE metric only differ by 1cm ADE and few centimeters FDE on the ETH [24] and UCY [14] datasets, one of the most commonly used datasets in this area. The 1cm difference between a previous SOTA model and the next one is so subtle and tiny that it can be an annotation error or an outlier sampling. Thus, there is a need for a new metric that can evaluate the whole predicted samples and have a sense of where the whole generated distribution is regarding the ground truth. Also, there is a need to quantify the uncertainty of the generated samples giving a view regarding the confidence of the model, something that is needed in real-life applications. For this, we introduce the usage of Mahalanobis Distance [21] as a metric in this domain. We introduce two metrics, the Average Mahalanobis Distance (AMD) which evaluates how close a generated distribution is with respect to the ground truth, and the Average Maximum Eigenvalue (AMV) that evaluates the confidence of the predictions. The AMD quantifies how close a ground truth point is to a predicted distribution in a sense of standard deviation units. Also, AMD connects with χ^2 distribution, helping us to determine the confidence of our predictions when the generated distribution degree of freedom is known. The AMV depends on the maximum magnitude of the eigenvalues of the covariance matrix of the predicted distribution. It quantifies the spread of the prediction. Thus, we can tell if a model is more confident than another model by using it. So, our goal is to achieve a model that generates a distribution which is close to the ground truth and has a small spread of samples around the ground truth. This aim leads us to rethink the nature of generative models used in training motion prediction models. We can classify the used generative techniques into parametric and non-parametric ones. Parametric ones use Maximum Likelihood Estimation (MLE) to model the predicted trajectories as Gaussian or Gaussian Mixture Models (GMM). Generative Adversarial Networks (GANs) [7] is an examples of non-parametric distributions. These approaches learn the distribution of the observed trajectories in order to generate the future ones. Yet, the primary goal of trajectory prediction models is the generated samples themselves. The MLE needs plenty of samples to converge, something we do not have in practice. While the GANs rely on the design of the discriminator and VAEs need to optimize the Evidence Lower Bound (ELBO). So, we needed a generative approach that only focuses on the generated samples and does not come with extra hassles. In this work, we show that Implicit Maximum Likelihood Estimation (IMLE) technique is an effective alternative to these approaches. IMLE focuses directly on the predicted trajectories, simplifying the optimization function. By using IMLE to train our introduced model Social-Implicit, the predicted trajectories improve in terms of quality and accuracy in comparison with prior works. Social-Implicit is a memory efficient deep model with only 5.8K parameters almost 55x less than the closest SOTA and runs in real-time almost 8.5x faster than the closest SOTA. This work is organized as follows: We start by literature review of recent relative works. Then we formulate the motion prediction problem, followed by an intro-

duction and discussion for both new metrics AMD and AMV. Then we introduce the Trajectory Conditioned IMLE mechanism used in training our model Social-Implicit. We follow this by explaining the architecture of Social-Implicit. Lastly, we analyze the results of the new metrics on our model and recent SOTA ones accompanied with a sensitivity analysis.

2 Literature Review

Trajectory Forecasting Models Recent works have proposed various models to forecast future trajectories. Based on their output formats, they can be roughly grouped into two categories. Explicitly modeling the future as a parametric distribution, or implicitly modeling the future as a non-parametric distribution. In the first category, methods model the future explicitly as continuous or discrete distribution [1,23,27,3,30,32,4,17,38,13,26,36,37]. For example, S-LSTM [1] and S-STGCNN [23] use Gaussian distribution to model the future trajectory that are trained by Maximum Likelihood Estimation (MLE). Gaussian distribution is single-mode and cannot catch the multi-modality of the future. To tackle this issue, PRECOG [27], Trajectron++ [30], ExpertTraj [37], and AgentFormer [36] learn a latent behavior distribution, which can be either discrete [30,37] or continuous [27,36], to represent the agent multi-modality intent. In these works, the predicted Gaussian distribution is generated conditioned on the sampled latent intent. This type of method is usually based on Conditional VAE [31]. Besides continuous distributions like Gaussian methods like MTP [4] and LaneGCN [17] use a discrete distribution to represent the future. These methods predict a fixed number of deterministic trajectories as the future candidates and use a categorical distribution to model their possibilities. In the second category, some methods model the future distribution in an implicit way. For example, S-GAN [8], SoPhie [29], S-BiGAT [11] and DiversityGAN [9] follows a Conditional GAN [6] architecture. Instead of generating a distribution as the model output they predict a deterministic trajectory that is conditioned on a random sampled noise and is trained by an adversarial loss mechanism. Our proposed method Social-Implicit models the future distribution implicitly by training it using IMLE [15] avoiding additional hassles like the discriminator in a GAN training mechanism.

Trajectory Forecasting Metrics Most of the trajectory forecasting methods are evaluated by the metric Average Displacement Error (ADE) [24] or Final Displacement Error (FDE) [1]. These two metrics are based on the L_2 distance of the whole temporal horizon (ADE) or the last time step (FDE) between the prediction and the ground truth trajectory. When the model generates a distribution as the output, the Best-of-N trick [8] is applied to evaluate the best trajectory only from N sampled predictions. The mean ADE/FDE can be also used to evaluate the predictions, it is mostly suitable in single modality predictions and when the predictions are close to a Gaussian distribution. In multi-modality, when for example the predictions are contradictory to each other (turning left, turning right) the mean ADE/FDE will fail because it is deterministic in nature. Another

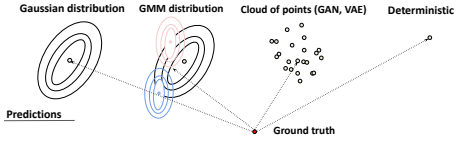

Fig. 2. Different motion prediction models output. The Gaussian and GMM are examples of parametric models. The GAN and VAE are examples of non-parametric models. The last category is a deterministic model output. A unified metric is needed to evaluate all of these models.

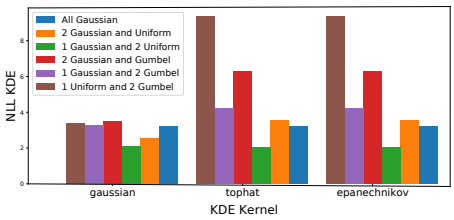

Fig. 3. Different mixture models NLL KDE vs the choice of the KDE Kernel. The lower the better. We also notice that the results vary depending on the kernel.

way to evaluate the distribution quality is Kernel Density Estimate (KDE), first used in [10]. KDE fits a kernel-based distribution from the prediction samples and estimates the negative log-likelihood of the ground truth as the evaluated score. Quehl et al. [25] propose a synthesized metric that is a weighted sum of different similarity metrics to alleviate the metric bias. But their metric is only suitable for deterministic models. We propose two new metrics Average Mahalanobis Distance (AMD) and Average Maximum Eigenvalue (AMV) that are a better alternative for the BoN ADE/FDE in evaluating the predictions.

3 The Average Mahalanobis Distance (AMD) metric

We define the problem of trajectory prediction as follows: Given an observed trajectories of N agents across a sequence of observed time steps T_o, the goal is to predict the next T_p prediction time steps. The observed trajectories contain P points, where each point indicates the spatial location of an agent. In the pedestrian trajectory prediction problems the P is a 2D Cartesian locations (x,y). We denote the set of observation to be $d_o = \{p_t |\ t \in T_o\}$ and the set of predictions to be $d_p = \{p_t |\ t \in T_p\}$.

To highlight the issue in the current BoN ADE/FDE, we start with Fig. 2 that illustrates the different types prediction models outputs. For deterministic models, it is straightforward to compute the ADE/FDE metrics defined in Equation 1. But for generative models, the ADE/FDE is being computed by the BoN approach. The BoN works by sampling N (usually 20) samples, selecting the closest sample to the ground truth, then using this sample to calculate the ADE/FDE. We can criticize this BoN approach in multiple aspects. The major concern is that it does not quantify the whole generated samples and only focuses on the closest one. This might disadvantage a model with a density that is surrounding the ground truth against another model with a density that is completely off the ground truth, but has one sample that is close to the ground truth. We can see this illustrated in the teaser Fig. 1. We base the other concern that with this method of BoN, one can run the metric a couple of times, getting

a result that is 1 cm better than another model. In some extreme cases, a lucky random run might have a very low BoN ADE/FDE. The work of [10] noticed this issue and introduced the usage of A Kernel Density Estimate (KDE) defined in Equation 1. The KDE is a kernel based tool that gets a non-parametric representation of the predictions' probability density. Then, the negative log likelihood of the ground truth is calculated and reported in logarithmic units (nats). Yet, there is a mix of limitations and concerns with the KDE metric. The main concern is that the KDE metric is sensitive to the choice of a kernel under the settings of low number of samples, which is the case in real-life datasets. Fig. 3 illustrates the different choices of the kernel used in the KDE versus a variety of mixtures of distributions. We notice that when a Gaussian kernel is being used; it does not differentiate between different samples and might favour a model with a full GMM output in comparison with other outputs. We also notice, we might get a mixed results whenever a different choice of kernel is being used, such as with tophat kernel versus a Gaussian kernel. The work of [10] was using KDE metrics with a Gaussian kernel. The other limitation of the KDE kernel is that it does not contain analytical properties that are easy to interpret. This limitation is because of the non-parametric nature of the KDE. Such properties of interest might be the probability moments and the confidence intervals.

$$\text{ADE} = \frac{1}{N \times T_p} \sum_{n \in N} \sum_{t \in T_p} \|\hat{p}_t^n - p_t^n\|_2, \text{FDE} = \frac{1}{N} \sum_{n \in N} \|\hat{p}_{T_p}^n - p_{T_p}^n\|_2, \text{KDE} = \frac{-1}{N \times T_p} \sum_{n \in N} \sum_{t \in T_p} \log \text{KDE}(\hat{p}_t^n, p_t^n) \tag{1}$$

Where p_t^n is ground truth location of agent $n \in N$ at predicted time step $t \in T_p$ and $\hat{p}_t^n$ is the predicted location. The new metric needs to be parametric that allows further analysis and insensitive to the way it calculates the distance. Thus, we introduce the usage of Mahalanobis distance. Mahalanobis distance can measure how far a point from a distribution is, while correlating the distance with the variance of the predictions. It also has analytical properties that connect it with the Chi-square distribution, in which one can evaluate the confidence of the predictions. Lastly, it depends on Gaussian distribution, which allows further analysis of the predicted moments. The Mahalanobis distance (MD) is defined as: $M_D\left(\hat{\mu}, \hat{\Sigma}, p\right) = \sqrt{(p-\hat{\mu})^T \hat{\Sigma}^{-1} (p-\hat{\mu})}$. Where, $\hat{\mu}$ is the mean of the prediction, $\hat{\Sigma}$ is the variance of the predicted distribution and p is the ground truth location. Originally, Mahalanobis distance was not designed for a GMM distribution. Yet, the work of [33] extended MD into a GMM by formulating it as:

$$M_D\left(\hat{\mu}_{\text{GMM}}, \hat{G}, p\right) = \sqrt{(\hat{\mu}_{\text{GMM}} - p)^T \hat{G} (\hat{\mu}_{\text{GMM}} - p)} \tag{2}$$

Where $\hat{G}$, the inverse covariances of each mixture components averaged and weighted probabilistically, is defined as: $\hat{G} = \frac{\sum_{k=1}^{K} \hat{\Sigma}_k^{-1} \hat{\pi}_k \int_{\hat{\mu}_{\text{GMM}}}^{p} p(x|k)dx}{\sum_{k=1}^{K} \hat{\pi}_k \int_{\hat{\mu}_{\text{GMM}}}^{p} p(x|k)dx}$, where K is the number of mixture components, $\hat{\pi}_k$ is the weight of the kth component and the mean of the GMM is defined as: $\hat{\mu}_{\text{GMM}} = \sum_{k=1}^{K} \hat{\pi}_k \hat{\mu}_k$. The integral term in $\hat{G}$ is tractable, as noted in [33]. We notice that if the GMM contains only one component, the G will be the $\hat{\Sigma}^{-1}$, thus the Tipping's MD is a more

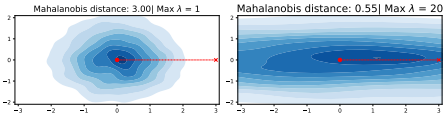

Fig. 4. The Mahalanobis distance is being measured for a test point marked by x. Two Bi-Variate Gaussian distributions are shown, the one on the left has a lower variance than the one on the right. λ stands for the maximum absolute eigenvalue of the distributions covariances.

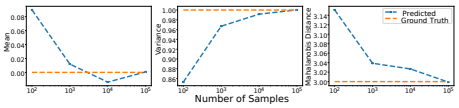

Fig. 5. GMM fit exhibit an error and need a lot of samples to converge into the true mean, something might be a disadvantage to non-parametric models. At the 1000 samples mark, GMM starts to converge to the true mean and variance with stable MD.

generalized version of the original MD. Our approach is the following, whatever distribution or output produced by a model, we fit into a GMM. A question will be raised regarding the number of optimal mixture components K. This can be easily solved by using the Bayesian information criterion (BIC): BIC $= m \ln n - 2 \ln \hat{L}_{\text{GMM}}$, where m is the number of parameters of the GMM model, n is the number of observed data points and $\hat{L}_{\text{GMM}}$ is the likelihood function of the model. The lower the BIC is the better the fitted GMM model representing the data points. The best GMM is chosen automatically based on the BIC. Looking into the reason for sampling from a model that is already predicting the mean and variance of trajectories such as [1,23], that we want to be fair. Fitting a GMM will carry out a sort of error, thus we want this error to be incorporated into all modes of measurements to have a unified metric. Fig. 5 show this error.

Because a deterministic model does not have a variance, we need a representation of the error in the model. We can train the deterministic model multiple times and fit the predictions to a GMM. Another proposal is to calculate the ensemble mean and variance and directly apply the MD distance without the GMM fit. The later approach might have an error that is equivalent to the GMM fit error, making the metric more fair. In the supplementary, we discuss both cases. We believe that evaluating a deterministic model versus a generative model is an open question that needs a further research and it is a limitation similar to the KDE [10] limitation. Now, we define the Average Mahalanobis Distance (AMD):

$$\text{AMD} = \frac{1}{N \times T_p} \sum_{n \in N} \sum_{t \in T_p} M_D \left(\hat{\mu}^n_{\text{GMM},t}, \hat{G}^n_t, p^n_t \right) \tag{3}$$

4 The Average Maximum Eigenvalue (AMV) Metric

A major concern of the AMD metric is that it is highly correlated with the variance of the distribution. A model might predict future trajectories, with a huge on-practical variance having the ground truth close to the mean. This will lead to a very low AMD in comparison with another model with a higher

variance. Another example is a model that predicts a huge variance that is in meters covering all the predicted points. This also will lead to an optimal AMD value. To counter this false behavior, we need our models to have a low AMD accompanied with a low variance aka a more certain model. Also, in practical application, we need to quantify the overall uncertainty of the predictions to have a holistic view of the performance. Thus, we introduce the usage of the eigenvalues of the covariance matrix. The largest magnitude eigenvalue of the covariance matrix is an indicator of the spread of the covariance matrix. Fig. 4 illustrates two distributions, the one on the left has a smaller variance than the one on the right. We notice that the MD of a fixed point with respect to the left distribution is much higher when compared to the right distribution. Yet, the largest magnitude eigenvalue of the left distribution is way less than the right distribution, showing the spread of the predictions. So, to properly evaluate the models we need both of the AMD and a measurement of the spread. And as we discussed we can have a measurement of the spread directly from the prediction covariance matrix. Because of the framework we introduced in the AMD metric, we have a covariance matrix of the prediction. Something that was missing in the KDE metric. Now, we can introduce the AMV metric:

$$\text{AMV} = \frac{1}{N \times T_p} \sum_{n \in N} \sum_{t \in T_p} \lambda_1^{\downarrow}(\hat{\Sigma}^n_{\text{GMM},t}) \tag{4}$$

Where $\lambda_1^{\downarrow}$ is the eigenvalue with the largest magnitude from the matrix eigenvalues. The $\hat{\Sigma}_{\text{GMM}}$ is the covariance matrix of the predicted GMM distribution defined as:$\hat{\Sigma}_{\text{GMM}} = \sum_{k=k}^{K} \hat{\pi}_k \hat{\Sigma}_k + \sum_{k=1}^{K} \hat{\pi}_k (\hat{\mu}_k - \hat{\mu}_{\text{GMM}})(\hat{\mu}_k - \hat{\mu}_{\text{GMM}})^T$. Thus, the AMV becomes a metric that evaluates the overall spread of the predicted trajectories. A model with low AMD will have a predicted distribution that is closer to the ground truth. And a model with low AMV will be more certain in their predictions. *Thus, a model with both low AMD/AMV average is preferred when compared with another model with a higher AMD/AMV average.* For this we use the $\frac{AMD+AMV}{2}$ as an indicator of a good model.

5 Trajectory Conditional Implicit Maximum Likelihood Estimation (IMLE) Mechanism

By induction from the goal of the AMD/AMV metric to have a model that generates samples that is close to the ground truth with low spread, we need a training mechanism that allows full control over the predicted samples as the main optimization goal. Typical training mechanism such as Maximum Likelihood Estimation (MLE) or its variants like maximizing evidence lower bound (ELBO) encourages the prediction samples to be close to some (ground truth) data sample. In this way, some data examples may be missed and lead to a mode dropping [15]. Other methods, such as GANs, need to introduce additional modules like the discriminator and their training is usually unstable and need to be carefully tuned to reach a proper Nash's equilibrium. The work of [15] introduced

the concept of Implicit Maximum Likelihood Estimation (IMLE). IMLE encourages every target ground truth to be close to some predicted samples. Therefore, it leads to a predicted distribution that is better in covering the ground truth unlike MLE. IMLE trains a model via a simple mechanism: inject a noise into the model's input to predict multiple samples, select the one closest to the ground truth, and back propagate using this sample. Unlike other generative approaches, IMLE does not load the optimization objective with a specific training technique and keeps the training stable due to the simple distance-minimization-based optimization. Using IMLE as a training mechanism aligns with the AMD/AMV goals and focuses on the important product, the predicted output. Another point of view for IMLE is that it is a more advanced neural technique in comparison with estimation techniques such as Kalman filter where the process and measurement noises drive the model. We refer the reader to the original IMLE paper [15] for a further discussion. The training mechanism is shown in Alg.1:

Algorithm 1 Trajectory Conditional Implicit Maximum Likelihood Estimation (IMLE) algorithm

Require: The dataset $D = (d_o^i, d_p^i)_{i=1}^n$ and the model $\theta(.)$ with a sampling mechanism conditioned on the input
Require: Choose a proper loss function $\mathcal{L}(.)$ such as mean squared error or L_1
  Initialize the model
  for $e = 1$ **to** Epochs **do**
    Pick a random batch (d_o, d_p) from D
    Draw i.i.d. samples $\tilde{d_p}^1, \ldots, \tilde{d_p}^m$ from $\theta(d_o)$
    $\sigma(i) \leftarrow \arg\min_i \mathcal{L}\left(d_p - \tilde{d_p}^i\right)\ \forall i \in m$
    $\theta \leftarrow \theta - \eta \nabla_\theta \sigma(i)$
  end for
  return θ

6 The Social-Implicit Model

In this section, we present the Social-Implicit model. The Social-Implicit is tiny in memory size with only 5.8K parameters with real run-time of 588Hz. The method comprises three concepts, Social-Zones, Social-Cell and Social-Loss.
The Social-Zones: The Social-Zones cluster the observed agents trajectories based on their maximum change of speed. The average pedestrian speed is 1.2m/s [12]. We noticed that we can cluster the motion of pedestrians into four groups. The first group is the motion-less group, where the pedestrian is waiting at the traffic light as an example. This group's maximum speed change is between 0-0.01m/s. While the second group is pedestrians with minimal motion, aka someone who is shaking in place or a group of pedestrians greeting each other, typically this group's maximum change of speed is between 0.01-0.1m/s.

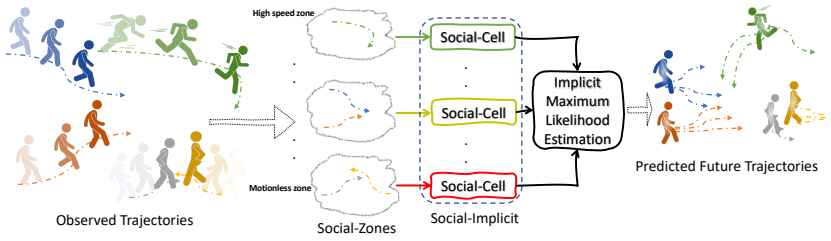

Fig. 6. Social-Implicit model concept. The Social-Zones cluster the observed trajectories based on their maximum observed speed. Then each Social-Zone is processed by a Social-Cell. The model is trained using IMLE.

The third group are pedestrians with an average walking speed, these pedestrians motion is between 0.1-1.2m/s. The last group is the running pedestrians, typically with a speed above the 1.2m/s average. When a deep model is trained on the stationary pedestrians alongside the faster ones, a bias towards the moving ones will exist in the predictions. This will force the model to predict the non-moving objects as moving ones. It is a sort of data imbalance, or in other terms, a zero(motionless)-inflated data issue. Hence, the concept of Social-Zones is needed to solve this issue. Empirically, we show that our model with the Social-Zones performs better than without it. The input to the Social-Zones is the observed trajectories and the output is clusters of pedestrians, each cluster is a graph of dimensions $P \times T_o \times N$.

The Social-Cell: The fundamental building unit of the Social-Implicit model is the Social-Cell. The Social-Cell is a 4 layers deep model that is simple and directly dealing with the spatio-temporal aspects of the observations. Fig. 7 illustrates the structure of the Social-Cell. We notice the cell has two components, one that deals with each individual agent at a local level and one that deals with the whole agents at a global level. We generate the final output of the cell by combining the local and global streams via self-learning weights. Both local and global streams are two consecutive residually connected CNN layers. The first CNN is a spatial CNN, which creates an embedding of the spatial information of the observed agents. The second layer is a temporal CNN that treats the time aspect of the observed trajectories. It treats the time as a feature channel, allowing us to predict the next T_p time steps without using a recurrent network [23]. We found out that this simple architecture is as effective as much larger and complex models, resulting in a small memory size and real-run time capabilities. Each Social-Cell deals with a specific Social-Zone. The input is $P \times T_o \times N$ and the output is $P \times T_p \times N$. The operations are shown in Fig. 7. **The Social-Loss:** The loss function of Social-Implicit exhibits several parts. The first part is the direct optimization objective of the IMLE mechanism that we discussed before. The second part is a triplet loss. This triplet loss considers the anchor to be the closest sample $\tilde{d}_p^1$ to the ground truth. The positive example is the next closest example $\tilde{d}_p^2$ to the ground truth. The negative example $\tilde{d}_p^m$ is the farthest sample

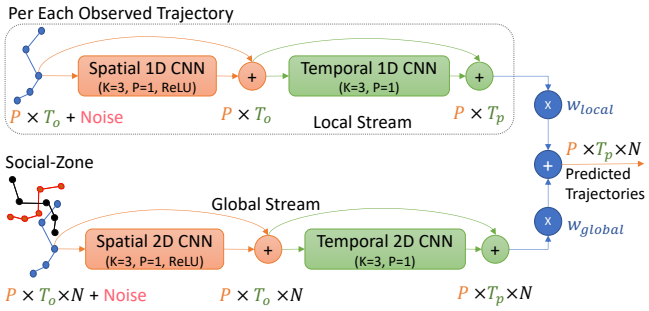

Fig. 7. Social-Cell model. The local and global stream has only two CNNs. P is the observed location, T_o and T_p is the observed and predicted time steps and N is the count of agents. Where K and P of the CNNs is the kernel and padding size.

from the ground truth. This helps in grouping the samples closer to the ground truth, resulting in a tighter distribution around the real trajectory. The last part of the loss is a geometric loss function that treats the predicted locations as a polygon. First, it ensures that the intra-distance between the predicted location matches the intra-distance between the ground truth locations. Second, it makes sure that the angles between the predicted points are the same as the angles between the ground truth points. It ensures that the predicted scene geometrically looks like the ground truth. We defines these losses in Equation 5. The social aspects of the scene can be addressed beyond what we introduced which is an open research area.

$$\begin{aligned} \mathcal{L}_{\text{triplet}} &= \|\tilde{d}_p^1 - \tilde{d}_p^2\|_1 - \|\tilde{d}_p^1 - \tilde{d}_p^m\|_1 \\ \mathcal{L}_{\text{G-distance}} &= \frac{1}{\frac{T_p(T_p-1)}{2}} \sum_{t \in T_p} \sum_{j \in T_p, j>t} \Big| \|p_t - p_j\|_2 - \|\tilde{p}_t - \tilde{p}_j\|_2 \Big|_1 \\ \mathcal{L}_{\text{G-angle}} &= \frac{1}{\frac{T_p(T_p-1)}{2}} \sum_{t \in T_p} \sum_{j \in T_p, j>t} \Big| \angle(p_t, p_j) - \angle(\tilde{p}_t, \tilde{p}_j) \Big|_1 \end{aligned} \tag{5}$$

Thus we define the Social-Loss as:

$$\mathcal{L} = \|d_p - \tilde{d}_p^1\|_1 + \alpha_1 \mathcal{L}_{\text{triplet}} + \alpha_1 \mathcal{L}_{\text{G-distance}} + \alpha_3 \mathcal{L}_{\text{G-angle}} \tag{6}$$

Where $\alpha_1 = 0.0001, \alpha_2 = 0.0001$ or $0.00001, \alpha_3 = 0.0001$.

7 Experiments and Analysis

We analyze the behavior of the metrics on common pedestrian motion prediction models in terms of overall performance and sensitivity analysis. Then we analyze our model in terms of design components and performance.

7.1 Metrics Sensitivity Analysis & Evaluation

We show that the BoN ADE/FDE metric is not sensitive to the change or shift in the distribution, while the AMD and KDE can quantify such a change. Fig. 1 illustrates this concept. We tested different models by shifting their predicted samples using different amounts, specifically $\pm$1cm and $\pm$10cm. In all of the models, the BoN ADE/FDE metric did not change at all or had a very tiny subtle change. Unlike the metrics that measure the whole distribution, like AMD and KDE the shift of the predicted distribution is reflected into the metric. This concludes that the ADE/FDE metric is not sensitive to the change of the whole distribution, even on a tremendous change of 10cm, which sometimes can define a new SOTA model over another. So, the BoN ADE/FDE metric is incapable of evaluating the whole predicted trajectories. Also, the AMV metric stayed the same, this was expected as only shifting the predictions does not change the variance. We notice that the KDE of Trajectron++ is -ve unlike other models because Trajectron++ output is a GMM distribution which is a bias in the KDE metric due to the kernel choice as we discussed earlier.

To evaluate the metrics quantitatively, we report the AMD/AMV, KDE and ADE/FDE metrics on different motion prediction models using the ETH/UCY datasets. We chose classic ones such as S-GAN [8] and S-STGCNN [23]. We also chose more recent ones such Trajectron++ [30] and ExpertTraj [37]. From Tab. 1 we notice that the last two models, which are considered SOTA, differ by a few centimetres on the ADE/FDE metric. Yet, when we evaluate both of them using the AMD/AMV metrics, we notice that Trajectron++ is performing much better than the ExpertTraj model. From the AMD/AMV metric, ExpertTraj generates a tight distribution that does not surround the ground truth, which results in a higher AMD unlike the Trajectron++. Though, both of ExpertTraj and Trajectron++ have very close ADE/FDE metrics, the quality of the whole predicted samples is completely different. Examining the results of our model, Social-Implicit, we see that it has the lowest AMD/AMV. By digesting the results, the ADE/FDE metric is not an indicative of the overall performance of the models which correlates with the aforementioned sensitivity analysis.

7.2 Ablation Study of Social-Implicit

We conduct an ablation study of the Social-Implicit components. Specifically, the Social-Zones and the Social-Loss. Tab. 2 illustrates the results. We noticed that the existence of the Social-Zones enhanced the AMD metric by almost 40%. It also led to a good AMV value, which enhanced the overall AMD/AMV performance. We notice that the triplet loss alone with the Social-Zones leads improves AMD/AMV. The effect of geometric angle loss is more than the geometric distance loss in improving the AMD/AMV. While both work better together.

7.3 Inference and Memory analysis

Social-Implicit besides being the most accurate model when compared with other models on the AMD/AMV metrics, it is the smallest and the fastest in terms

Table 1. For all metrics, the lower the better. The results are on the ETH/UCY datasets. M is a non reported model. NaN is failed computation. ExpertTraj ADE/FDE were taken from their paper. We notice sometimes, even if a model has a low ADE/FDE the AMD/AMV contradicts this by evaluating the overall generated samples.

	ETH	Hotel	Univ	Zara1	Zara2	Mean	(AMD+AMV)/2
	ADE/ FDE AMD/ AMV KDE						
S-GAN [8]	0.81/1.52 3.94/0.373 5.02	0.72/1.61 2.59/0.384 3.45	0.60/1.26 2.37/0.440 2.03	0.34/0.69 1.79/0.355 0.68	0.42/0.84 1.66/0.254 -0.03	0.58/1.18 2.47/0.361 2.23	1.42
S-STGCNN [23]	0.64/1.11 3.73/0.094 6.83	0.49/0.85 1.67/0.297 1.56	0.44/0.79 3.31/0.060 5.65	0.34/0.53 1.65/0.149 1.40	0.30/0.48 1.57/0.103 1.17	0.44/0.75 2.39/0.104 3.32	1.26
Trajectron++ [30]	0.39/0.83 3.04/0.286 1.34	0.12/0.21 1.98/0.114 -1.89	0.20/0.44 1.73/0.228 -1.08	0.15/0.33 1.21/0.171 -1.38	0.11/0.25 1.23/0.116 -2.43	0.19/0.41 1.84/0.183 **-1.09**	1.01
ExpertTraj [37]	0.30/0.62 61.77/0.034 NaN	0.09/0.15 21.30/0.003 NaN	0.19/0.44 M/M M	0.15/0.31 32.14/0.005 NaN	0.12/0.24 M/M M	**0.17/0.35** 38.4/0.004 NaN	19.20
Social-Implicit(ours)	0.66/1.44 3.05/0.127 5.08	0.20/0.36 0.58/0.410 0.59	0.31/0.60 1.65/0.148 1.67	0.25/0.50 1.72/0.078 1.24	0.22/0.43 1.16/0.106 0.69	0.33/0.67 **1.63/0.174** 1.85	**0.90**

Table 2. Ablation study of Social-Implicit components.

$\mathcal{L}_{\text{triplet}}$	$\mathcal{L}_{\text{G-distance}}$	$\mathcal{L}_{\text{G-angle}}$	Zones	AMD/KDE/AMV	AMD/AMV	ADE/FDE
				2.06/2.29/0.110	1.09	0.32/0.66
			✓	2.04/1.86/0.104	1.07	0.32/0.64
✓			✓	1.96/2.16/0.097	1.03	0.56/1.08
✓	✓		✓	2.13/2.32/0.092	1.11	0.32/0.62
✓		✓	✓	1.84/1.78/0.094	0.97	0.78/1.38
✓	✓	✓		2.29/2.76/0.090	1.16	0.35/0.71
✓	✓	✓	✓	1.63/1.85/0.174	**0.90**	0.33/0.67

of parameters size and inference time. Table 3 shows these results. The closest SOTA is ExpertTraj which Social-Implicit is 55x smaller and 8.5x faster.

7.4 Social-Zones ablation

Tab. 4 shows the ablation of the number of zones. Different zones affects the model's performance. Also, the model is sensitive to the zone's speeds. For example, when we changed the last zone from 1.2m/s to 0.6m/s the results changed. The 1.2m/s reflects human average walking speed, thus the 0.6m/s does not suit the data, hence it leads to poor performance in comparison with the 1.2m/s.

Table 3. Parameters counts and mean inference speed reported, benchmarked on the GTX1080Ti.

	Parameters count	Speed (s)
S-GAN [8]	46.3K (7.98x)	0.0968 (56.9x)
S-STGCNN [23]	7.6K (1.3x)	0.0020 (1.2x)
Trajectron++ [30]	128K (22.1x)	0.6044 (355.5x)
ExpertTraj [37]	323.3k (55.74x)	0.0144 (8.5x)
Social-Implicit (ours)	**5.8K**	**0.0017**

Table 4. Number of zones and their speed effect on our model accuracy.

#Zones	ADE/FDE	KDE	AMD/AMV	AVG
1	0.35/0.71	2.76	2.29/0.090	1.16
2	0.36/0.73	3.09	2.32/0.088	1.20
3	0.34/0.70	2.31	2.08/0.085	1.08
4@0.6m/s	0.34/0.69	2.85	2.31/0.080	1.19
4@1.2m/s	0.33/0.67	1.85	1.63/0.174	**0.90**

7.5 Qualitative Results

In Fig.8, we list two qualitative examples of our method and baseline models. In the first row, we see a pedestrian turns right at the end of the ground truth future. We notice that Social-Implicit and Trajectron++ cover the ground truth future well, whereas S-GAN and ExpertTraj give us an early turning prediction and concentrate away from the ground truth. The second row shows a zigzag walking pedestrian. Baseline models like S-STGCNN, Trajectron++, and ExpertTraj cannot generate good distributions to cover the ground truth trajectory, unlike ours and S-GAN. Although the prediction of ExpertTraj is close to the ground truth, ExpertTraj is over confident contradicting the ground truth. Qualitative results show that our predicted distribution are better.

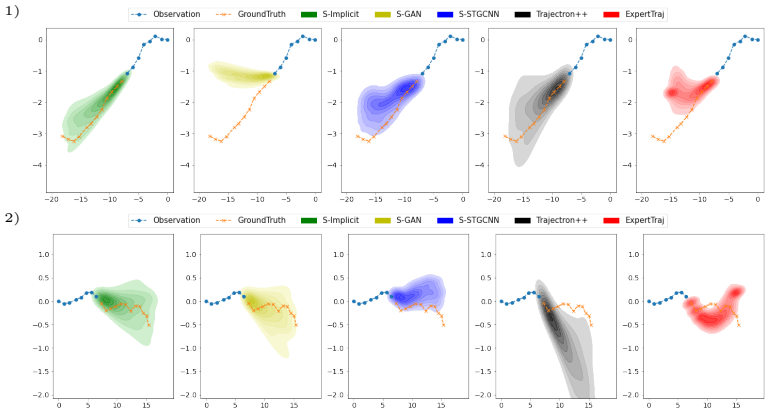

Fig. 8. Visualization of the predicted trajectories on the ETH/UCY datasets.

8 Conclusions

We introduced the AMD and AMV metrics that evaluate the distribution generated by a trajectory prediction model. We showed that the BoN ADE/FDE metric gives out an inadequate evaluation of the generated distribution. Based on the objective of AMD/AMV metrics to have a model that generates samples that are close to the ground truth with a tight variance, we introduced the usage of IMLE to train our model, Social-Implicit. We showed that Social-Implicit is a memory efficient model that runs in real-time and relies on several concepts such as Social-Zones, Social-Cell and Social-Loss to enhance the performance. Overall, we invite the motion prediction community to adapt the AMD/AMV to have a better evaluation of their methods.

References

1. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social lstm: Human trajectory prediction in crowded spaces. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 961–971 (2016)
2. Bütepage, J., Kjellström, H., Kragic, D.: Anticipating many futures: Online human motion prediction and generation for human-robot interaction. In: 2018 IEEE international conference on robotics and automation (ICRA). pp. 4563–4570. IEEE (2018)
3. Chai, Y., Sapp, B., Bansal, M., Anguelov, D.: Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. arXiv preprint arXiv:1910.05449 (2019)
4. Cui, H., Radosavljevic, V., Chou, F.C., Lin, T.H., Nguyen, T., Huang, T.K., Schneider, J., Djuric, N.: Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 2090–2096. IEEE (2019)
5. Deo, N., Trivedi, M.M.: Convolutional social pooling for vehicle trajectory prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 1468–1476 (2018)
6. Gauthier, J.: Conditional generative adversarial nets for convolutional face generation. Class Project for Stanford CS231N: Convolutional Neural Networks for Visual Recognition, Winter semester **2014**(5), 2 (2014)
7. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Advances in neural information processing systems **27** (2014)
8. Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., Alahi, A.: Social gan: Socially acceptable trajectories with generative adversarial networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2255–2264 (2018)
9. Huang, X., McGill, S.G., DeCastro, J.A., Fletcher, L., Leonard, J.J., Williams, B.C., Rosman, G.: Diversitygan: Diversity-aware vehicle motion prediction via latent semantic sampling. IEEE Robotics and Automation Letters **5**(4), 5089–5096 (2020)
10. Ivanovic, B., Pavone, M.: The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2375–2384 (2019)
11. Kosaraju, V., Sadeghian, A., Martín-Martín, R., Reid, I., Rezatofighi, S.H., Savarese, S.: Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks. arXiv preprint arXiv:1907.03395 (2019)
12. Laplante, J.N., Kaeser, T.P.: The continuing evolution of pedestrian walking speed assumptions. Institute of Transportation Engineers. ITE Journal **74**(9), 32 (2004)
13. Lee, N., Choi, W., Vernaza, P., Choy, C.B., Torr, P.H., Chandraker, M.: Desire: Distant future prediction in dynamic scenes with interacting agents. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 336–345 (2017)
14. Lerner, A., Chrysanthou, Y., Lischinski, D.: Crowds by example. In: Computer graphics forum. vol. 26, pp. 655–664. Wiley Online Library (2007)
15. Li, K., Malik, J.: Implicit maximum likelihood estimation. arXiv preprint arXiv:1809.09087 (2018)

16. Li, X., Ying, X., Chuah, M.C.: Grip: Graph-based interaction-aware trajectory prediction. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC). pp. 3960–3966. IEEE (2019)
17. Liang, M., Yang, B., Hu, R., Chen, Y., Liao, R., Feng, S., Urtasun, R.: Learning lane graph representations for motion forecasting. In: European Conference on Computer Vision. pp. 541–556. Springer (2020)
18. Limmer, M., Forster, J., Baudach, D., Schüle, F., Schweiger, R., Lensch, H.P.: Robust deep-learning-based road-prediction for augmented reality navigation systems at night. In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC). pp. 1888–1895. IEEE (2016)
19. Liu, H., Wang, L.: Human motion prediction for human-robot collaboration. Journal of Manufacturing Systems **44**, 287–294 (2017)
20. Liu, Y., Yan, Q., Alahi, A.: Social nce: Contrastive learning of socially-aware motion representations. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 15118–15129 (2021)
21. Mahalanobis, P.C.: On the generalized distance in statistics. National Institute of Science of India (1936)
22. Mangalam, K., An, Y., Girase, H., Malik, J.: From goals, waypoints & paths to long term human trajectory forecasting. In: Proc. International Conference on Computer Vision (ICCV) (Oct 2021)
23. Mohamed, A., Qian, K., Elhoseiny, M., Claudel, C.: Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14424–14432 (2020)
24. Pellegrini, S., Ess, A., Schindler, K., Van Gool, L.: You'll never walk alone: Modeling social behavior for multi-target tracking. In: 2009 IEEE 12th International Conference on Computer Vision. pp. 261–268. IEEE (2009)
25. Quehl, J., Hu, H., Taş, Ö.Ş., Rehder, E., Lauer, M.: How good is my prediction? finding a similarity measure for trajectory prediction evaluation. In: 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). pp. 1–6. IEEE (2017)
26. Rhinehart, N., Kitani, K.M., Vernaza, P.: R2p2: A reparameterized pushforward policy for diverse, precise generative path forecasting. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 772–788 (2018)
27. Rhinehart, N., McAllister, R., Kitani, K., Levine, S.: Precog: Prediction conditioned on goals in visual multi-agent settings. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2821–2830 (2019)
28. Rudenko, A., Palmieri, L., Herman, M., Kitani, K.M., Gavrila, D.M., Arras, K.O.: Human motion trajectory prediction: A survey. The International Journal of Robotics Research **39**(8), 895–935 (2020)
29. Sadeghian, A., Kosaraju, V., Sadeghian, A., Hirose, N., Rezatofighi, H., Savarese, S.: 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. pp. 1349–1358 (2019)
30. Salzmann, T., Ivanovic, B., Chakravarty, P., Pavone, M.: Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16. pp. 683–700. Springer (2020)
31. Sohn, K., Lee, H., Yan, X.: Learning structured output representation using deep conditional generative models. Advances in neural information processing systems **28**, 3483–3491 (2015)

32. Tang, C., Salakhutdinov, R.R.: Multiple futures prediction. Advances in Neural Information Processing Systems **32**, 15424–15434 (2019)
33. Tipping, M.E.: Deriving cluster analytic distance functions from gaussian mixture models. In: 1999 Ninth International Conference on Artificial Neural Networks ICANN 99.(Conf. Publ. No. 470). vol. 2, pp. 815–820. IET (1999)
34. Westphal, C.: Challenges in networking to support augmented reality and virtual reality. IEEE ICNC (2017)
35. Wu, P., Chen, S., Metaxas, D.N.: Motionnet: Joint perception and motion prediction for autonomous driving based on bird's eye view maps. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11385–11395 (2020)
36. Yuan, Y., Weng, X., Ou, Y., Kitani, K.: Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. arXiv preprint arXiv:2103.14023 (2021)
37. Zhao, H., Wildes, R.P.: Where are you heading? dynamic trajectory prediction with expert goal examples. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7629–7638 (2021)
38. Zhu, D., Zahran, M., Li, L.E., Elhoseiny, M.: Motion forecasting with unlikelihood training in continuous space. In: 5th Annual Conference on Robot Learning (2021)