Summary

This paper seeks to explore the effectiveness of non-contrastive learning in comparison to contrastive learning. This research is conducted by developing a feature extractor called NCLTP, that is used together with a state-of-the-art trajectory prediction model to observe that the use of non-contrastive learning can improve the accuracy and precision of existing trajectory prediction models by creating better and more meaningful feature representations.

Trajectory prediction for pedestrians, cars and other entities is an essential task for many tasks such as robot navigation and autonomous driving. However, predicting the future movement of these agents is not a trivial task. Due to the complexity of the problem, many different approaches have been attempted to try and solve the problem or further improve previous methods. Some of these methods are currently attempting to solve the problem using contrastive learning. This is an interesting approach that works by comparing two different samples and thereby learning by comparing the difference between the two samples. This approach has proved great results on a variety of different tasks, however, this approach can also be troublesome, as it can be hard to find the correct negative samples. This is due to the method requiring the negative samples to be of a certain difficulty. If the negative samples are too close to the positive samples, then the model will not be able to differentiate between them and likewise if they are too far from each other, then the model will not be able to learn anything meaningful. Therefore, the ability to select appropriate negative samples is a difficult task.

With this information, I propose a method using non-contrastive learning, which only utilizes positive samples, thereby neglecting the problem of mining for good negative samples. This method is based on performing different data augmentations and then letting the model learn the similarities between the two instead of looking for dissimilarities.

In the paper, I try three different data augmentation methods. The first one is applying noise to the coordinates of agent. The intuition is that agents nearby should behave similarly. The second one is scaling up the size of the bounding box of the agent, making them appear further away or closer. The intuition behind this is that an agent should behave similarly no matter if they are very close to the camera that recorded the footage or far away. The last augmentation is a shift, where a random time step is cut out of the observation sequence. This should change the speed of the agent as they now would be moving faster or slower during the observation length. To select the best augmentation method I perform an ablation study, where I test NCLTP, together with the trajectory prediction model, with each of the augmentations. Then I also try to make combinations of the different augmentations methods, for instance applying noise and then scaling the bounding boxes of the agents. Through this ablation study I was able to produce the data augmentation that performed the best.

The approach that I develop in this paper, NCLTP, was compared against multiple state-of-the-art models, and notably, a method that uses human labeled annotated data with action classes, meaning what the pedestrians are doing at a certain time frame, for instance standing or walking. Labeling data manually is a slow and potentially expensive task, which is why I suggest an approach that does not utilize them unless they provide a benefit that cannot be improved without them. However, through my experiments I show that the non-contrastive method that I suggest in this paper produces competitive, and in some cases outperforms, the current state-of-the-art models on both first-person view datasets and bird's-eye view datasets. Notably NCLTP also outperforms the method that uses additional information about the pedestrians, suggesting that the method in this paper can produce feature representations that are competitive and in many cases better than the current state-of-the-art contrastive learning trajectory prediction model. This paper provides a foundation for further exploration on this topic as I contemplate even better results can be produced given further research and additional experiments.

NCLTP: Non-Contrastive Learning for Trajectory Prediction

Søren Hjorth Boelskifte
Aalborg University
Department of Computer Science
Denmark, 9220 Aalborg
sboels18@student.aau.dk

***Abstract*—The ability to predict the trajectories of pedestrians and cars is an important task for tasks such as autonomous driving and navigation for robots. Many current state-of-art methods are trained using contrastive methods that require human labelled data about the pedestrians, for instance their current action e.g., walking or standing. Human labelled data is both expensive and time-consuming to produce. In this study, I will present a method using a non-contrastive method, which produces competitive results without the need for human labelled data. Instead of comparing the action labels of pedestrians, the model uses different augmentations of the data to learn similar representations. Experiments for the proposed method are conducted on both first-person view (FPV) datasets and bird's-eye view (BEV) datasets. This method provides competitive results to existing state-of-the-art methods, including methods that make use of human labeled annotations. The results of this paper should provide further research a base to work from and expand further upon this topic.**

1. Introduction

The ability to predict future trajectories of other pedestrians and vehicles is essential for many real world applications. Being able to predict future trajectory of other vehicles for instance allows an autonomous vehicle to plan safer paths and avoid potential collisions. However, predicting future trajectories is not a trivial task.

The trajectory of an agent depends on a long range of factors. These could include environmental factors, the type of action they are performing for instance running or walking, or it could be related to their social context. With the amount of factors that affect an agent's trajectory, many different approaches have been attempted to predict the future trajectory of the agent. Contrastive learning is an interesting approach that have been used by multiple previous works [1] [2] [3]. [4] attempted to include the actions that agents are performing to shape the latent space and thereby improve the feature representation that are used to predict the future trajectories. This approach improved an existing state-of-the-art model by using their framework jointly. However, this approach assumes that these action labels are available. These labels are expensive to acquire as they are often human annotated, or predicted by a model and then verified by a human, to ensure high accuracy of the labels [5] [6].

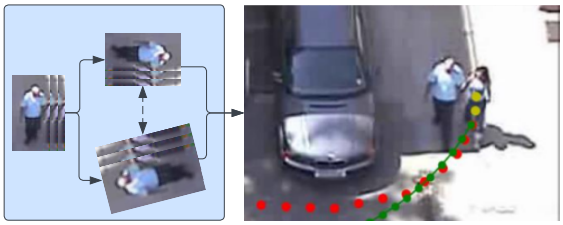

Figure 1: The use of two augmentations of the past trajectory to create positive samples that are compared to find similarities as proposed. The feature representations that are learned are then fed to a trajectory prediction model, which will then predict the future trajectory of the agent.

However, in this work I argue that these labels are not required, and non-contrastive learning can be utilized to avoid the need for these labels. Non-contrastive learning avoids one of the main challenges of contrastive learning, which is negative mining. This is how different samples are deemed to be similar or different. In [4] they use the action labels. Negative mining is a difficult task and will have a great effect on the effectiveness [7]. This is because if the negative samples are too easy or hard, then the model will not be able to learn what really distinguishes them.

Because of this I propose using a method similar to [8] [9] [10] [11] [12] [13] where only positive samples are used. This is done by using data augmentations and then training the model to see the similarities between the two augmentations instead of comparing the sample to a positive and negative sample [3]. This way negative sampling is avoided. I propose using this method as a pre-trained feature extractor for the downstream task of trajectory prediction. The feature extractor will be trained before and is integrated with the existing trajectory prediction model, which will be able to generate improved predictions based on the improved features from the feature extractor. This process is illustrated in Fig. 1.

I evaluate my method on both first-person view and bird's-eye view datasets and compare to existing methods. The datasets include both vehicles and pedestrians to potentially determine if this approach performs better for one type of agent. I show that the method suggested in this paper improves the performance of a previous state-of-the-art model when used together with the proposed method, NCLTP. The experiments show that the proposed method provide competitive results, and in many cases outperform, approaches that utilize contrastive learning and the human annotated labels in addition to previous state-of-the-art methods.

2. Related work

Unsupervised learning. One of the interesting aspects of unsupervised learning is that it can be performed without any labeled data compared to supervised learning [14] [15]. This provides a few advantages such as it allows for discovery of hidden patterns in the data. Since there are no explicit labels or pre-defined variables, a search for patterns in the data happens instead. This way patterns that might not be obvious to humans can be discovered. This can make this approach better at generalizing and can make it more scalable as larger datasets can be used, as there is no need to pre-label the data, which can be an expensive task [16] [17]. Contrastive learning is a promising technique that has proven to be effective, in the field of unsupervised learning, for a wide range of tasks [3] [18] [19] [20]. Contrastive learning works by learning data representations where positive (similar) samples are placed close together and negative (dissimilar) samples are placed further away from each other. Previous works [1] [2] [4] have attempted to use contrastive learning within trajectory prediction and proved great results. [4] used the action that the pedestrians are performing to separate samples in the feature space. A limitation with this approach is that there is a need for these labels.

Compared to the above methods which require labels or other forms of negative mining for their contrastive learning to shape the feature space, my approach does not require negative samples and only uses the positive samples and compares them to find similarities. My approach is inspired by previous works [8] [9] [10] [11] [12] [13], who have proved great performance from using self-supervised learning that do not have a need for negative samples. The benefit of this approach is that no labels are required, and no hard negative mining is required. This is a huge benefit as negative mining is a difficult task and will have a great effect on the effectiveness of the method [7]. If the negative samples are too easy or hard, then the model will not be able to learn what really distinguishes them. Finding the right approach to mining for negative samples is troublesome and therefore this approach has the benefit of not requiring negative samples. Instead, it compares different data augmentations of the same sample and finds similarities instead of differences.

Trajectory prediction. As mentioned above a trajectory prediction model is required for the use of the above mentioned approach. Because of this related works for trajectory prediction is needed to be explored shortly to select a model to use for the downstream task.

Within the task of trajectory prediction there are many different approaches. Some previous works have focused on uni-modal trajectory prediction [21] [22] [23] [24] [25] which is where they only output a single prediction for the future trajectory. The issue with this approach is that an agent has multiple different future trajectories which are plausible. However, a single prediction might also be what is required when using the model for an actual real-world task of for instance robot navigation. Some other recent works [26] [27] [28] [29] [30] [31] [32] are focused on multi-modal trajectory prediction. This is where multiple future trajectories are predicted. [33] [34] [35] produces multiple possible future trajectories by using a Conditional Variational Autoencoder (CVAE) and sampling from the latent space.

Some of the previous state-of-the-art models [36] [28] [37] [38] [33] have focused on the goal of the agent to help predict the future trajectory. This has includes predicting the long term end-goal [33], separating the prediction task into multiple stages and predicting potential targets and the likelihood of each [37] and combining the approximated goal with the past trajectory [38]. Because of the great results and being a state-of-the-art model, [33] is chosen as the trajectory prediction that will be combined with the above-mentioned approach, NCLTP, that I propose in this paper.

3. Problem Formulation

The trajectory prediction problem I formally formulate as:

For each available agent, an observed past trajectory is available at time step t, which is defined as $X_t = [x_1, ..., x_{t-1}, x_t]$. For the first-person view datasets each x includes the bounding box coordinates of the agent. This is defined as the center coordinate of the box and the width and height of the bounding box in pixels. For the bird's-eye view datasets x includes the x, y coordinates of the agent. Given X_t I predict the future positions of the agent $\hat{Y}_t = [\hat{y}_{t+1}, \hat{y}_{t+2}, ..., \hat{y}_T]$, where T is the number of frames that are to be predicted. The overall objective is to reduce the difference between predicted future trajectory Y and the ground truth future trajectory.

The training data is split into N training samples and for each of these samples $i \in [1, ..., N]$ the past trajectory X_t and the ground truth future truth trajectory Y_t is available.

4. NCLTP

In this section NCLTP will be further described in more detail. First, I will provide a short description of the trajectory prediction model that I have chosen to use as the base for NCLTP. This will be followed by the feature extractor module, NCLTP, that will be pre-trained before being used

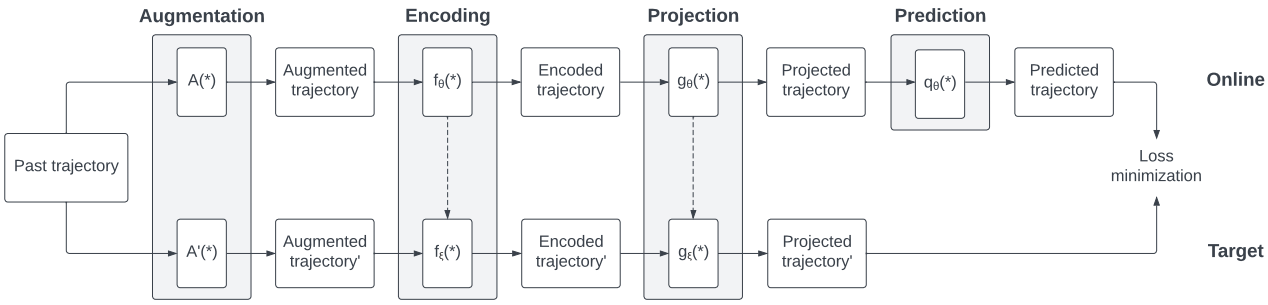

Figure 2: An overview of NCLTP. NCLTP tries to reduce the loss between the predicted trajectory from the online network and the projected trajectory from the target network. The online network is the upper process, and the target network is the bottom process. The weights in the target network are trained as an exponential moving average based on the online network, which is indicated by the dotted lines. When NCLTP is used for the downstream task of trajectory prediction then only the encoder is kept, and the encoded trajectory is used as the feature representation.

with the described trajectory prediction model. Then lastly I will highlight the augmentation methods that have been used.

4.1. Trajectory prediction model

Given the nature of the feature extractor, the actual trajectory prediction model that is used in conjunction is not restricted. The architecture of the trajectory prediction model could be vastly different, and the feature extractor could be combined with other trajectory prediction models. However, for the purpose of studying the effectiveness of the feature extractor, I have chosen to use the trajectory prediction model BiTrap [33]. I have chosen to use this model for two reasons. The first is that they produce great results and is a state-of-the-art trajectory prediction model. The second is that [4] have based their contrastive framework on the BiTrap code. By using the same base prediction model a fair comparison between their results and the results from the non-contrastive method I propose can be provided. This allows to highlight the promising nature of the proposed method, where action labelled data is not required. Bi-Trap's [33] implementation is based on a common encoder-decoder prediction model. I will extend their implementation with NCLTP to highlight the effectiveness of the proposed method.

4.2. Feature extractor

NCLTP is used together with a trajectory prediction model as a feature extractor that improves the performance of that trajectory prediction model. NCLTP is pre-trained independently before being used together with the trajectory prediction model. The goal of pre-training NCLTP is to learn a meaningful latent space from unlabeled trajectories. This is done by having an online and target network, where the online network tries to make predictions and the target network is used as a reference to compute a loss. The architecture of NCLTP is shown in Fig. 2. Pre-training the feature extractor starts with the past trajectory. This trajectory is then augmented by the augmentation module by both the online and target network. The augmentation block will be explained further detail in Section 4.3. This block is very important as it controls the effectiveness of how well the model will be able to learn. The output of the augmentation block is two different augmented versions of the past trajectory. These two augmented trajectories are the reason that no negative sampling is required, as now two different versions of the same trajectory is acquired instead. These augmented trajectories are then encoded by the encoders and used as a representation. Note that these encoders are slightly different and that the target network is trained as an exponential moving average based on the online network. This is indicated by the dotted lines in Fig. 2. The representation is then used to get a projection of the trajectory which is a lower-dimensional space. Similarly to the target network encoder, the target network projector is also an exponential moving average of the online network projector. Based on the projection of each of the trajectories the online network predictor tries to make a prediction of target network projection. The similarity between these two is used to train the model. The loss is calculated as:

$$L_{\theta,\xi} = 2 - 2 \cdot (z \cdot t) \tag{1}$$

Equation 1: NCLTP loss function component

z represents the normalized online trajectory prediction from the online network. Likewise, t represents the normalized target projection from the target network. $\cdot$ represents the inner product. Both augmentations of the past trajectory is passed through both the online and target network. This is done to help stabilize the networks and make the model more robust. This should also help the model learn to make better representations. Because of this the total loss for

NCLTP is calculated as follows:

$$\text{Loss} = L_{\theta,\xi} + L'_{\theta,\xi} \tag{2}$$

Equation 2: NCLTP loss function.

where $L_{\theta,\xi}$ represents the loss function component described in Eq. 1 with the first augmentation of the past trajectory. Likewise $L'_{\theta,\xi}$ represents the loss function component described in Eq. 1 with the other augmented past trajectory. The goal for the loss is the minimize the loss function with respect to the online network. The target is updated as a slowly moving exponential average as mentioned above. It is calculated as follows:

$$\xi \leftarrow \tau\xi + (1-\tau)\theta \tag{3}$$

Equation 3: Exponential moving average used to update the target network based on the online network.

where τ represents the target decay rate. The specific value used is mentioned in Section 5.2.

4.3. Augmentation methods

In this section the first block in Fig. 2, augmentation, will be described and explored further. The different augmentation methods that can be applied have an impact on the abilities of the model as a feature extractor. Determining which perform best and to what degree they influence will be discussed further in Section 5.5.

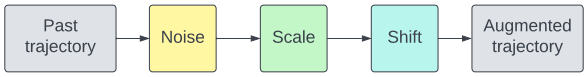

Figure 3: A more detailed view at the augmentation block in Fig. 2. When augmenting the past trajectory it is sequentially passed through the following augmentations before the final augmented trajectory is produced.

In Fig. 3 is a more detailed view of the exact augmentation methods that are applied to the trajectories during training. Each step is passed sequentially to each other meaning that the output of the noise augmentation is fed to the scale augmentation and so on. Each of these augmentation steps will now be highlighted separately.

Noise. The noise step is the first step in the augmentation block. The past trajectory is received as input and white noise is added to the bounding box of the agent for the first-person view datasets and white noise is applied to the x,y coordinates for the bird's-eye view datasets. Mathematically it is calculated as illustrated in Eq. 4, where x is the past trajectory, $\hat{x}$ is x with noise applied and $\mathcal{N}(0, w^2)$ is the Gaussian noise with mean 0 and standard deviation w applied to each element of the bounding box coordinates in x for the first-person view datasets and for the x,y coordinates for the bird's-eye view datasets. The value of w is explored further in Section 5.5. The intuition for adding white noise to the bounding box or coordinates of the agents is that the agents remaining relatively similar and therefore would likely behave similarly. This is what is trying to be captured when adding the noise augmentation.

$$\hat{x} = x + \mathcal{N}(0, w^2) \tag{1}$$

Equation 4: Noise augmentation

Scale. The scale step is the second step in the augmentation block. The past trajectory with noise is received as input and is then being scaled. This scaling is applied to the bounding box width and height. Mathematically it is calculated as illustrated in Eq. 5, where $\hat{w}$ and $\hat{h}$ represents the augmented width and height values respectively of the agent's bounding box. w and h represents the original width and height values of the agent's bounding box. The scale factor is a randomly chosen value drawn between a $lower$ and $upper$ bound. The values of $lower$ and $upper$ are further explored in Section 5.5. The output of this augmentation step is then the noised trajectory with a random scale applied to the bounding boxes of the trajectories. The intuition for adding scale as an augmentation is that the model should be able to handle agents being far away from the camera, meaning their bounding box is small, or very close to the camera, meaning their bounding box is very large. However, the distance from the camera does not necessarily speak to the behavior of the agent, which is why the scale is being applied so that the model can learn to ignore this and become more robust.

$$\hat{w} = w \cdot \text{scale\_factor} \tag{1}$$
$$\hat{h} = h \cdot \text{scale\_factor} \tag{2}$$

Equation 5: Scale augmentation

Shift. The shift step is the third and final step of the augmentation block. The past trajectory with both noise and scale applied is received as input. Then a random step in the observation length is chosen and cut from the observation. All the remaining bounding boxes for the first-person view datasets and x,y coordinates for the bird's-eye view datasets in the observation length is shifted and a new bounding box or x,y coordinate is then added to the end of the observation length, to preserve the length. It is calculated as illustrated in Eq. 6, where $rand(0, i-1)$ is a randomly chosen number between the first observation in the observation length and the last observation. $\hat{x}$ represents the new augmented trajectory with the shift applied. x is the past trajectory with the previous augmentations applied. x_{-2} represents the last two elements in the observation length of the trajectory. $\hat{x}_{-1}$ represents the last element in the observation length of the augmented trajectory. The intuition

for the shift augmentation is that the model should be able to handle varying speeds. By cutting out a time step the agent is moving faster in a short burst of time, in the span of one time step instead of the previous two, which could help the model become more robust and handle varying speeds and sudden movements. The output of the shift augmentation is the final augmentation that is applied to the trajectory as illustrated on Fig. 3. This augmentation is then passed along to the encoder as shown in Fig. 2.

$$\text{cut\_idx} = rand(0, i-1) \tag{1}$$
$$\hat{x} = x_i, \quad \text{for } i < \text{cut\_idx} \tag{2}$$
$$\hat{x} = x_{i+1}, \quad \text{for } i \geq \text{cut\_idx} \tag{3}$$
$$\text{last\_two} = \hat{x}_{-2} \tag{4}$$
$$\text{difference} = \text{last\_two}_1 - \text{last\_two}_0 \tag{5}$$
$$\text{new\_last} = \text{last\_two}_1 + \text{diff} \tag{6}$$
$$\hat{x}_{-1} = \text{new\_last} \tag{7}$$

Equation 6: Shift augmentation

5. Experiments and Results

In this section I will perform experiments using NCLTP as a feature extractor with a state-of-the-art trajectory prediction model BiTrap. First, I will describe the different datasets that have been used during the experiments. Then shortly the implementation details will be listed that were used during the experiments. This will be followed by a brief overview of the evaluation metrics that were used to evaluate NCLTP. An ablation study will then be conducted, where the effectiveness of the different augmentations will be explored. The results using the mentioned evaluation metrics will then be highlighted and discussed to show the benefits of using NCLTP by comparing against the current state-of-the-art models. Then lastly qualitative results will be highlighted.

5.1. Datasets

To evaluate my method, I have used both first-person view and third-person view datasets. The datasets used for each type of view will now briefly be mentioned.

First-Person View dataset. Joint Attention for Autonomous Driving (JAAD) [5] and Pedestrian Intention Estimation (PIE) [6] are two first-person datasets that have been used. Both datasets are recorded at 30 frames per second (fps). JAAD consists of 2800 pedestrian trajectories. PIE consists of 1835 pedestrian trajectories. I have followed the approach that was used in [6] [4] [24] [39] [33], where the dataset was split into a train, a validation, and a test set for both JAAD and PIE. The ratios were used are 50% train, 10% validation and 40% test. I use 0.5 seconds of observations (15 frames) to predict future trajectories of 0.5 (15 frames), 1.0 (30 frames) and 1.5 (45 frames) seconds.

Bird's-Eye View dataset. ETH [40] and UCY [41] are the third-person datasets that have been used. Both datasets are recorded at 2.5 frames per second (fps). They consist of 1,536 pedestrians in five different sets of data with four unique scenes. I follow the approach used in [42] [32] [33], which is a leave-one-out strategy to split the data into train and test sets. I use 3.2 seconds of observations (8 frames) to predict future trajectories of 4.8 seconds (12 frames).

5.2. Implementation details

NCLTP uses the following augmentations as illustrated in Fig. 3. First a small amount of noise, 5%, is applied to the past trajectory. Then the trajectory is scaled. The scale is set to a random amount between 80% to 120% of the original size. Then lastly a random step is cut from the observation length. Then a new step is inserted, which is based on the last two time-steps in the observation. The values for these augmentations are further explored in Section 5.5. For the bird's-eye view datasets the same augmentations are applied as described above except the scale step is skipped. The target decay rate τ is set to 0.99. The model was trained for 50 epochs with a batch size of 128.

I used the same hyper-parameters as in [33] for the BiTrap trajectory prediction model. This is done to properly illustrate the benefit of NCLTP in isolation.

5.3. Baselines

A range of models is used as a baseline to compare NCLTP against. This is done to show how NCLTP performs in comparison to existing state-of-the-art models. I evaluate the performance against BiTrap [33], which is the trajectory prediction model that NCLTP uses and should improve the effectiveness of. Then notably NCLTP is compared against ABC+ [4], which is a state-of-the-art model that uses contrastive learning and human annotated labels, in the form of action-classes, to learn feature representations. Similarly to NCLTP, ABC+ also acts as a feature extractor and is then used in conjunction with a trajectory prediction model, where they chose BiTrap. This comparison directly highlights the difference between using the human annotated and labelled data compared to the approach suggested in this paper as both methods are based on the same base trajectory prediction model, BiTrap. Additionally, the model is also compared against: A linear Kalman filter [6], Basic LSTSM [6], Bayesian-LSTM model [24], FOL-X [39], PIE$_{traj}$ [6], Social-LSTM [43], Social-GAN [32], MATF [44], FvTraj [45], STAR [46], Trajectron++ [42], CGNS [47], PECNet [28], Sophie [30] and DSCMP [48].

5.4. Evaluation Metrics

On JAAD and PIE, similar to previous works [33] [4] [6] [24] [39], I use average displacement error (ADE) and final displacement error (FDE). ADE measures the accuracy of the prediction along the entire trajectory. FDE on the other

Noise	Scale	Shift	JAAD			PIE		
			ADE ↓ (0.5s / 1.0s / 1.5s)	C_{ADE}↓ (1.5s)	CF_{FDE}↓ (1.5s)	ADE ↓ (0.5s / 1.0s / 1.5s)	C_{ADE}↓ (1.5s)	CF_{FDE}↓ (1.5s)
0.01			36 / 91 / 215	174	533	22 / 46 / 99	77	212
0.05			35 / 90 / 213	173	521	**15 / 37** / 88	65	197
0.20			36 / 91 / 214	173	539	17 / 43 / 104	81	253
	0.5-1.5		41 / 96 / 209	162	483	16 / 40 / 91	68	210
	0.8-1.2		37 / 92 / 208	165	489	**15 / 37** / 87	65	202
	0.9-1.1		36 / 93 / 215	172	529	17 / 40 / 95	72	234
		✓	35 / 89 / 205	163	503	16 / 39 / 92	69	220
0.05	0.8-1.2		38 / 94 / 218	176	538	16 / 42 / 102	80	254
0.05		✓	39 / 98 / 231	186	595	17 / 42 / 103	81	276
	0.8-1.2	✓	37 / 92 / 210	166	**478**	16 / 41 / 100	77	259
0.05	0.8-1.2	✓	**34 / 88 / 208**	**164**	494	**15 / 37 / 85**	**63**	**182**

Table 1: Ablation study of NCLTP on both JAAD and PIE. Lower is better which is denoted as ↓. Bold denotes the lowest value.

hand only measures the accuracy at the final position of the trajectory. ADE is calculated by the upper-left and lower-right coordinates of the bounding box. Following previous works [33] [4] [6] [24] [39] I also measure center average displacement error (C_{ADE}) and center final displacement error (CF_{FDE}). Results from JAAD and PIE are measured in pixels. On ETH/UCY, similarly to previous works [33] [28] [32] [30] [47] [44] [45] [48] [46] [42], I also use ADE and FDE similarly to JAAD and PIE. However, for ETH/UCY ADE and FDE is calculated in meters.

ADE is calculated as illustrated in Eq. 7.

$$ADE = \frac{1}{T}\sum_{t=1}^{T}\sqrt{(\hat{x}_t - x_t)^2 + (\hat{y}_t - y_t)^2} \quad (1)$$

Equation 7: Average displacement error.

where t is the current time step, and T is the total time. x_t and y_t represents the ground truth coordinates of the agent at time t. $\hat{x}_t$ and $\hat{y}_t$ represents the predicted agent location at time t. FDE is calculated as illustrated in Eq. 8.

$$FDE = \sqrt{(\hat{x}_T - x_T)^2 + (\hat{y}_T - y_T)^2} \quad (1)$$

Equation 8: Final displacement error.

where $\hat{x}_T$ and $\hat{y}_T$ represents the predicted agent location at the final time step T and x_T and y_T represents the ground truth agent location at the final time step T.

5.5. Ablation Study

To choose the best augmentation block as shown in Fig. 3, an ablation study was performed. The purpose of this ablation study is to define what augmentations are helpful and to what degree. The results of the ablation study are illustrated in Table 1.

The ablation study was performed by training NCLTP using different augmentations, as shown in Table 1, and then using NCLTP together with the BiTrap trajectory prediction model to observe the effects. The results are measured in ADE, C_{ADE} and CF_{FDE}. As previously mentioned the ablation study had two goals, figuring what augmentations worked and to what degree. Because of this the ablation study experiments with applying different values and ranges to the different augmentations. This together with results of the combinations of the different augmentations should provide a clear view into the effectiveness of augmentation methods. The ablation study provides some interesting results as it can be observed that the base cases, with only a single augmentation method, generally performed well. It can also be observed how different values and ranges affect the effectiveness of the augmentation method. For instance comparing using noise with 5% and 20% shows the clear benefit of only using a small amount of noise, however as the noise with 1% shows that using too little also negatively affects the effectiveness. Interestingly the combination of some of the augmentation methods performed worse than their the base augmentation counterparts alone. However, using all three augmentations together, as shown in Table 1, provided the best results on both JAAD and PIE. The results of this ablation study defined the shape of the augmentation block as illustrated in Fig. 3.

5.6. Comparison with the State-of-the-art

In the following I will evaluate NCLTP's performance when predicting both single and multiple possible future trajectories for the agent. This will be done on both the first-person view datasets, JAAD, and PIE, as well as the bird's-eye view datasets, ETH/UCY.

Uni-modal results on first-person view datasets. In Table 2 it is illustrated that NCLTP outperforms current

Method	JAAD			PIE		
	ADE ↓ (0.5s / 1.0s / 1.5s)	C_{ADE}↓ (1.5s)	CF_{FDE}↓ (1.5s)	ADE ↓ (0.5s / 1.0s / 1.5s)	C_{ADE}↓ (1.5s)	CF_{FDE}↓ (1.5s)
Linear [6]	233 / 857 / 2303	1565	6111	123 / 477 / 1365	950	3983
LSTM [6]	289 / 569 / 1558	1473	5766	172 / 330 / 911	837	3352
B-LSTM [24]	159 / 539 / 1535	1447	5615	101 / 296 / 855	811	3259
FOL-X [39]	147 / 484 / 1374	1290	4924	47 / 183 / 584	546	2303
PIE_{traj} [6]	110 / 399 / 1248	1183	4780	58 / 200 / 636	596	2477
BiTrap-D [33]	93 / 378 / 1206	**1105**	4565	41 / 161 / 511	481	1949
NCLTP	**89 / 366 / 1165**	1112	**4419**	**30 / 137 / 480**	**446**	**1837**

Table 2: Comparison of uni-modal results on JAAD and PIE. Results are noted in ADE, C_{ADE} and CF_{FDE}. Lower is better which is denoted as ↓. Bold denotes the lowest value.

Method	ADE (4.8s) ↓ / FDE (4.8s) ↓					
	ETH	HOTEL	UNIV	ZARA1	ZARA2	Avg
Social-LSTM [43]	1.09 / 2.35	0.79 / 1.76	0.67 / 1.40	0.47 / 1.00	0.56 / 1.17	0.72 / 1.54
Social-GAN [32]	1.13 / 2.21	1.01 / 2.18	0.60 / 1.28	0.42 / 0.91	0.52 / 1.11	0.74 / 1.54
MATF [44]	1.33 / 2.49	0.51 / 0.95	0.56 / 1.19	0.44 / 0.93	0.34 / 0.73	0.64 / 1.26
FvTraj [45]	0.62 / 1.23	0.53 / 1.10	0.57 / 1.19	0.42 / 0.89	0.38 / 0.79	0.50 / 1.04
STAR-D [46]	**0.56 / 1.11**	0.26 / 0.50	0.52 / 1.15	0.41 / 0.90	0.31 / 0.71	0.41 / 0.87
Trajectron++ [42]	0.71 / 1.68	**0.22 / 0.46**	0.41 / 1.07	0.30 / 0.77	0.23 / **0.59**	0.37 / 0.91
NCLTP	0.60 / 1.39	**0.22** / 0.54	**0.37 / 0.96**	**0.28 / 0.72**	**0.22** / 0.60	**0.33 / 0.84**

Table 3: Comparison of uni-modal results on ETH and UCY. Results are noted in ADE and FDE. Lower is better which is denoted as ↓. Bold denotes the lowest value.

Method (Best of 20)	JAAD			PIE		
	ADE ↓ (0.5s / 1.0s / 1.5s)	C_{ADE}↓ (1.5s)	CF_{FDE}↓ (1.5s)	ADE ↓ (0.5s / 1.0s / 1.5s)	C_{ADE}↓ (1.5s)	CF_{FDE}↓ (1.5s)
BiTrap-GMM [33]	153 / 250 / 585	501	998	38 / 90 / 209	171	368
BiTrap-NP [33]	38 / 94 / 222	177	565	23 / 48 / 102	81	261
ABC+ [4]	40 / 89 / **189**	**145**	**409**	16 / 38 / 87	65	191
NCLTP	**34 / 88** / 208	164	494	**15 / 37 / 85**	**63**	**182**

Table 4: Comparison of multi-modal results on JAAD and PIE. Results are noted in ADE, C_{ADE} and CF_{FDE}. Lower is better which is denoted as ↓. Bold denotes the lowest value.

state of the art models when only predicting a single future trajectory. On all metrics NCLTP performs better than the state-of-the-models except for C_{ADE} on the JAAD dataset, where it performs competitively. On the PIE dataset NCLTP outperforms the previous state-of-the-art trajectory prediction model BiTrap, which also happens to be the model that is used with NCLTP. NCLTP increases the effectiveness of BiTrap significantly, which shows the potential of the method that is proposed in this paper.

Uni-modal results on bird's-eye view datasets. Table 3 demonstrates NCLTP performance on ETH/UCY when only predicting a single future trajectory. It can be observed that NCLTP outperforms the previous state-of-the-art models on UNIV, ZARA1 and achieves competitive results on ETH, HOTEL and ZARA2. On average NCLTP performs better than the previous state-of-the-art models.

Multi-modal results on first-person view datasets. In Table 4 it is similarly illustrated that NCLTP outperforms the current state-of-the-art models, except for a few metrics where it performs competitively, when predicting multiple possible future trajectories. Compared to [4] on JAAD, NCLTP performs competitively on C_{ADE} and CF_{FDE} and even outperforms on $ADE0,5s$ and $ADE1.0s$. Compared to [4] on PIE, NCLTP has lower or equal ADE, C_{ADE} and CF_{FDE}. This is significant as [4] uses additional information in the form of manual human annotated labels, whereas NCLTP does not. This suggests that it is possible to learn equal, and possibly better, feature representations by using the approach suggested in this paper. Additionally, simply comparing against the base trajectory prediction model, BiTrap, NCLTP displays significant improvements, and outperforms the base BiTrap model on all evaluation metrics.

Multi-modal results on bird's-eye view datasets. Table 5 shows that NCLTP provides outperforms the previous state-of-the-art models when predicting multiple possible future trajectories. Comparing the previous state-of-the-art model BiTrap to NCLTP it can be observed that NCLTP out-

Method (Best of 20)	ADE (4.8s) ↓ / FDE (4.8s) ↓					
	ETH	HOTEL	UNIV	ZARA1	ZARA2	Avg
Social-GAN [32]	0.81 / 1.52	0.72 / 1.61	0.60 / 1.26	0.34 / 0.69	0.42 / 0.84	0.58 / 1.18
Sophie [30]	0.70 / 1.43	0.76 / 1.67	0.54 / 1.24	0.30 / 0.63	0.38 / 0.78	0.54 / 1.15
CGNS [47]	0.62 / 1.40	0.70 / 0.93	0.48 / 1.22	0.32 / 0.59	0.35 / 0.71	0.49 / 0.97
MATF GAN [44]	1.01 / 1.75	0.43 / 0.80	0.44 / 0.91	0.26 / 0.45	0.26 / 0.57	0.48 / 0.90
FvTraj [45]	0.56 / 1.14	0.28 / 0.55	0.52 / 1.12	0.37 / 0.78	0.32 / 0.68	0.41 / 0.85
DSCMP [48]	0.66 / 1.21	0.27 / 0.46	0.50 / 1.07	0.33 / 0.68	0.28 / 0.60	0.41 / 0.80
PECNet [28]	0.54 / 0.87	0.18 / 0.24	0.35 / 0.60	0.22 / 0.39	0.17 / 0.30	0.29 / 0.48
STAR [46]	0.36 / 0.65	0.17 / 0.36	0.31 / 0.62	0.26 / 0.55	0.22 / 0.46	0.26 / 0.53
Trajectron++ [42]	0.43 / 0.86	0.12 / 0.19	0.22 / 0.43	0.17 / 0.32	0.12 / 0.25	0.21 / 0.41
BiTrap-GMM [33]	0.40 / 0.74	0.13 / 0.22	0.19 / 0.40	0.14 / 0.28	0.11 / 0.22	0.19 / 0.37
BiTrap-NP [33]	0.37 / 0.69	0.12 / **0.21**	0.17 / 0.37	**0.13 / 0.29**	0.10 / **0.21**	0.18 / 0.35
NCLTP	**0.34 / 0.63**	**0.11** / 0.22	**0.16 / 0.36**	**0.13 / 0.29**	**0.09 / 0.21**	**0.16 / 0.34**

Table 5: Comparison of multi-modal results on ETH and UCY. Results are noted in ADE and FDE. Lower is better which is denoted as ↓. Bold denotes the lowest value

(a) (b) (c) (d)

Figure 4: Qualitative results of both the uni-modal and multi-modal trajectory prediction model on the JAAD dataset. The uni-modal trajectory model prediction is illustrated as the top row. The multi-modal trajectory model prediction is shown as the bottom row. The past observed trajectory of the agent is illustrated in dark blue. The ground truth future trajectory of the agent is shown in red. The predicted future trajectories by the trajectory prediction model using NCLTP is illustrated with green.

performs on ETH, HOTEL, UNIV and ZARA2 and provides competitive results on HOTEL and ZARA1. This provides clear evidence that NCLTP increases the effectiveness of BiTrap on bird's-eye view datasets, which helps indicate the promise of the method proposed in this paper.

5.7. Qualitative results

Fig. 4 illustrates four examples of NCLTP on the JAAD dataset. The top row illustrates the uni-modal results. The bottom row shows the multi-modal results. These examples show the robustness of NCLTP and how it handles different situations that the agent might be in. For instance, when observing (a) and (b) the model is able to handle different speeds of the agent as in (a) the agent is moving slow, which both the uni-modal and multi-modal model captures well. Similarly, in (b) the agent is moving quite quickly, and the uni-modal model does well at capturing this movement. The multi-modal model captures the other options that the agent could take, which results in a further spread of possible future locations. This is quite reasonable as the agent could move in many different ways. (c) and (d) shows an agent as they are about to or in the middle of crossing the street. Again, both the uni-modal and multi-modal model is doing a good job at capturing the behavior of the agent. Many of the predicted future trajectories are quite close to the actual ground truth trajectory and bounding box of the agent for both the uni-modal model but also the multi-modal model. This should help suggest the stability of NCLTP in a variety of cases.

6. Conclusion

In this paper, I have addressed the challenge of using expensive and time-consuming human labels for trajectory prediction. I proposed a method that utilizes only positive samples, compared to contrastive methods that use both positive and negative samples. This has removed the need

for negative mining, which is a difficult task as the effectiveness of the method using contrastive learning is highly dependent on how well the negative samples are mined [7]. Additionally, to only using positive samples, the proposed method also use data augmentations to learn similarities instead of comparing positive and negative samples. These samples are what often requires human labels, where the data augmentations can be made, importantly, automatically and quickly compared to the time-consuming process of manually labeling data. Therefore, the proposed method in this paper should provide an alternative to using expensive and time-consuming human labels when further research is done for trajectory prediction. Using the proposed method in this paper I have been able to produce a feature extractor that provides competitive, and in many cases even better, results to methods that use human annotated labels. The feature extractor was used in conjunction with a previous state-of-the-art model, BiTrap, and my experiments show that when combined with NCLTP, the model's performance is improved significantly. The model is performing competitively, and in many cases even better, than ABC+ [4] who is a current state-of-the-art model, that uses human annotated labels and is used jointly with BiTrap just as NCLTP. Additionally, BiTrap when used together with NCLTP outperforms previous state-of-the-art models on both first-person view datasets as well as bird's-eye view datasets. This indicates that the proposed method in this paper is both effective and robust. This research provides a foundation for future exploration and expansion upon this topic. Additionally, this work should illustrate a possible way of overcoming the challenge of expensive and often manual human labeling that is required for many state-of-the-art methods.

6.1. Future work

In this paper three different augmentation methods were used. It could be interesting to try to run experiments where additional augmentations were tested. It could prove that there are augmentation methods that perform even better than the ones tested in this paper. Additionally, as a small side detail it could be interesting to investigate if the order of the augmentations plays a role in the effectiveness of the model. For instance, in this paper only the order as shown in Fig. 3 was tested; therefore, it could be interesting to test the reserve order for instance or another random order. These findings in combination with new and additional augmentation methods could further improve the results that were found in this paper. Additionally, it could also be interesting to test the model on additional datasets to see if it performs better in different scenarios, which could include additional bird's-eye view datasets such as Stanford Drone Dataset [49]. It could also be interesting to try datasets that have even more complex annotated labels such as TITAN [50]. Lastly, it could be interesting to try and further improve the non-contrastive architecture as illustrated in Fig. 2. Perhaps it could be improved further by drawing inspiration from other non-contrastive methods from other areas than trajectory prediction such as [8] [9] [11] [12] [13].

Acknowledgements

I would like to thank my supervisors, Jilin Hu and Miao Zhang, for their supervision throughout this project.

Bibliography

1. Makansï, O., Ozgün, M., Içek, C., Marrakchi, Y. & Brox, T. On Exposing the Challenging Long Tail in Future Prediction of Traffic Actors.
2. Liu, Y., Yan, Q. & Alahí, A. A. Social NCE: Contrastive Learning of Socially-aware Motion Representations.
3. Khosla, P. *et al.* Supervised Contrastive Learning. https://t.ly/supcon.
4. Halawa, M., Hellwich, O. & Bideau, P. Action-based Contrastive Learning for Trajectory Prediction.
5. Kotseruba, I., Rasouli, A. & Tsotsos, J. K. Joint Attention in Autonomous Driving (JAAD). http://www.mobileye.com/.
6. Rasouli, A., Kotseruba, I., Kunic, T. & Tsotsos, J. K. PIE: A Large-Scale Dataset and Models for Pedestrian Intention Estimation and Trajectory Prediction. http://data.nvision2..
7. Jaiswal, A., Babu, A. R., Zadeh, M. Z., Banerjee, D. & Makedon, F. A SURVEY ON CONTRASTIVE SELF-SUPERVISED LEARNING.
8. Chen, X. & He, K. Exploring Simple Siamese Representation Learning.
9. Caron, M. *et al.* Emerging Properties in Self-Supervised Vision Transformers. https://github.com/facebookresearch/dino.
10. Grill, J.-B. *et al.* Bootstrap Your Own Latent A New Approach to Self-Supervised Learning. https://github.com/deepmind/deepmind-research/tree/master/byol.
11. Niizumi, D., Takeuchi, D., Ohishi, Y., Harada, N. & Kashino, K. BYOL for Audio: Self-Supervised Learning for General-Purpose Audio Representation.
12. Elbanna, G., Scheidwasser-Clow, N., Kegler, M., Beckmann, P. & Hajal, K. E. BYOL-S: Learning Self-supervised Speech Representations by Bootstrapping. *Proceedings of Machine Learning Research* **1,** 2022–2021. https://neuralaudio.ai/ (2022).
13. Cho, J. *et al.* Non-Contrastive Self-Supervised Learning of Utterance-Level Speech Representations. https://github.com/hyperion-ml/hyperion..
14. Marcus, G. *et al.* Deep Learning: A Critical Appraisal. http://www.nytimes.com/2012/11/24/science/scientists-see-advances-in-deep-learning-a-part-of-artificial-.
15. Noroozi, M. & Favaro, P. Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles.
16. Domingos, P. A Few Useful Things to Know about Machine Learning.
17. Dissertation, A & Tong, S. ACTIVE LEARNING: THEORY AND APPLICATIONS (2001).
18. Chen, X., Fan, H., Girshick, R. & He, K. Improved Baselines with Momentum Contrastive Learning.

19. Chen, T., Kornblith, S., Norouzi, M. & Hinton, G. A Simple Framework for Contrastive Learning of Visual Representations. https://github.com/google-research/simclr. (2020).
20. He, K., Fan, H., Wu, Y., Xie, S. & Girshick, R. Momentum Contrast for Unsupervised Visual Representation Learning. https://github.com/facebookresearch/moco.
21. Monti, A. *et al.* How many Observations are Enough? Knowledge Distillation for Trajectory Forecasting.
22. Bartoli, F., Lisanti, G., Ballan, L. & Bimbo, A. D. Context-Aware Trajectory Prediction in Crowded Spaces.
23. Yagi, T., Mangalam, K., Yonetani, R. & Sato, Y. Future Person Localization in First-Person Videos.
24. Bhattacharyya, A., Fritz, M. & Schiele, B. Long-Term On-Board Prediction of People in Traffic Scenes under Uncertainty.
25. Sun, J. *et al.* Human Trajectory Prediction with Momentary Observation.
26. Xu, C., Mao, W., Zhang, W. & Chen, S. Remember Intentions: Retrospective-Memory-based Trajectory Prediction. https://github.com/MediaBrain-.
27. Li, L., Pagnucco, M. & Song, Y. Graph-based Spatial Transformer with Memory Replay for Multi-future Pedestrian Trajectory Prediction. https://github.com/Jacobieee/ST-MR..
28. Mangalam, K. *et al.* It is not the Journey but the Destination: Endpoint Conditioned Trajectory Prediction. https://karttikeya.github.io/publication/htf/.
29. Wang, J., Ye, T., Gu, Z. & Chen, J. LTP: Lane-based Trajectory Prediction for Autonomous Driving.
30. Sadeghian, A. *et al.* SoPhie: An Attentive GAN for Predicting Paths Compliant to Social and Physical Constraints.
31. Lee, M. *et al.* MUSE-VAE: Multi-Scale VAE for Environment-Aware Long Term Trajectory Prediction.
32. Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S. & Alahi, A. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks.
33. Yao, Y., Atkins, E., Johnson-Roberson, M., Vasudevan, R. & Du, X. BiTraP: Bi-directional Pedestrian Trajectory Prediction with Multi-modal Goal Estimation. https://github.com/umautobots/.
34. Lee, N. *et al.* DESIRE: Distant Future Prediction in Dynamic Scenes with Interacting Agents.
35. Xu, C., Li, M., Ni, Z., Zhang, Y. & Chen, S. GroupNet: Multiscale Hypergraph Neural Networks for Trajectory Prediction with Relational Reasoning. https://github.com/MediaBrain-.
36. Rhinehart, N., Mcallister, R., Kitani, K. & Levine, S. PRECOG: PREdiction Conditioned On Goals in Visual Multi-Agent Settings. https://sites.google.com/view/precog..
37. Zhao, H. *et al.* TNT: Target-driveN Trajectory Prediction.
38. Deo, N. & Trivedi, M. M. Trajectory Forecasts in Unknown Environments Conditioned on Grid-Based Plans. https://github.com/nachiket92/P2T..
39. Yao, Y. *et al.* Egocentric Vision-based Future Vehicle Localization for Intelligent Driving Assistance Systems.
40. Pellegrini, S, Ess, A, Schindler, K & Van Gool, L. You'll Never Walk Alone: Modeling Social Behavior for Multi-target Tracking.
41. Leal-Taixé, L. *et al.* Learning an image-based motion context for multiple people tracking.
42. Salzmann, T., Ivanovic, B., Chakravarty, P. & Pavone, M. Trajectron++: Multi-Agent Generative Trajectory Forecasting With Heterogeneous Data for Control. https://github.com/StanfordASL/.
43. Alahi, A. *et al.* Social LSTM: Human Trajectory Prediction in Crowded Spaces.
44. Zhao, T. *et al.* Multi-Agent Tensor Fusion for Contextual Trajectory Prediction.
45. Bi, H., Zhang, R., Mao, T., Deng, Z. & Wang, Z. How Can I See My Future? FvTraj: Using First-person View for Pedestrian Trajectory Prediction.
46. Yu, C., Ma, X., Ren, J., Zhao, H. & Yi, S. Spatio-Temporal Graph Transformer Networks for Pedestrian Trajectory Prediction. https://github.com/Majiker/STAR.
47. Li, J., Ma, H. & Tomizuka, M. Conditional Generative Neural System for Probabilistic Trajectory Prediction.
48. Tao, C. Dynamic and Static Context-aware LSTM for Multi-agent Motion Prediction.
49. *Computational Vision and Geometry Lab* https://cvgl.stanford.edu/projects/uav_data/.
50. *TITAN - Honda Research Institute USA* https://usa.honda-ri.com/titan.