Human Trajectory Prediction in Socially Interacting Crowds

Haruki Nishimura and Bilan Yang
{hnishimura, bilany}@stanford.edu

December 17, 2018

Abstract

This project aims to predict human trajectories in dense crowds for robotic motion planning applications. Inspired by prior work [1] and a recent success in Convolutional Neural Networks (CNN) for sequence-to-sequence tasks, we propose a data-driven approach that employs CNN for trajectory prediction of multiple pedestrians at once. The performance evaluation on a publicly available dataset [2] suggests that our approach successfully predicts reasonable trajectories for multiple pedestrians in different scenes. The extensive error analysis and the trajectory analysis also reveal the cases where our model has difficulty in capturing unusual pedestrian behavior or social interactions.

1 Introduction

Despite recent advancements in planning and sensing technology for mobile robots, they are still incapable of achieving fully autonomous navigation in urban environments, especially in crowded scenes with multiple pedestrians. One of the key challenges in such densely populated scenes is trajectory forecasting of the pedestrians, which is crucial for safe and efficient robot navigation. A key aspect of the pedestrian motion is social compliance; with no explicit communication humans are already good at moving towards individual goal locations without hindering the paths of the others, even when there are no traffic lanes or rules in the scene. This now raises the question of how the robots can incorporate this notion of social compliance into the prediction model to enhance the safety and efficiency of the planning.

To address this challenging problem this paper proposes a novel data-driven approach to human trajectory prediction, especially in crowded scenes with multiple pedestrians. Given a history of the motion of pedestrians, our goal is to predict which trajectories they will follow in the near future up to a certain prediction horizon. In order to model this complex behavior, we use a deep learning technique to capture both salient and subtle features rather than manually designing them. Specifically, we propose a novel architecture based on Convolutional Neural Networks (CNN) that takes multiple pedestrian trajectories as its input and outputs the corresponding predictions. We train the network on a publicly available pedestrian motion dataset [2] and report the prediction performance.

2 Related work

Traditional methods for trajectory prediction use hand-crafted features to handle the interpersonal aspect of the human behavior. The Social Force model is one example that aims at simulating such behavior with a combination of attractive forces guiding the pedestrians towards their goals and repulsive forces encouraging collision avoidance [3]. Although a variety of extensions are proposed based on hand-crafted energy potentials established by relative distance and acceptable social norms, these methods have been outperformed by the recent data-driven approaches with the rise of deep neural networks. The learning-based methods are considered to be more capable of handling the stochastic nature of human behavior as compared to feature-matching methods [4].

The Recurrent Neural Networks (RNN) [5] are one of the most widely-used learning techniques to tackle the trajectory prediction problem. Highlighted by the capability to perform sequence-to-sequence modeling, RNNs have been successful in learning temporal sequences like future pedestrian trajectories conditioned on the trajectory history [6]. Combined with the Long Short-Term Memory (LSTM) networks [7], the RNN model retains long-term dependencies and avoids the vanishing and exploding gradient problems. Extensions of the RNN-based approach include the RNN Encoder-decoder framework with a Conditional Variational Autoencoder (CVAE) for trajectory prediction [8, 9]. This method has an advantage in handling the multimodality of trajectories by generating diverse hypotheses representing the multimodal probability distributions over plausible future actions. Generative Adversarial Networks (GANs) have been also employed to approximate socially-accepted motion trajectories in crowds [10, 11, 12].

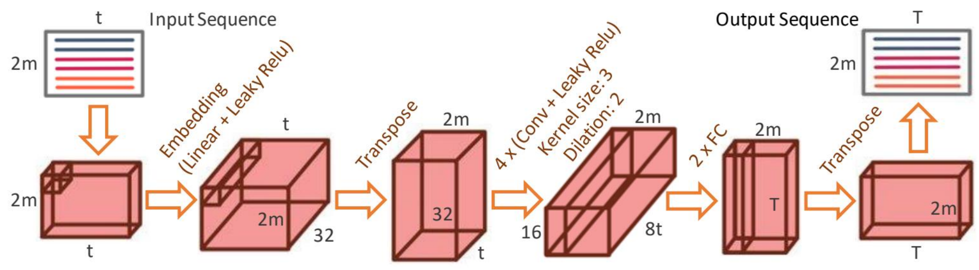


Figure 1: The proposed CNN-based architecture takes as inputs the first t frames of the trajectories of $2m$ pedestrians and outputs the next T frames of the predicted trajectories.

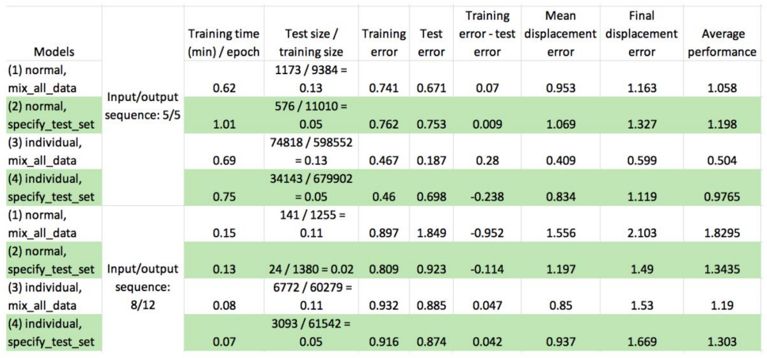
Besides RNNs, it has been recently reported that CNN-based models are well-suited to perform sequence-to-sequence tasks [13]. While the predictions with RNNs are inherently sequential and the later time-steps cannot be computed until the earlier ones are completed, CNN takes advantage of its parallelism and can perform training and inference in a more time efficient manner. Inspired by this recent success, we propose a CNN-based method to perform inference over future pedestrian trajectories when presented the corresponding history. Our work is closely related to [1], where the authors use a CNN for predicting pedestrian motion in crowded scenes. However the proposed framework in this paper is different from theirs in that it is designed to capture spatio-temporal interactions among pedestrians whereas their method can only handle individual pedestrian motion. This is possible by carefully designing the convolution layers so that the convolution operation runs in the space dimensions as well as the time dimension.

3 Dataset and Features

We use a publicly available dataset from Stanford Trajectory Forecasting Benchmark [2]. The dataset consists of 37 text files representing different crowd scenes, each of which contains the Cartesian coordinates of the pedestrians in the same scene at different time-steps along with the unique pedestrian IDs and the frame IDs. The data are originally collected from surveillance and drone camera videos taken at various outdoor public locations (e.g. Gates building on Stanford campus). The data are already manually labeled and pre-processed so that the coordinates of the pedestrians are extracted from the images and represented in the world frame corresponding to the 2D plane on the ground. Note that we did not use actual camera images or video streams in the training but only the coordinates.

From this data we build a custom input pipeline ourselves so the input (observed) trajectories and the output (predicted) trajectories are both represented as a 2D matrix, which is favorable for our CNN-based model. Specifically, each row of the matrix corresponds to either x or y coordinates of a single pedestrian whereas each column represents different time-steps. A single input-output pair is made of $t + T$ consecutive frames where t is the input sequence length and T is the prediction horizon. To construct the matrices from these frames we employ the following three distinct methods, each for a different purpose as explained below and in Section 4.2. In the first method, we concatenate m pedestrians that exist in the first t frames all the time. The training example is valid if the second T frames all contain exactly the same pedestrians. This is for the network to capture the inter-pedestrian features as explained in Section 4.1 in more detail. Note that the value of m varies across different examples in this case. In the second method, on the other hand, we only use a single pedestrian in each example. This is to test the basic performance of the network in predicting a single pedestrian motion as well as to increase the number of training examples that can be used. Lastly, the third method extends the first one and allows some pedestrians to appear or disappear during the $t + T$ frames. The missing coordinates are filled with 0. This is to give the network a flexibility to use incomplete sequences (i.e. some pedestrians having shorter lengths) as inputs. For all those three methods the coordinates of all the examples are uniformly normalized so that they range between -1.0 and 1.0.

Unfortunately, the size of the original dataset has been insufficient to effectively train our CNN model. In order to increase the number of examples we perform data augmentation by rotating and flipping the coordinates. The rotation was performed before the normalization to ensure that the potentially different normalization factors in x and y coordinates does not affect the rotated data. For the first and the third method of constructing matrices, we also permute the order of pedestrians that exist in the rows. For each augmented example the permutation of the input matrix is consistent with the output matrix so that the joint trajectories remain the same even after the representation changes.



Models		Training time (min) / epoch	Test size / training size	Training error	Test error	Training error - test error	Mean displacement error	Final displacement error	Average performance
(1) normal, mix_all_data	Input/output sequence: 5/5	0.62	1173 / 9384 = 0.13	0.741	0.671	0.07	0.953	1.163	1.058
(2) normal, specify_test_set		1.01	576 / 11010 = 0.05	0.762	0.753	0.009	1.069	1.327	1.198
(3) individual, mix_all_data		0.69	74818 / 598552 = 0.13	0.467	0.187	0.28	0.409	0.599	0.504
(4) individual, specify_test_set		0.75	34143 / 679902 = 0.05	0.46	0.698	-0.238	0.834	1.119	0.9765
(1) normal, mix_all_data	Input/output sequence: 8/12	0.15	141 / 1255 = 0.11	0.897	1.849	-0.952	1.556	2.103	1.8295
(2) normal, specify_test_set		0.13	24 / 1380 = 0.02	0.809	0.923	-0.114	1.197	1.49	1.3435
(3) individual, mix_all_data		0.08	6772 / 60279 = 0.11	0.932	0.885	0.047	0.85	1.53	1.19
(4) individual, specify_test_set		0.07	3093 / 61542 = 0.05	0.916	0.874	0.042	0.937	1.669	1.303

Figure 2: Training and testing results

4 Methods and Experiments

4.1 Network Architecture

Fig.1 illustrates the proposed CNN-based architecture. The input matrix first goes through the embedding layer to form a 3D tensor, which is needed to perform the convolution at a later step. The tensor is transposed before the convolution layers; this is to ensure that 1) the convolution operation runs over the spatial dimension across different pedestrians and the temporal dimension so the network captures spatio-temporal relationship of the pedestrians and 2) the network is agnostic to the value of m. The dilated convolution with dilation 2, kernel size 3 and stride 1 is used with padding $(0, 2)$. The dilation size 2 is chosen so that x-coordinates of pedestrian 1 is directly correlated to x-corrdinates of pedestrian 2, etc. After the convolution operation, two fully connected layers unroll the tensor into a matrix, which is transposed back to retain the output matrix. All the nonlinear activations are leaky-Relu, which has better performance than Relu activations in our case. Batch Normalization [14] is applied after each convolution layer to mitigate the covariate shift. Furthermore, Dropout [15] with rate 0.06 is performed between the two fully connected layers to reduce overfitting. The network was trained using Adam Optimizer [16] with learning rate 0.001 and weight decay 0.001 to minimize the standard L2 loss between the input matrix and the output matrix. This loss is calculated in the unnormalized coordinates to aim for accurate prediction in the original world frame. All the hyperparameters have been experimented with different values and the ones associated with the best performance were used for actual training.

4.2 Training Procedure

Our training models are categorized based on (1) the length of input and output trajectory sequence, (2) the way of how the dataset were prepared to feed into the proposed network, and (3) how data was assigned into training, development and test set.

(1) *Two types of input and output sequence.* Each input and output sequence pair is associated with different numbers of time steps in that sequence: $(t, T) = (5, 5)$ and $(t, T) = (8, 12)$.

(2) *Two methods of constructing one training example.* The first method ("normal") includes the valid trajectories of all pedestrians in one camera frame as one example so that pedestrians with potential social interactions are in the same example to be fed into the CNN architecture. However, since the number of trajectories can essentially be different from one example to another, only one example can be trained at a time (stochastic gradient descent) due to the dimension inconsistency. The second method ("individual") includes one trajectory in one example and completely disregards the temporal relationship among pedestrians.

(3) *Two ways of assigning data into training, dev, and test sets.* The first way guarantees data consistency by randomly mixing data from various crowd scenes and allocating them into the three sets with different ratios. This way of distribution is primarily used for diagnosing if a gap between the training and dev errors is from the insufficiency of training model generalization. The second way leaves out data from chosen crowd scenes before data mixing. The mixed data forms the training set; the leave-out data is split into dev and test sets.

The proposed CNN-based architecture[1] was constructed as described in Section 4.1 using the PyTorch framework [17]. For "normal" examples, stochastic gradient descent method was used with 500 epochs; for "individual" examples, a mini batch method with 100 batches was implemented with 1500 epochs. The batch size was selected to be large enough in order for the model to be sufficiently distinct from the that with "normal" examples.

[1]https://github.com/biy001/social-cnn-pytorch

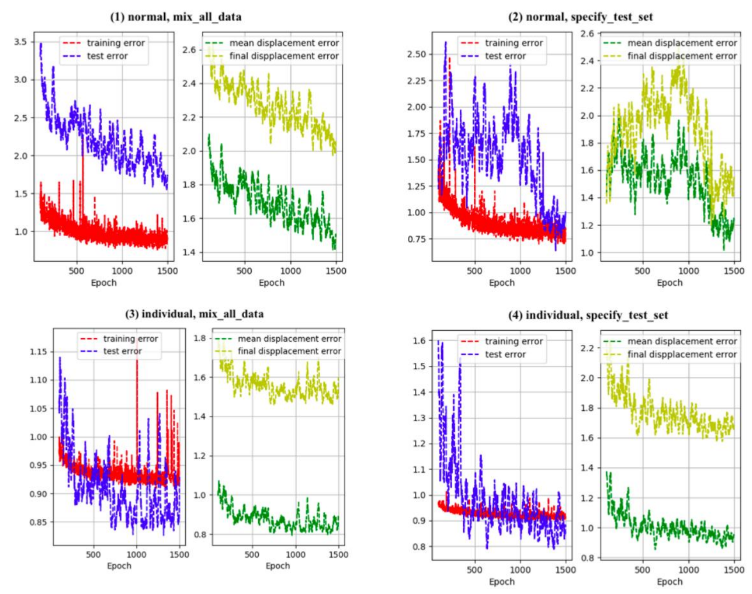


Figure 3: Loss curves of $(t, T) = (8, 12)$ and the associated accuracy metrics

5 Results and Discussion

The results associated with the 8 models are summarized as a table in Fig. 2. Training and test errors were averaged over the last 250 epochs for $(t, T) = (8, 12)$, and the last 100 epochs for $(t, T) = (5, 5)$ to avoid oscillations. The mean and final displacement errors are presented as accuracy metrics. The average performance is the average value of the mean and final displacement errors. The dev set is not shown since it has the same distribution as the test set. Loss curves of $(t, T) = (8, 12)$ are shown in Fig. 3 while those of $(t, T) = (5, 5)$ are dismissed due to limited space. Observations from the table and the loss curves include: (1) models of $(t, T) = (5, 5)$ always perform better than those of $(t, T) = (8, 12)$. (2) Datasets that were constructed with "individual" examples give much smaller errors in the case of $(t, T) = (5, 5)$ but may not have an advantage with $(t, T) = (8, 12)$. (3) The choice of whether or not to mix all data before allocating the training, dev, and tests does not have a clear impact on the performance. Meanwhile, this choice affects the size of the sets, which may be responsible for the difference in performance.

The performance of the models is largely impacted by the amount of data available for each model. The "normal" examples represent a larger dataset size but their associated models are trained faster than others. The mean and final displacement errors are generally correlated to the test error. The error analysis section discusses the training and test errors of different models; the trajectory analysis section uses trajectory plots to back up the error analysis and show the quality of predictions.

5.1 Error Analysis

That models with $(t, T) = (5, 5)$ outperforms those with $(t, T) = (8, 12)$ can be attributed to the fact that $(t, T) = (8, 12)$ simply represents a more complicated prediction task. It is always harder to predict longer sequence as the uncertainty of the pedestrian movement pattern keeps increasing along the time. Another reason could be that the case of $(t, T) = (5, 5)$ has a larger amount of training examples available, which simply facilitates learning.

With $(t, T) = (5, 5)$, the training error has nearly 40% decrease when the model changes from using "normal" examples to "individual" examples. The test errors and displacement errors all decrease accordingly, yet not as significant as the training errors. This boost in performance can be explained by the sheer size of the dataset with "individual" examples, which is as 7 times large as that with "normal" examples. With $(t, T) = (8, 12)$, however, "individual" examples give a slightly larger training error but actually have a smaller test error. It seems that the "normal" model over-fits with insufficient amount of training data but captures the social context quite well in training with a smaller training error.

The choice of mixing data before assigning to the three sets results in a slightly larger training error regardless of the length of the input and output sequence, which may be because the training dataset is smaller in this case. Generally, it does not noticeably affect the variance with training and test errors very close to each other. Exceptions include the case of $(t, T) = (8, 12)$ with "normal" examples. Besides the discussion in the previous paragraphs, that the test set is extremely small passes on instability to the test performance.

There are several other cases that have a gap between the training and the test errors. The case of $(t, T) = (5, 5)$ with specified test set experiences the high variance issue. Reasons could include that the test set has a relatively ratio to the training set or the chosen test set is particularly difficult for $(t, T) = (5, 5)$ with "normal" examples. Another unusual case of test error being apparently lower than the training error occurs with $(t, T) = (5, 5)$, "individual" examples, and mixed datasets.

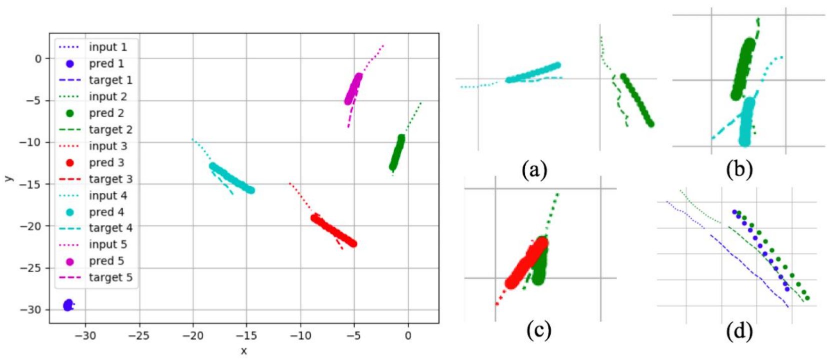


Figure 4: Left: examples of the typical output trajectory from the test set; right: examples of challenging predictions

Possible explanations could be that the test set is not diversified enough and is actually fairly "easy" for the prediction task. Some datasets of particular crowd scenes contain a lot more raw data trajectories than others and thus have a higher impact on the performance. Undesirable predictions could occur when the data are not shuffled sufficiently, which leads to inconsistent distributions in training and test sets.

5.2 Trajectory Analysis

Our models successfully predict the trend of the pedestrian trajectory with reasonable mean and final displacement errors. All trajectory examples shown in this section are with $(t, T) = (8, 12)$. The trajectory plot on the left of Fig.4 is a typical example of the predicted trajectory from the test set. The challenge lies on foreseeing the future trajectory with a sense of uncertainty inherent from the human movement mode as well as capturing the social interactions among pedestrians. On the right side of Fig.4 are observations from some interesting cases that give insight on the capability of the predictor. Fig.4(a) shows a case when a pedestrian frequently changes the direction of movement, yet the predictor outputs a linear trajectory based on the moving trend of the input sequence. Fig.4(b) is an example showing that two interacting input sequences will not affect the output sequence. Fig.4(c), however, presents an opposite case the predicted trajectories seem to be too close to each other, which poses a risk of collision. Fig.4(d) gives a perfect example of what happens for the performance of model (2) "normal, specify_test_set" in the table shown in Fig. 2 to have a relatively low final displacement error but a high mean displacement error. It is an unusual case that the prediction starts relatively far from the end of the input sequence but manages to go closer to the actual trajectory at the end. Future work will include detailed investigations on prediction failures.

5.3 Other Experiments

Besides the models described in Section 4.2, experiments have been performed with dataset that have missing coordinates filled with 0, which is described as the third method of representing the input and output sequences in Section 3. The dataset constructed with this method has a very large size, which makes the training extremely slow. At an epoch of 200, the corresponding training error seems to stabilized at a value about 20 times as large as that from any other model. The learning failure may be attributed to the confusion of the network for treating the filled zero values simply as un-learnable space-holders or as actually physical coordinates. When there are too many zero values being filled into the training examples, the network is discouraged from high-quality learning.

We have also considered to compare the performance of our model with other existing methods but several difficulties have hindered us from measuring results of other models. First, we initially considered using the benchmarking tool at Stanford Trajectory Forecasting Benchmark website with a set of quantitative metrics for performance evaluation [2]. However, a website issue seems to prevent effective submission of our predictions. Secondly, we tried running the social LSTM [6] and social GAN models [11] but they either output incorrect accuracy metrics or have unrevealed data preprocessing procedures. A lot of other methods even do not provide open source code.

6 Conclusion and Future Work

In this work we have presented a novel data-driven approach for human trajectory prediction using CNN. The proposed model's capability to learn reasonable predictions given a trajectory history of multiple pedestrians was demonstrated. However, whether our model truly captures pedestrian interactions still needs to be verified in comparison to other methods, which should be all trained with much larger data. Also, further error analysis using raw image data would clarify the situations where our model had difficulty to predict the trajectories. In the future, we are interested in conducting a formal comparative study with other existing methods as well as extending the current approach to account for multimodality and scene-specific contexts to further improve the prediction performance.

Contributions

Haruki Nishimura: conducted literature review; modified existing social-lstm code, especially the input pipeline and the training pipeline; prepared input pipeline for the proposed model; wrote abstract, Sections 1-3, 4.1 and 6 of the final report. Bilan Yang: conducted literature review; built the training network; worked on loss curves and trajectory plots; wrote the rest of the report

Acknowledgements

The authors are grateful to Amir Sadeghian and the CS230 2018-2019 Autumn Quarter teaching staff, especially Abhijeet Shenoi for helpful guidance and discussions.

References

[1] N. Nikhil and B. T. Morris, "Convolutional neural networkfor trajectory prediction," *arXiv preprint arXiv:1809.00696*, 2018.

[2] A. Sadeghian, V. Kosaraju, A. Gupta, S. Savarese, and A. Alahi, "Trajnet: Towards a benchmark for human trajectory prediction," *arXiv preprint*, 2018.

[3] D. Helbing and P. Molnar, "Social force model for pedestrian dynamics," *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[4] Y. F. Chen, M. Everett, M. Liu, and J. P. How, "Socially aware motion planning with deep reinforcement learning," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 1343–1350, IEEE, 2017.

[5] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[6] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 961–971, 2016.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 336–345, 2017.

[9] E. Schmerling, K. Leung, W. Vollprecht, and M. Pavone, "Multimodal probabilistic model-based planning for human-robot interaction," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–9, May 2018.

[10] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[11] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social gan: Socially acceptable trajectories with generative adversarial networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, no. CONF, 2018.

[12] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, and S. Savarese, "Sophie: An attentive GAN for predicting paths compliant to social and physical constraints," *CoRR*, vol. abs/1806.01482, 2018.

[13] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *arXiv preprint arXiv:1705.03122*, 2017.

[14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[16] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.